Handcrafted Feature Detection and Extraction

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE GRANADA

DaSCI

Instituto Andaluz de Investigación en **Data Science** and **Computational Intelligence**

Readings

- ***Feature Detection***
  - Szeliski (2022), Chapter 7.1
  - Forsyth and Ponce (2012), Chapter 5.3
  - González and Woods (2018), Chapter 11
- ***Bag-of-Words***
  - Szeliski (2022), Chapter 6.2.1
  - Forsyth and Ponce (2012), Chapter 16.1.3, 16.1.4, 21.4

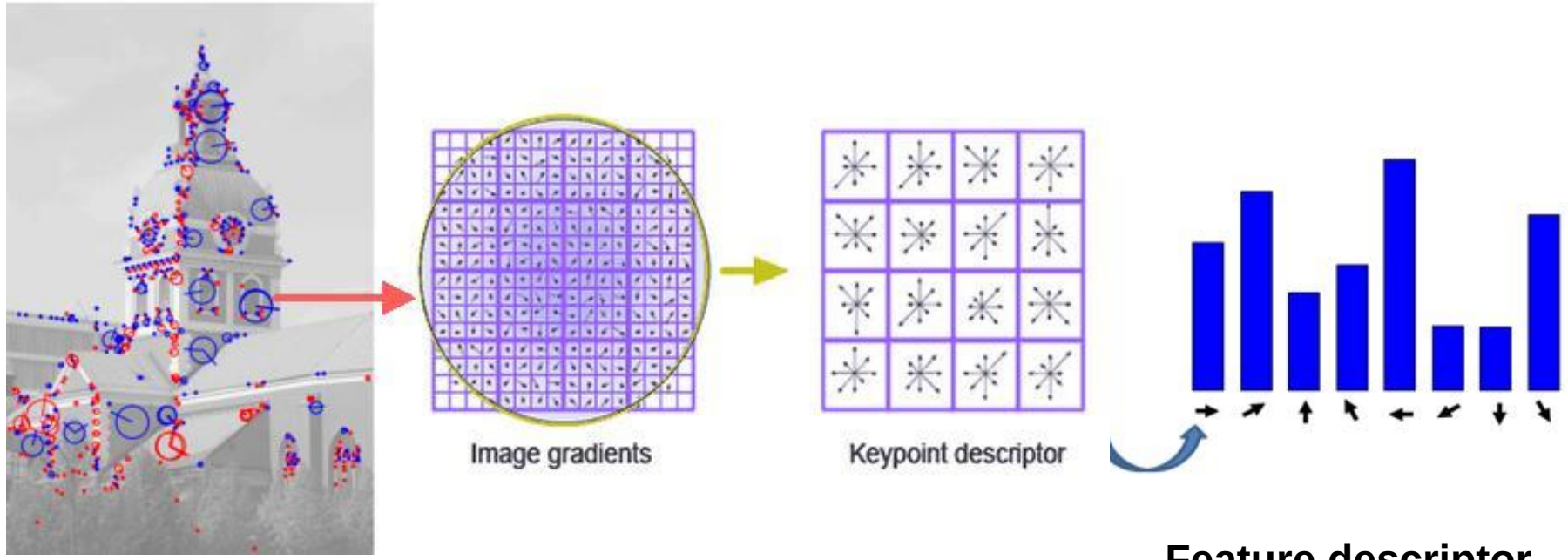
Keypoint Detection and Description

Interest Point Detection

- We want to **summarize the image** in some way.
  - Find **regions/keypoints that are distinctive**, and describe the image with a (generally) low dimensional representation.
  - Applications:
    - Motion tracking
    - Image recognition
    - 3D reconstruction
    - Image alignment (registration and stitching)
    - Image retrieval
    - Robot navigation
    - …

Interest Point Detection

Feature = keypoint + descriptor



keypoints

Feature descriptor

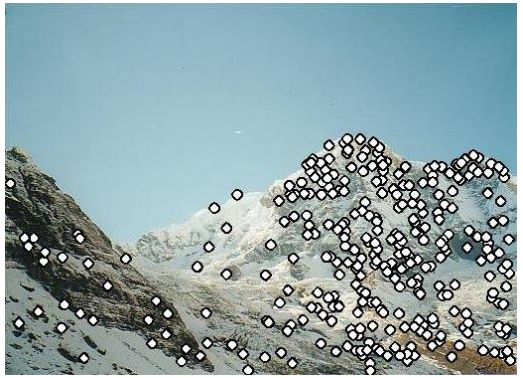
https://gilscvblog.com/2013/08/18/a-short-introduction-to-descriptors/
https://www.codeproject.com/Articles/619039/Bag-of-Features-Descriptor-on-SIFT-Features-with-O

