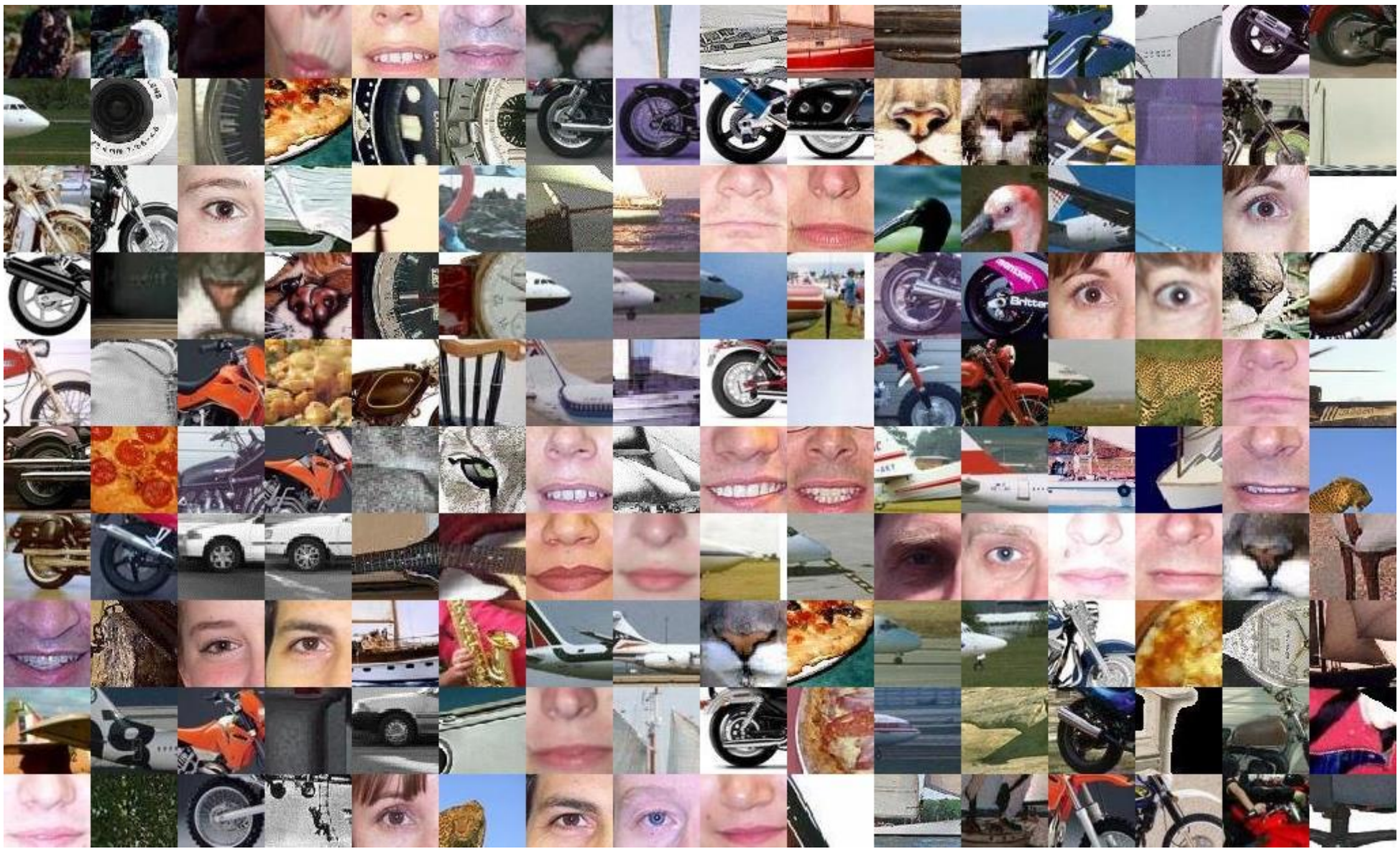
Feature detectors and descriptors



ICS 483 Computer Vision
Fall 2025, Lecture 9/10

Overview of today's lecture

- Why do we need feature descriptors?
- Designing feature descriptors.
- Local feature descriptors
  - SIFT descriptor.
- Global feature descriptors
  - GIST descriptor.
  - HoG descriptor

Slide credits

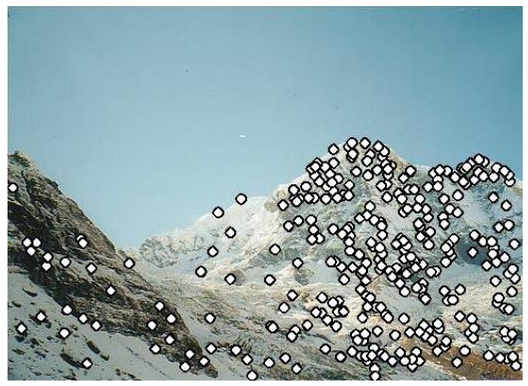
Most of these slides were adapted from:

- Matt O'Toole (CMU, 15-463, Fall 2022).
- Ioannis Gkioulekas (CMU, 15-463, Fall 2020).
- Kris Kitani (CMU, 15-463, Fall 2016).
- Noah Snavely (Cornell)

Recap

Local features: main components

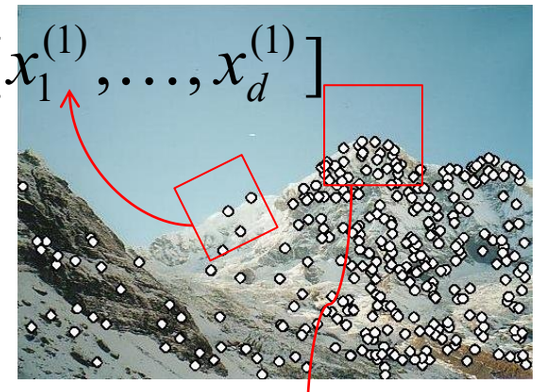
1) Detection: Identify the interest points



Last 2 lectures

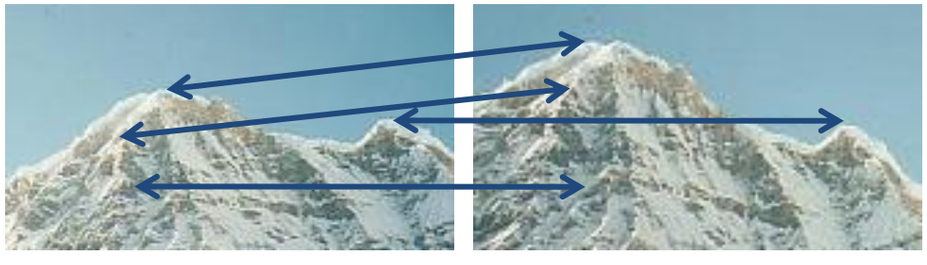
2) **Description: Extract vector feature descriptor surrounding each interest point.**

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

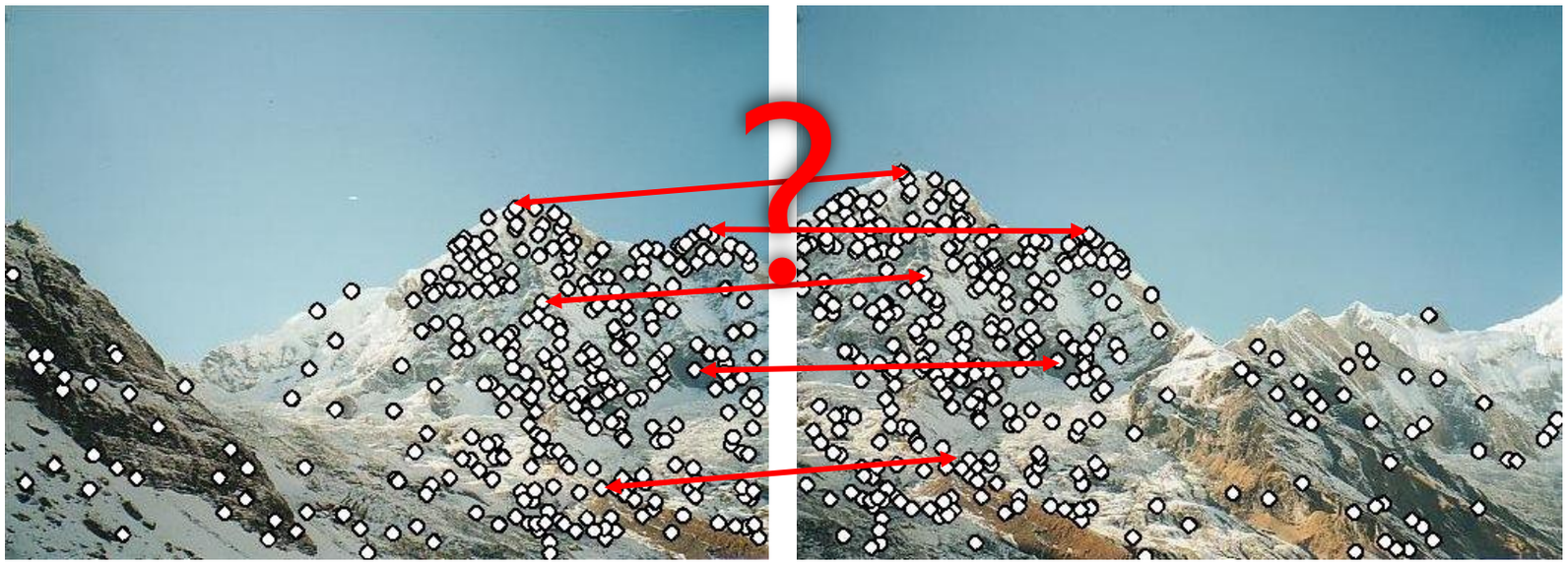
3) Matching: Determine correspondence between descriptors in two views



Kristen Grauman

Feature descriptors

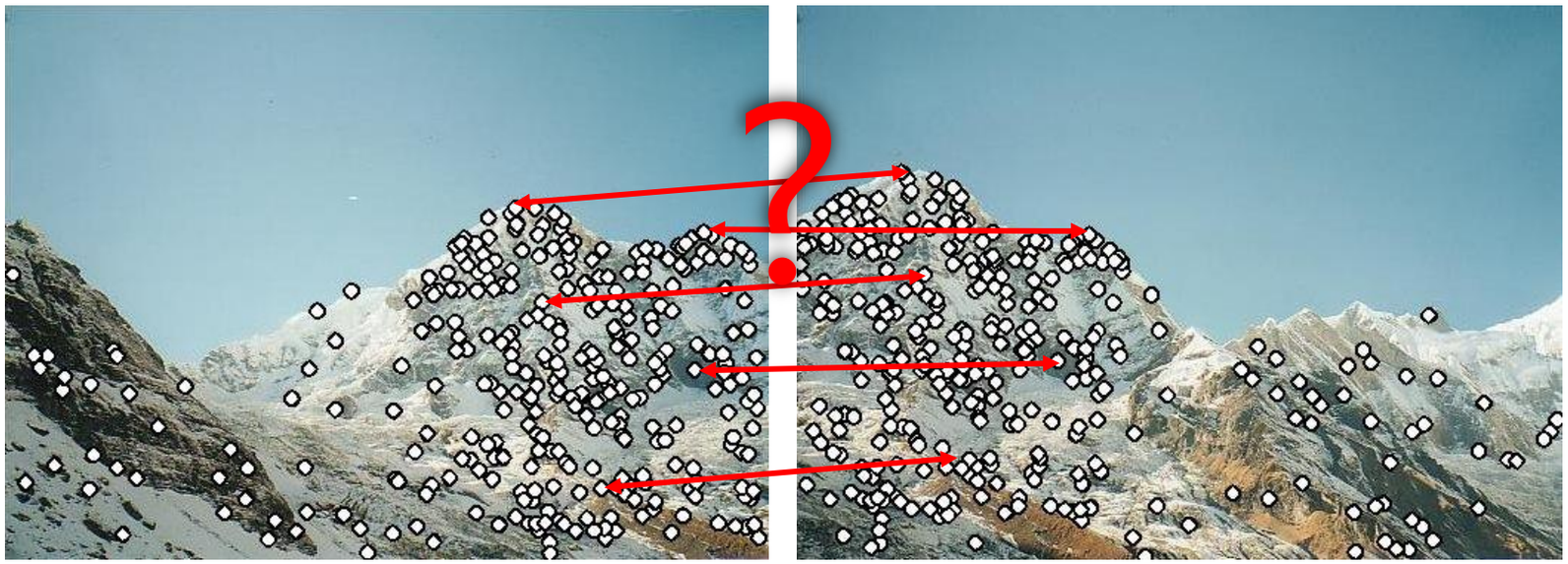
We know how to detect good points
Next question: **How to match them?**



Answer: Come up with a *descriptor* for each point, find similar descriptors between the two images

Feature descriptors

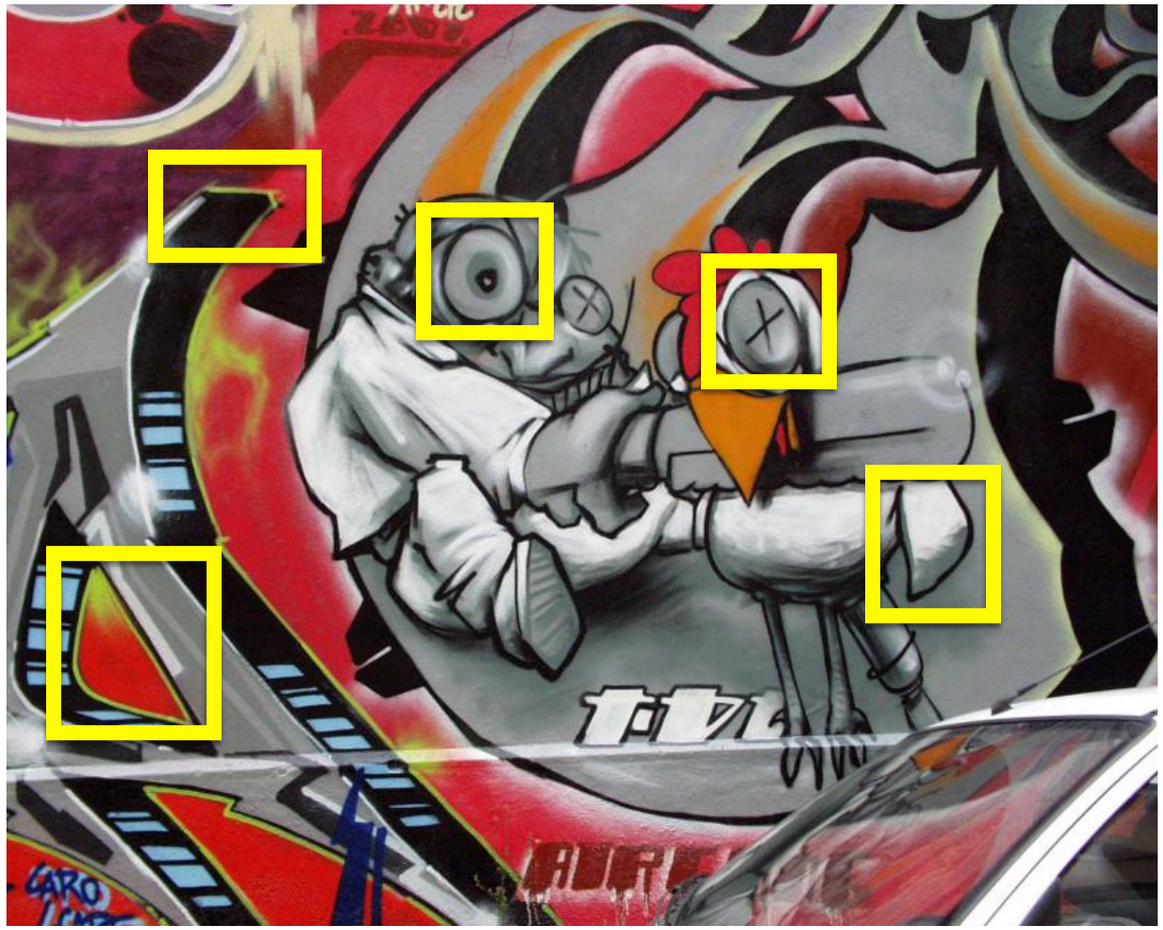
We know how to detect good points
Next question: **How to match them?**

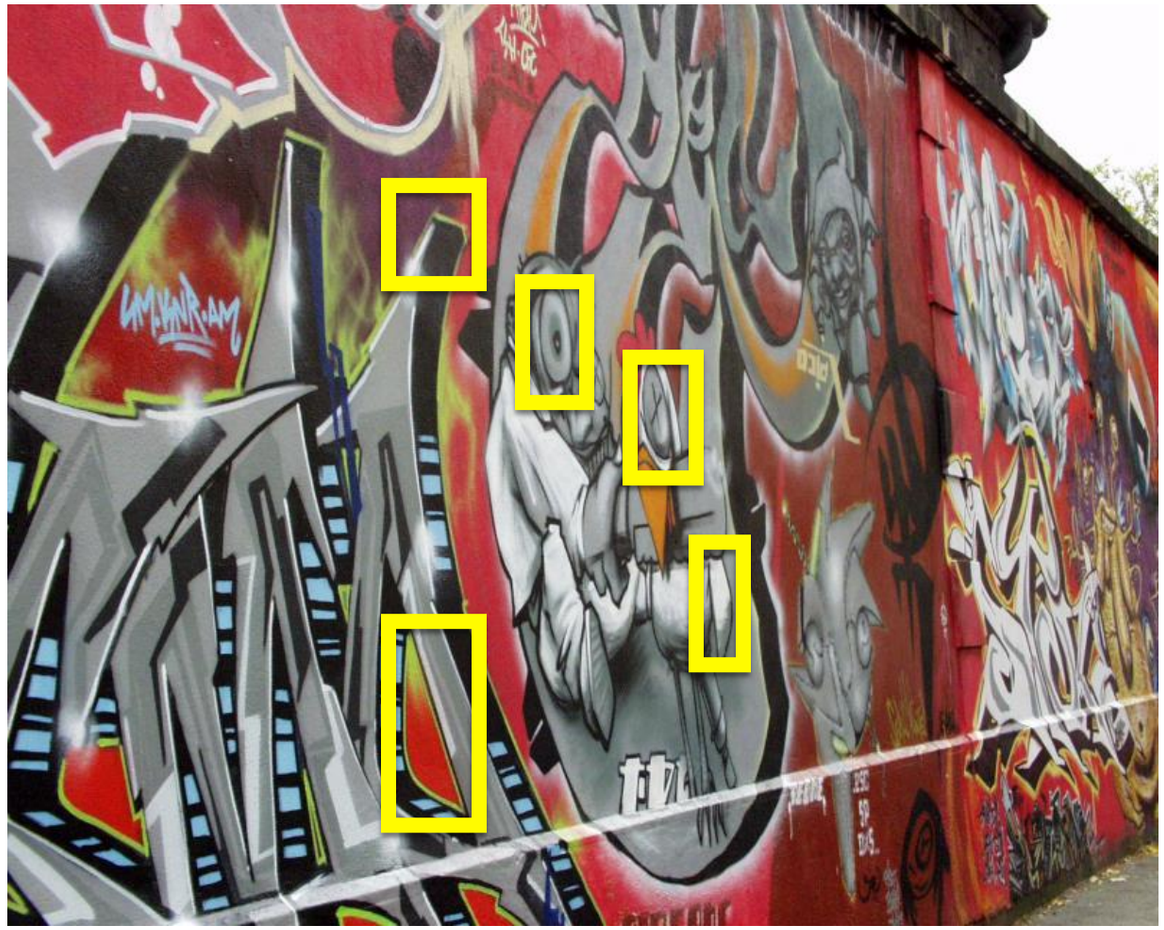


Lots of possibilities

- Simple option: match square windows around the point
- "State of the art" approach: SIFT
  - David Lowe, UBC http://www.cs.ubc.ca/~lowe/keypoints/

Why do we need feature descriptors?





If we know where the <u>good</u> features are,
how do we <u>match</u> them?

How do we describe an image patch?

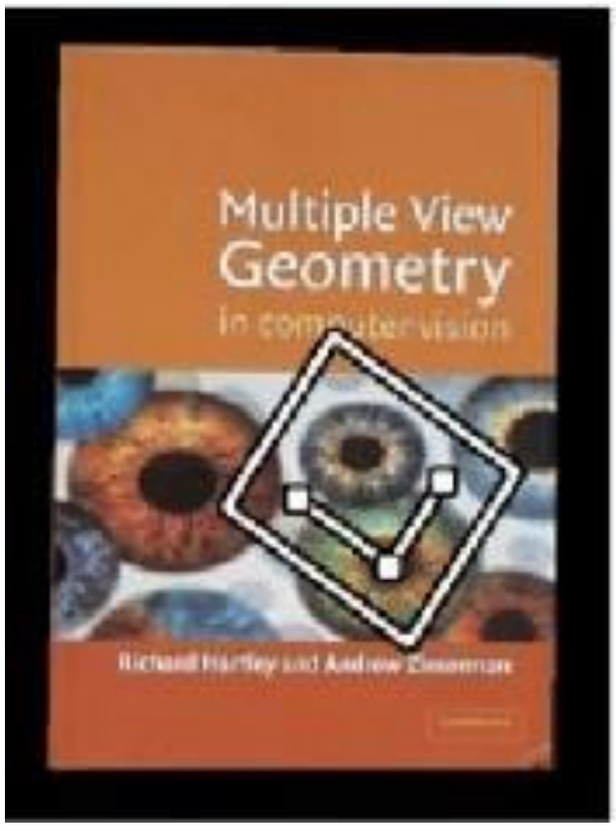
Patches with similar content should have similar descriptors.

Designing feature descriptors

What is the best descriptor for an image feature?

Photometric transformations

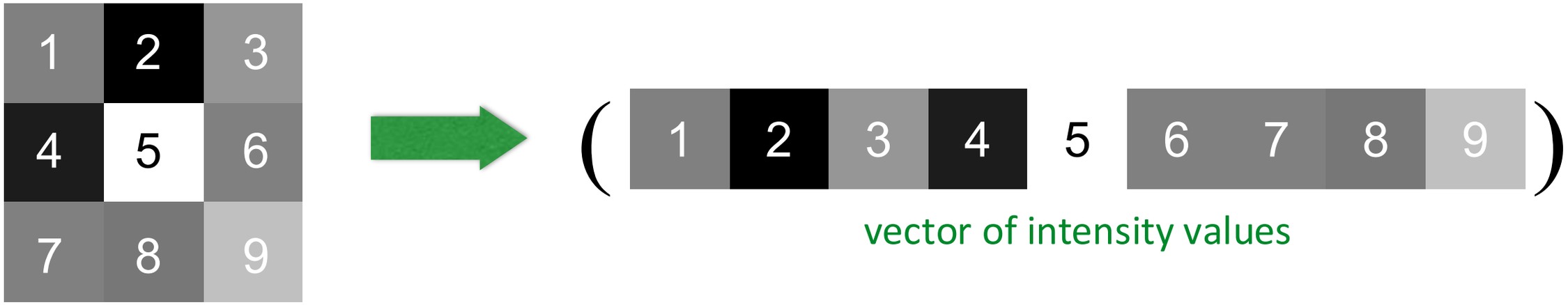
Geometric transformations



objects will appear at different scales, translation and rotation

Image patch

Just use the pixel values of the patch!



Perfectly fine if geometry and appearance is unchanged

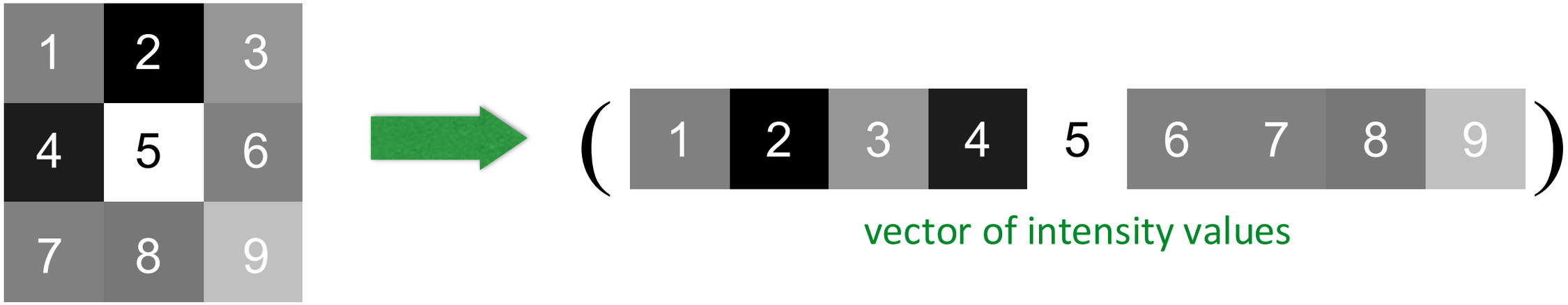
(a.k.a. template matching)

Tiny Images

Just down-sample it!
Simple, fast, robust to small affine transforms.

Image patch

Just use the pixel values of the patch!



vector of intensity values

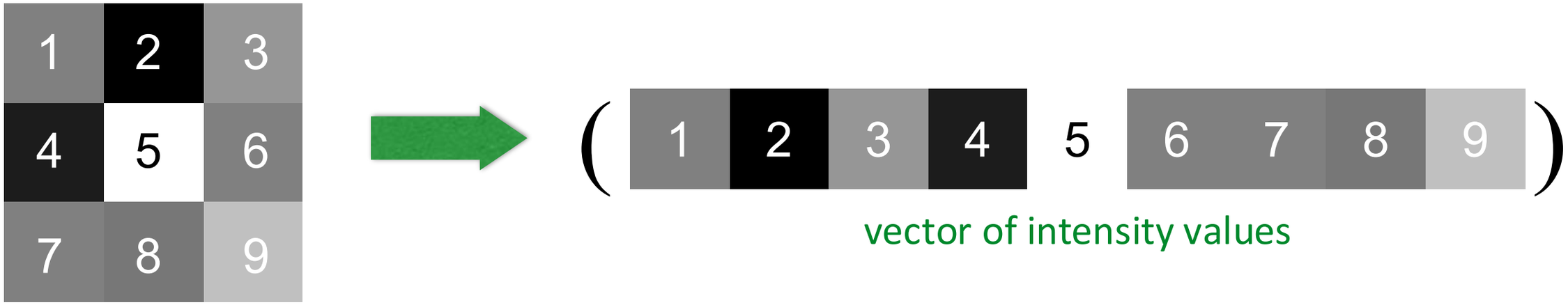
Perfectly fine if geometry and appearance is unchanged

(a.k.a. template matching)

What are the problems?

Image patch

Just use the pixel values of the patch!



Perfectly fine if geometry and appearance is unchanged

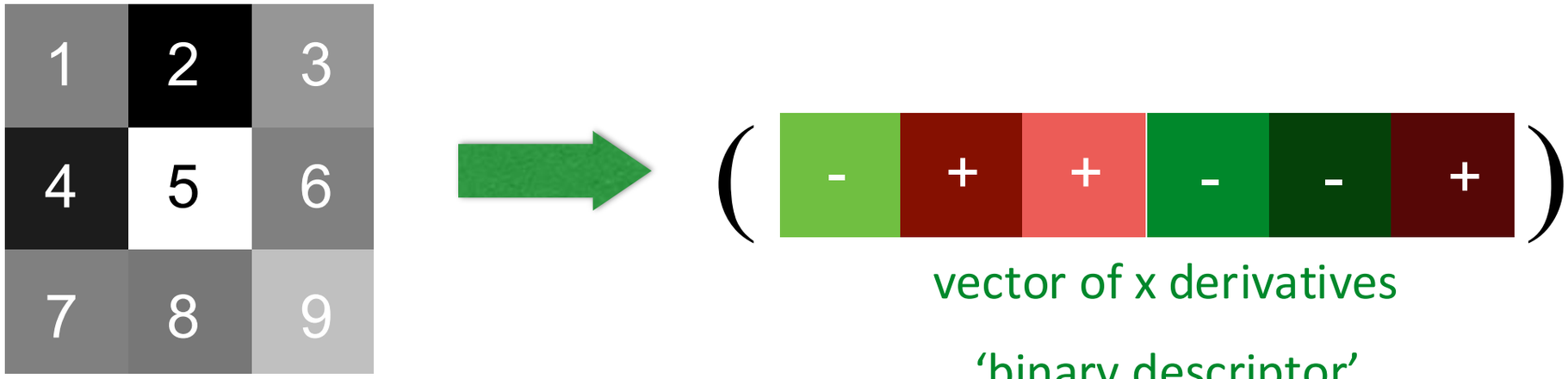
(a.k.a. template matching)

What are the problems?

How can you be less sensitive to absolute intensity values?

Image gradients

Use pixel differences

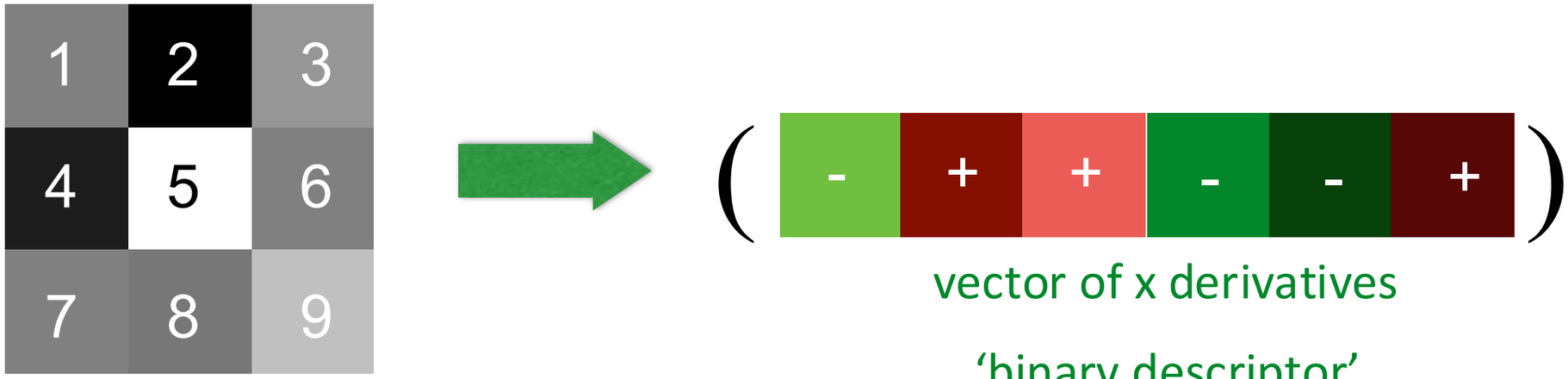


Feature is invariant to absolute intensity values

What are the problems?

Image gradients

Use pixel differences



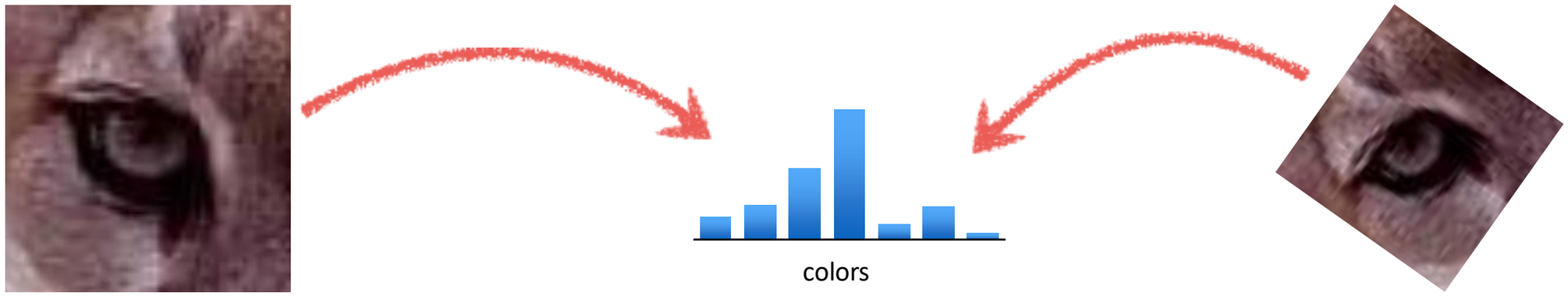
Feature is invariant to absolute intensity values

What are the problems?

How can you be less sensitive to deformations?

Color histogram

Count the colors in the image using a histogram

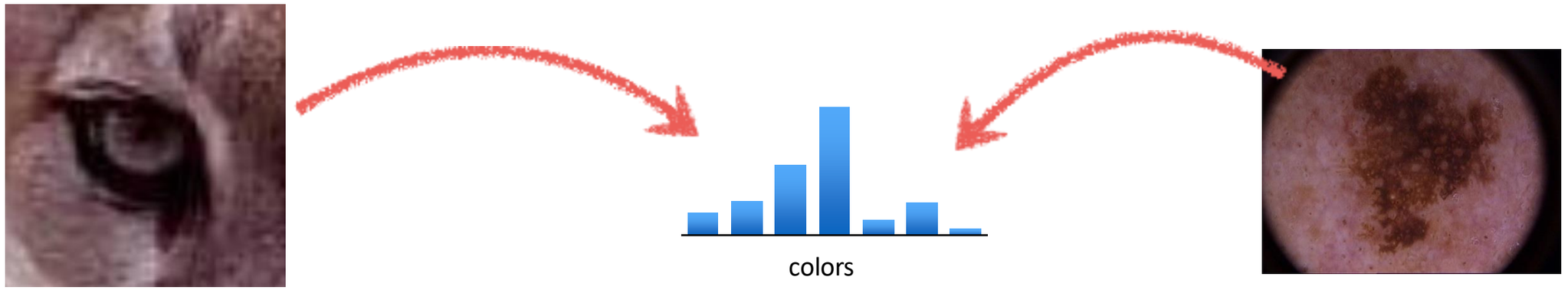


Invariant to changes in scale and rotation

What are the problems?