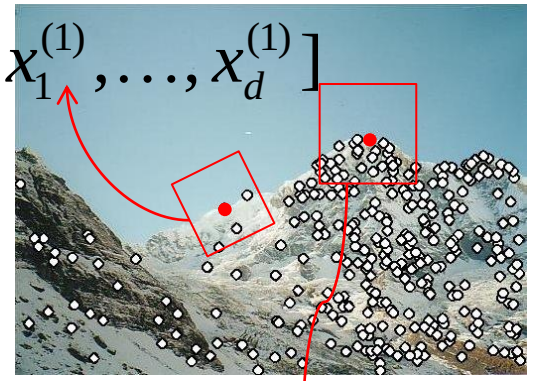
Interest Point Detection

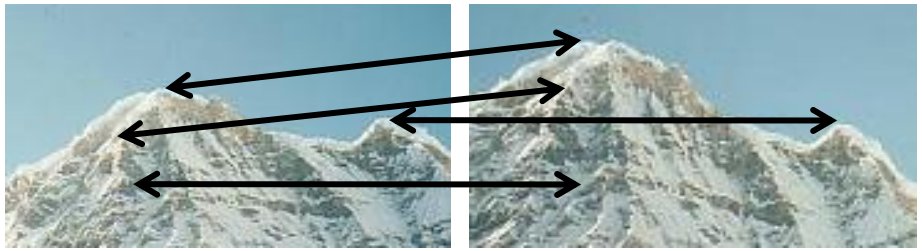
1) **Detection**: Identify the interest points

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

3) **Matching**: Determine correspondence between descriptors in two views



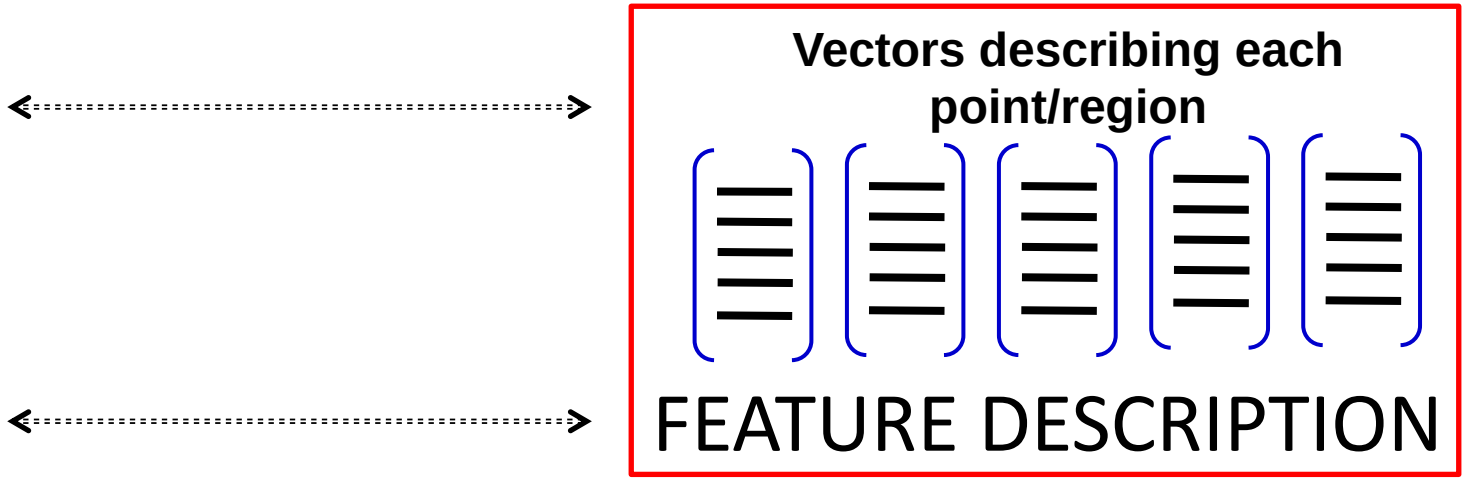




Slide: Kristen Grauman

Interest Point Detection

FEATURE DETECTION



Edge detection (Canny, Sobel,…)

Corner detection (Harris, FAST,…)

Blob detection (LoG, DoG,…)

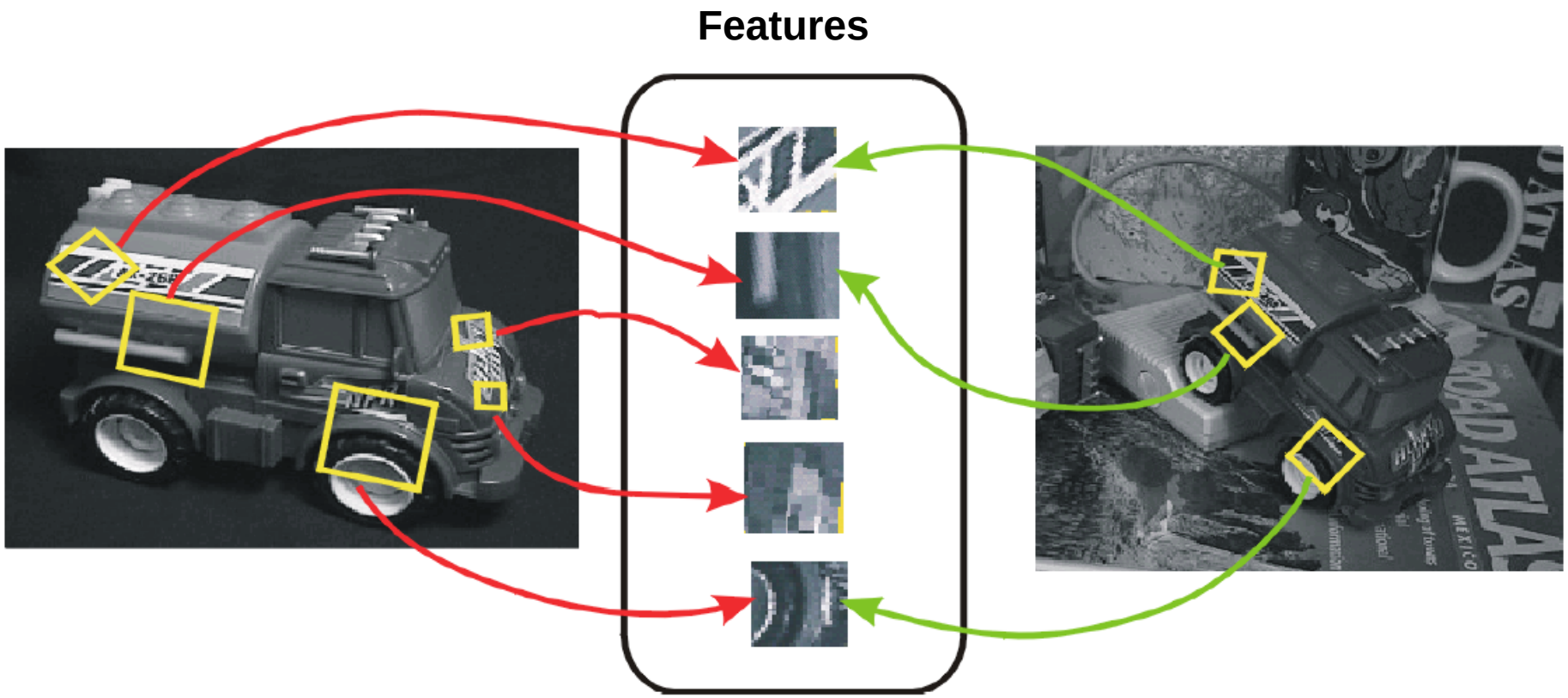
For example, SIFT is both a detector and a descriptor and, for the keypoint detection part, it uses the Difference of Gaussians on rescaled images

SIFT, SURF, HOG, KAZE,…

GLCM, LBP,…

More information about *Feature Detection* using *OpenCV*:
https://docs.opencv.org/4.4.0/db/d27/tutorial_py_table_of_contents_feature2d.html

From keypoint detection to feature description



Detection is *covariant*:

features(transform(image)) = transform(features(image))

Description is *invariant*:

features(transform(image)) = features(image)

Panoramas (*Image Stitching*)

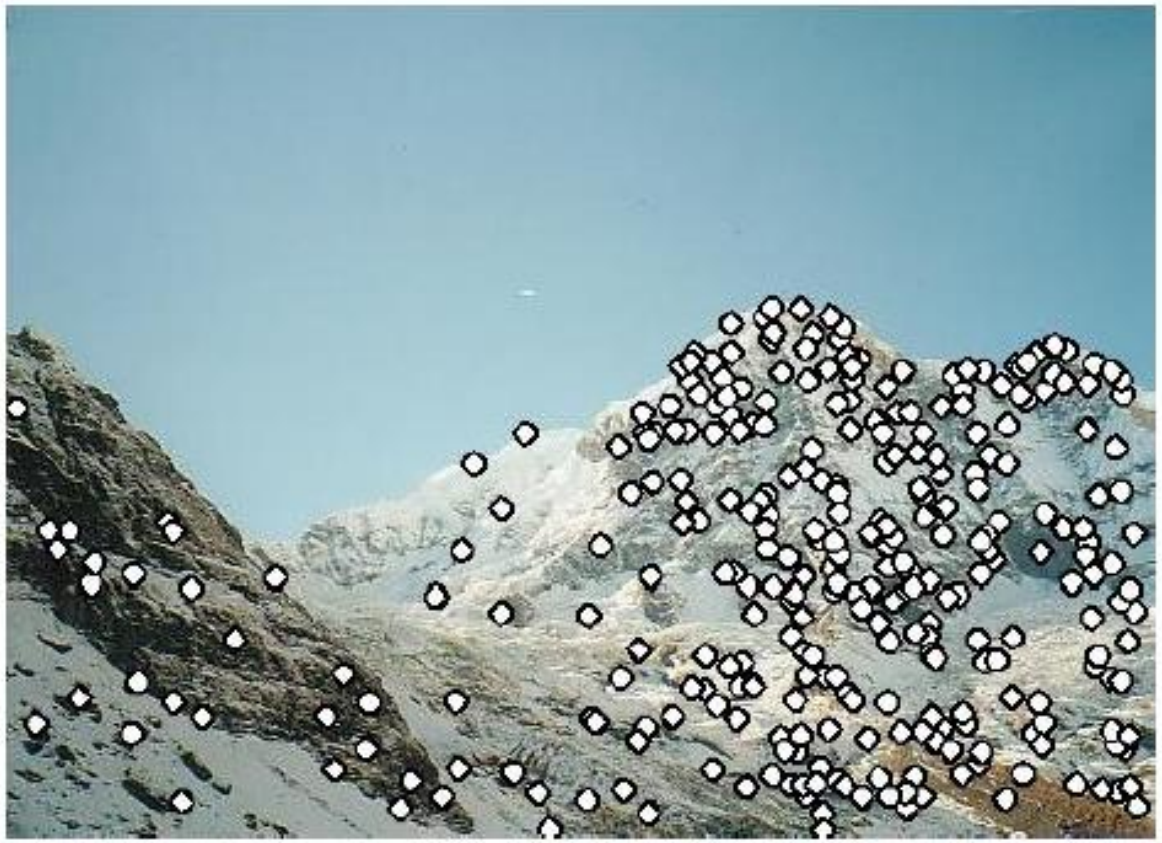
- Process of combining multiple partially overlapping images to produce a high-resolution panorama.