Color histogram

Count the colors in the image using a histogram

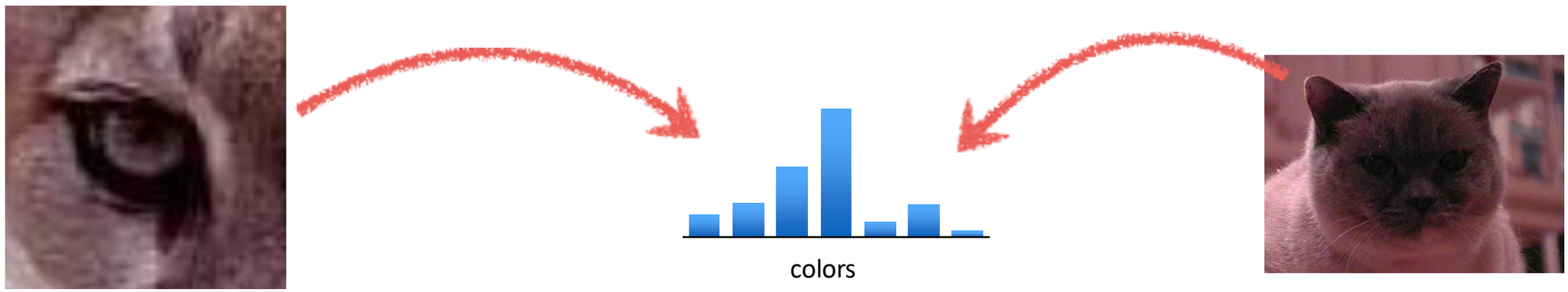


Invariant to changes in scale and rotation

What are the problems?

Color histogram

Count the colors in the image using a histogram



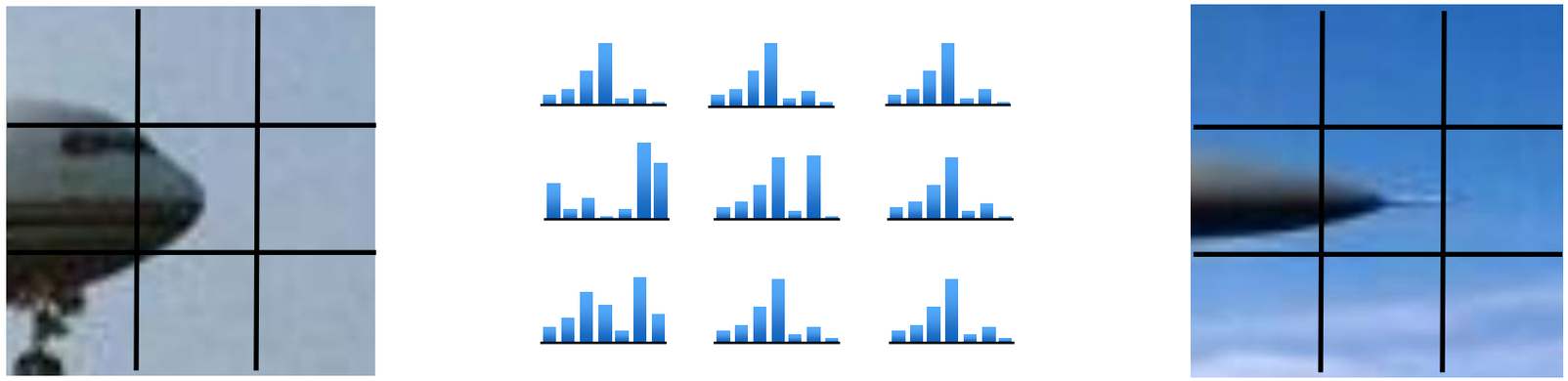
Invariant to changes in scale and rotation

What are the problems?

How can you be more sensitive to spatial layout?

Spatial histograms

Compute histograms over spatial ‘cells’



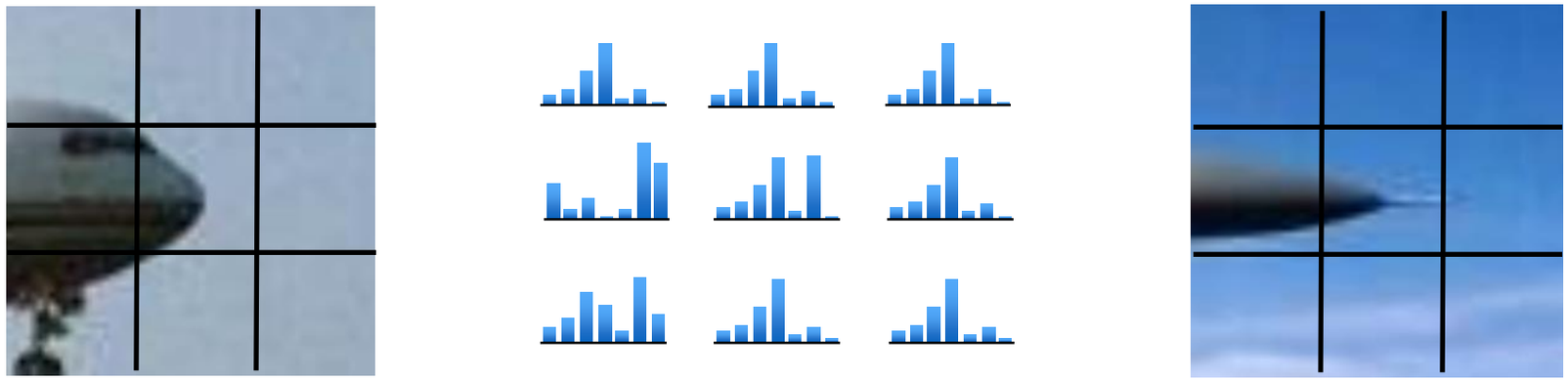
Retains rough spatial layout

Some invariance to deformations

What are the problems?

Spatial histograms

Compute histograms over spatial ‘cells’



Retains rough spatial layout

Some invariance to deformations

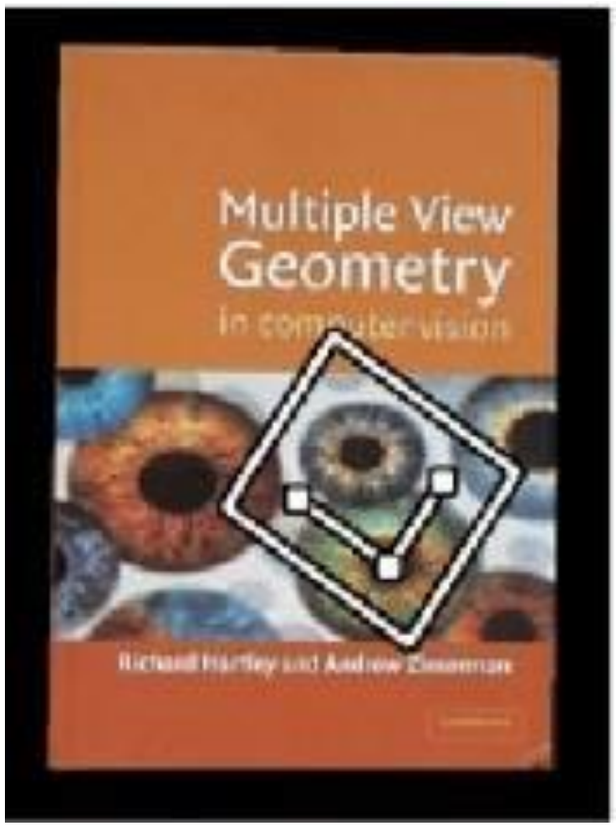
What are the problems?

How can you be completely invariant to rotation?

What is the best descriptor for an image feature?

Photometric transformations

Geometric transformations



objects will appear at different scales,
translation and rotation

Invariance vs. discriminability

- Invariance:
  - Descriptor shouldn’t change even if image is transformed

- Discriminability:
  - Descriptor should be highly unique for each point

SIFT

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations
- Shift the bins so that the biggest one is first

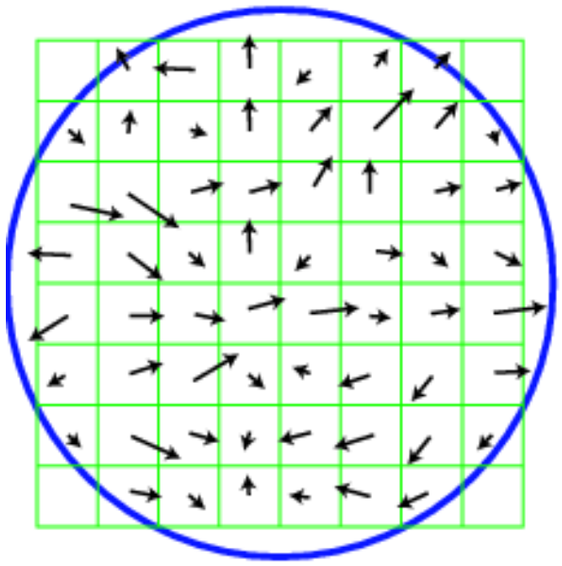
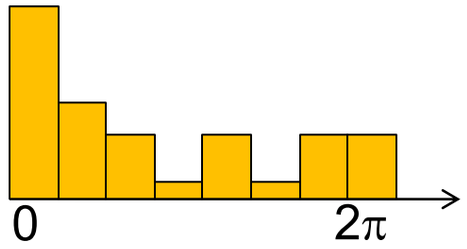
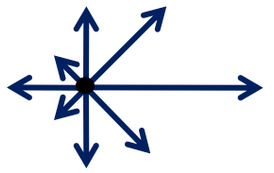


Image gradients



angle histogram



Adapted from slide by David Lowe

SIFT descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

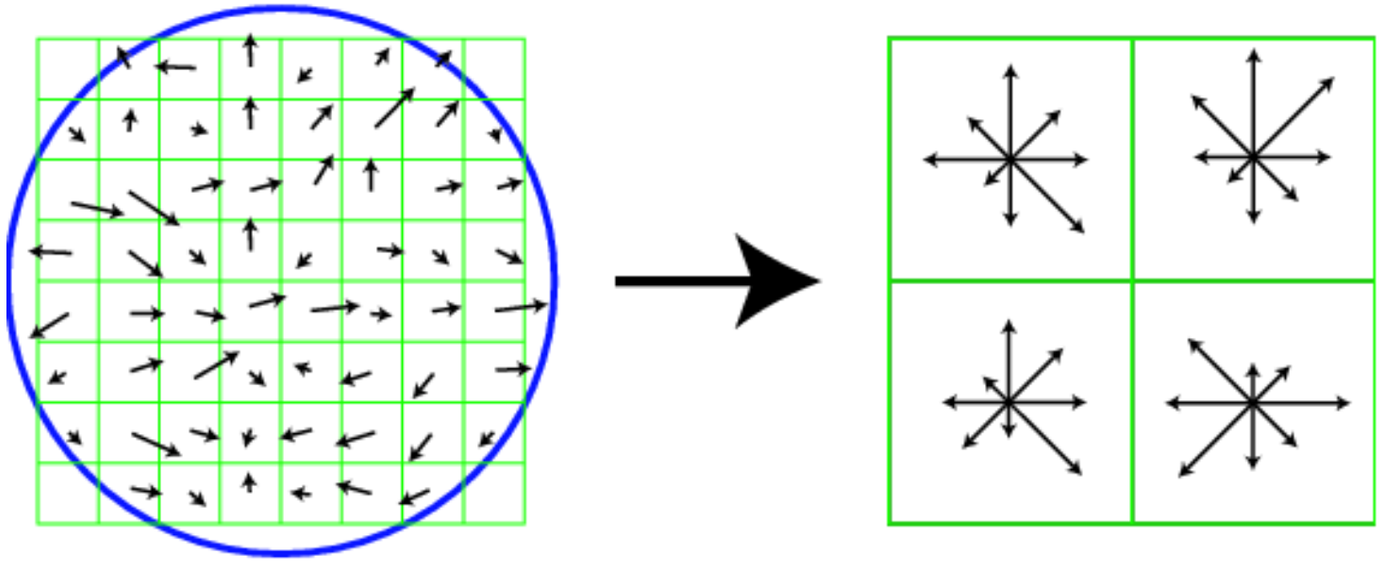
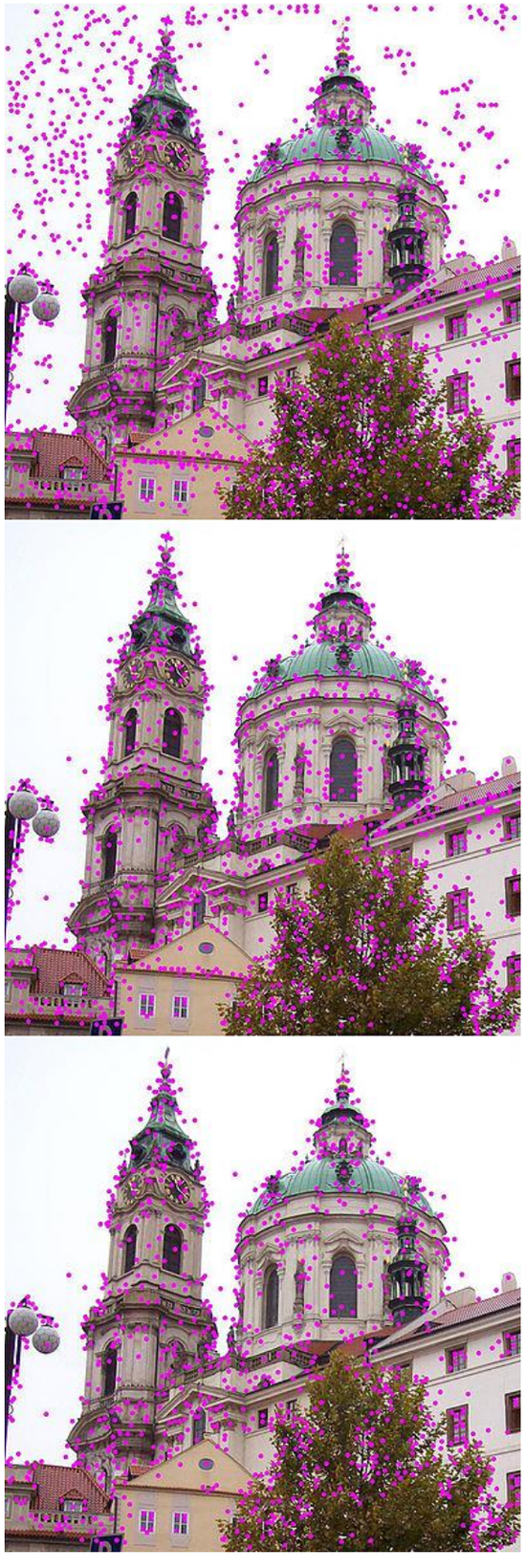


Image gradients

Keypoint descriptor

Adapted from slide by David Lowe



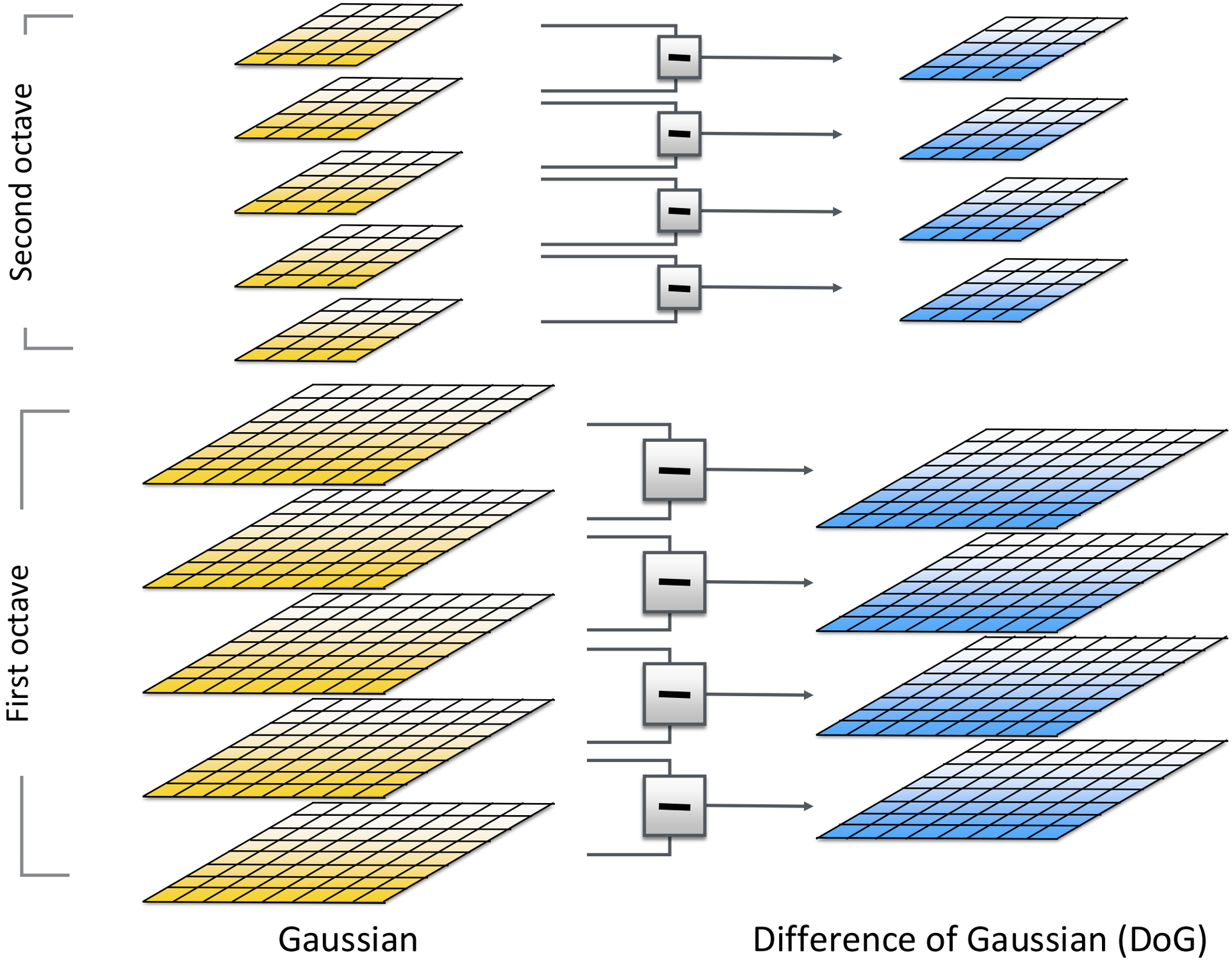
SIFT

(Scale Invariant Feature Transform)

SIFT describes both a **detector** and **descriptor**

1. Multi-scale extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

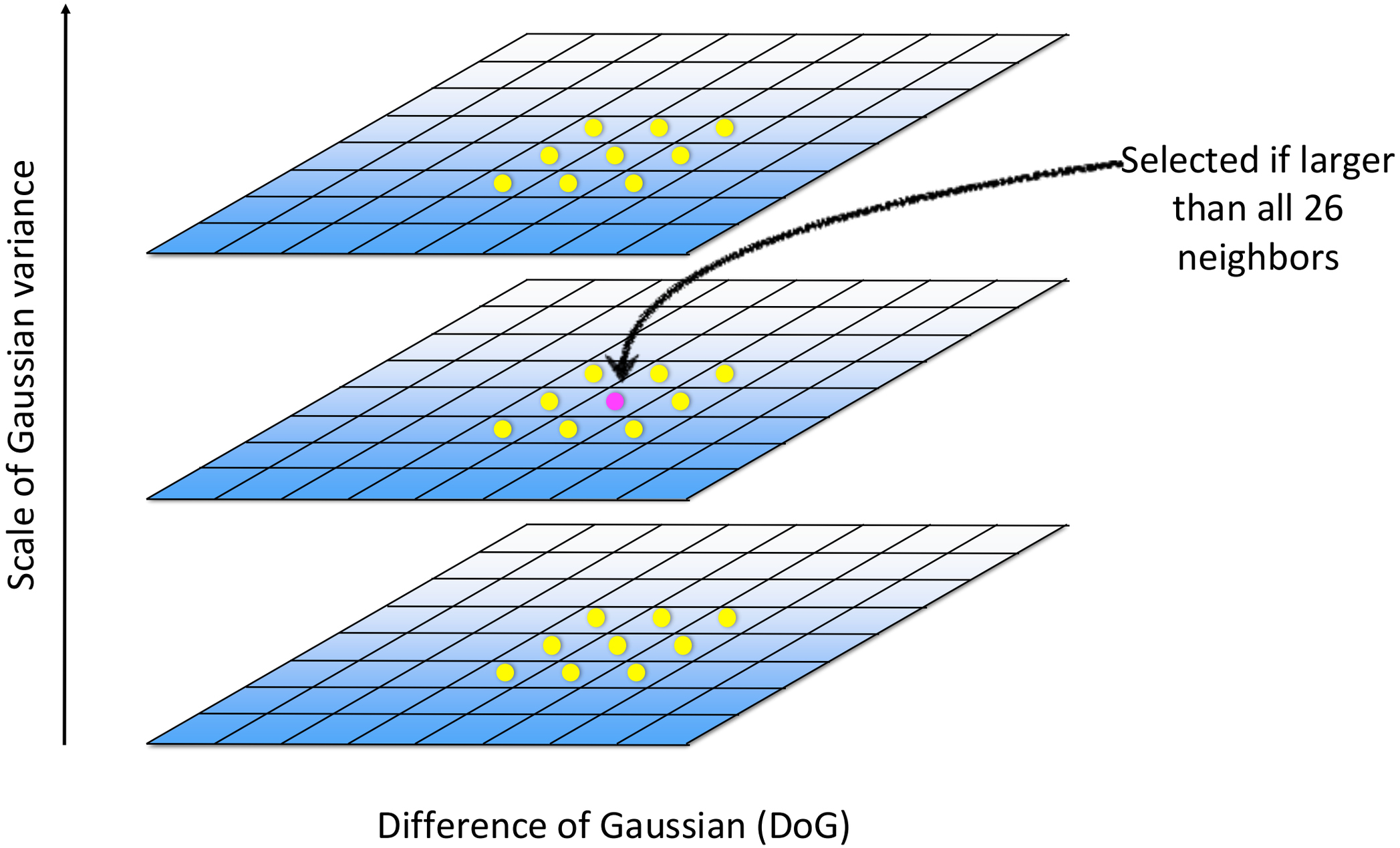
1. Multi-scale extrema detection



Gaussian

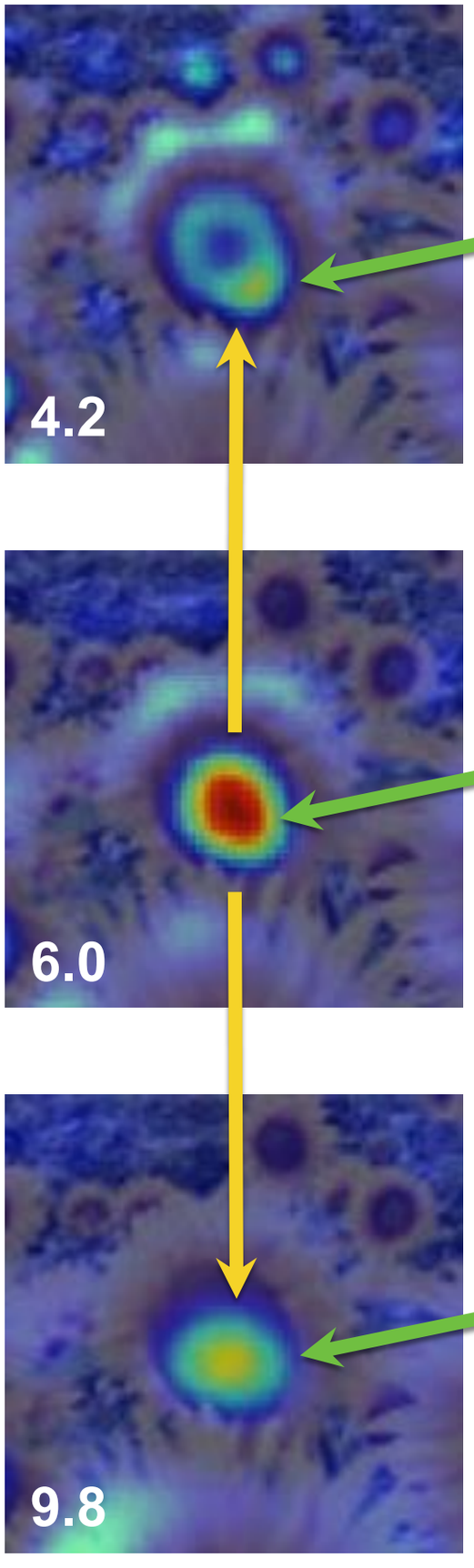
Laplacian

Scale-space extrema



Recall:

cross-scale maximum

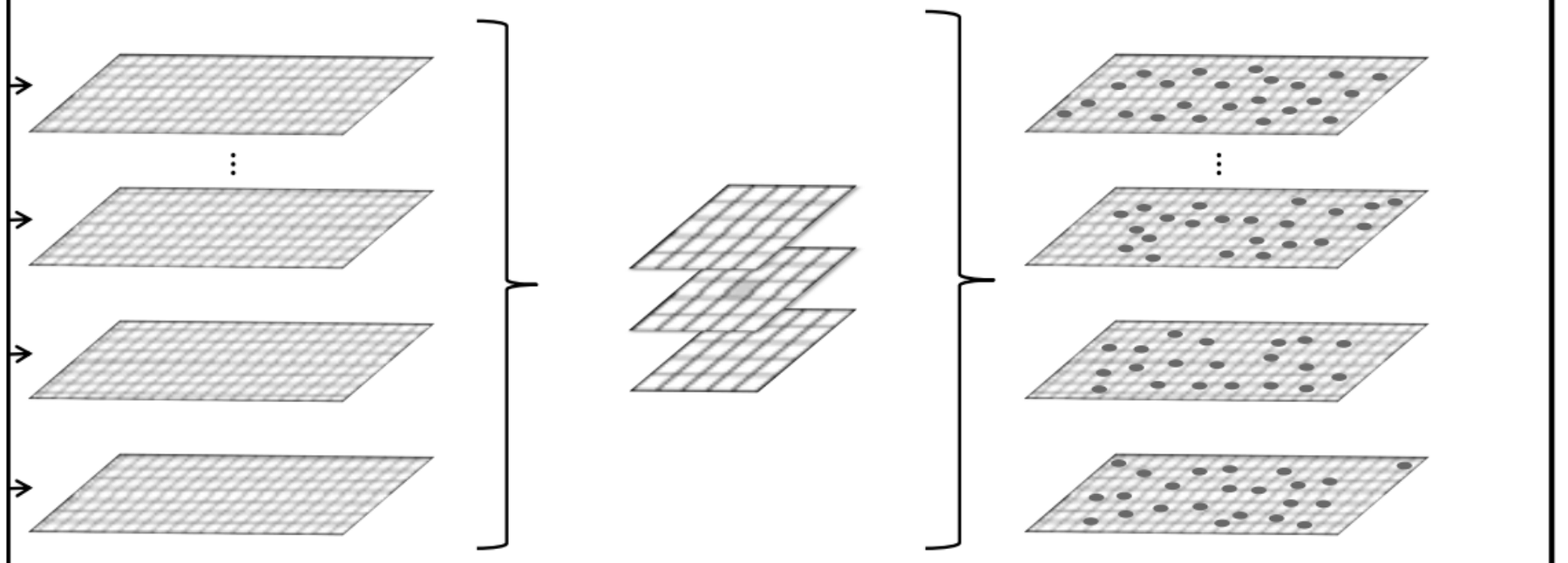


local maximum

local maximum

local maximum

Extracting SIFT Interest Points



Difference of Gaussians (DoG)

$\approx (s-1)\sigma^2\nabla^2 S(x,y,\sigma)$

Find Extremum in every 3x3x3 grid

Interest Point Candidates

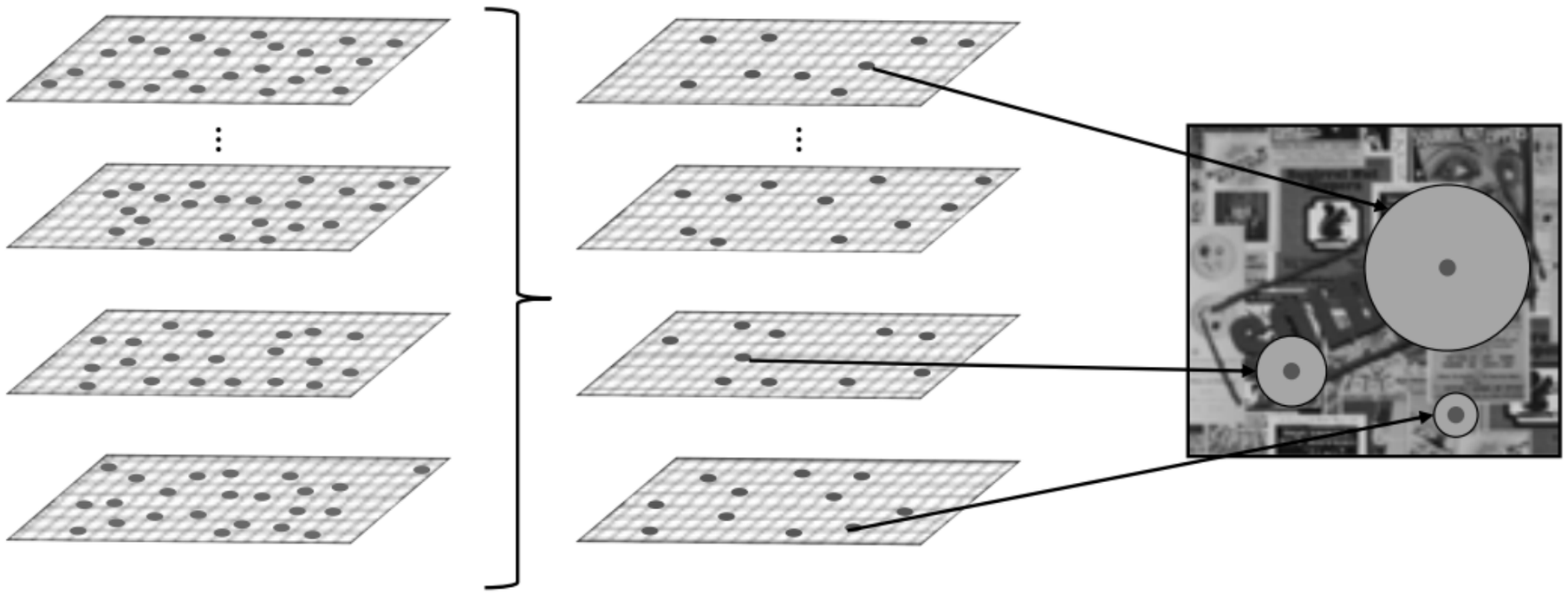
(includes weak extrema)

[Lowe 2004]

34

https://cave.cs.columbia.edu/Statics/monographs/SIFT%20Detector%20FPCV-2-3.pdf

Extracting SIFT Interest Points



Interest Point Candidates

(includes weak extrema)

SIFT Interest Points

(after removing weak extrema)

[Lowe 2004]

35

https://cave.cs.columbia.edu/Statics/monographs/SIFT%20Detector%20FPCV-2-3.pdf

Recall:

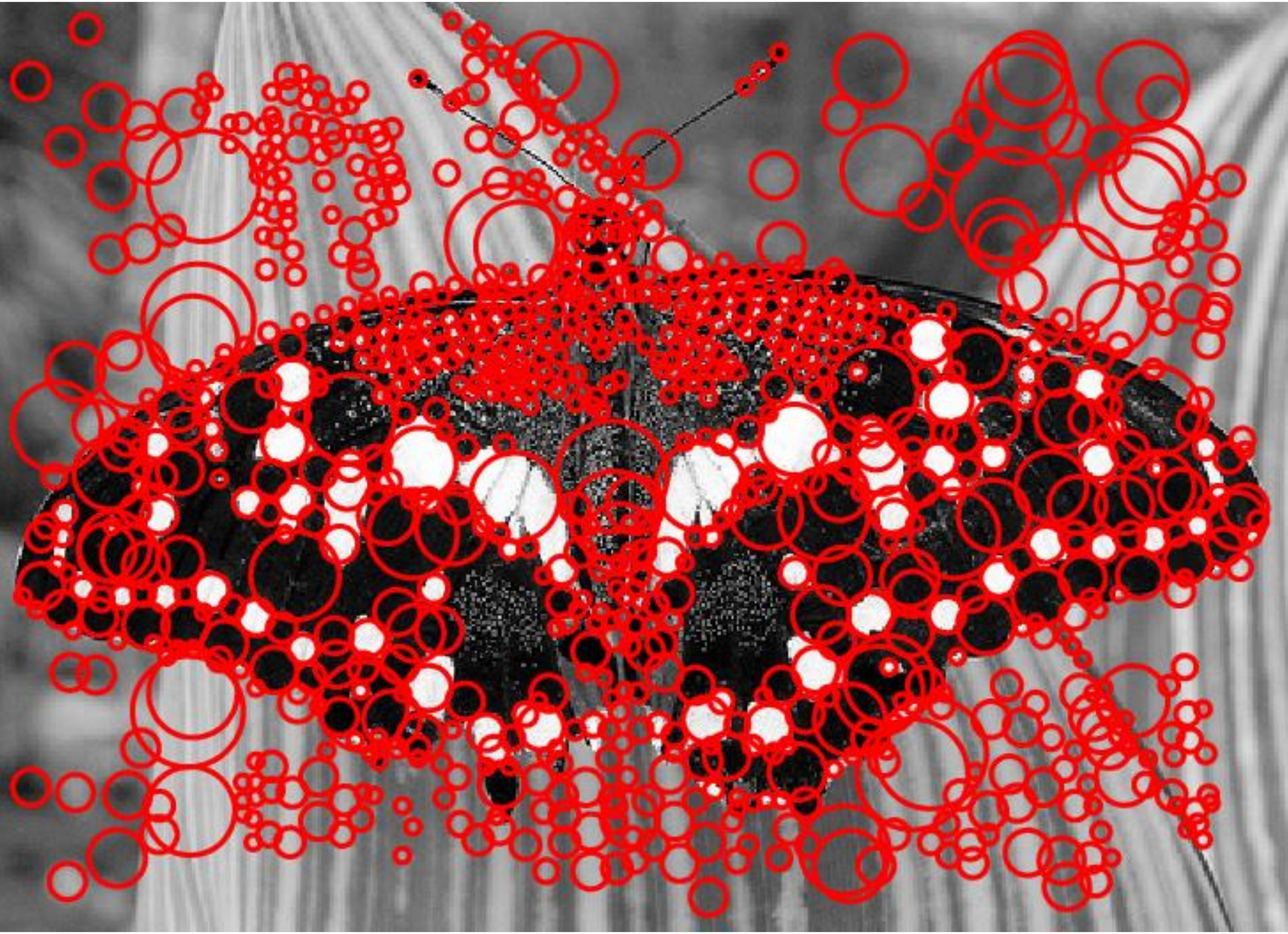
For each level of the Gaussian pyramid

compute feature response (e.g. Harris, Laplacian)