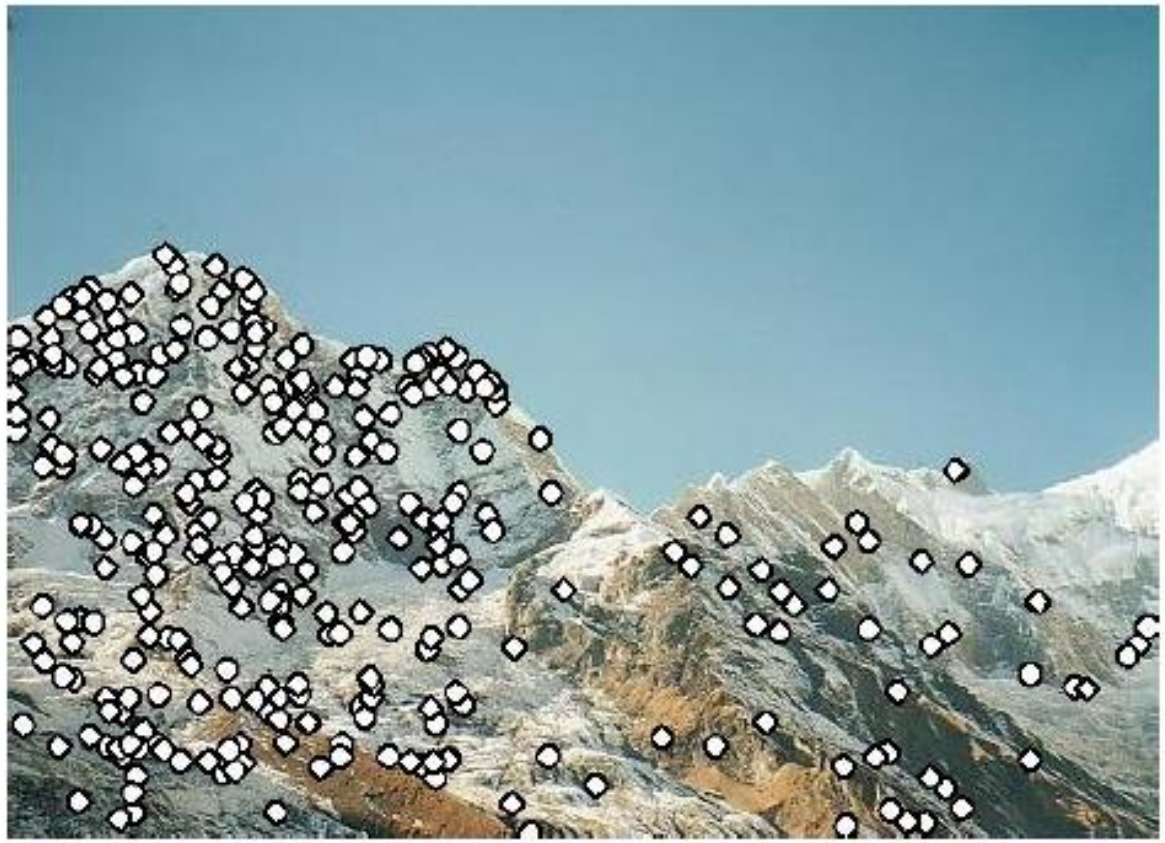
How do we build a panorama?

- We need to align images

How do we build a panorama?

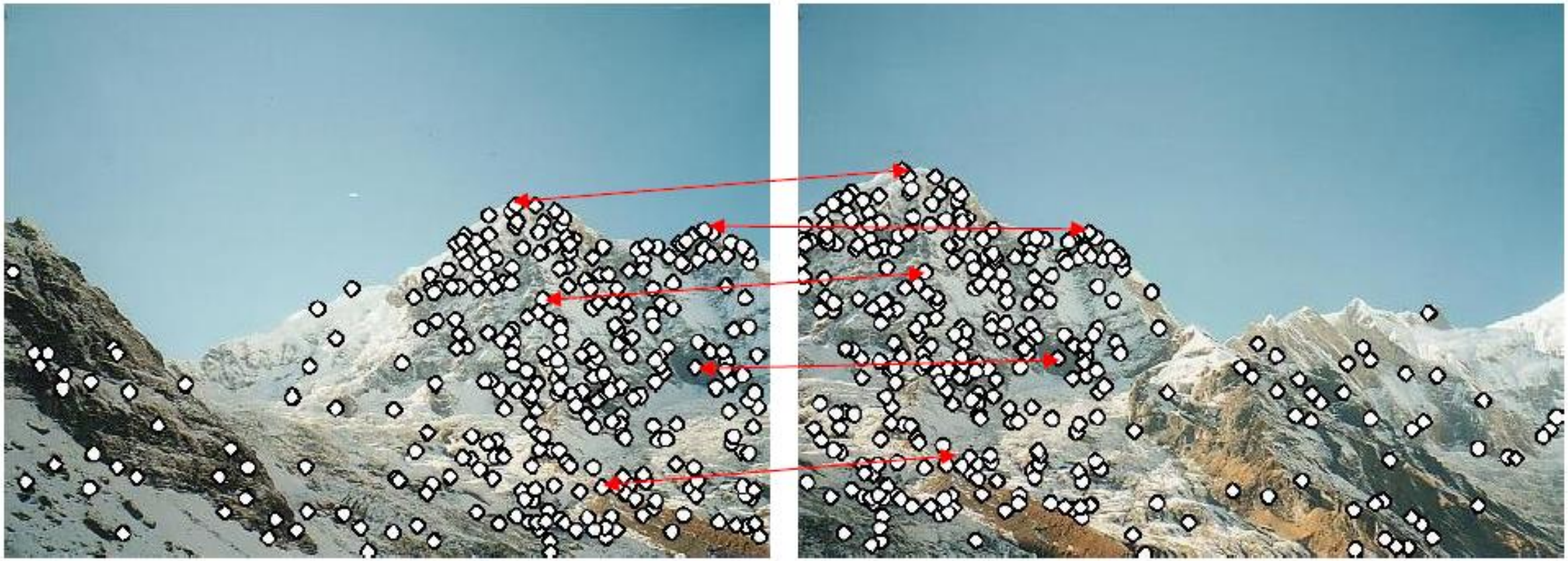
- First, we detect a bunch of keypoints on each image





How do we build a panorama?

- We need to find corresponding pairs



How do we build a panorama?

- We use those pairs to align the images

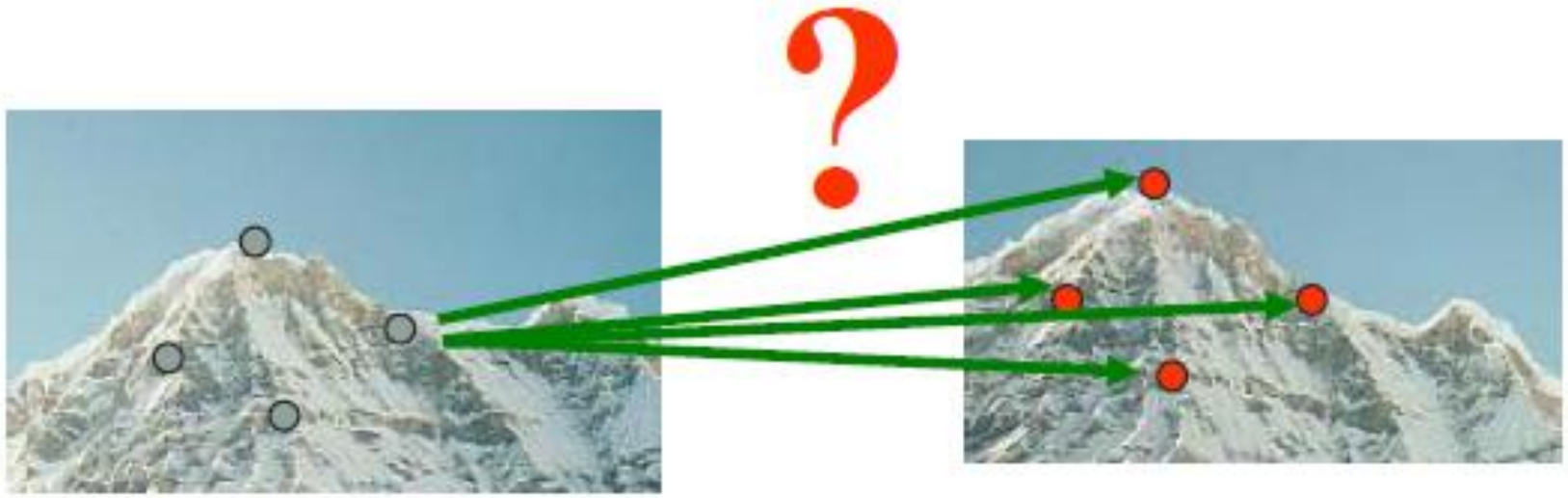
Problem 1

- We need to detect the same keypoint in both images
  - Repeatable/stable **detector**

no chance to match!