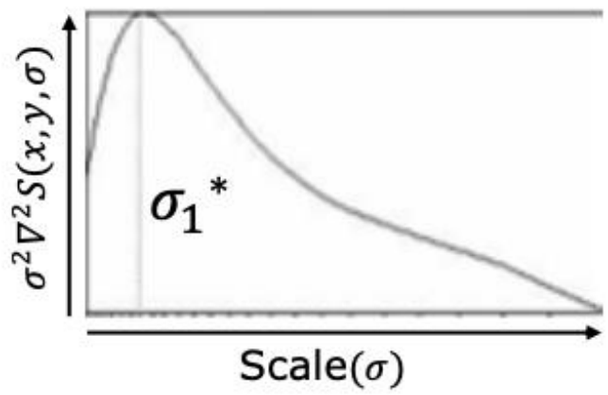
For each level of the Gaussian pyramid

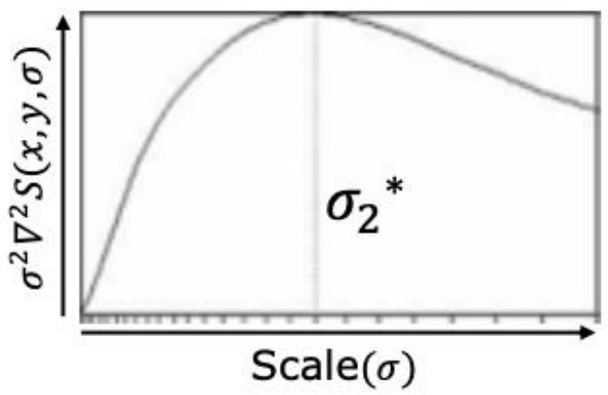
if local maximum and cross-scale

save scale and location of feature (x, y, s)



SIFT Scale Invariance





$\frac{\sigma_1{}^*}{\sigma_2{}^*}$: Ratio of Blob Sizes

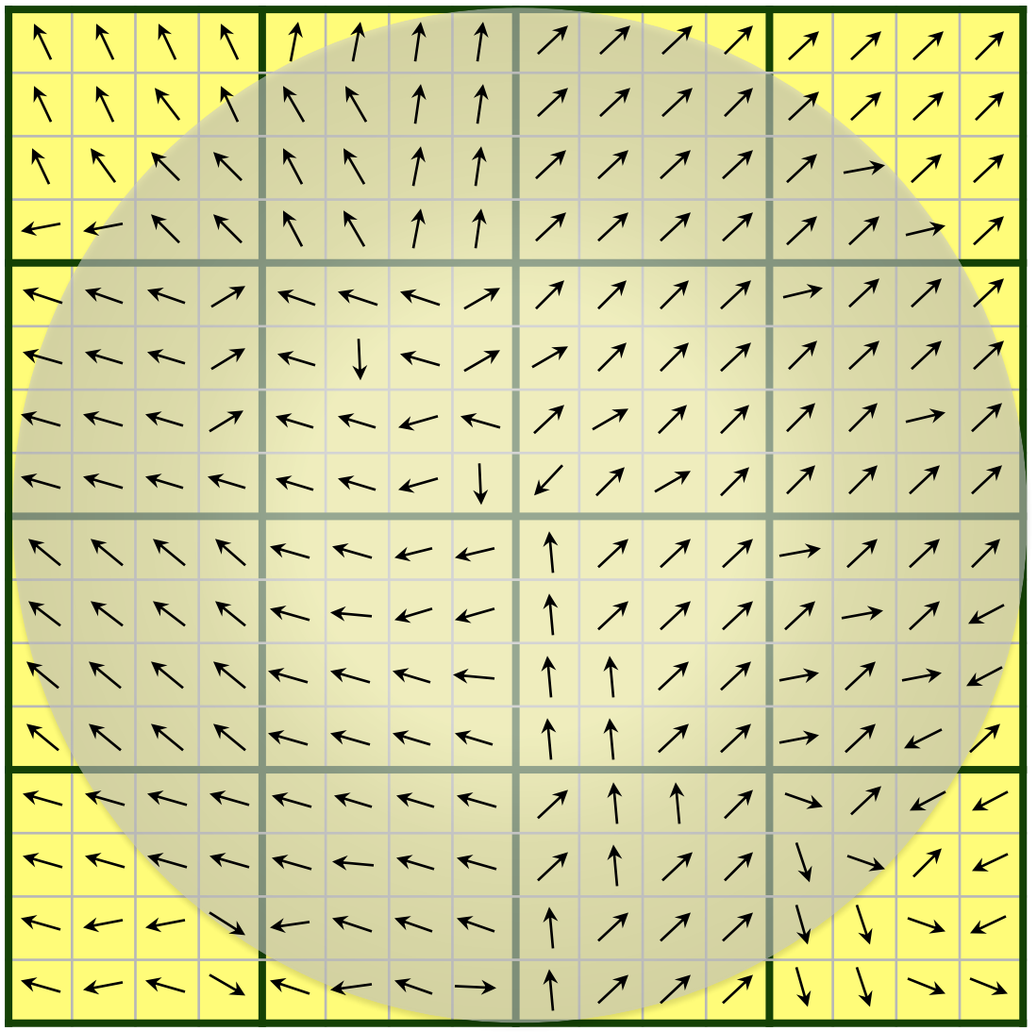
[Mikolajczyk 2002]

https://cave.cs.columbia.edu/Statics/monographs/SIFT%20Detector%20FPCV-2-3.pdf

Keypoint descriptor

Image Gradients

(4 x 4 pixel per cell, 4 x 4 cells)

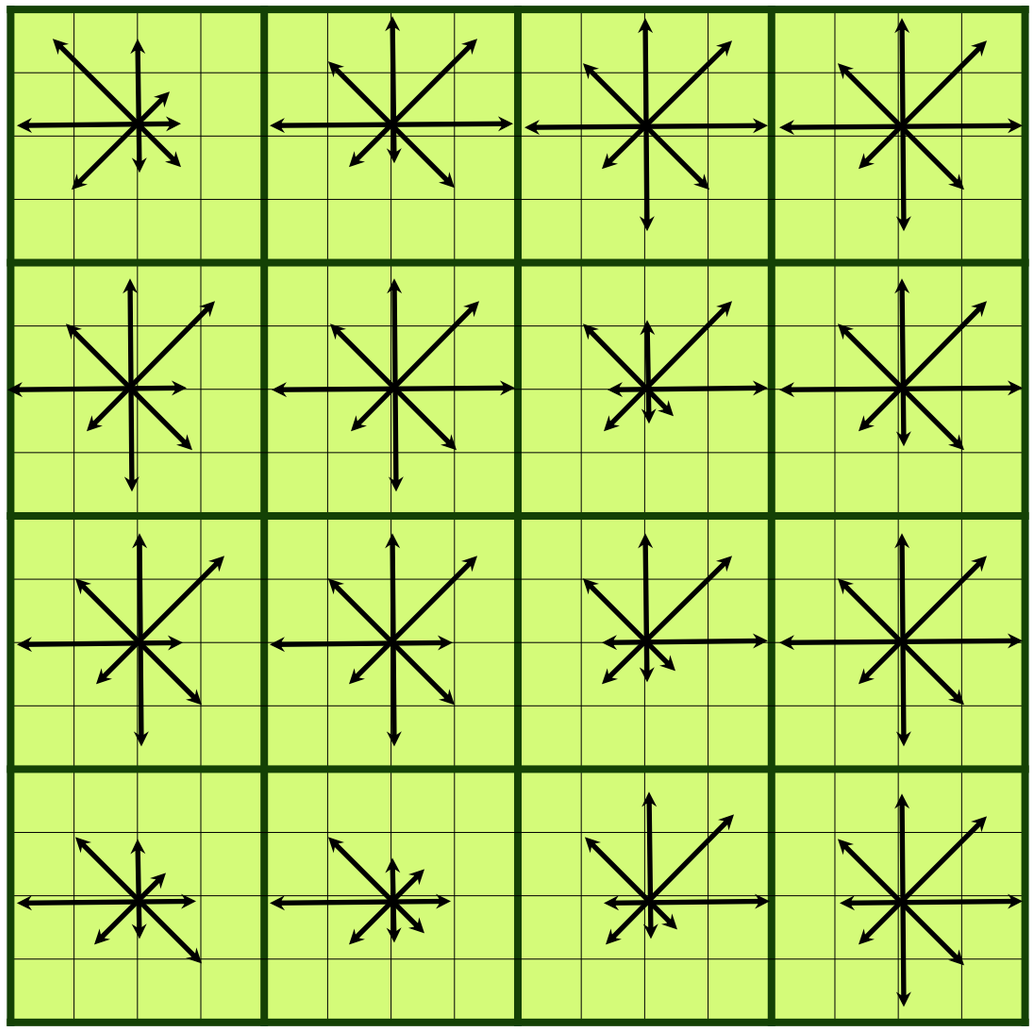


Gaussian weighting

(sigma = half width)

SIFT descriptor

(16 cells x 8 directions = 128 dims)



How’s rotation invariance handled?

Computing the Principal Orientation

Use the histogram of gradient directions

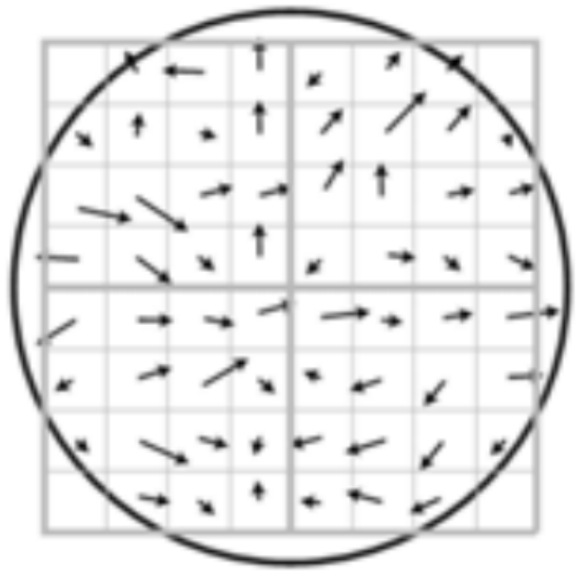
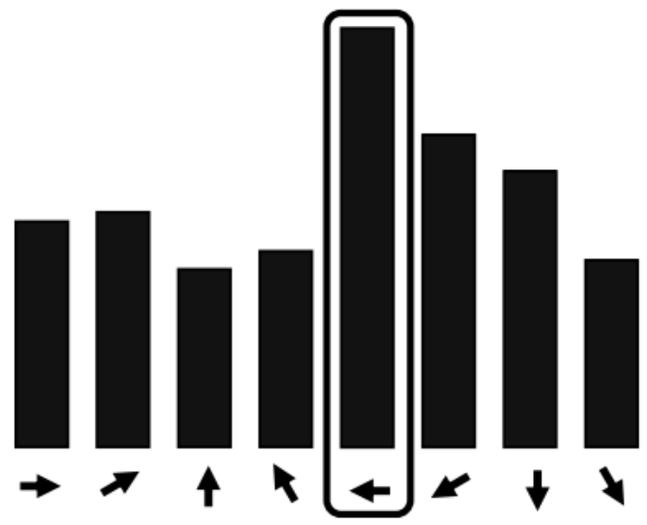


Image gradient directions

$$\theta = \tan^{-1}\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$$

Principal Orientation



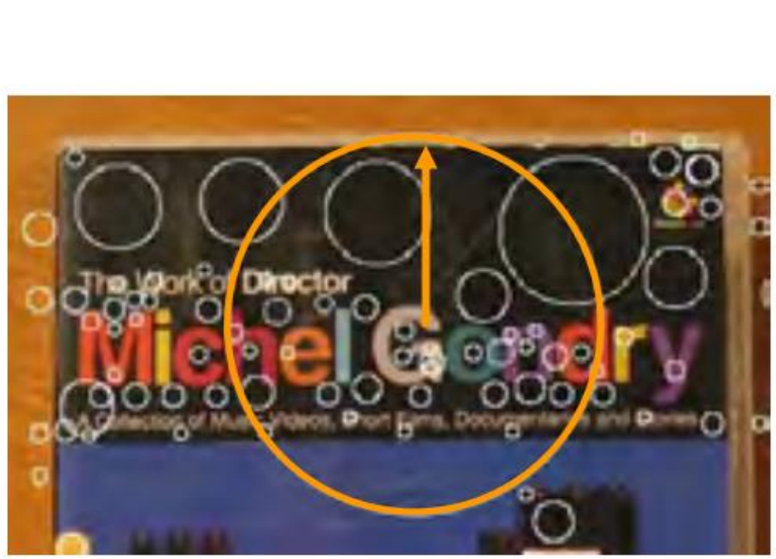
Choose the most prominent gradient direction

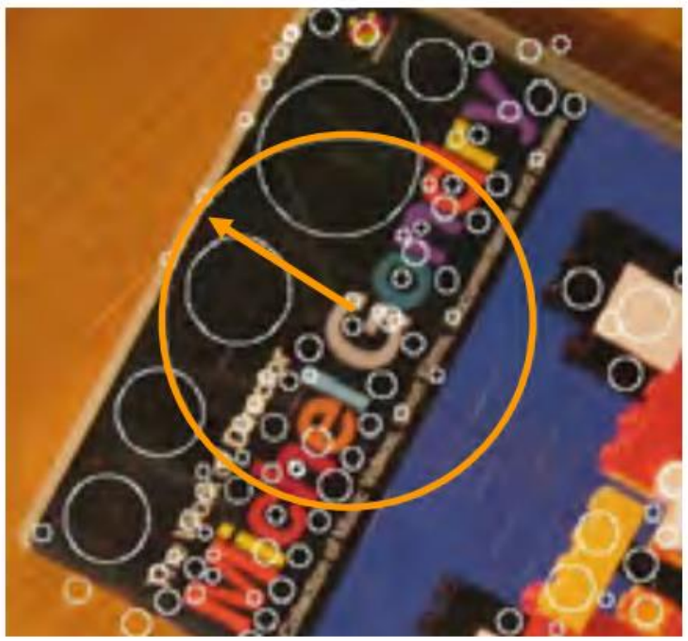
41

https://cave.cs.columbia.edu/Statics/monographs/SIFT%20Detector%20FPCV-2-3.pdf

SIFT Rotation Invariance

Use the principal orientation to undo rotation



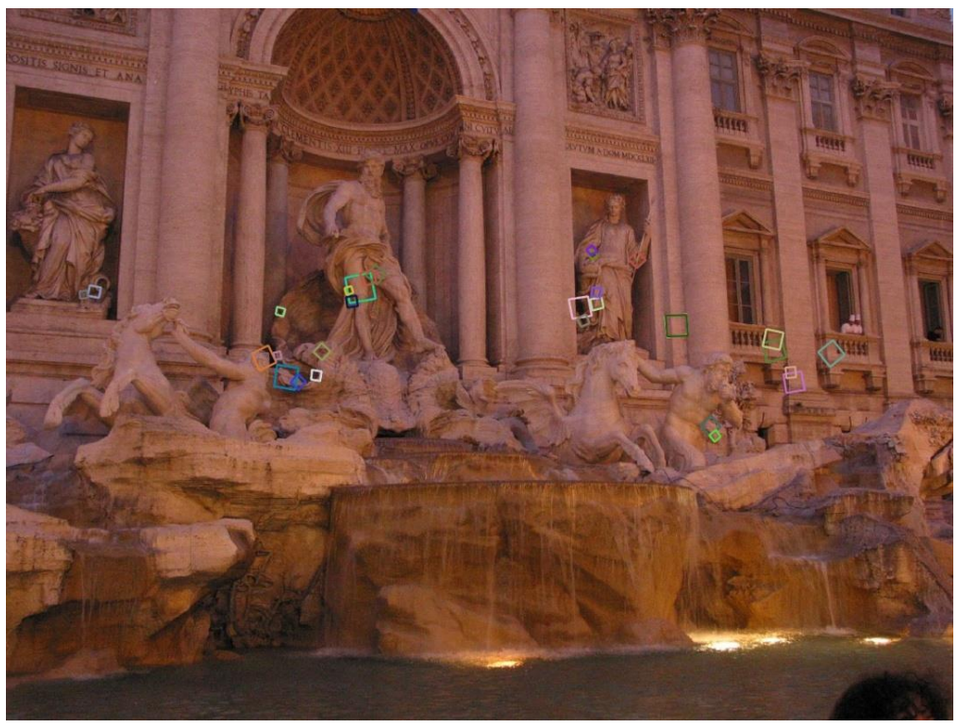


https://cave.cs.columbia.edu/Statics/monographs/SIFT%20Detector%20FPCV-2-3.pdf

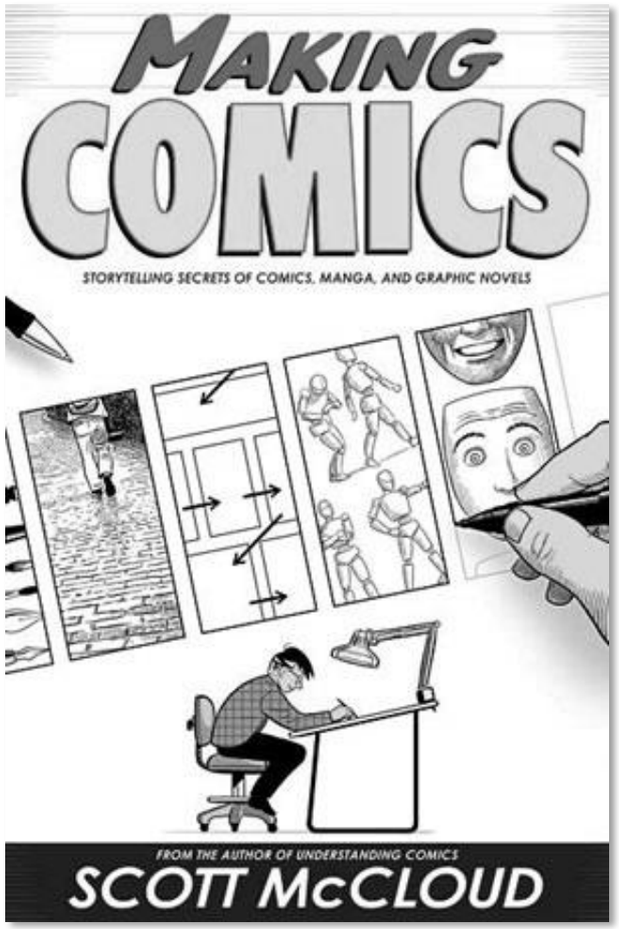
Properties of SIFT

Extraordinarily robust matching technique

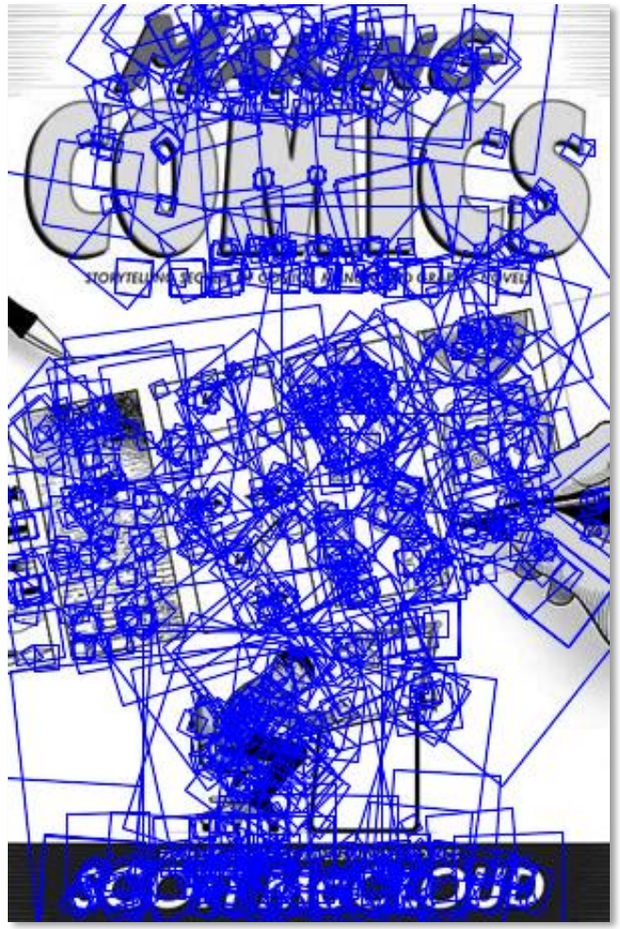
- Can handle changes in viewpoint (up to about 60 degree out of plane rotation)
- Can handle significant changes in illumination (sometimes even day vs. night (below))
- Pretty fast—hard to make real-time, but can run in <1s for moderate image sizes
- Lots of code available, patent expired as of now



SIFT Example



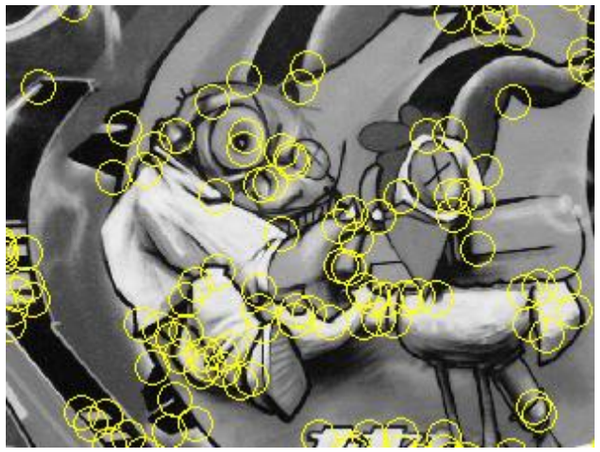
sift



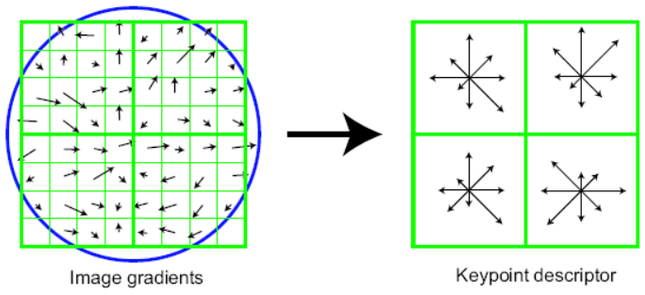
868 SIFT features

Summary

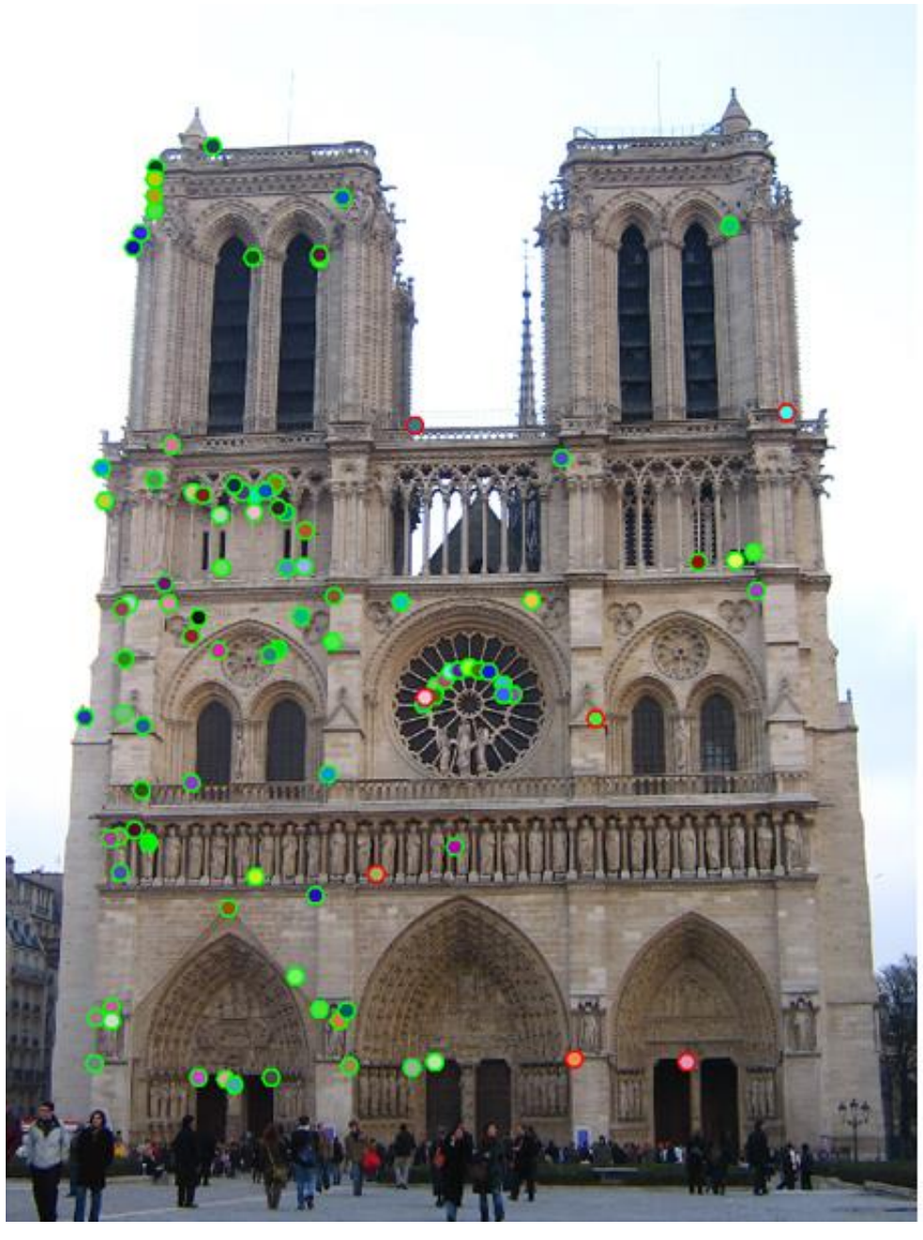
- Keypoint detection: repeatable and distinctive
  - Corners, blobs
  - Harris, DoG

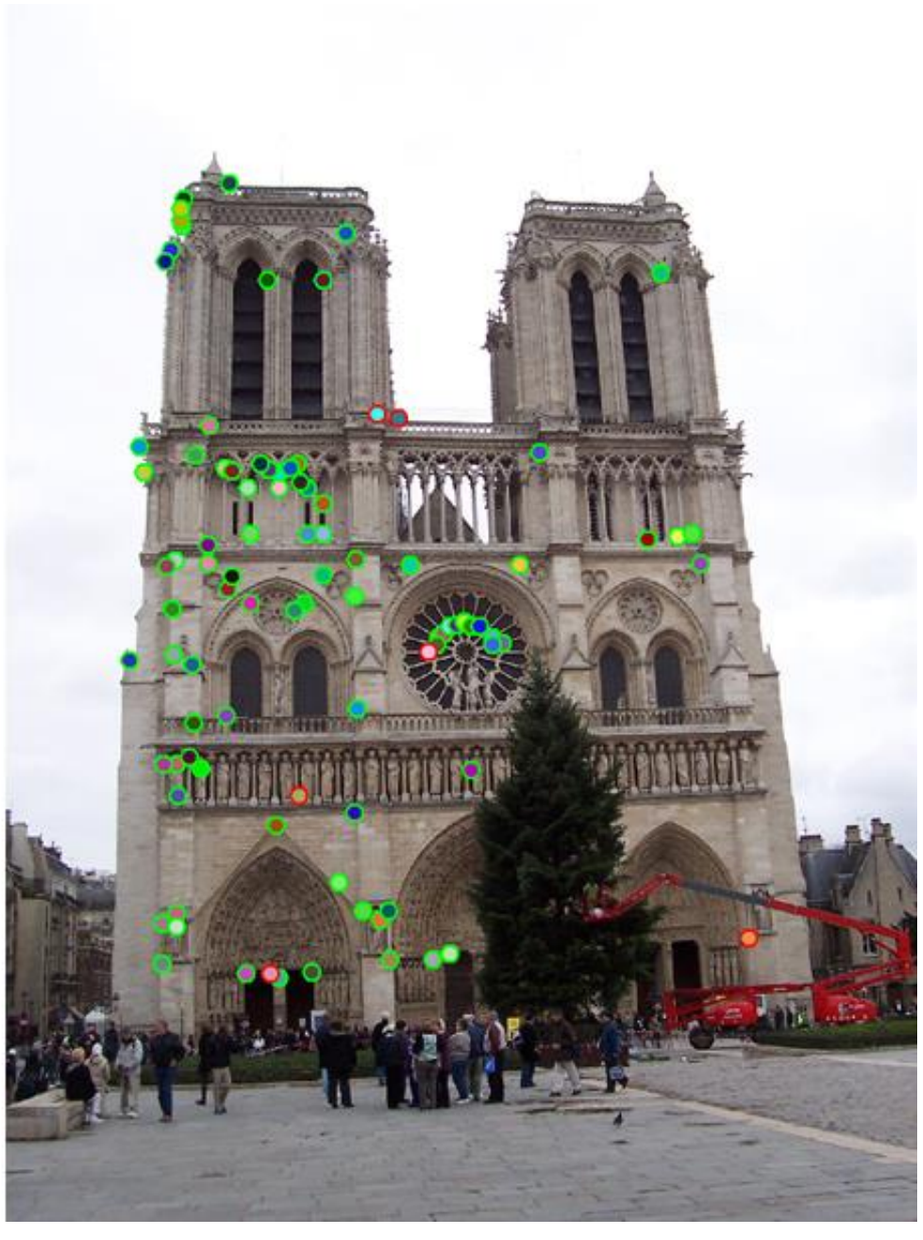


- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT and variants are typically good for stitching and recognition



Which features match?





Feature matching

Given a feature in I_1, how to find the best match in I_2?

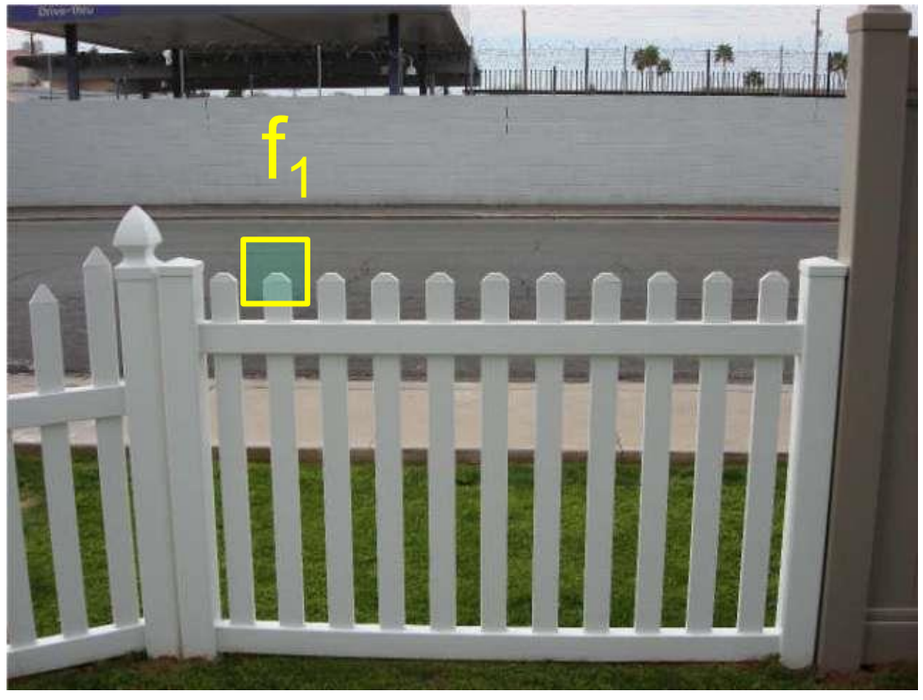
1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

(can be accelerated with a nearest neighbors search data structure, like a *kd*-tree)

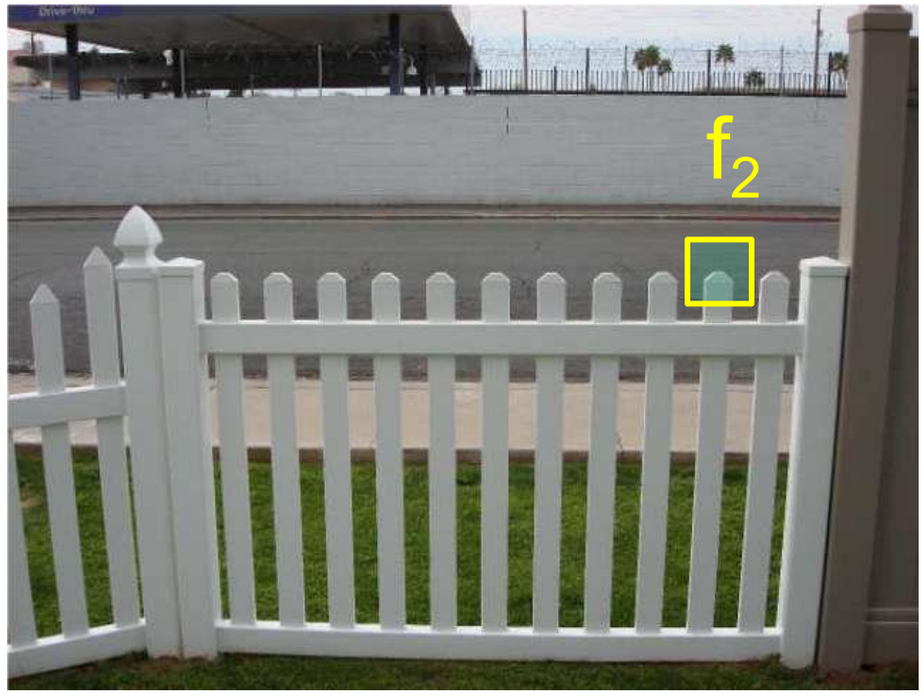
Feature distance

How to define the difference between two features f_1, f_2?