Problem 2

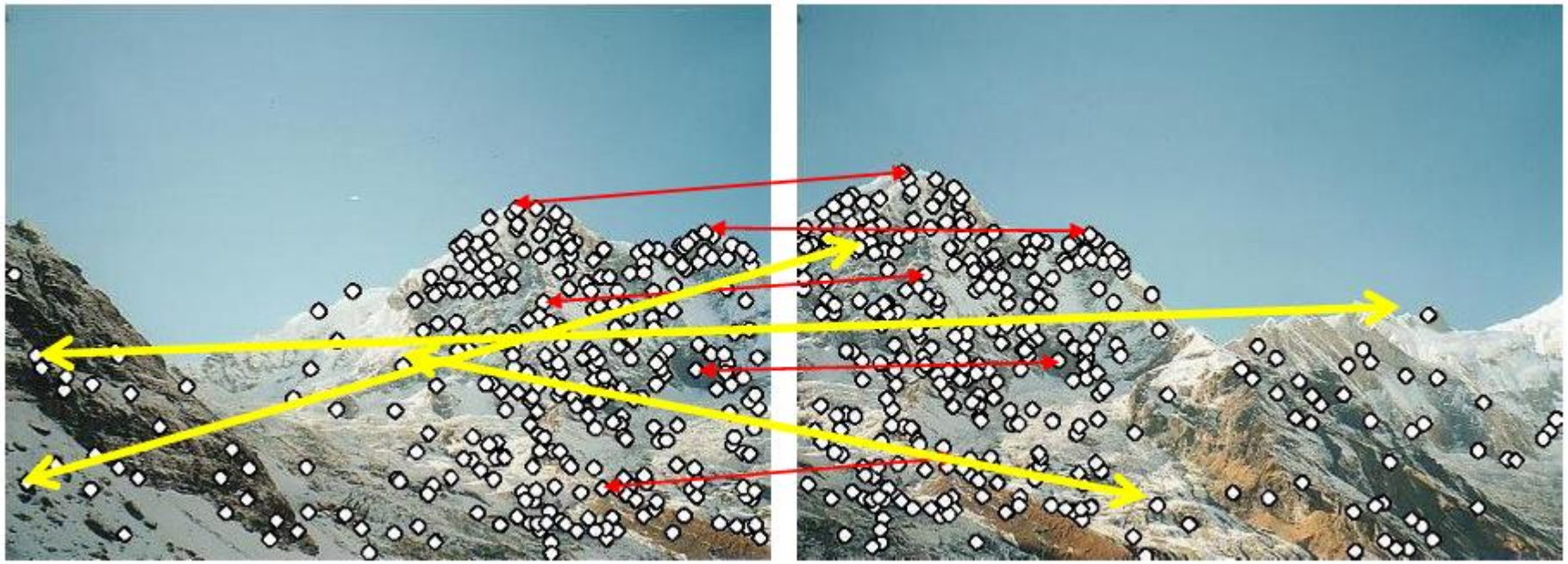
- For each point, we need to find its match in the other image
  - Reliable and distinctive **descriptor**



  - Must provide some **invariance** to **geometric** and **photometric** differences between the two views.

Problem 3

- Need to estimate transformation between images, despite erroneous correspondences

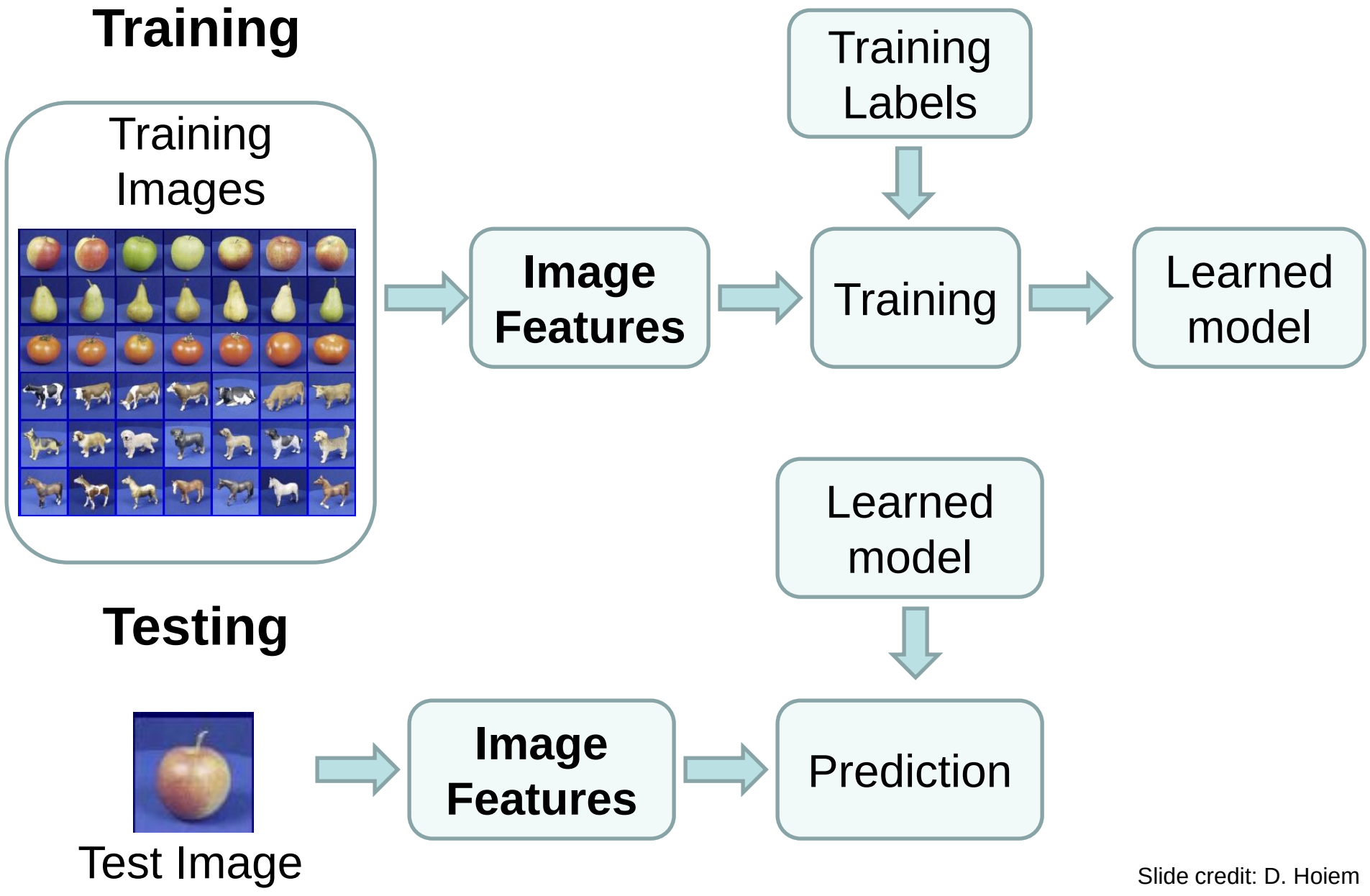


What is an interest point? Or, in other words, what is a good image feature?

- **Saliency**:
  - It has rich image content within the local window (**expressive texture**). E.g. points at which the direction of the object boundary changes abruptly (**corners**)
- It has a **well-defined representation** (signature or description) for matching/comparing with other points
- **Repeatability**
  - The same feature can be found in several images despite geometric and photometric transformations
- **Compactness and efficiency**
  - Many fewer features than image pixels
- **Locality**
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Slide: Rob Fergus

Handcrafted feature extraction for image classification

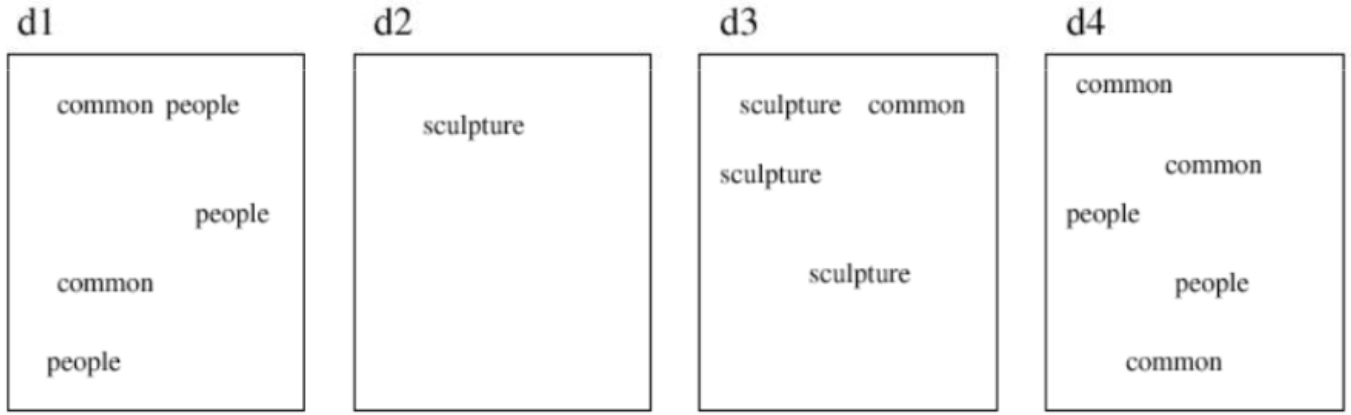


Slide credit: D. Hoiem

Bag-of-words (BoW) model in computer vision

Bag-of-Words Representation

- Orderless document representation:
  - **frequencies of words from a dictionary**
- Text classification to **determine document categories**



Bag-of-words

Common	2	0	1	3
People	3	0	0	2
Sculpture	0	1	3	0
…	…	…	…	…

Note: many of these slides are based on those by Cordelia Schmid and Kris Kitani

Bag-of-Words Representation

- Orderless document representation:
  - **frequencies of words from a dictionary**
- Text classification to **determine document categories**