- Simple approach: L_2 distance, $\| f_1 - f_2 \|$
- can give small distances for ambiguous (incorrect) matches



I_1

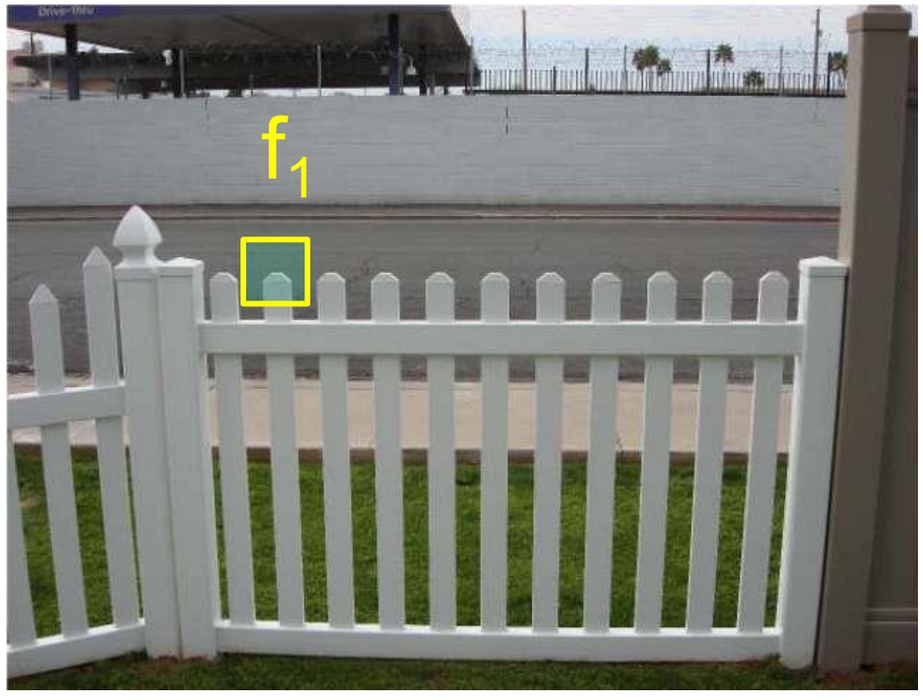


I_2

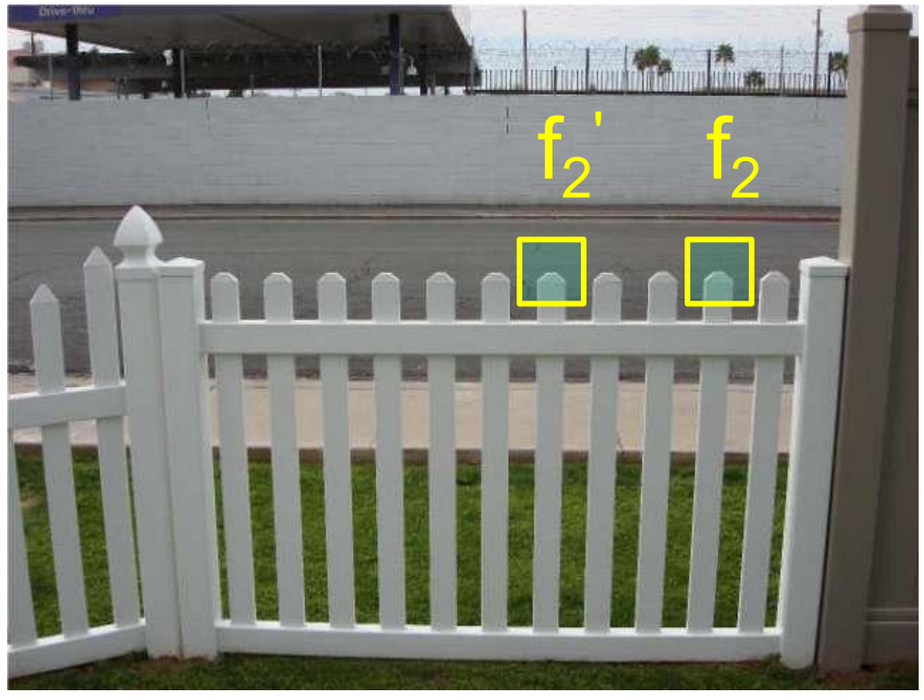
Feature distance

How to define the difference between two features f_1, f_2?

- Better approach: ratio distance = || f_1 - f_2 || / || f_1 - f_2' ||
  - f_2 is the best SSD match to f_1 in I_2
  - f_2' is the 2nd best SSD match to f_1 in I_2
  - gives large values for ambiguous matches



I_1

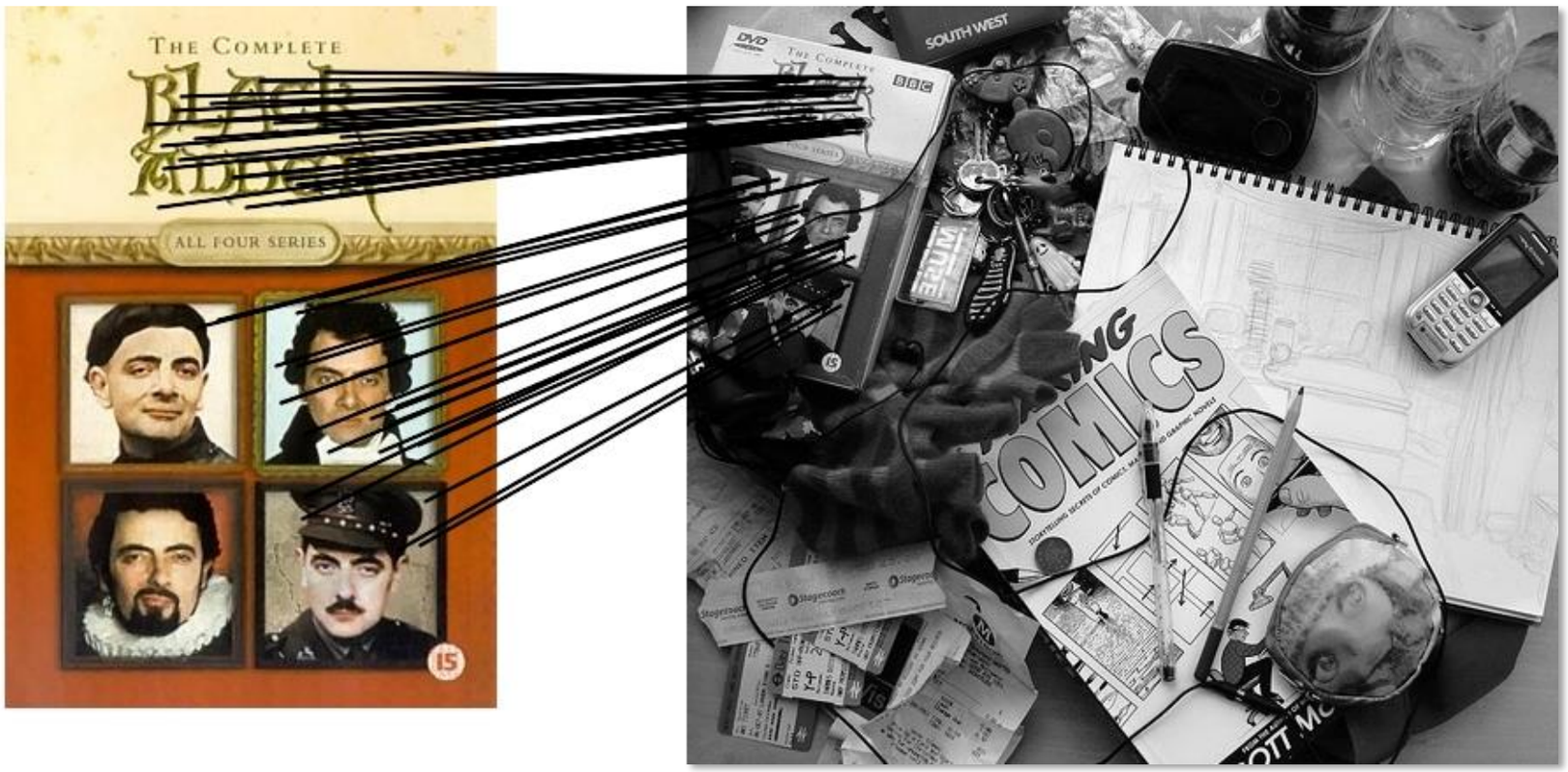


I_2

Feature distance

- Does the SSD vs “ratio distance” change the best match to a given feature in image 1?
- No, but it changes the distance, and it can change the ordering of matches from good to bad
- After we compute a set of matches, we *threshold* by distance (that is, throw out matches with distance > threshold)

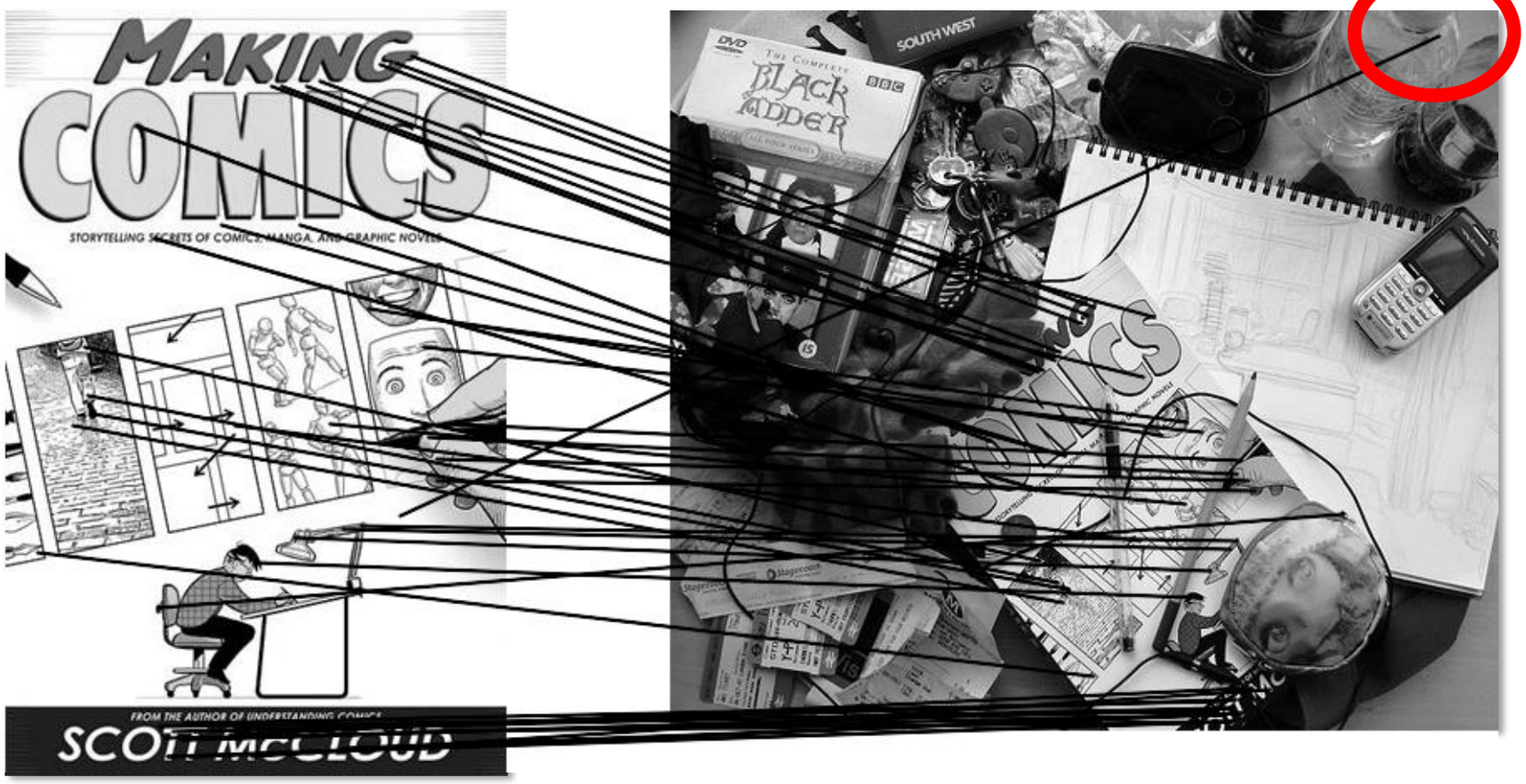
Feature matching example



58 matches (thresholded by ratio score)

Feature matching example

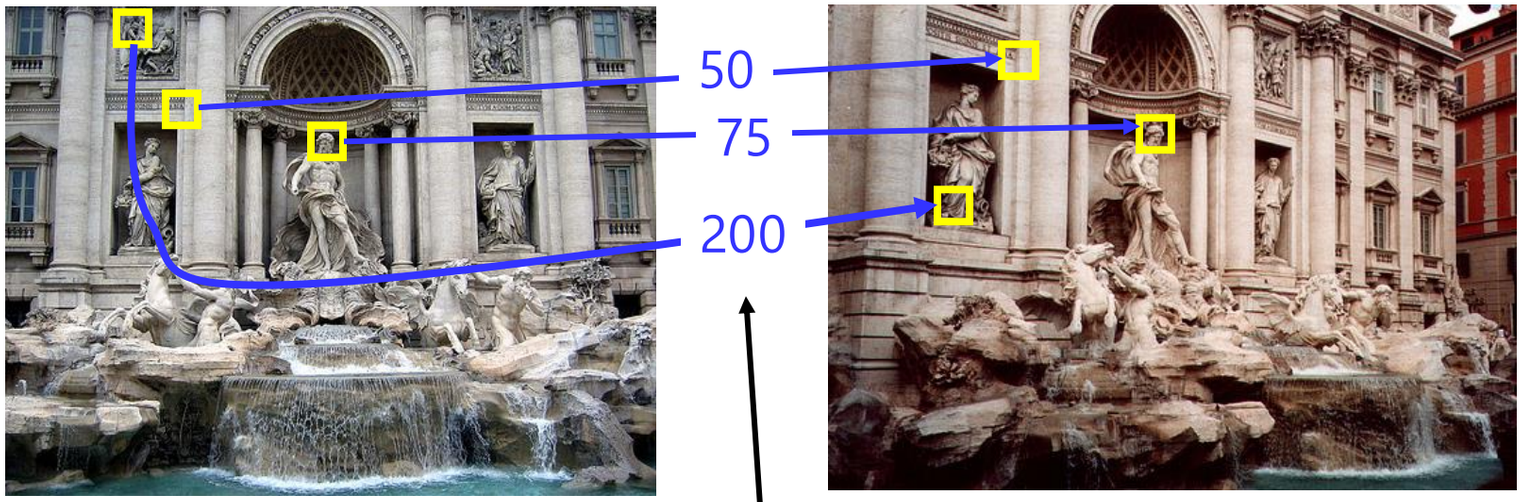
We'll deal with **outliers** later



51 matches (thresholded by ratio score)

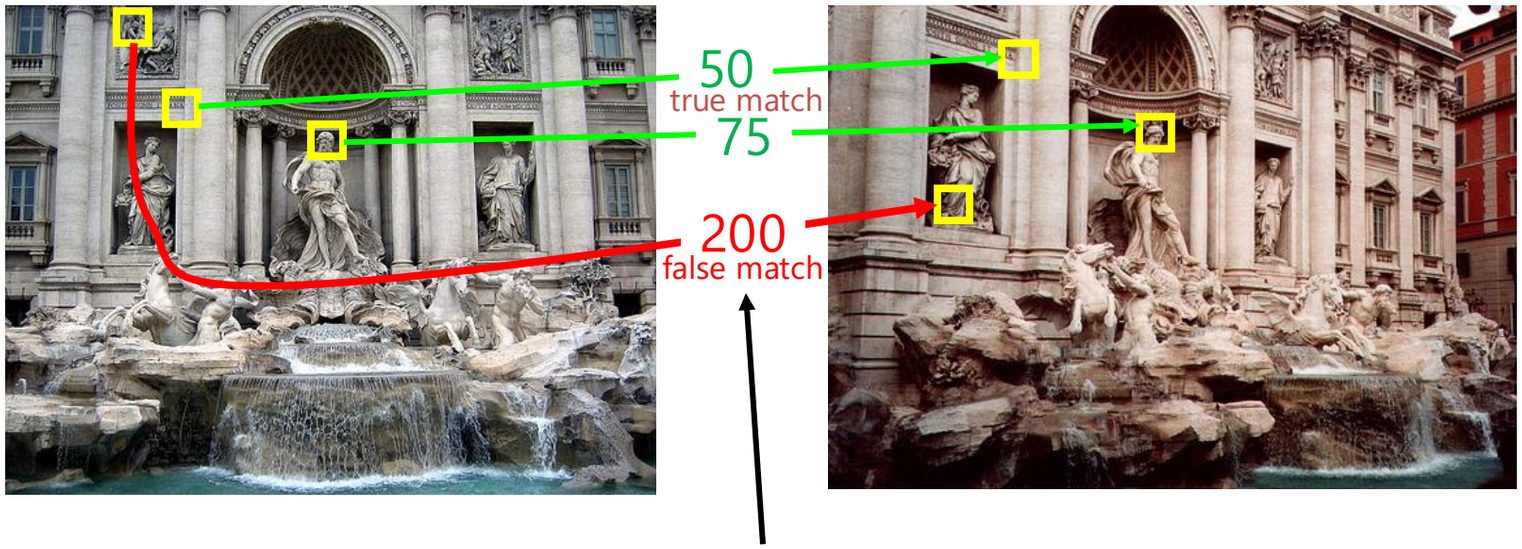
Evaluating the results

How can we measure the performance of a feature matcher?



True/false positives

How can we measure the performance of a feature matcher?

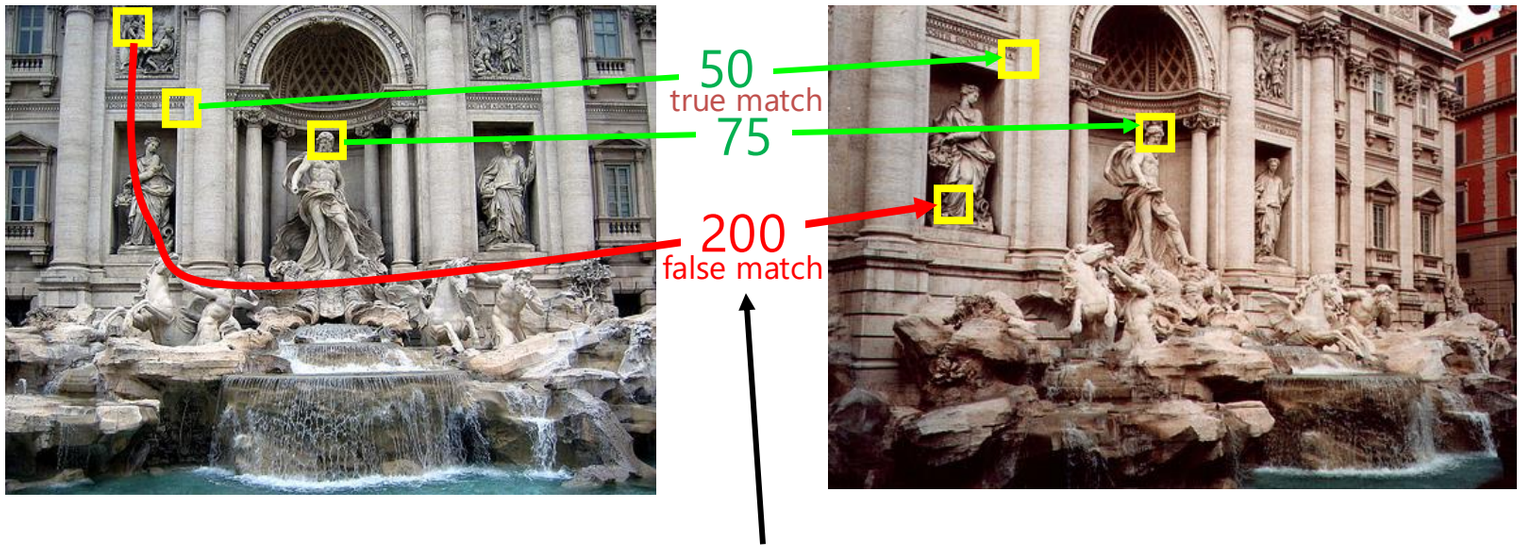


The distance threshold affects performance

- True positives = # of detected matches that survive the threshold that are correct
- False positives = # of detected matches that survive the threshold that are incorrect

True/false positives

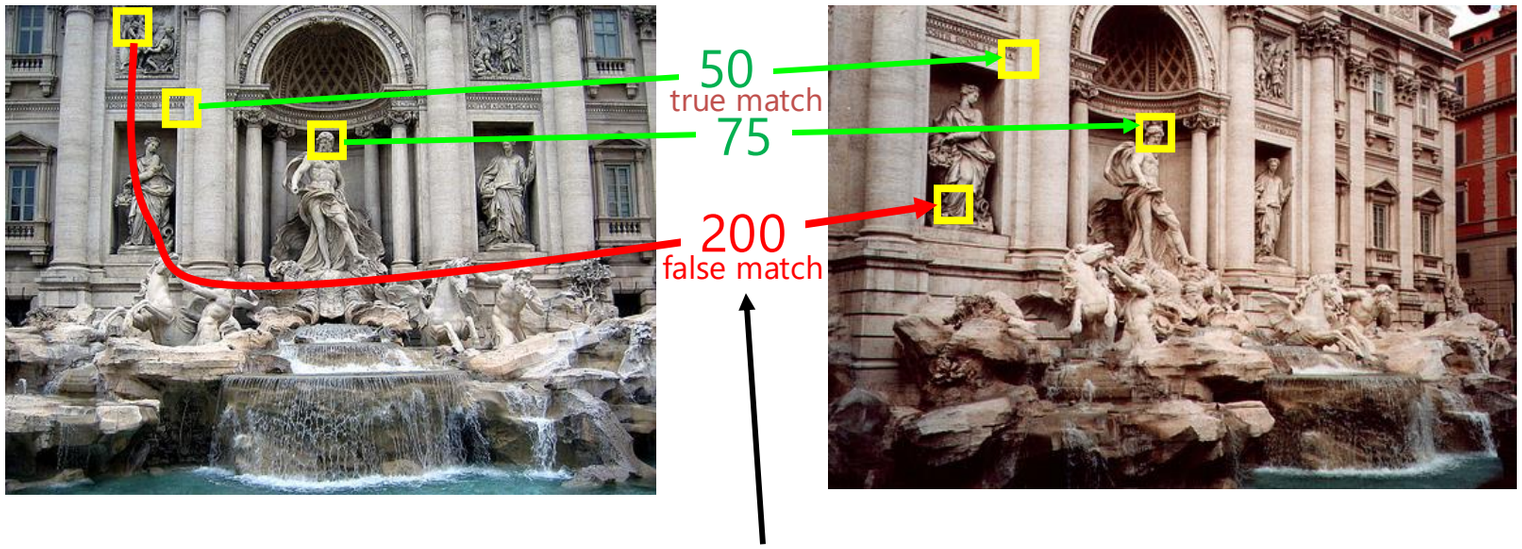
How can we measure the performance of a feature matcher?



Suppose we want to **maximize true positives**. How do we set the threshold? (Note: we keep all matches with distance below the threshold.)

True/false positives

How can we measure the performance of a feature matcher?



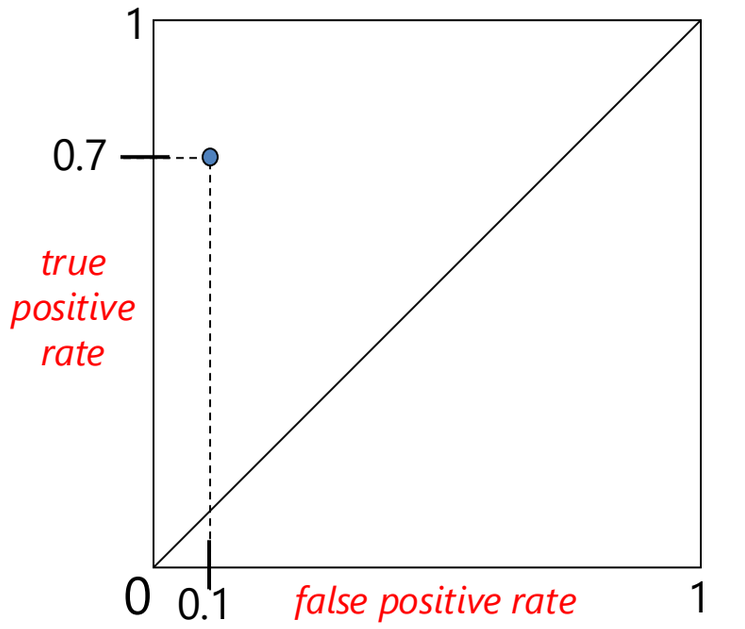
Suppose we want to **minimize false positives**. How do we set the threshold? (Note: we keep all matches with distance below the threshold.)

Example

- Suppose our matcher computes 1,000 matches between two images
  - 800 are correct matches, 200 are incorrect (according to an oracle that gives us ground truth matches)
  - A given threshold (e.g., ratio distance = 0.6) gives us 600 correct matches and 100 incorrect matches that survive the threshold
  - True positive rate = 600 / 800 = ¾
  - False positive rate = 100 / 200 = ½

Evaluating the results

How can we measure the performance of a feature matcher?



$$\frac{\text{\# true positives surviving threshold}}{\text{\# total correct matches (positives)}}$$

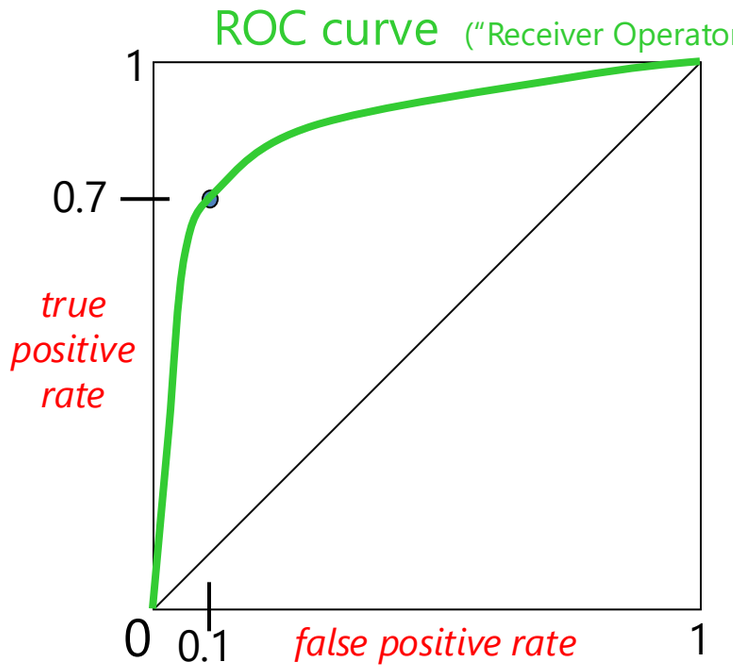
recall

$$\frac{\text{\# false positives surviving threshold}}{\text{\# total incorrect matches (negatives)}}$$

1 - *specificity*

Evaluating the results

How can we measure the performance of a feature matcher?



true positives surviving threshold
total correct matches (positives)

recall

false positives surviving threshold
total incorrect matches (negatives)

1 - *specificity*

Single number: Area Under the Curve (AUC)

E.g. AUC = 0.87

1 is the best

ROC curves – summary

- By thresholding the match distances at different thresholds, we can generate sets of matches with different true/false positive rates
- ROC curve is generated by computing rates at a set of threshold values swept through the full range of possible thresholds
- Area under the ROC curve (AUC) summarizes the performance of a feature pipeline (higher AUC is better)

More on feature detection/description

http://www.robots.ox.ac.uk/~vgg/research/affine/
http://www.cs.ubc.ca/~lowe/keypoints/
http://www.vision.ee.ethz.ch/~surf/

Publications

Region detectors

- *Harris-Affine & Hessian Affine*: K. Mikolajczyk and C. Schmid, Scale and Affine invariant interest point detectors. In IJC V 60(1):63-86, 2004. PDF
- *MSER*: J.Matas, O. Chum, M. Urban, and T. Pajdla, Robust wide baseline stereo from maximally stable extremal regions. In BMVC p. 384-393, 2002. PDF
- *IBR & EBR*: T.Tuytelaars and L. Van Gool, Matching widely separated views based on affine invariant regions . In IJCV 59(1):61-85, 2004. PDF
- *Salient regions*: T. Kadir, A. Zisserman, and M. Brady, An affine invariant salient region detector. In ECCV p. 404-416, 2004. PDF
- *All Detectors - Survey*: T. Tuytelaars and K. Mikolajczyk , Local Invariant Feature Detectors - Survey. In CVG, 3(1):1-110, 2008. PDF

Region descriptors

- *SIFT*: D. Lowe, Distinctive image features from scale invariant keypoints. In IJCV 60(2):91-110, 2004. PDF

Performance evaluation

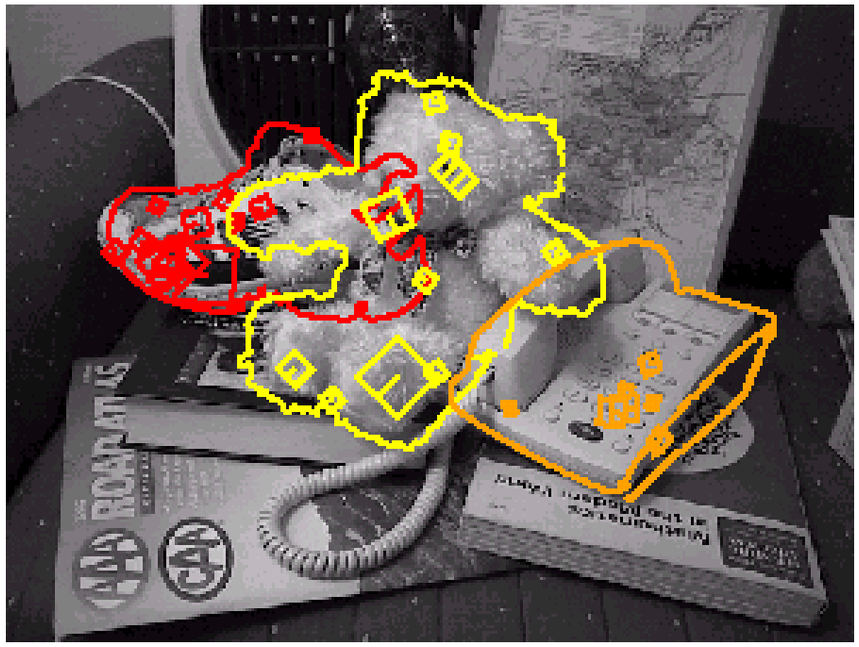
- K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir and L. Van Gool, A comparison of affine region detectors. In IJCV 65(1/2):43-72, 2005. PDF
- K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors. In PAMI 27(10):1615-1630 . PDF

Lots of applications

Features are used for:

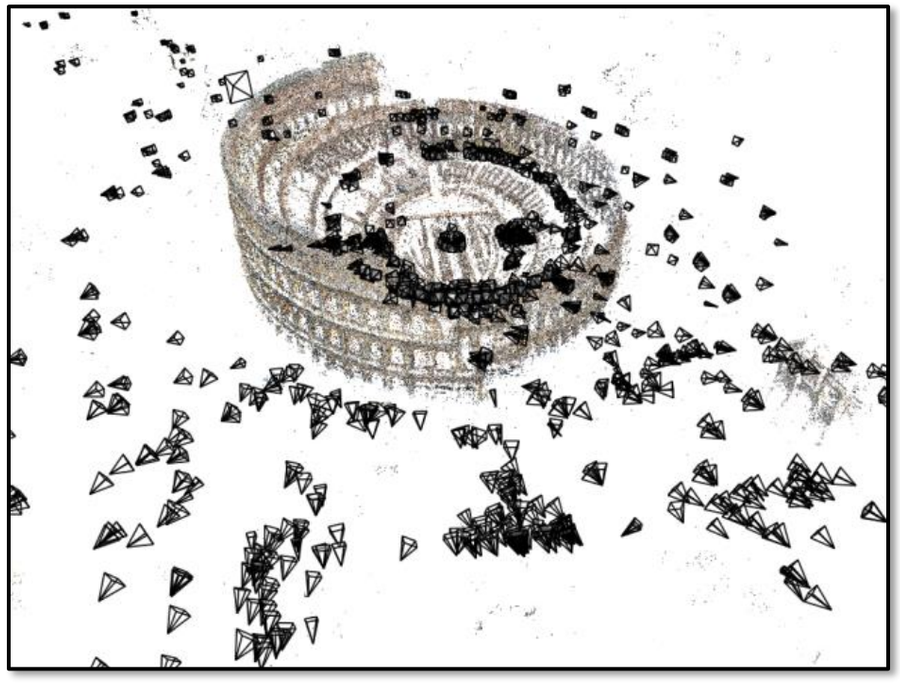
- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other

Object recognition (David Lowe)



3D Reconstruction

Internet Photos (“Colosseum”)



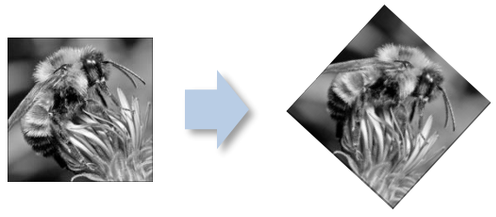
Reconstructed 3D cameras and points

Augmented Reality

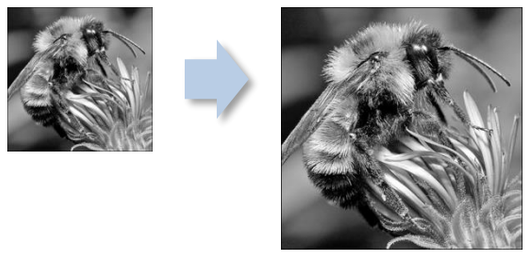
Image transformations revisited

- Geometric

Rotation



Scale



- Photometric

Intensity change

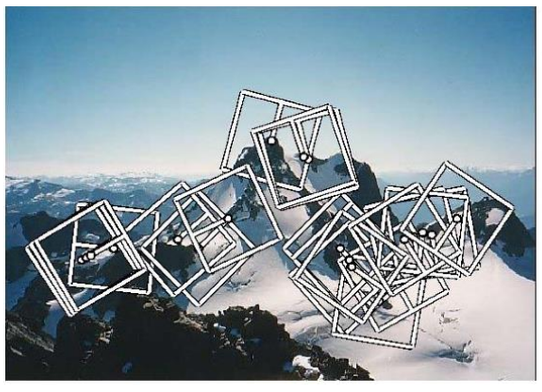
Invariant descriptors

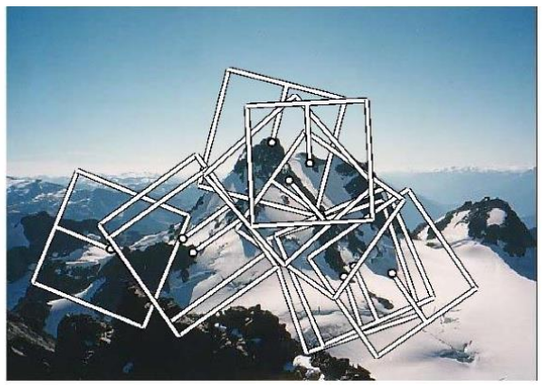
- We looked at invariant / equivariant **detectors**
- Most feature descriptors are also designed to be invariant to:
  - Translation, 2D rotation, scale
- They can usually also handle
  - Limited 3D rotations (SIFT works up to about 60 degrees)
  - Limited affine transforms (some are fully affine invariant)
  - Limited illumination/contrast changes

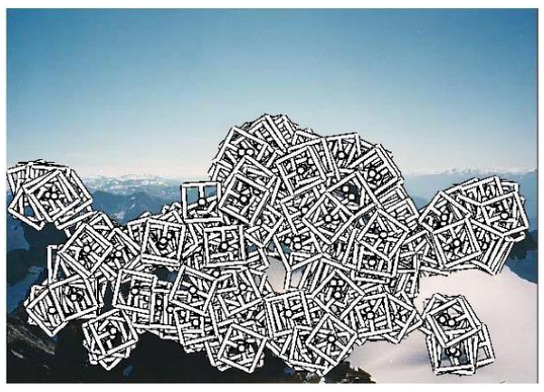
MOPS descriptor

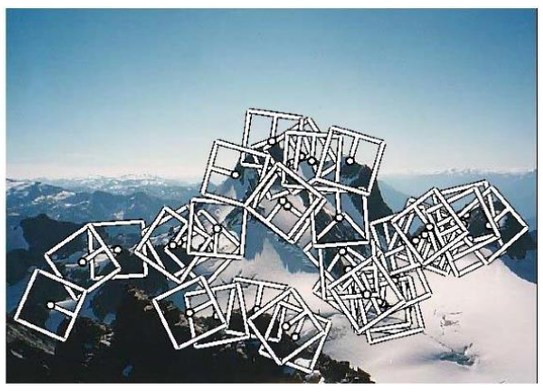
Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder. International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517









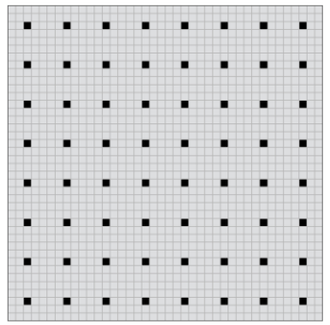
Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder. International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

Get 40 x 40 image patch, subsample every 5th pixel

(*what's the purpose of this step?*)



Subtract the mean, divide by standard deviation

(*what's the purpose of this step?*)

Haar Wavelet Transform

(*what's the purpose of this step?*)

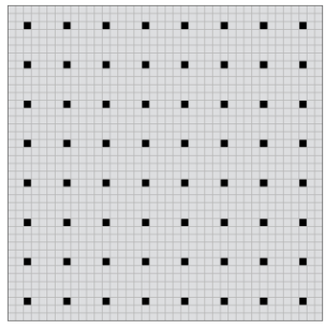
Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder. International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

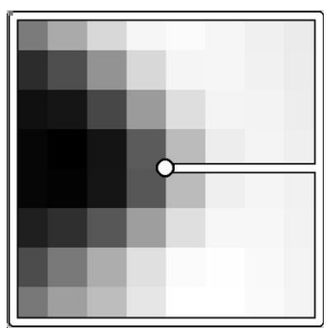
Get 40 x 40 image patch, subsample every 5th pixel

(low frequency filtering, absorbs localization errors)



Subtract the mean, divide by standard deviation

(*what's the purpose of this step?*)



Haar Wavelet Transform

(*what's the purpose of this step?*)

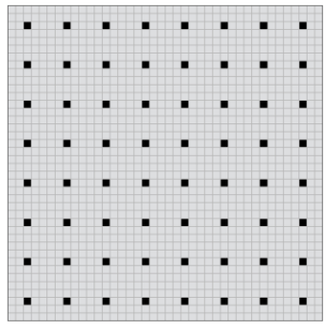
Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder. International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

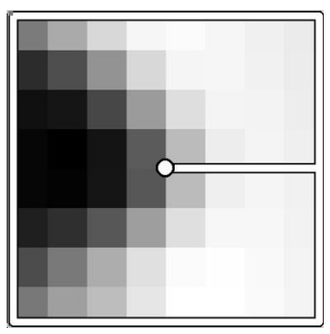
Get 40 x 40 image patch, subsample every 5th pixel

(low frequency filtering, absorbs localization errors)



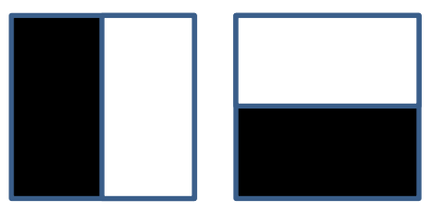
Subtract the mean, divide by standard deviation

(removes bias and gain)



Haar Wavelet Transform

(*what's the purpose of this step?*)



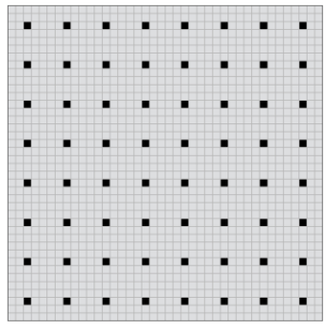
Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder. International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

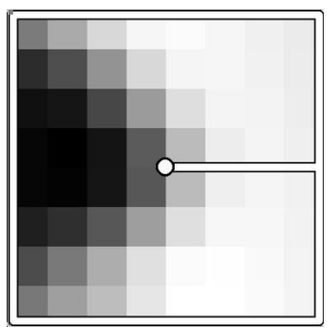
Get 40 x 40 image patch, subsample every 5th pixel

(low frequency filtering, absorbs localization errors)



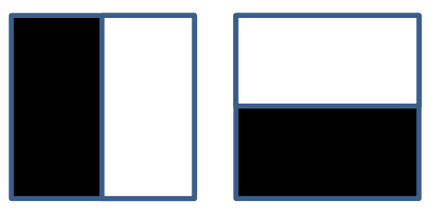
Subtract the mean, divide by standard deviation

(removes bias and gain)



Haar Wavelet Transform

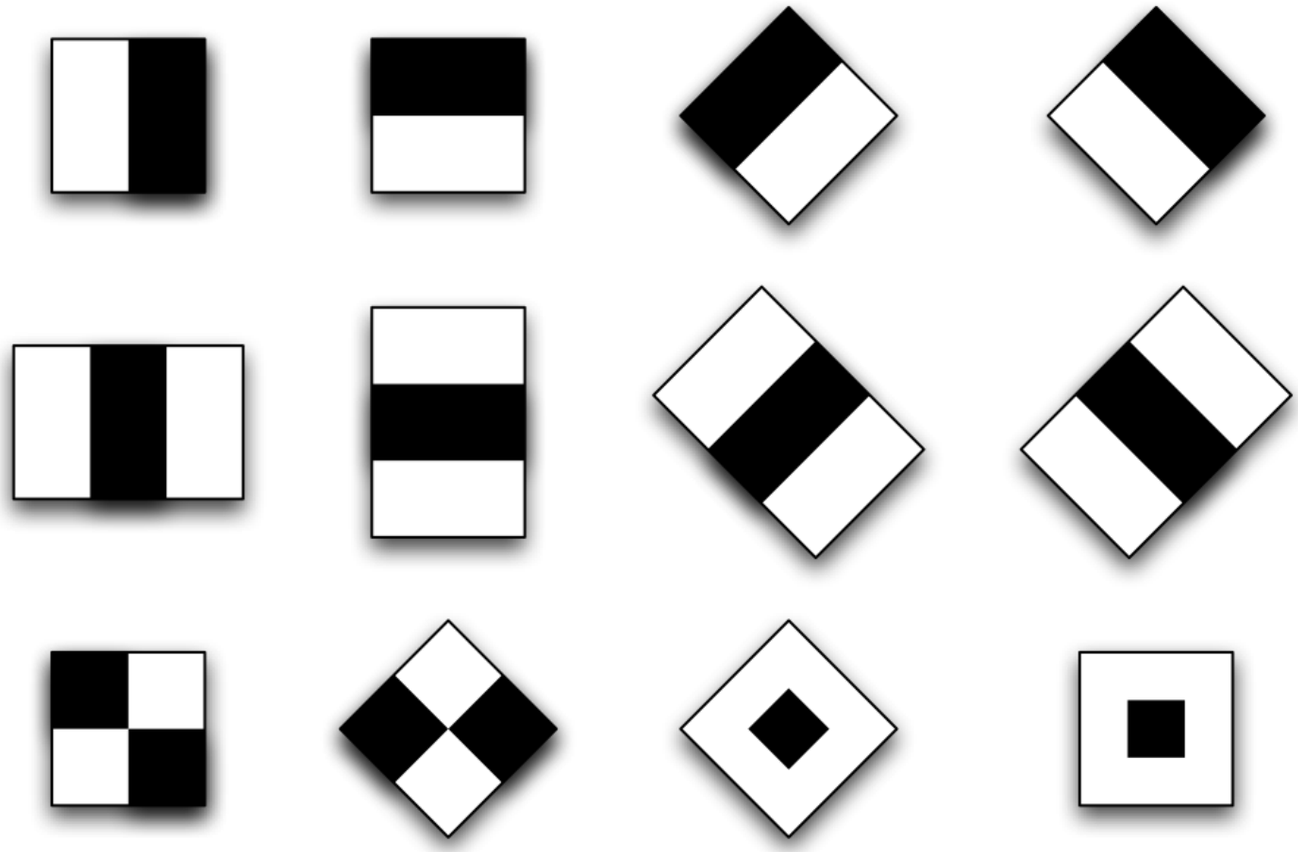
(low frequency projection)



Haar Wavelets

(actually, Haar-like features)

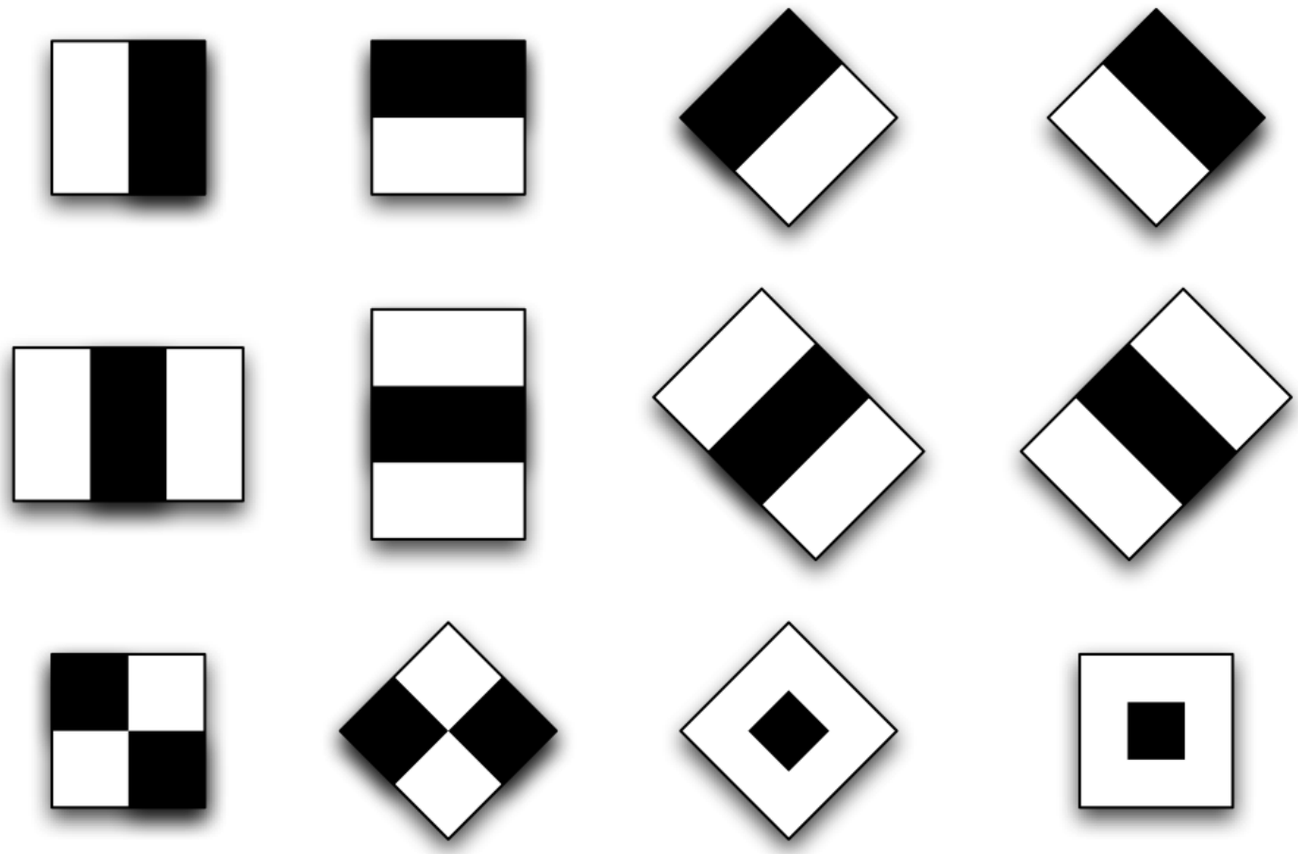
Use responses of a bank of filters as a descriptor



Haar Wavelets

(actually, Haar-like features)

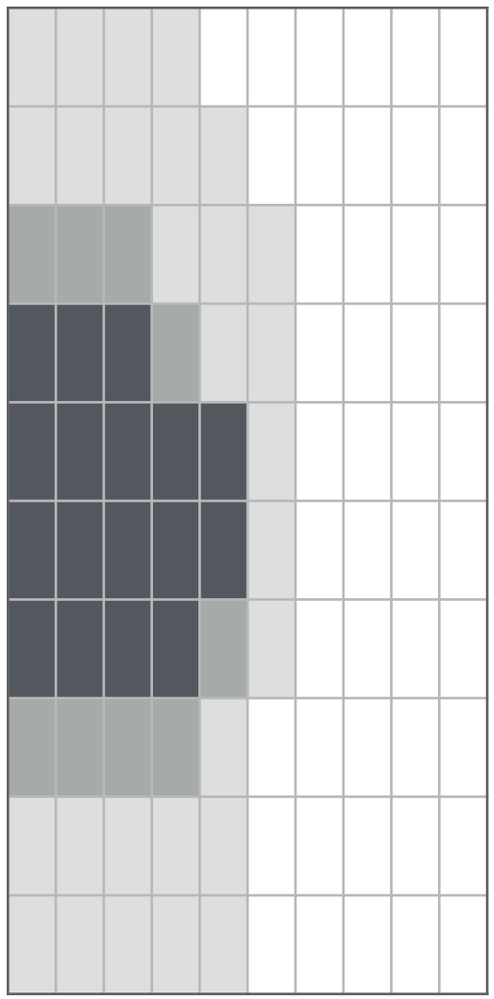
Use responses of a bank of filters as a descriptor



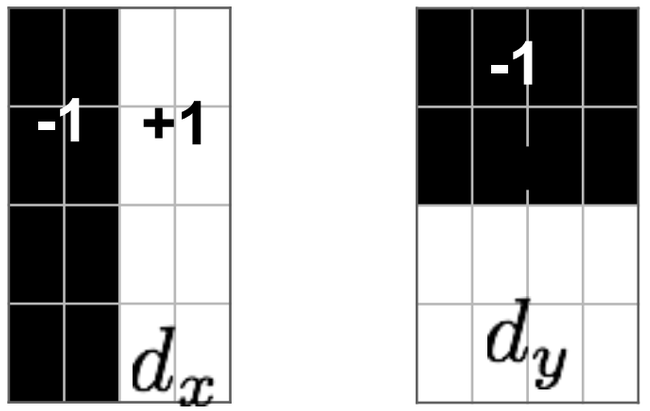
We will see later in class how to compute Haar wavelet responses **efficiently** (in constant time) with integral images

Haar wavelet responses can be computed with filtering

image patch



Haar wavelets filters



Haar wavelet responses can be computed **efficiently** (in constant time) with integral images

Integral Image

original image

$I(x, y)$

1	5	2
2	4	1
2	1	1

$A(x, y)$

1	6	8
3	12	15
5	15	19

integral image

$$A(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

Integral Image

original image

$I(x, y)$

1	5	2
2	4	1
2	1	1

$A(x, y)$

1	6	8
3	12	15
5	15	19

integral image

$$A(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

Can find the **sum** of any block using **3** operations

$$A(x_1, y_1, x_2, y_2) = A(x_2, y_2) - A(x_1, y_2) - A(x_2, y_1) + A(x_1, y_1)$$

What is the sum of the bottom right 2x2 square?

$$A(x_1, y_1, x_2, y_2) = A(x_2, y_2) - A(x_1, y_2) - A(x_2, y_1) + A(x_1, y_1)$$

$I(x, y)$

1	5	2
2	4	1
2	1	1

image

$A(x, y)$

1	6	**8**
3	12	15
5	15	**19**

integral image

$$\begin{aligned} A(1,1,3,3) &= A(3,3) - A(1,3) - A(3,1) + A(1,1) \\ &= 19 - 8 - 5 + 1 \\ &= 7 \end{aligned}$$

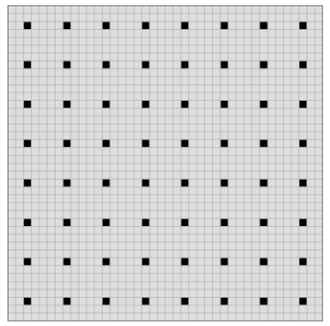
Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder. International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

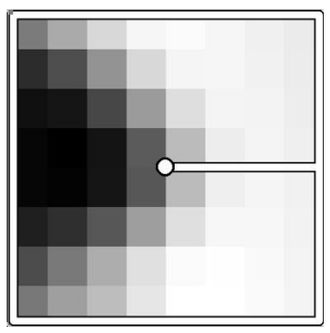
Get 40 x 40 image patch, subsample every 5th pixel

(low frequency filtering, absorbs localization errors)



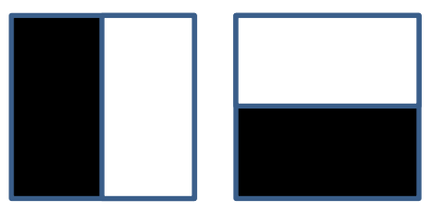
Subtract the mean, divide by standard deviation

(removes bias and gain)



Haar Wavelet Transform

(low frequency projection)



Global Descriptors

← Local

↓ Global

GIST descriptor

GIST

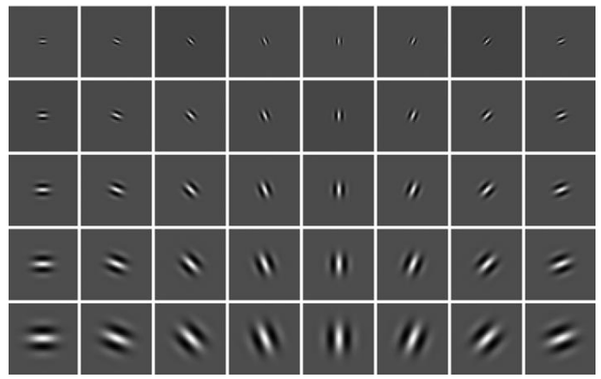
1. Compute filter responses (filter bank of Gabor filters)

2. Divide image patch into 4 x 4 cells

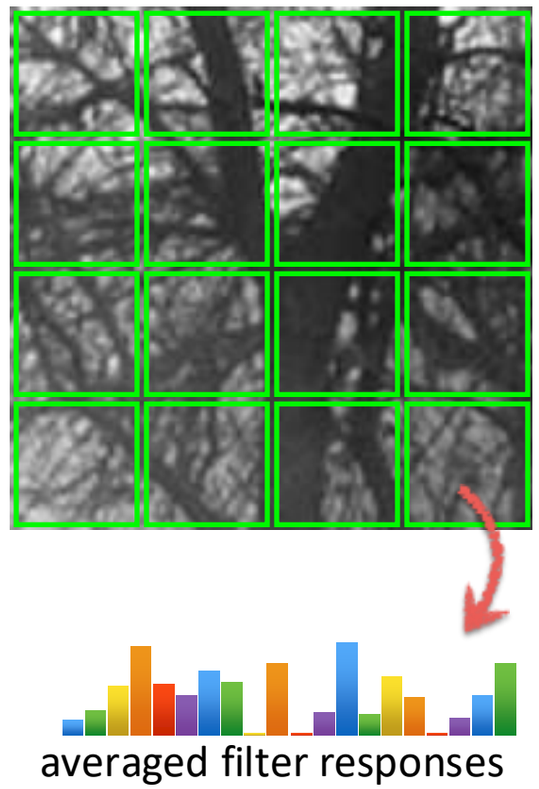
3. Compute filter response averages for each cell

4. Size of descriptor is 4 x 4 x N, where N is the size of the filter bank

Filter bank



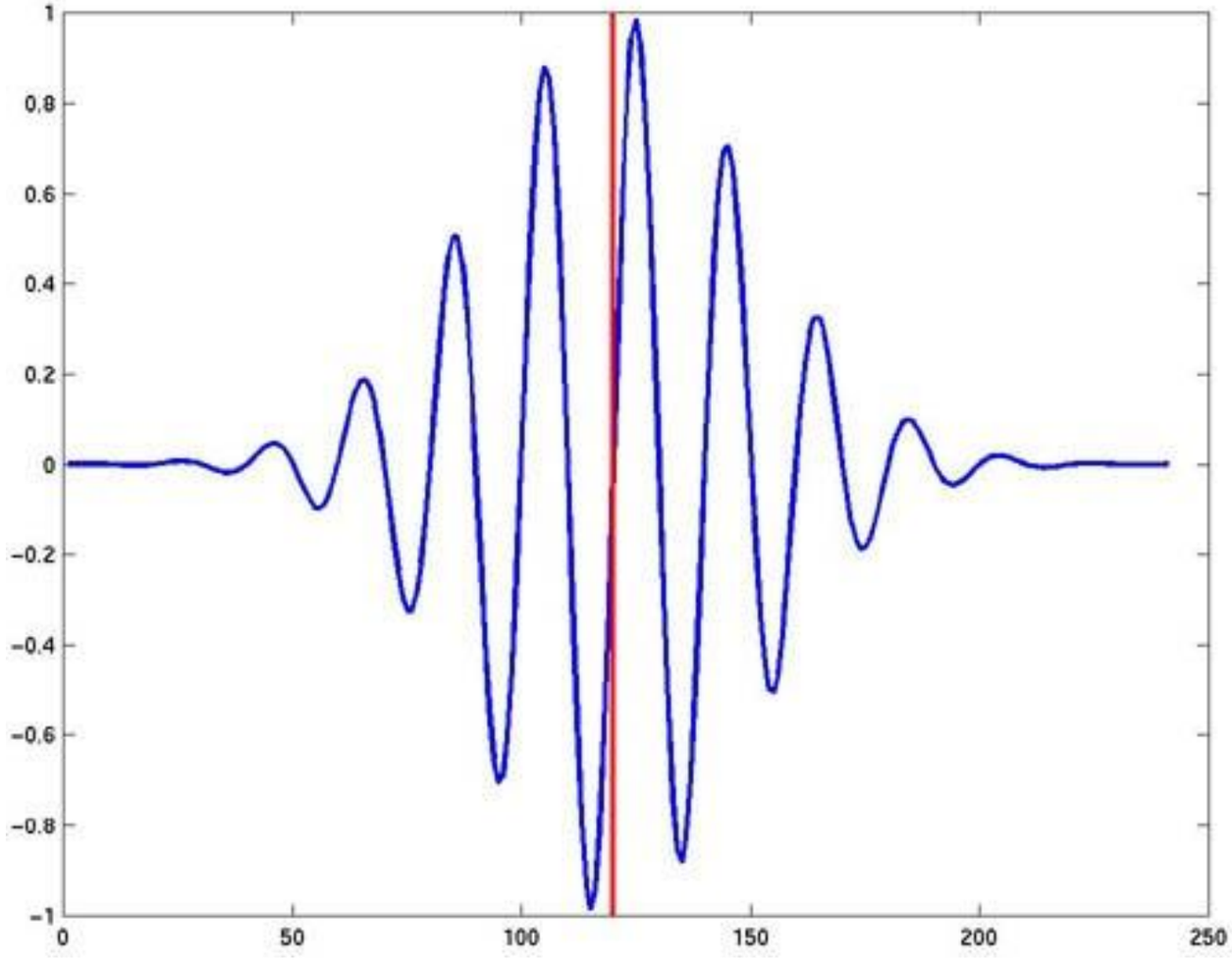
4 x 4 cell



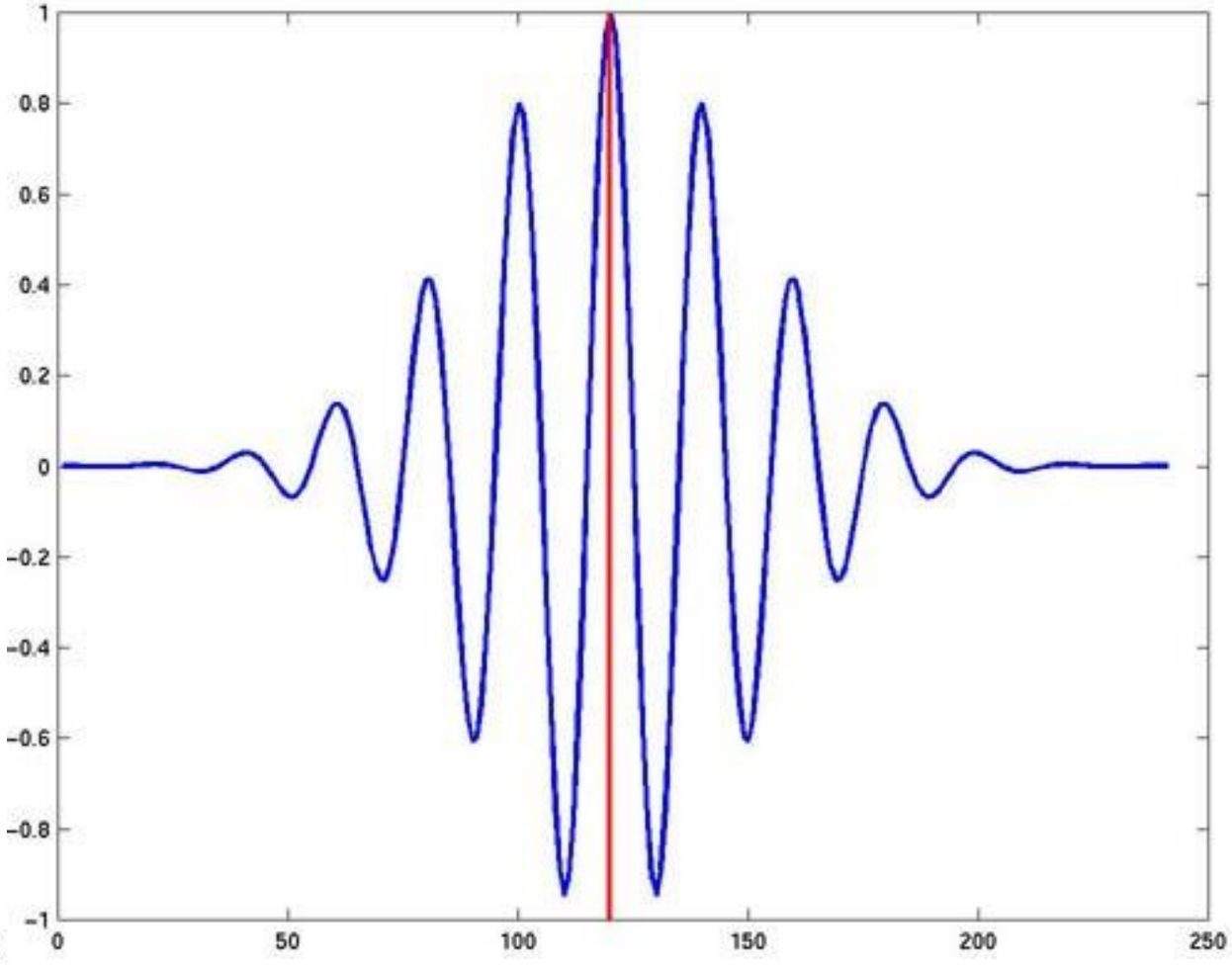
averaged filter responses

Gabor Filters

(1D examples)



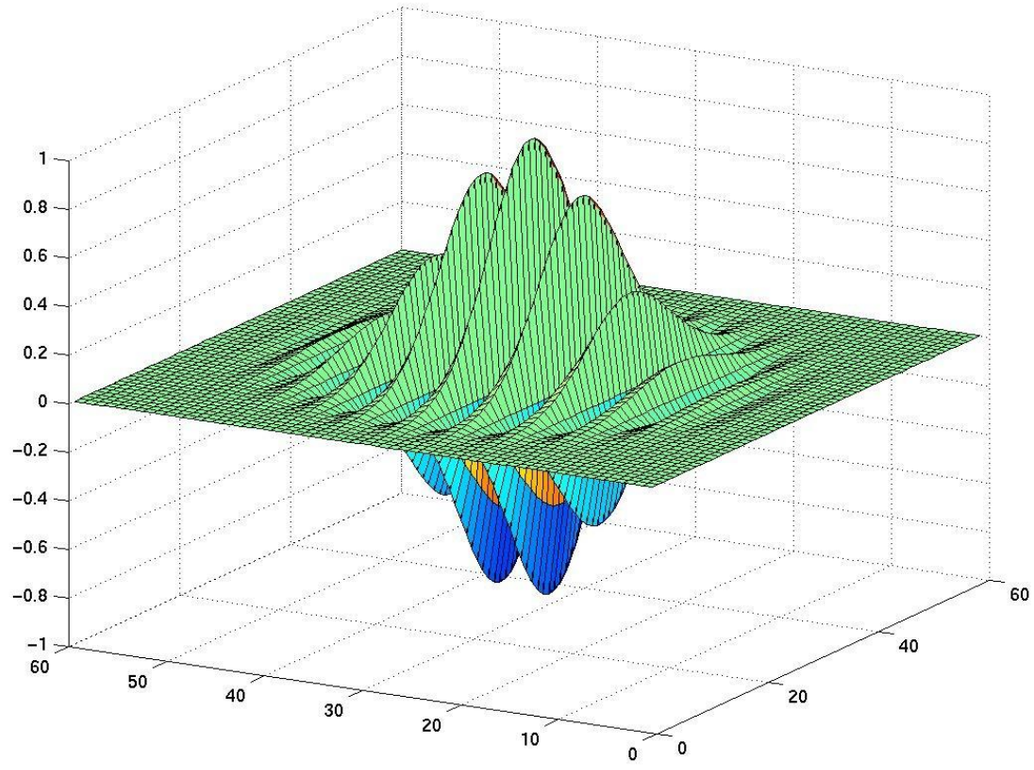
$$e^{-\frac{x^2}{2\sigma^2}} \sin(2\pi\omega x)$$

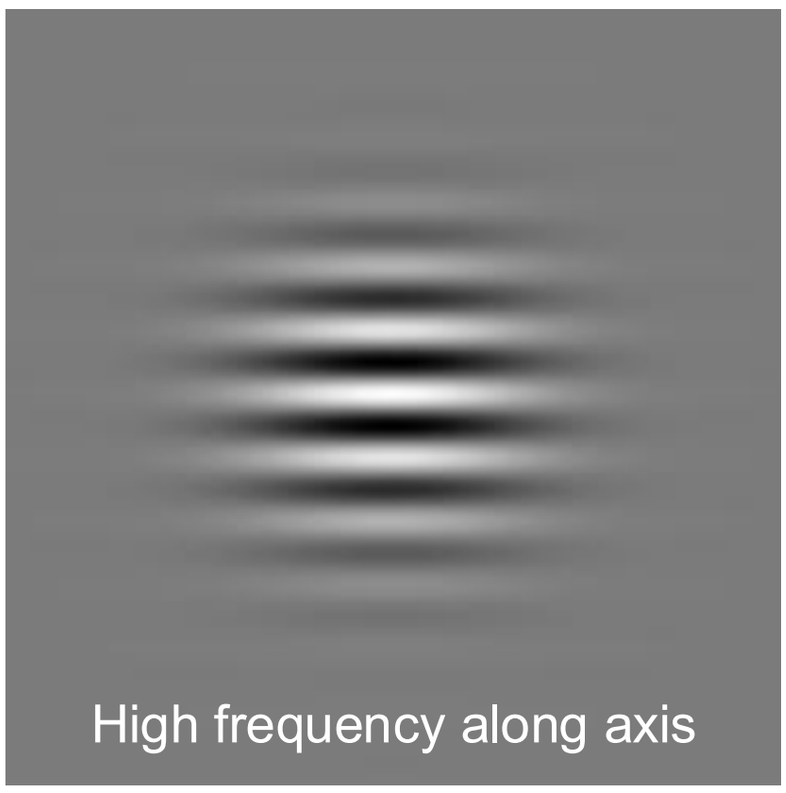


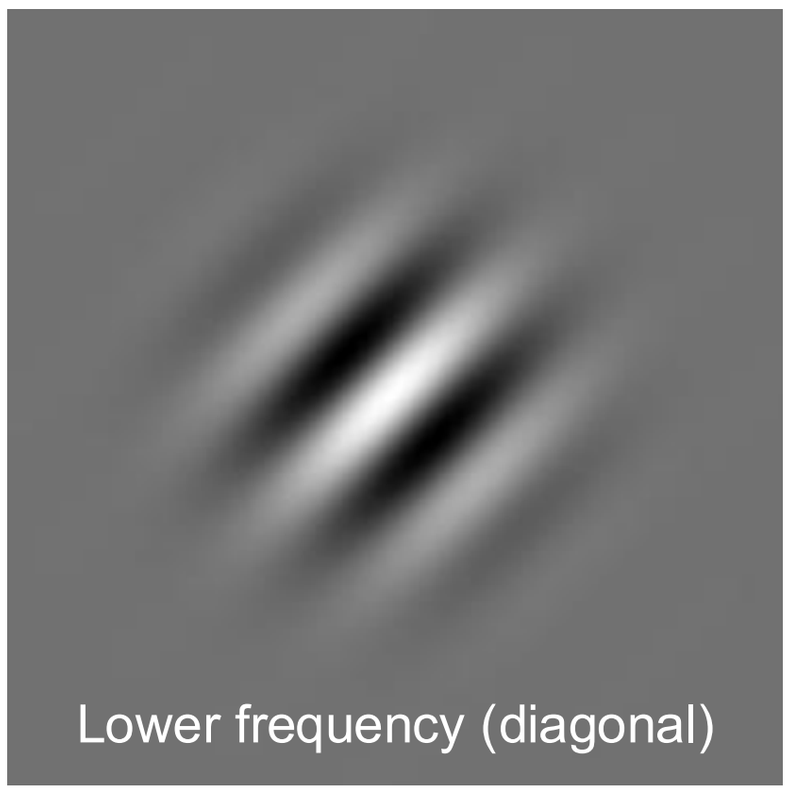
$$e^{-\frac{x^2}{2\sigma^2}} \cos(2\pi\omega x)$$

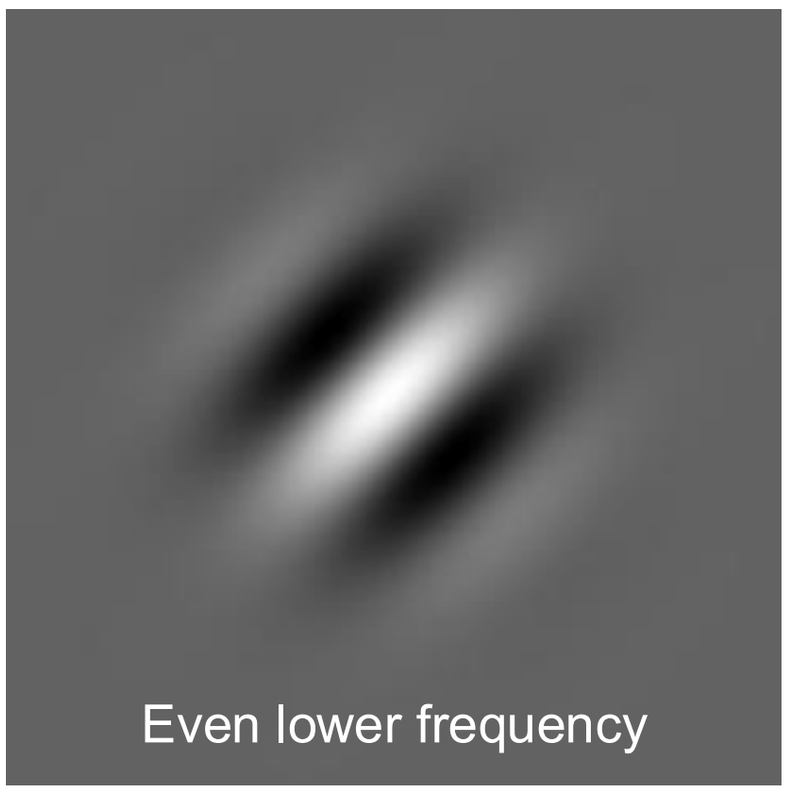
2D Gabor Filters

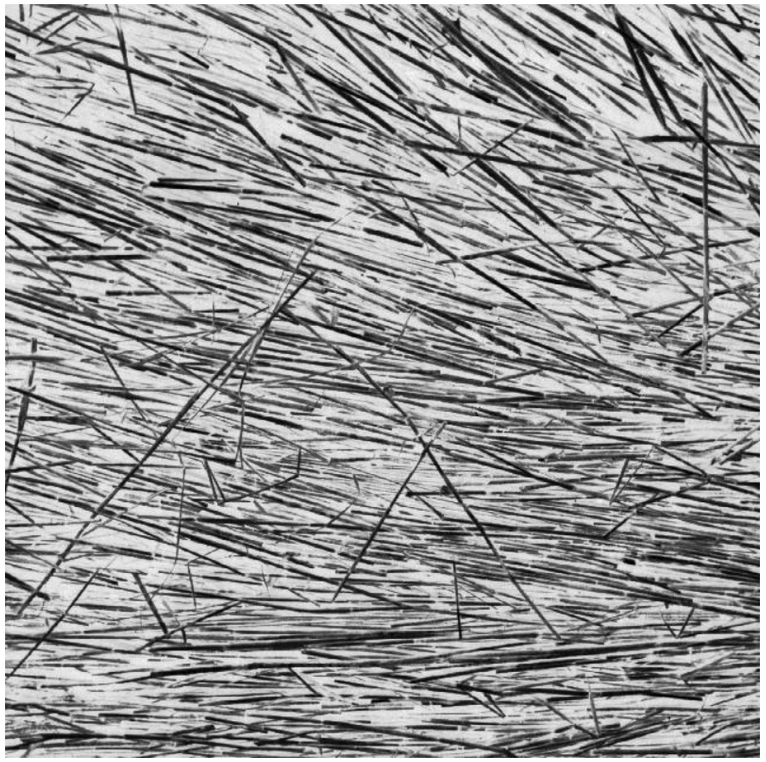
$$e^{-\frac{x^2+y^2}{2\sigma^2}} \cos\left(2\pi\left(k_x x + k_y y\right)\right)$$

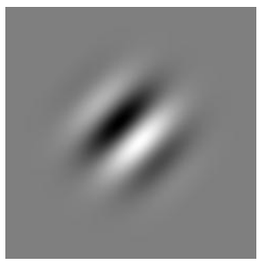


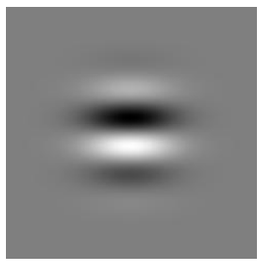


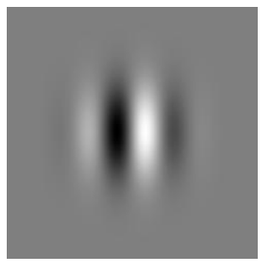


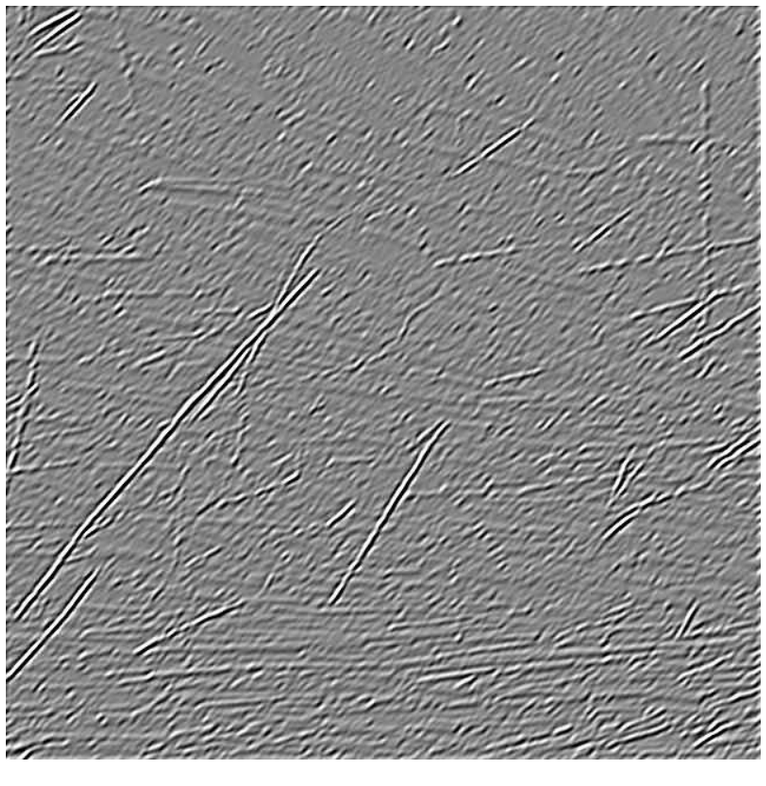


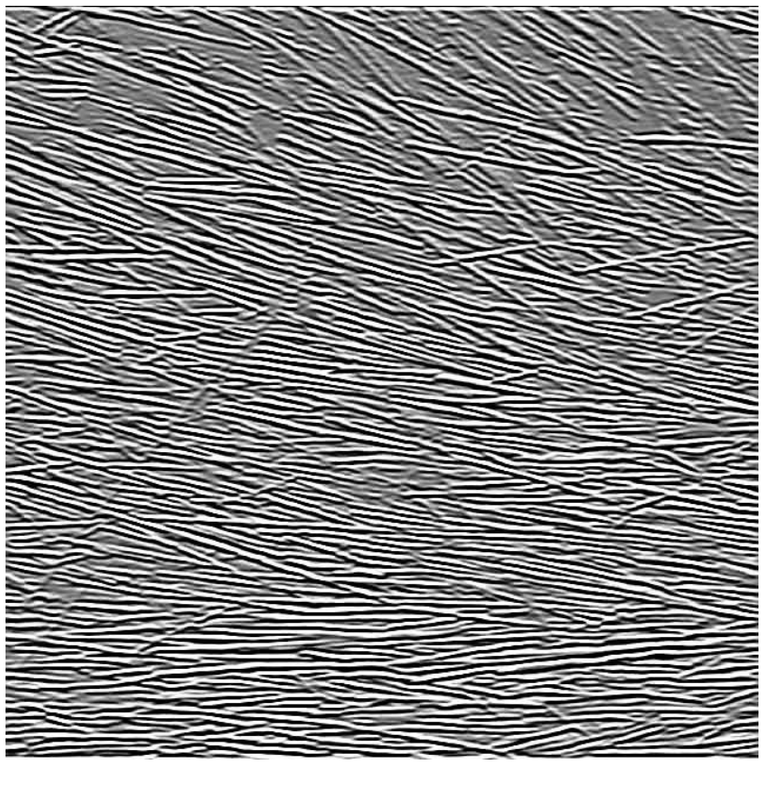


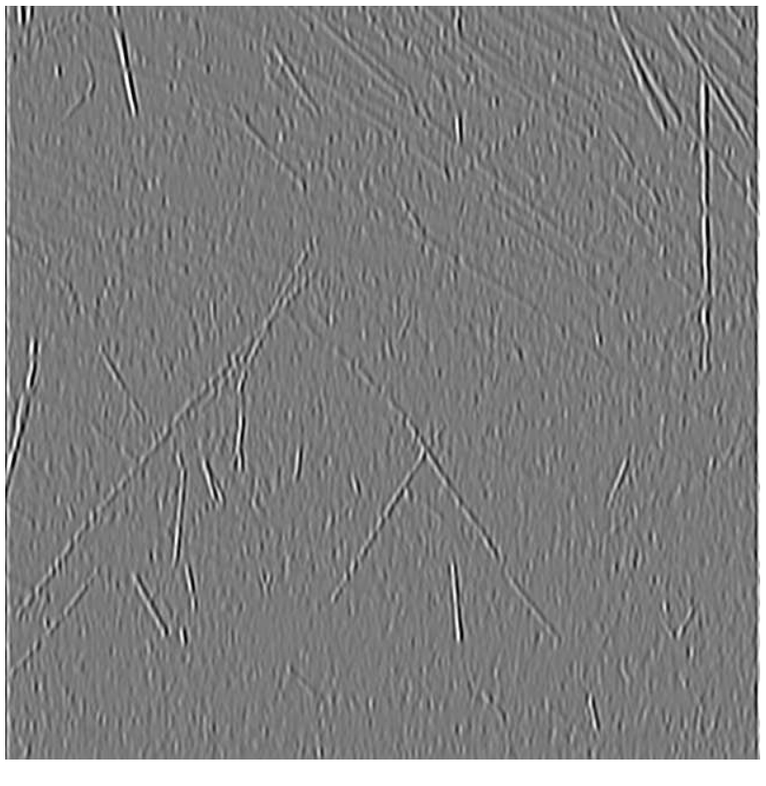


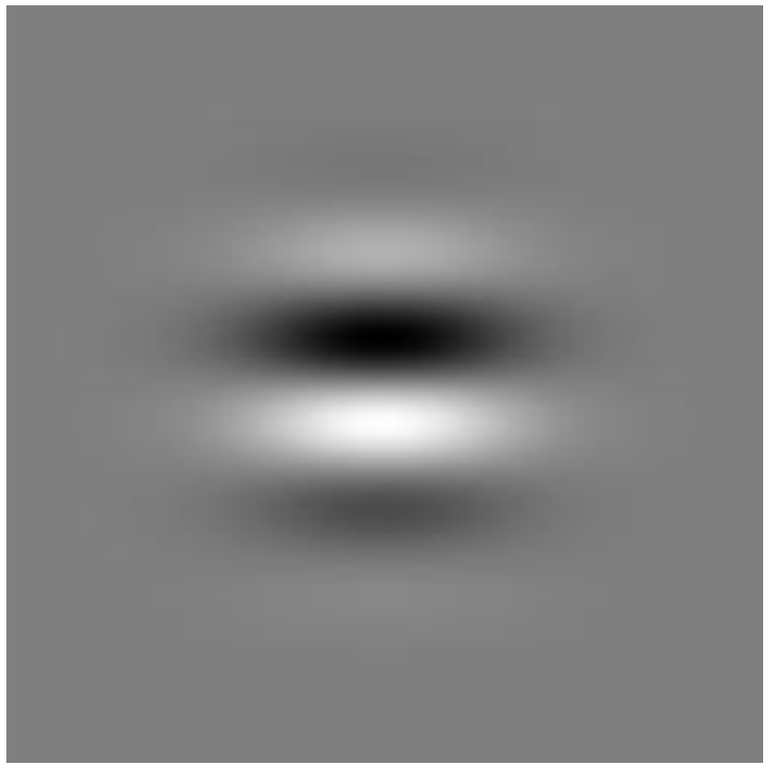




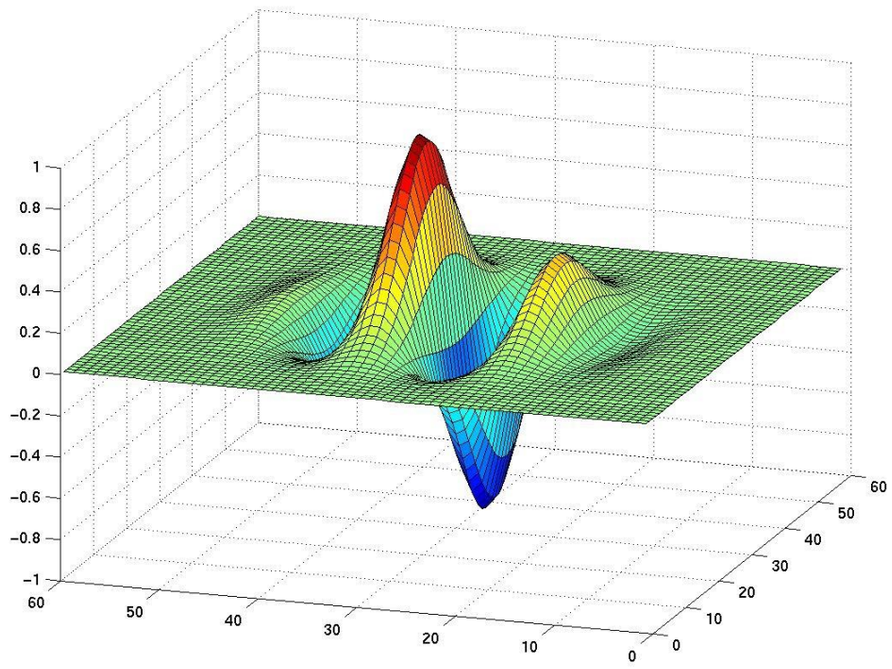




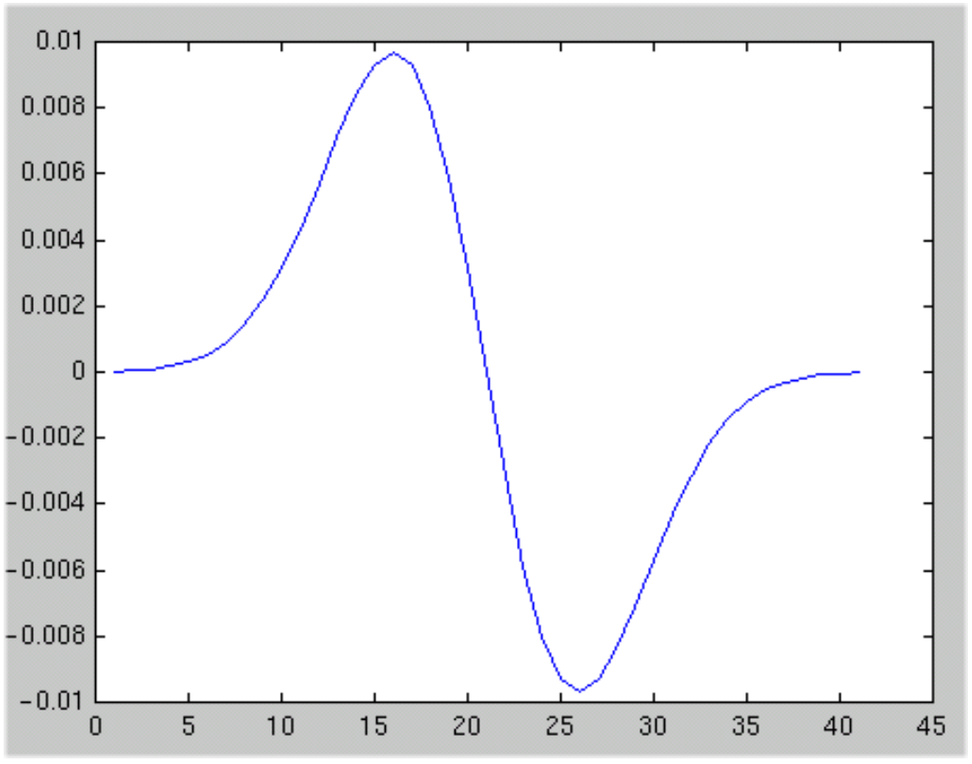




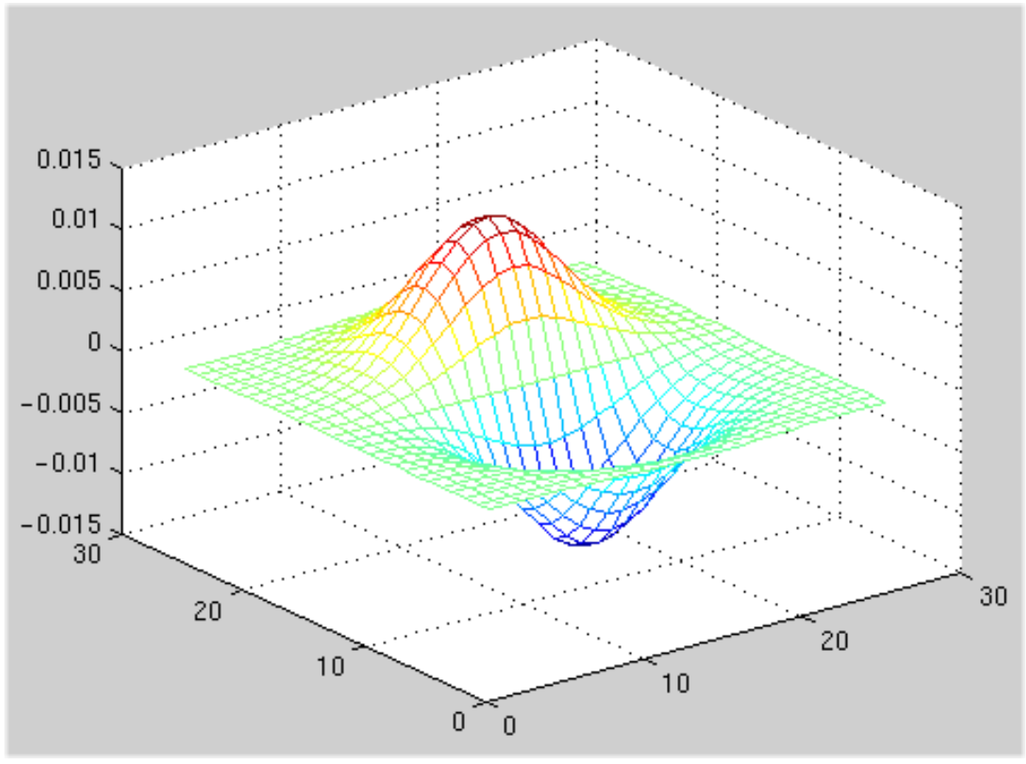
Odd Gabor filter

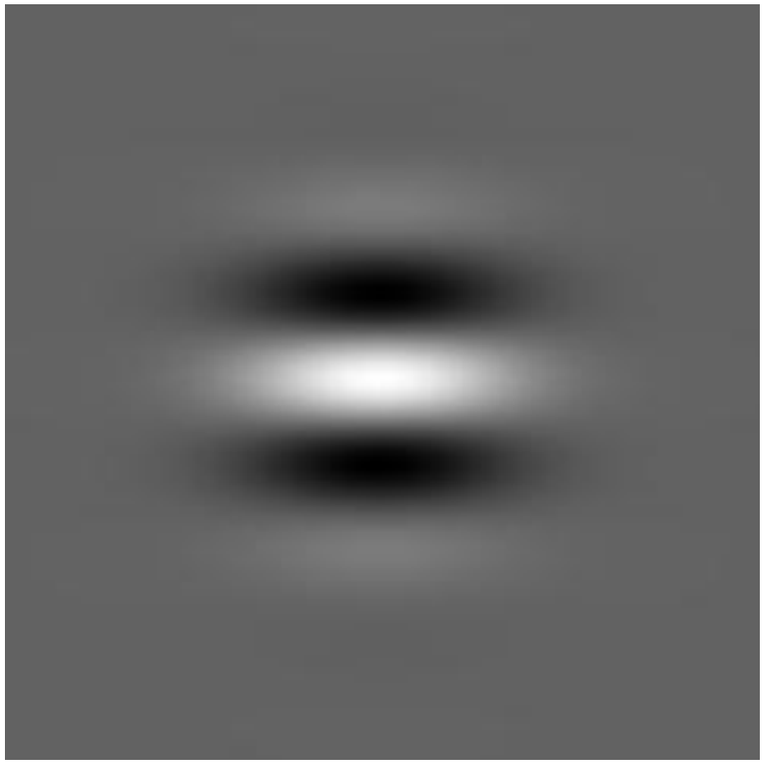


… looks a lot like…

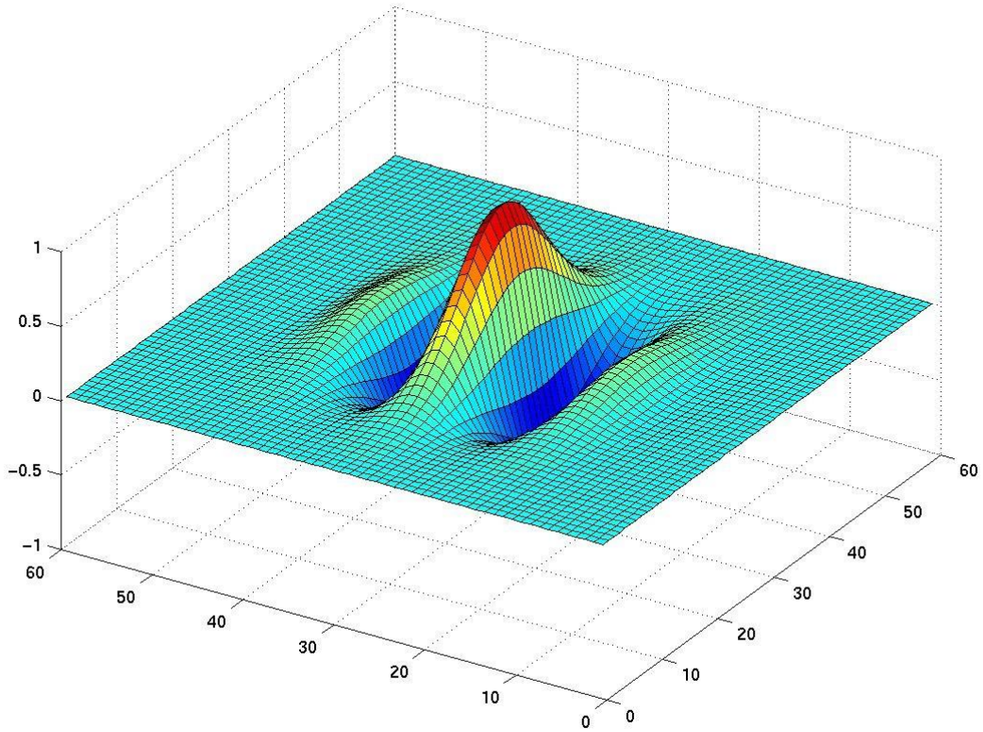


Gaussian Derivative

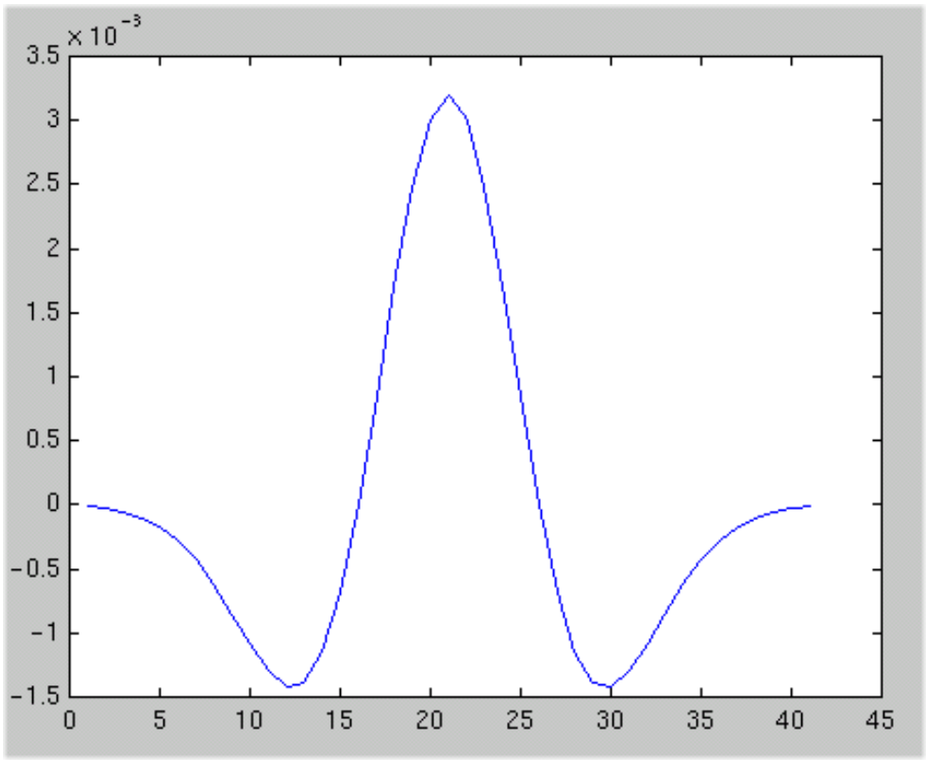




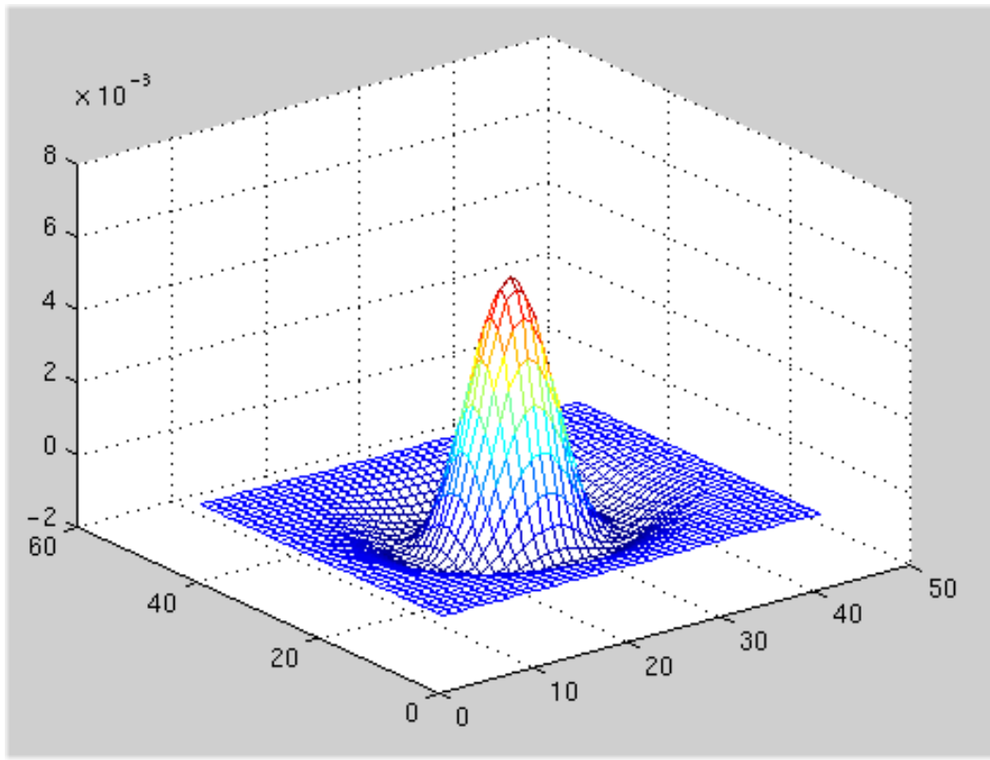
Even Gabor filter



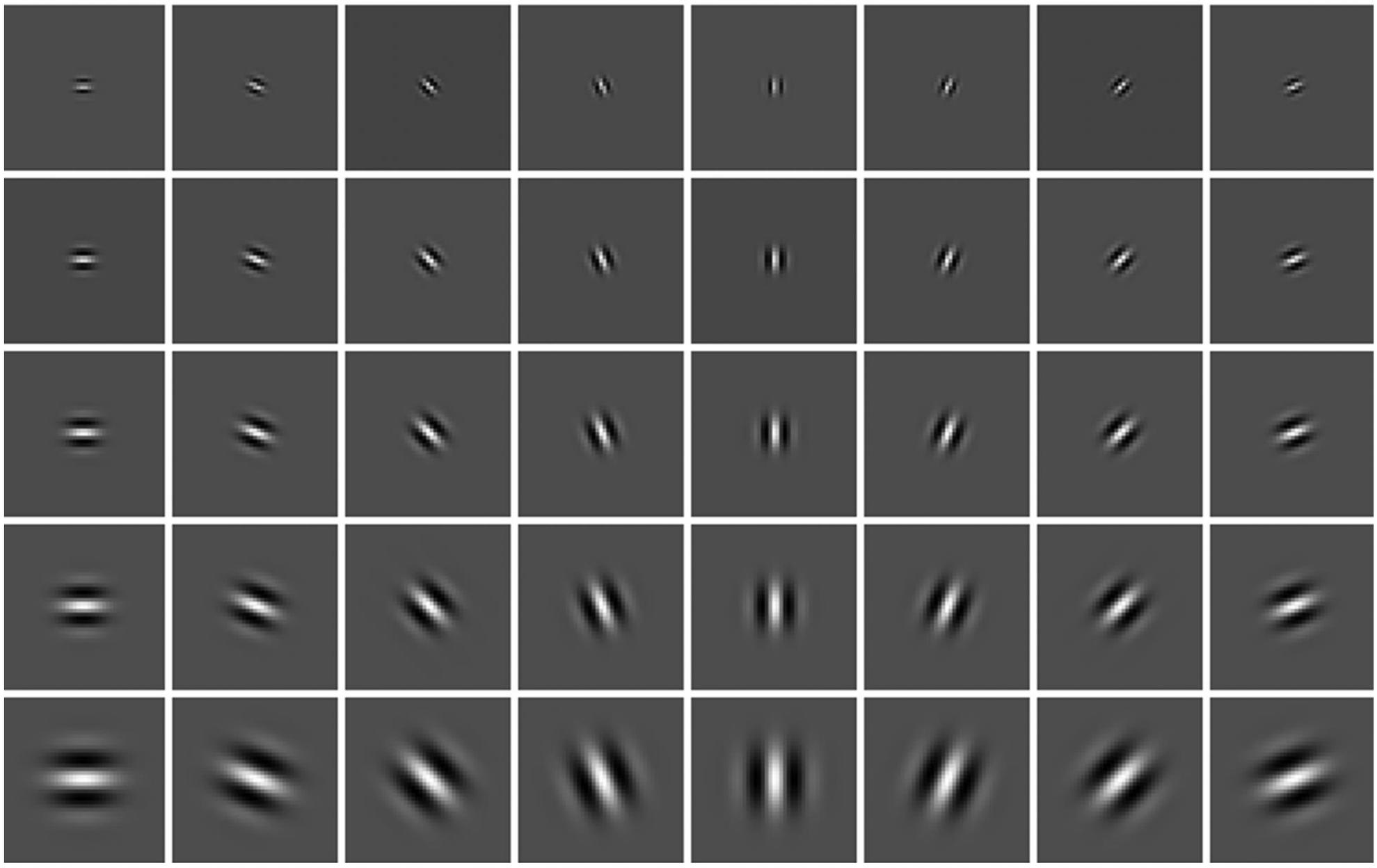
… looks a lot like…



Laplacian



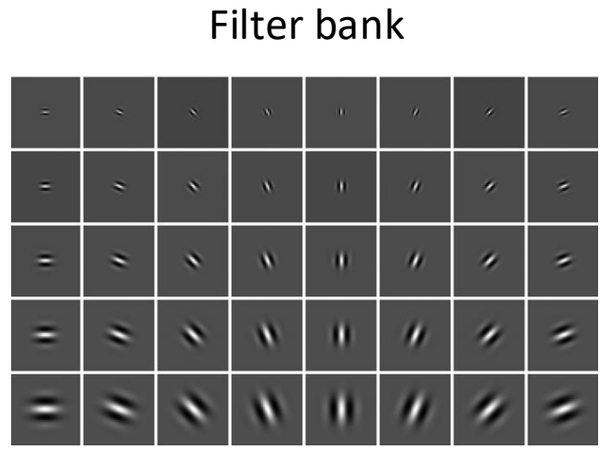
Directional edge detectors

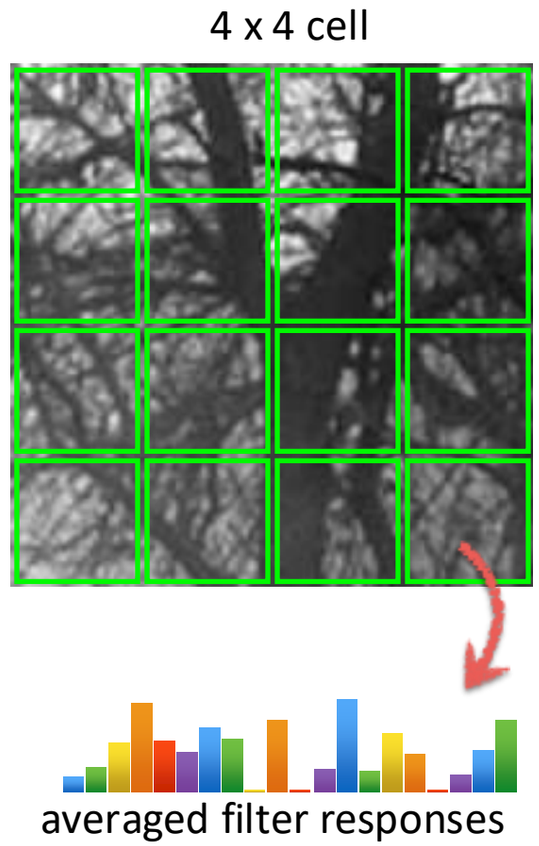


GIST

1. Compute filter responses (filter bank of Gabor filters)
2. Divide image patch into 4 x 4 cells
3. Compute filter response averages for each cell
4. Size of descriptor is 4 x 4 x N, where N is the size of the filter bank

What is the GIST descriptor encoding?

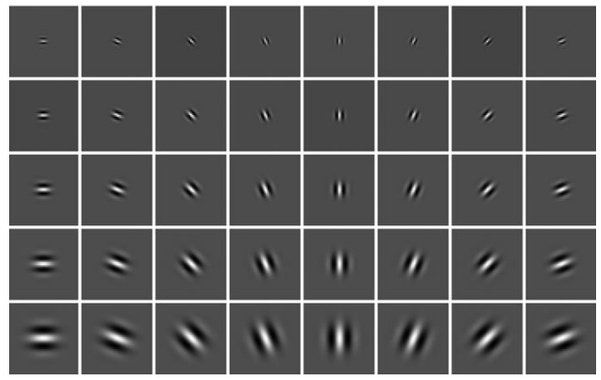




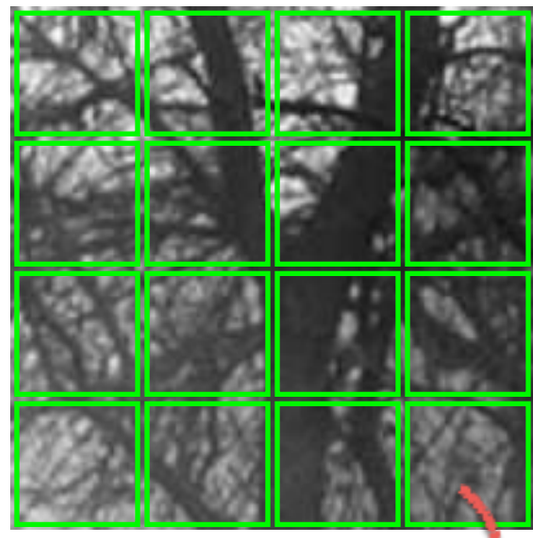
GIST

1. Compute filter responses (filter bank of Gabor filters)
2. Divide image patch into 4 x 4 cells
3. Compute filter response averages for each cell
4. Size of descriptor is 4 x 4 x N, where N is the size of the filter bank

Filter bank



4 x 4 cell



averaged filter responses

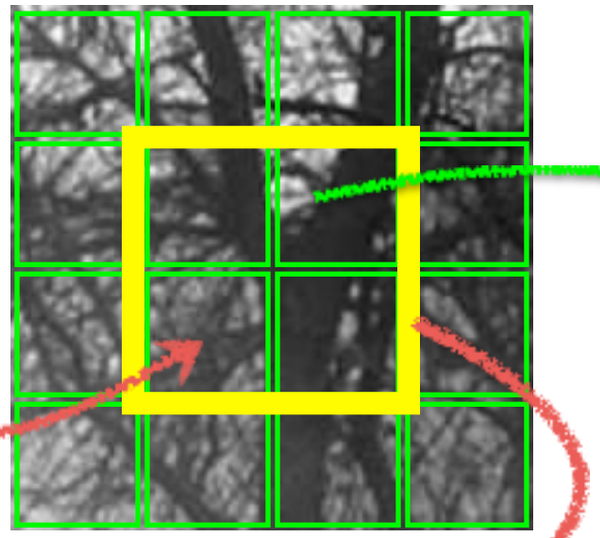
What is the GIST descriptor encoding?

Rough spatial distribution of image gradients

HOG descriptor

HOG

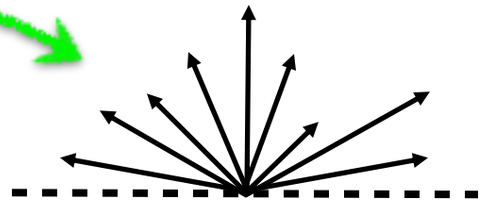
Dalal, Triggs. **Histograms of Oriented Gradients** for Human Detection. CVPR, 2005



Cell
(8x8 pixels)

Block
(2x2 cells)

histogram of ‘unsigned’ gradients



soft binning

gradient magnitude histogram
(one for each cell)

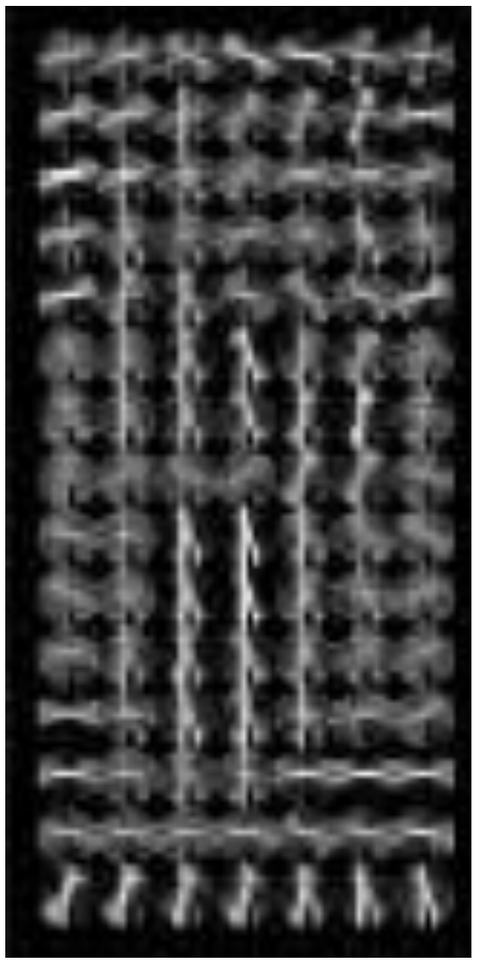
Concatenate and L-2 normalization

Single scale, no dominant orientation

Pedestrian detection

1 cell step size

visualization



128 pixels
16 cells
15 blocks

15 x 7 x 4 x 9 =
3780

64 pixels
8 cells
7 blocks

Redundant representation due to overlapping blocks

How many times is each inner cell encoded?





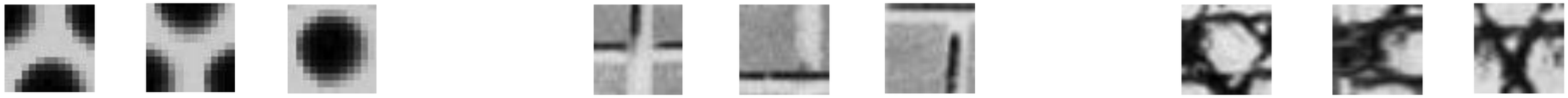
http://chrisjmccormick.wordpress.com/2013/05/09/hog-person-detector-tutorial/

Histogram of Textons descriptor

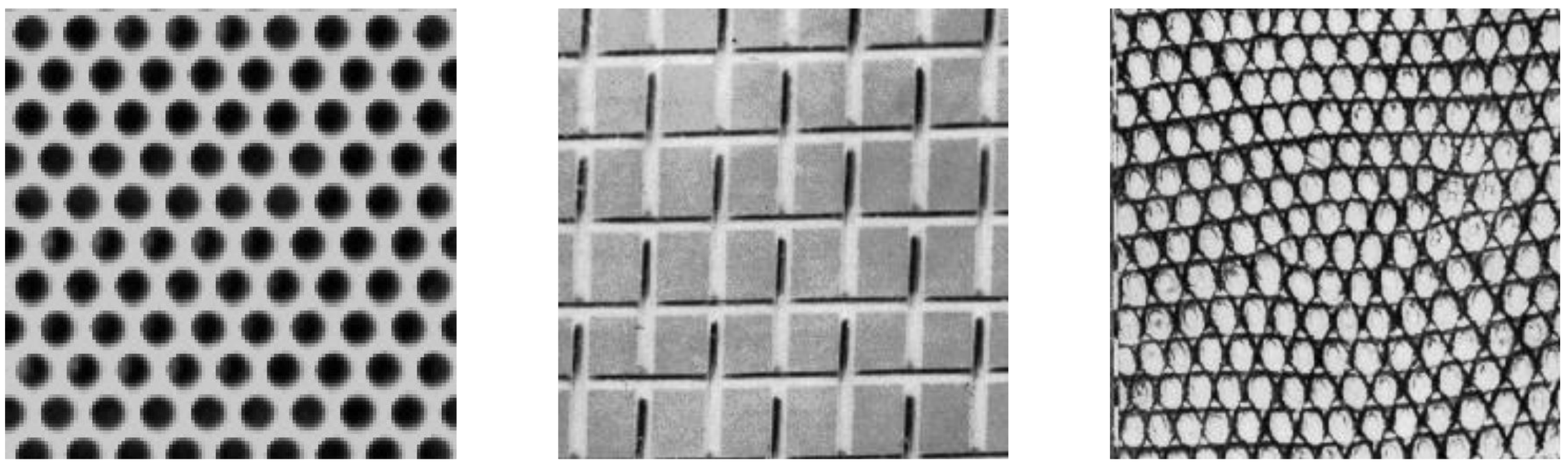
Textons

Julesz. Textons, the elements of texture perception, and their interactions. Nature 1981

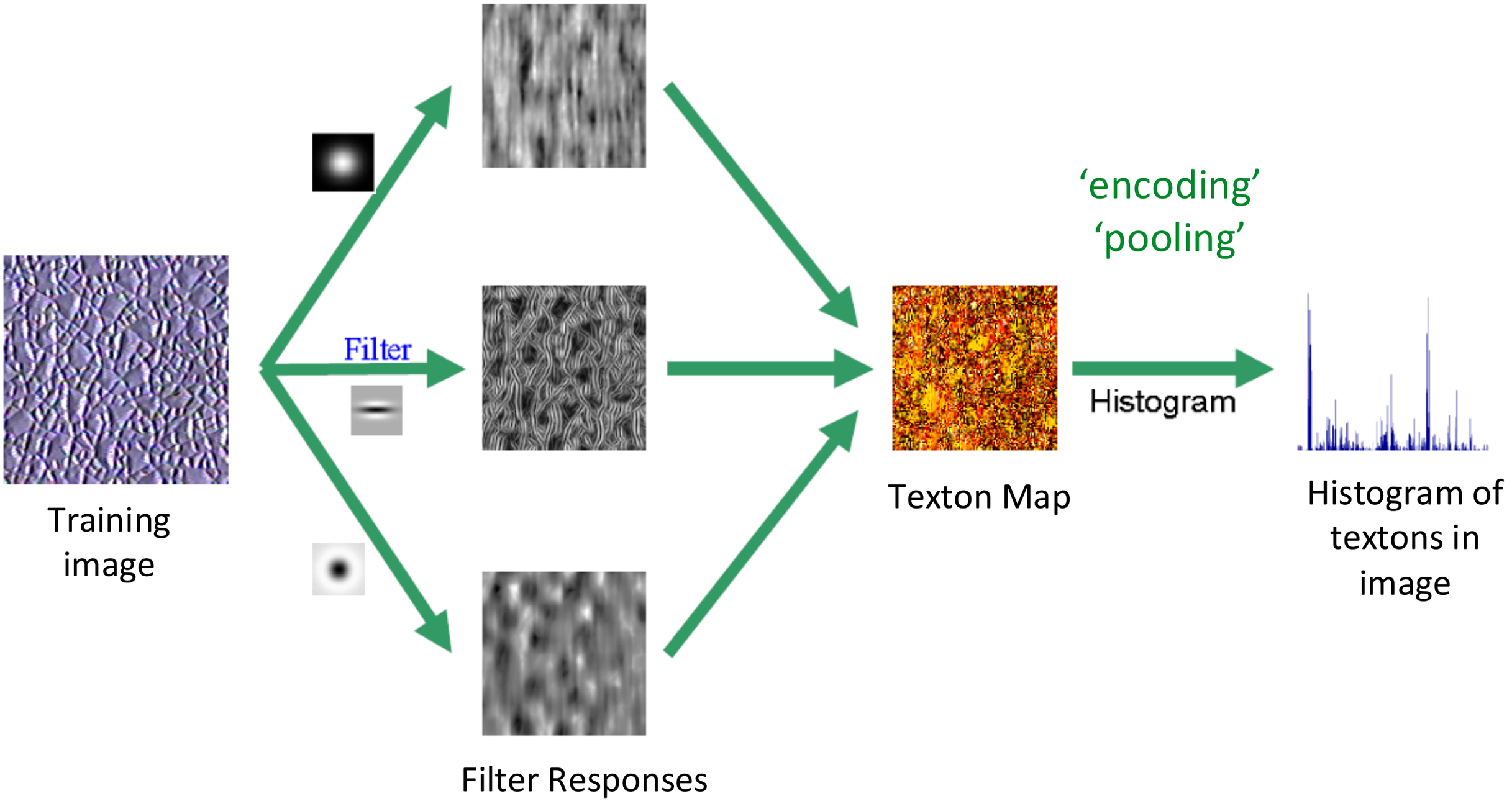
Texture is characterized by the repetition of basic elements or ***textons***



For stochastic textures, it is the identity of the ***textons***, not their spatial arrangement, that matters

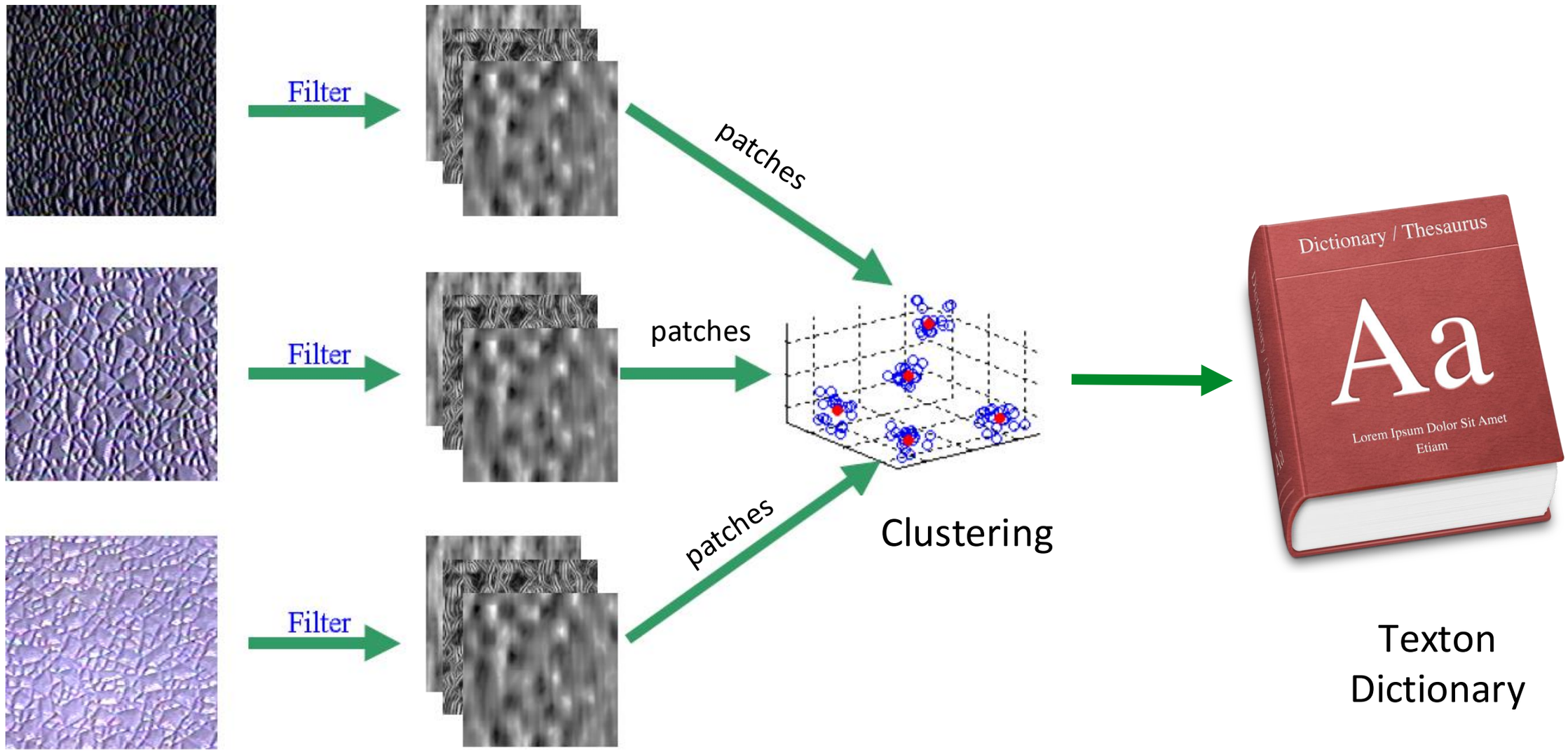


Histogram of Textons descriptor



Malik, Belongie, Shi, Leung. Textons, Contours and Regions: Cue Integration in Image Segmentation. ICCV 1999.

Learning Textons from data

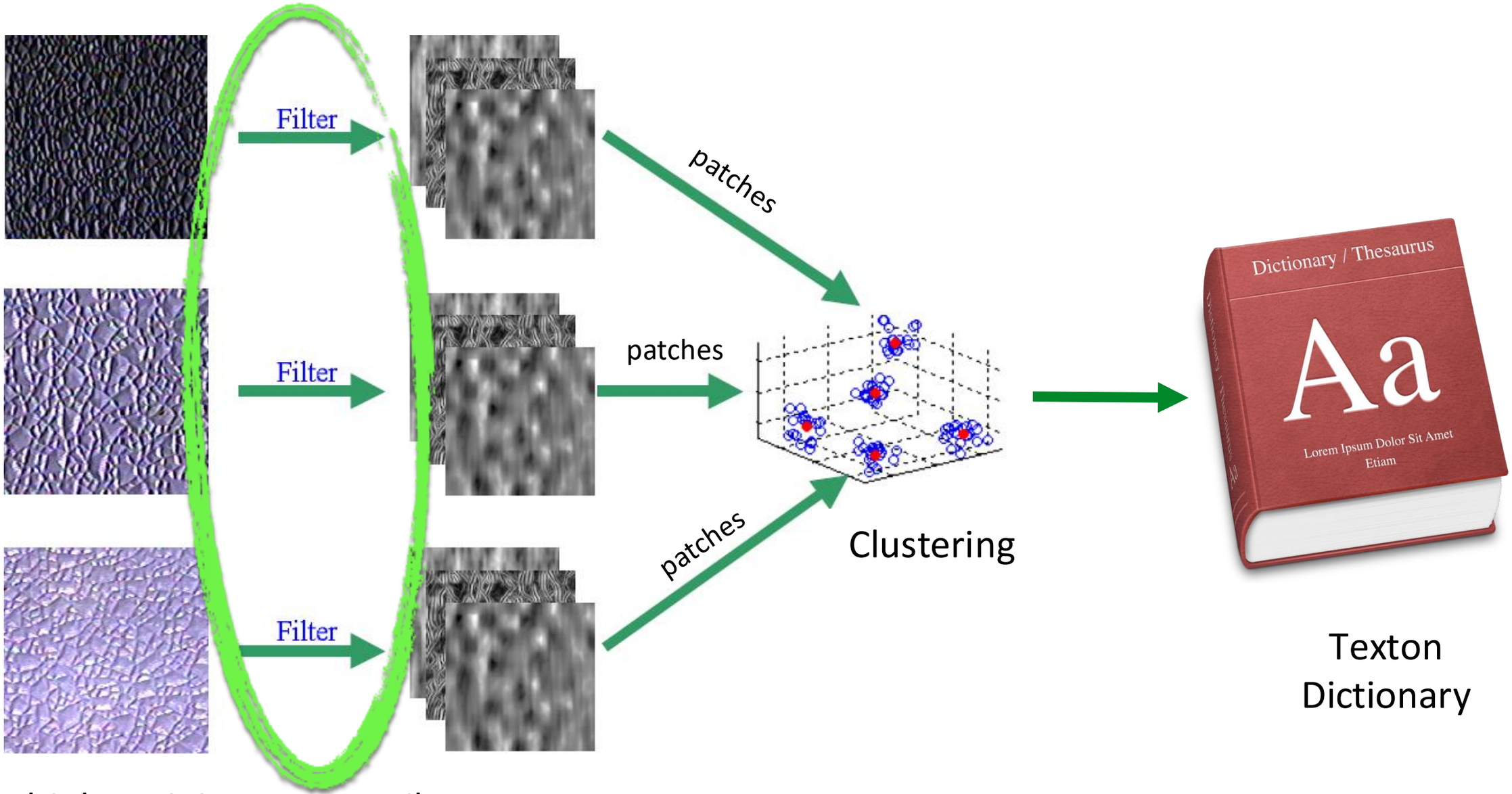


Multiple training images of the **same texture**

Filter response over a bank of filters

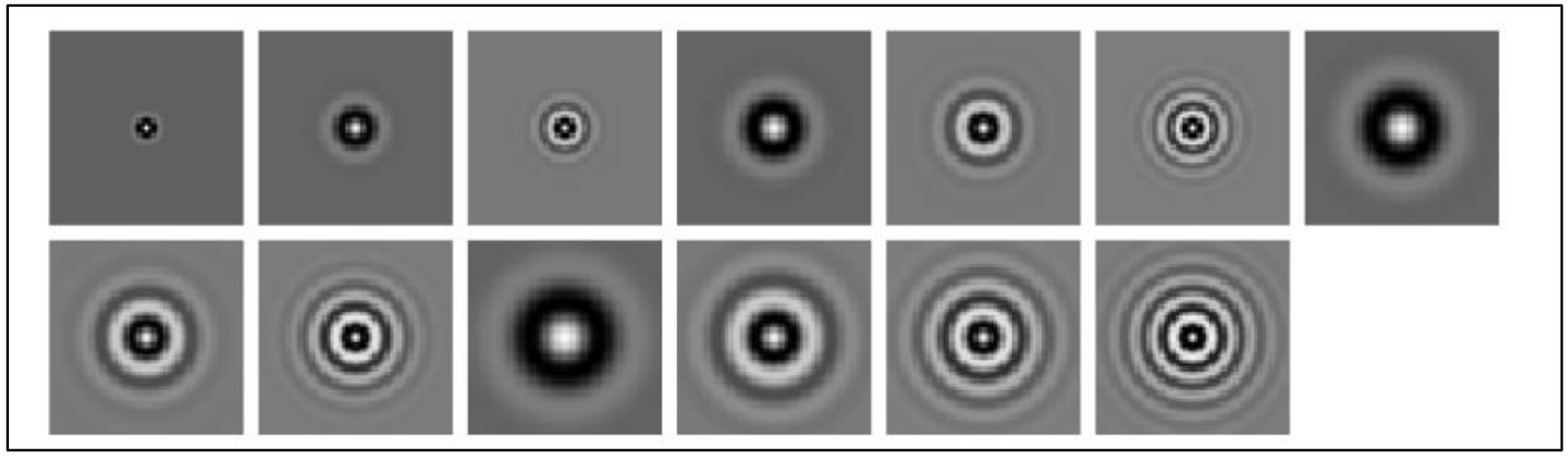
Texton Dictionary

Learning Textons from data



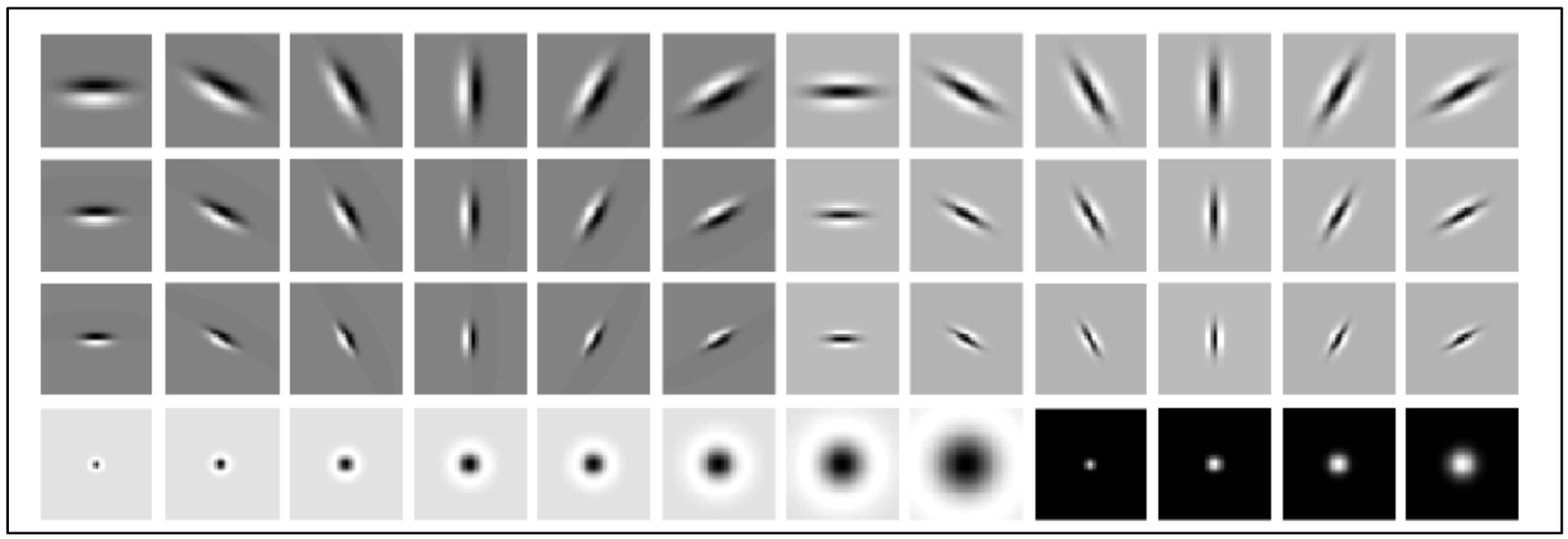
Example of Filter Banks

Isotropic Gabor

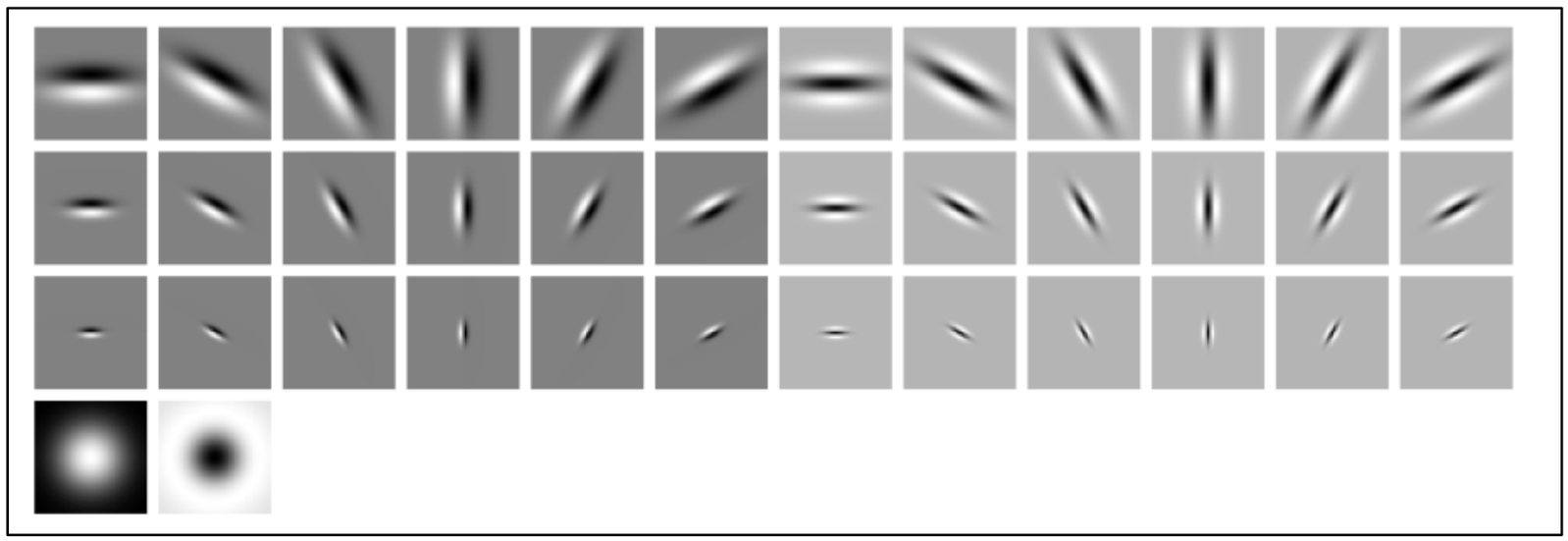


'S'

Gaussian derivatives at different scales and orientations

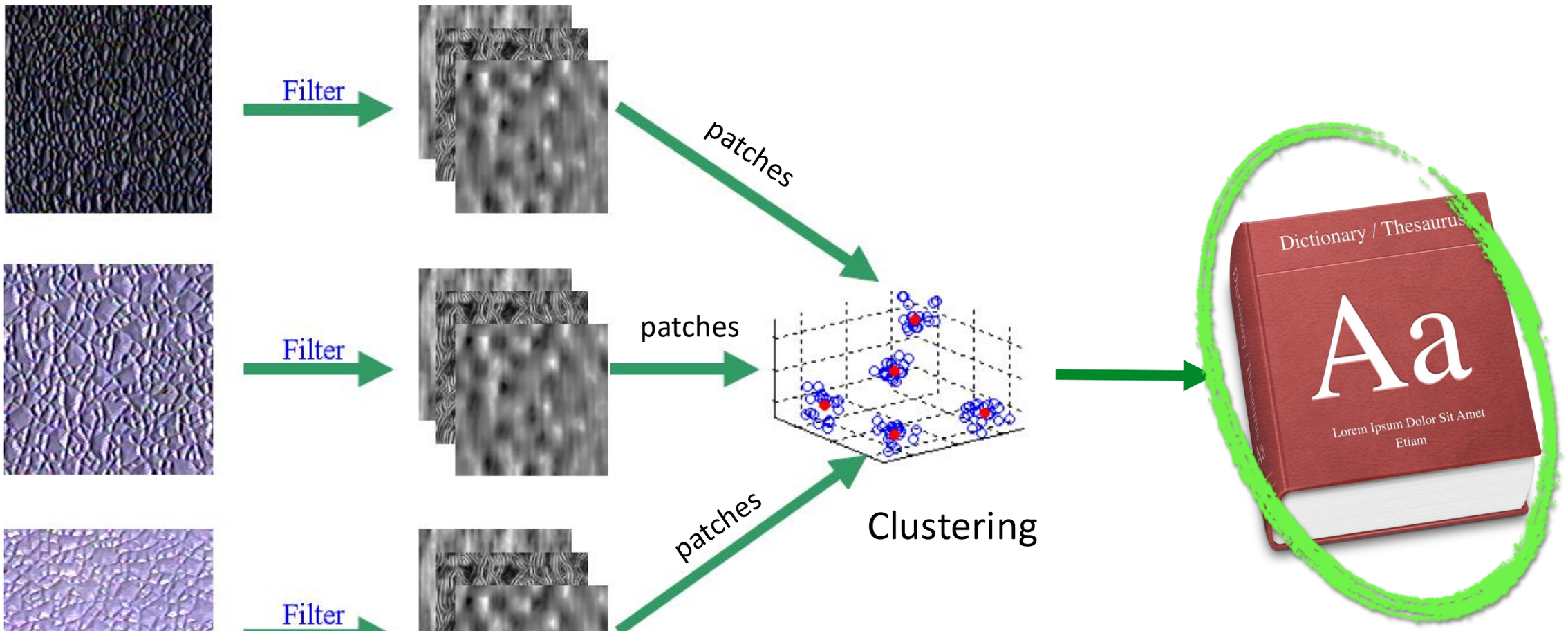


'LM'



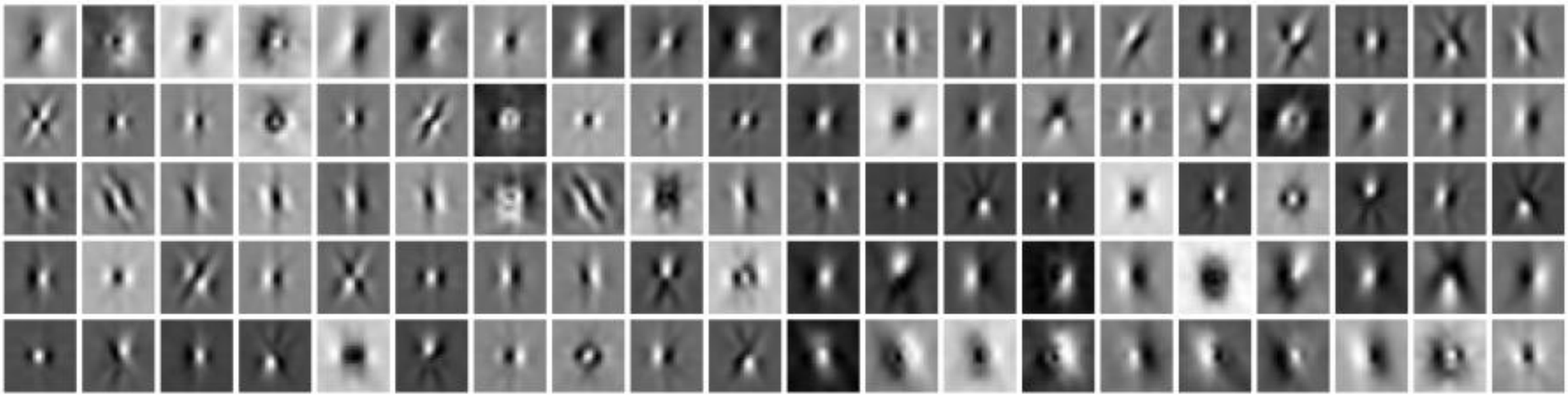
'MR8'

Learning Textons from data

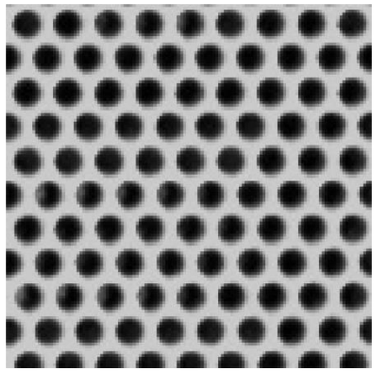


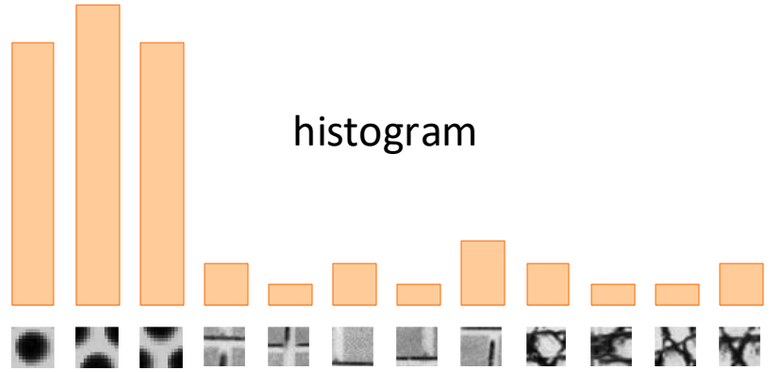
We will learn more about clustering later in class (Bag of Words lecture).

Texton Dictionary

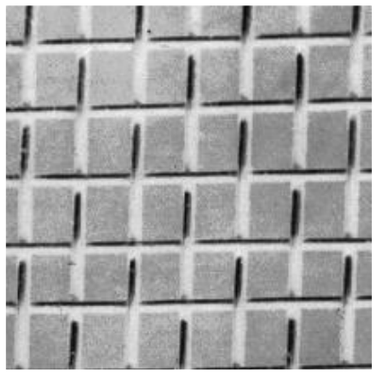


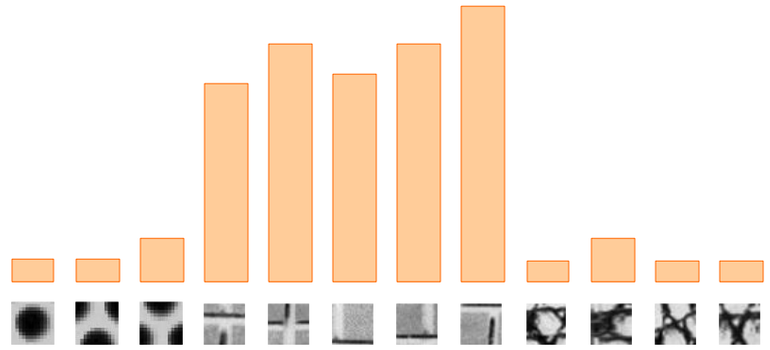
Malik, Belongie, Shi, Leung. Textons, Contours and Regions: Cue Integration in Image Segmentation. ICCV 1999.

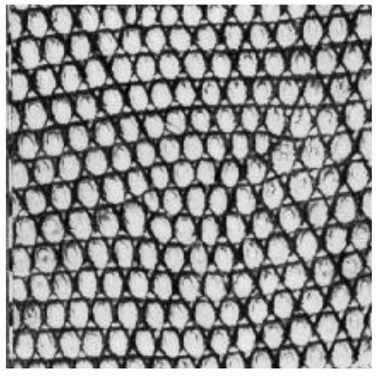


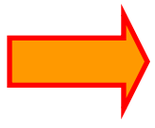


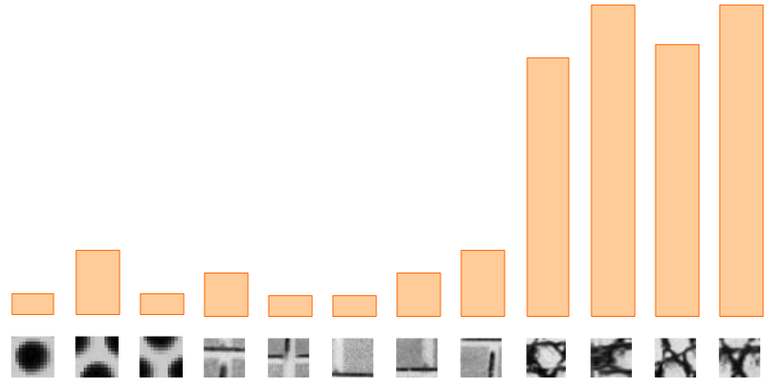
Universal texton dictionary











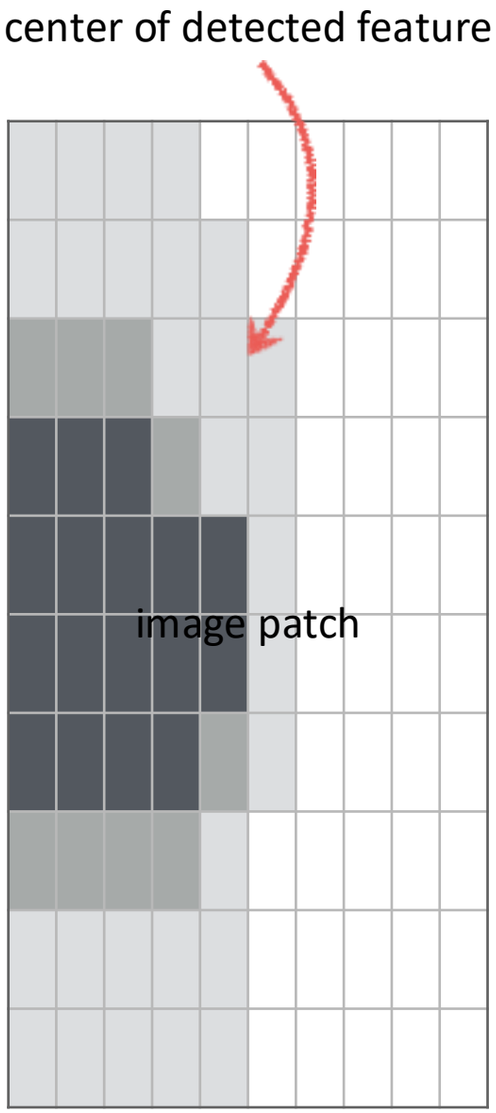
Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

SURF descriptor

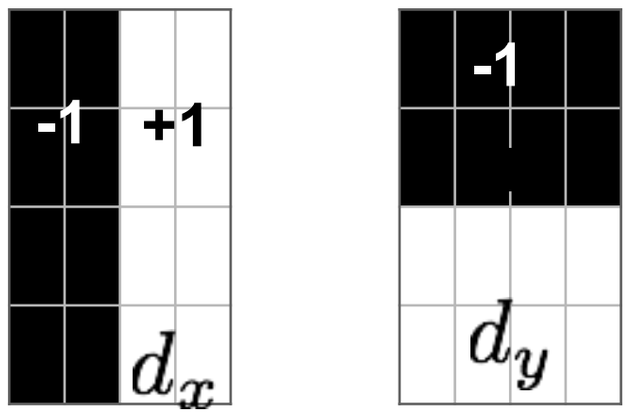
SURF

('Speeded' Up Robust Features)

Compute Haar wavelet response at each pixel in patch



Haar wavelets filters



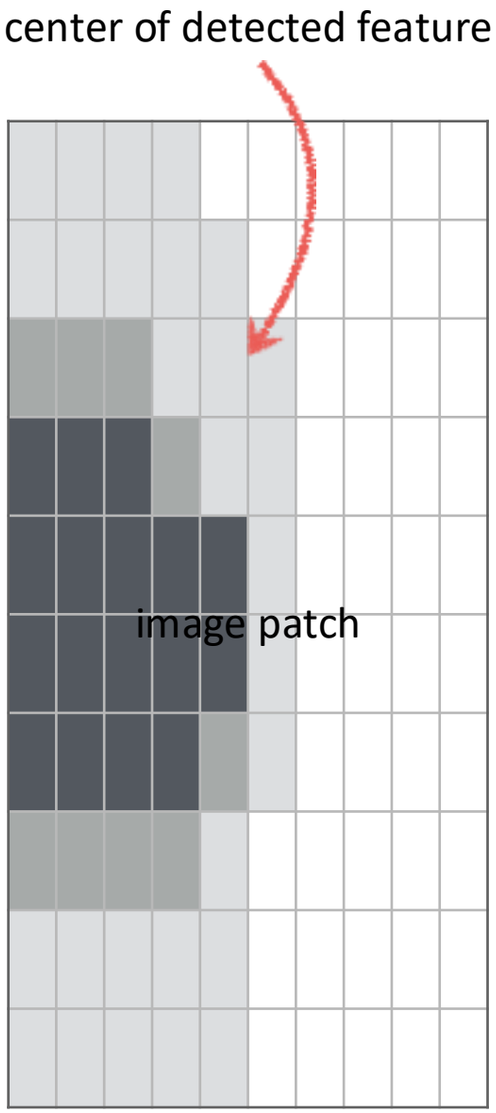
(Gaussian weighted from center)

How would do you compute the filter response?

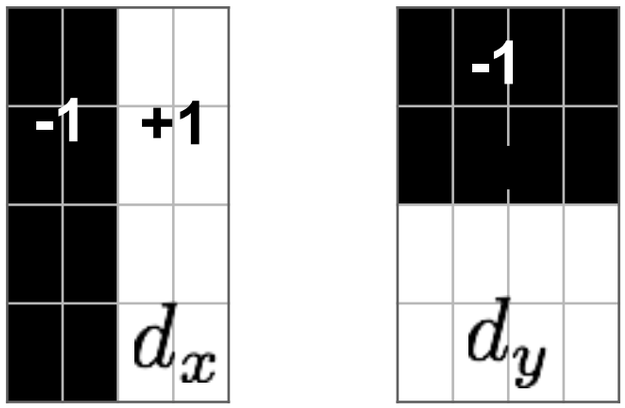
SURF

('Speeded' Up Robust Features)

Compute Haar wavelet response at each pixel in patch



Haar wavelets filters



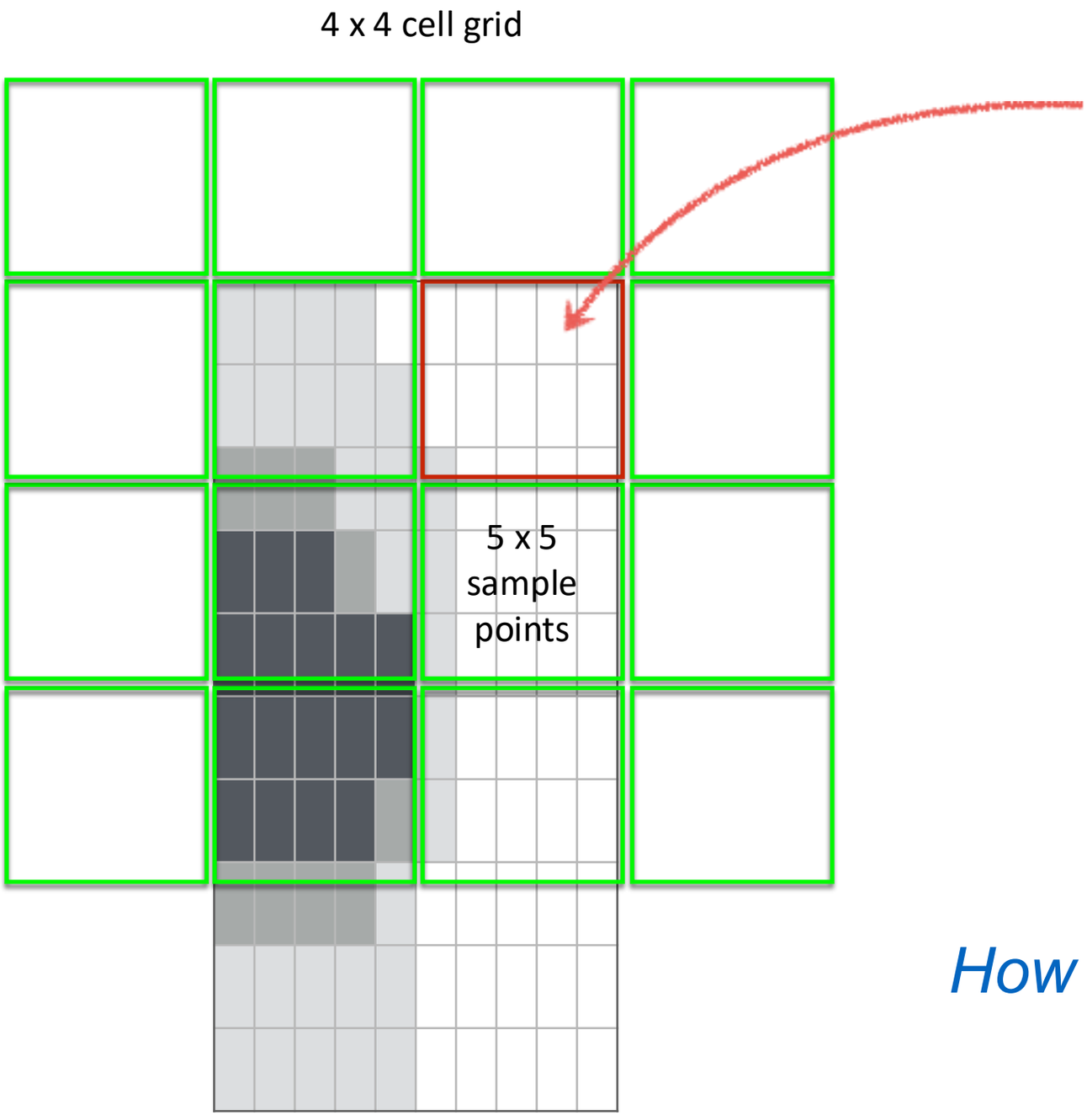
(Gaussian weighted from center)

How would do you compute the filter response?

Filtering using a sliding window can be slow
Haar wavelets are just sums over blocks
Use integral images for efficiency (6 operations)

SURF

(‘Speeded’ Up Robust Features)

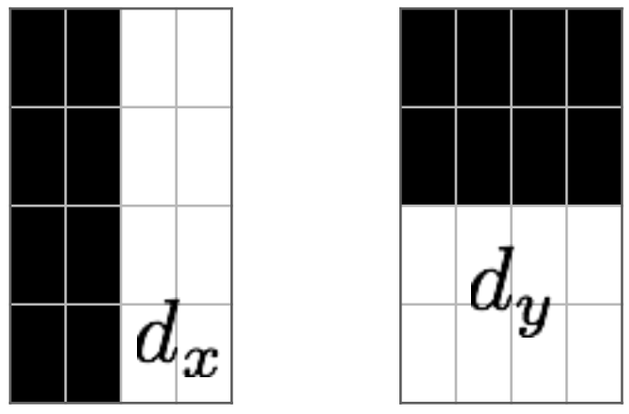


Each cell is represented by 4 values:

$$\left[\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right]$$

Haar wavelets filters

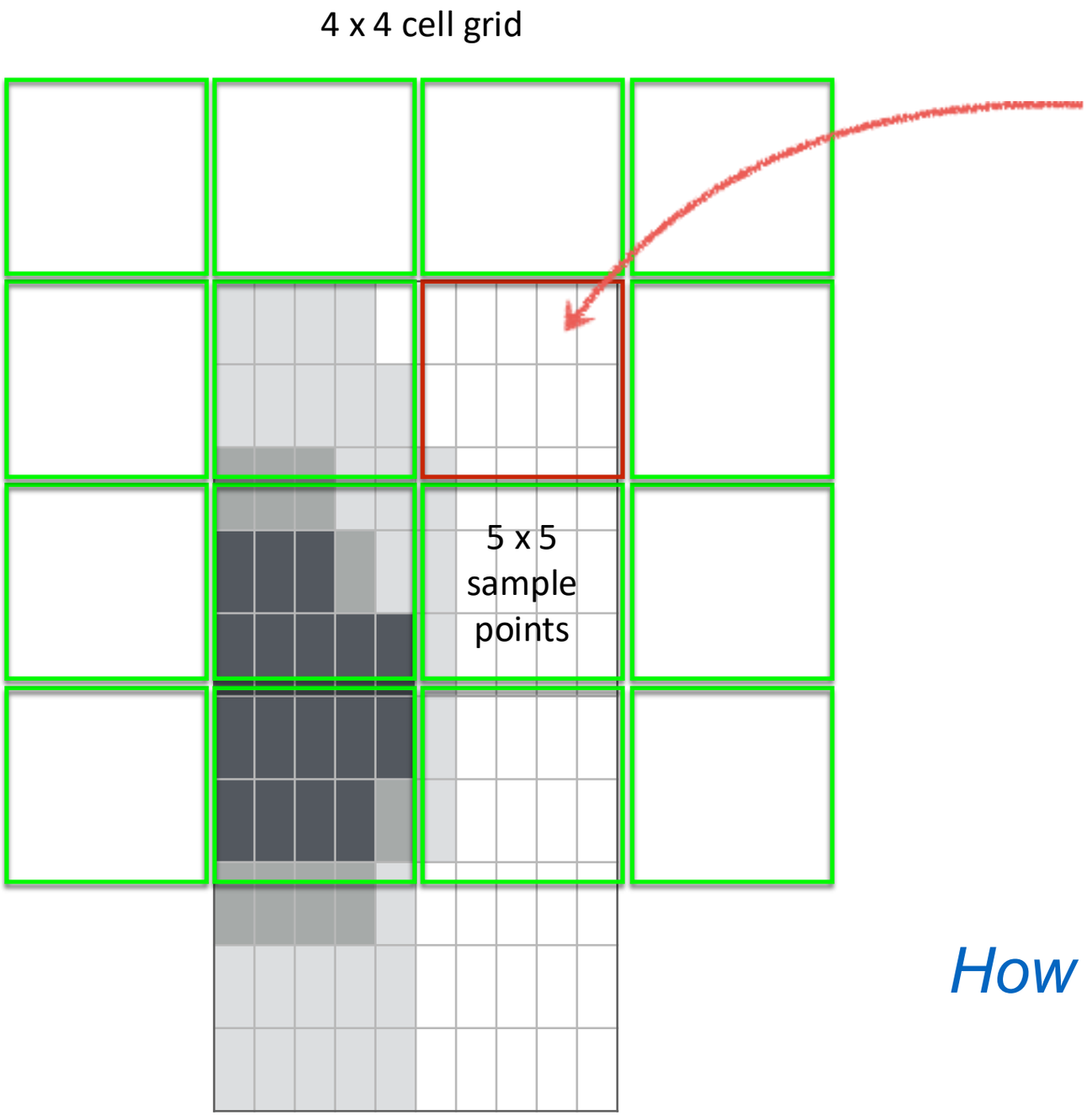
(Gaussian weighted from center)



How big is the SURF descriptor?

SURF

(‘Speeded’ Up Robust Features)

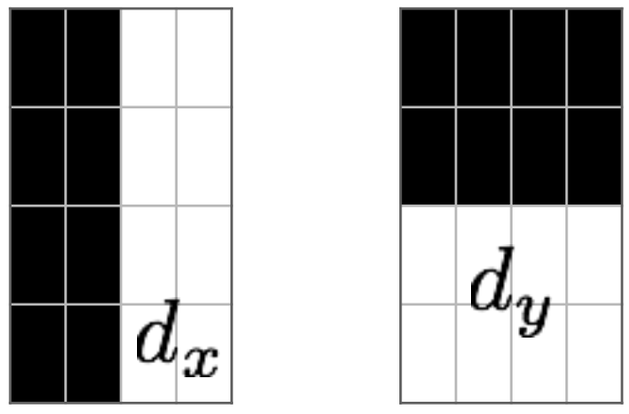


Each cell is represented by 4 values:

$$\left[\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right]$$

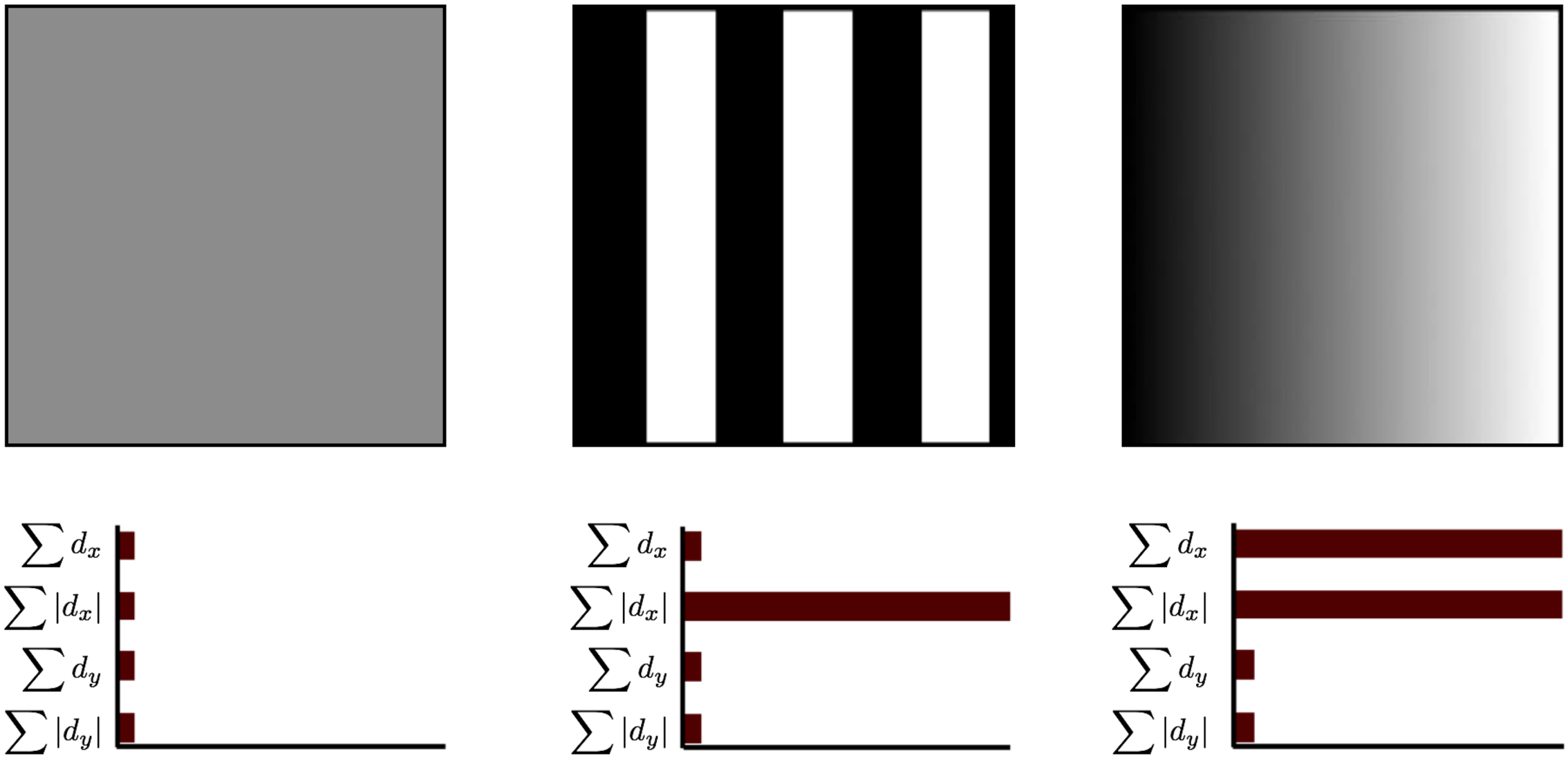
Haar wavelets filters

(Gaussian weighted from center)

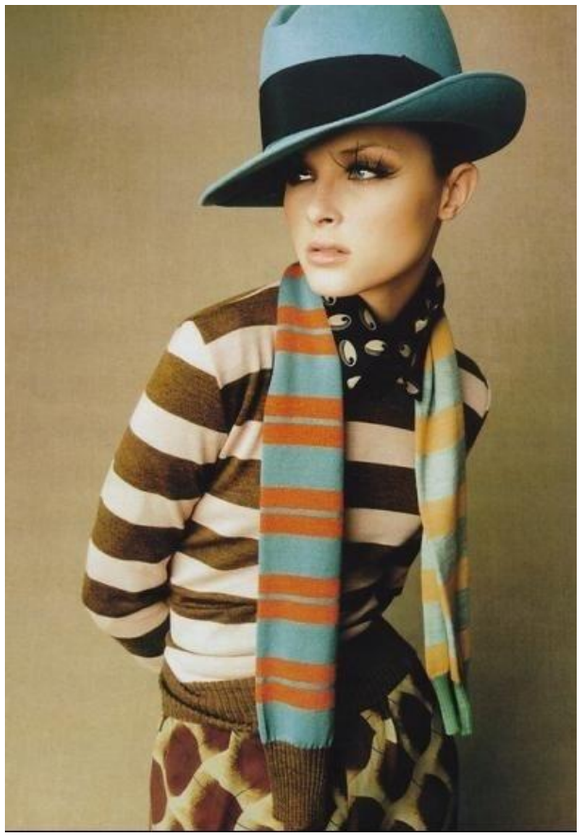


How big is the SURF descriptor?

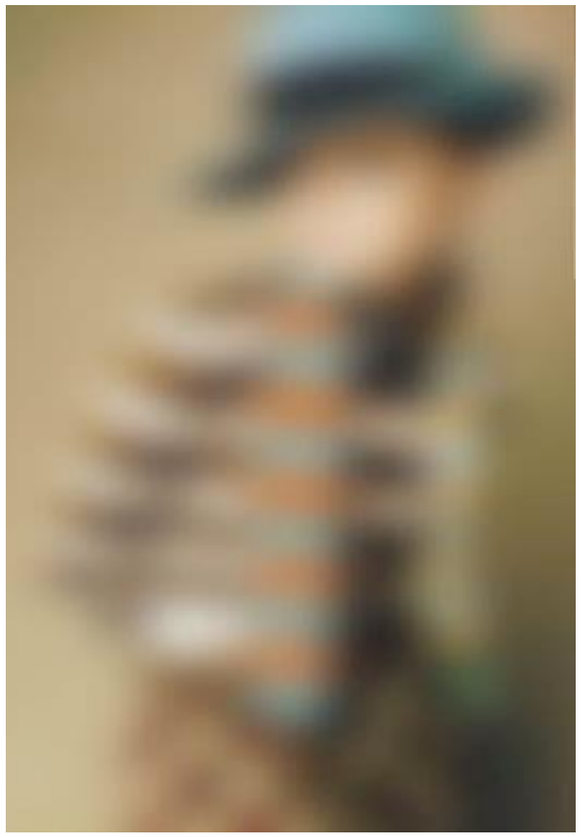
64 dimensions


$\sum d_x$
$\sum |d_x|$
$\sum d_y$
$\sum |d_y|$

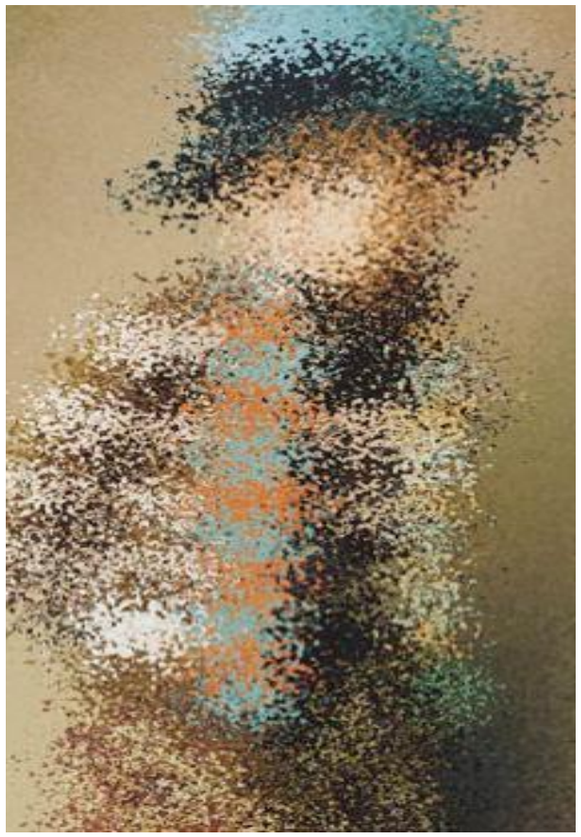
Discriminative power



Raw pixels



Sampled



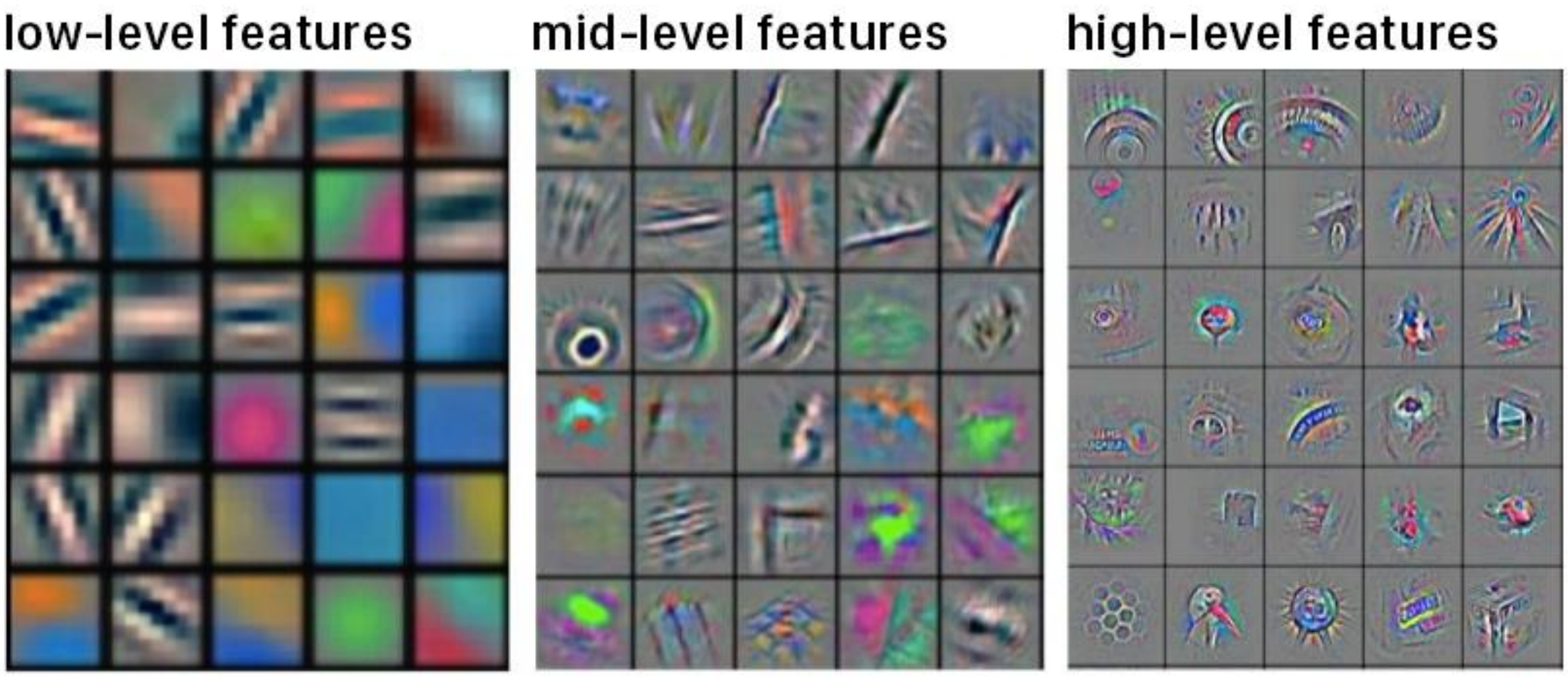
Locally orderless

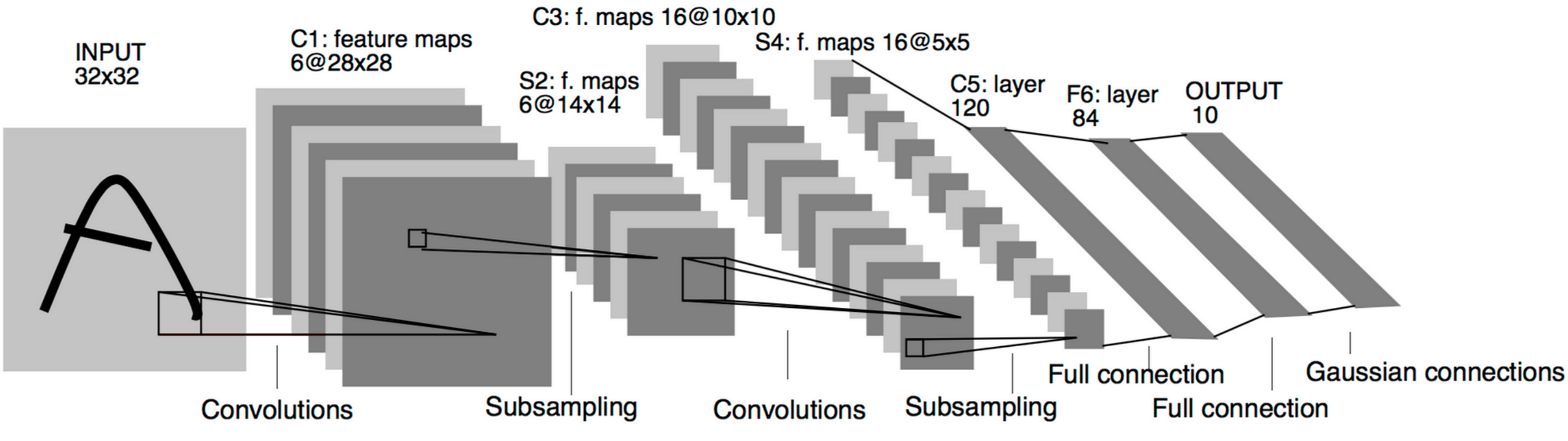
Global histogram

Generalization power

What do you think of handcrafted feature descriptors?

Deep learning: learning the features, instead of handcrafting





References

Basic reading:

- Szeliski textbook, Sections 4.1.2, 14.1.2.