US Presidential Speeches Tag Cloud
http://chir.ag/phernalia/preztags/

http://vision.stanford.edu/teaching/cs131_fall1718/files/14_BoW_bayes.pdf

Bag-of-Words Representation

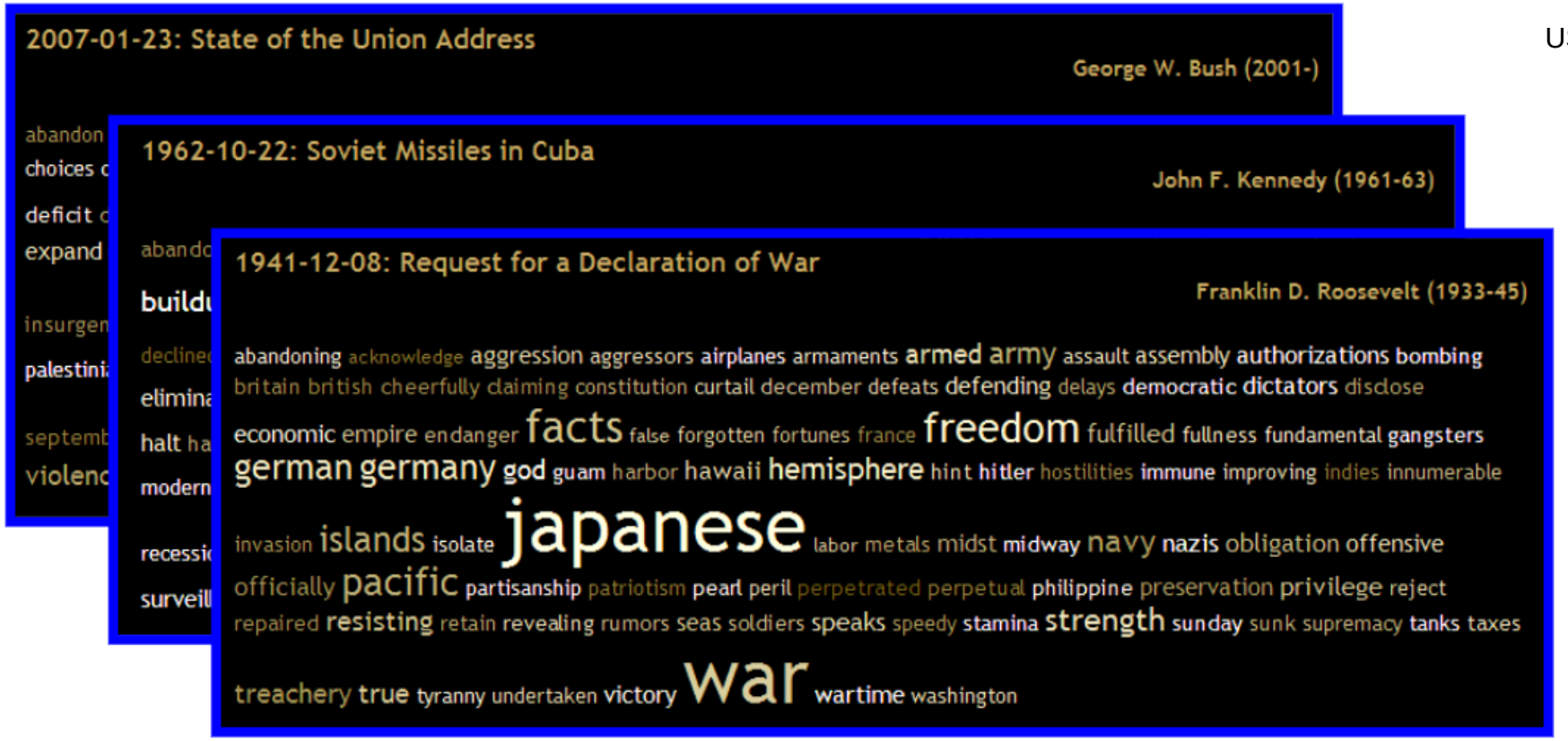
- Orderless document representation:
  - **frequencies of words from a dictionary**
- Text classification to **determine document categories**

US Presidential Speeches Tag Cloud
http://chir.ag/phernalia/preztags/

http://vision.stanford.edu/teaching/cs131_fall1718/files/14_BoW_bayes.pdf

Bag-of-Words Representation

- Orderless document representation:
  - **frequencies of words from a dictionary**
- Text classification to **determine document categories**



US Presidential Speeches Tag Cloud
http://chir.ag/phernalia/preztags/

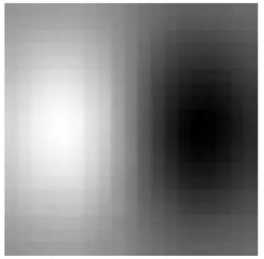
http://vision.stanford.edu/teaching/cs131_fall1718/files/14_BoW_bayes.pdf

Bag-of-Words Representation

- Take a large **corpus of text**:
  - Represent every **letter** by a 26 dimensional vector

$$a = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 0 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 0 \end{bmatrix}$$

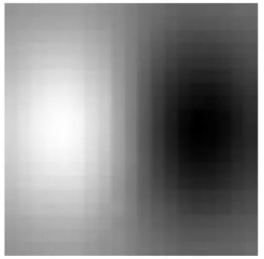
letter = filter response at image locations



Bag-of-Words Representation

- Take a large **corpus of text**:
  - Represent every **letter** by a 26 dimensional vector
  - Represent each **word** by an average of letter representations in it

letter = filter response at image locations



word = patch summary

	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
Win.#9	20	20

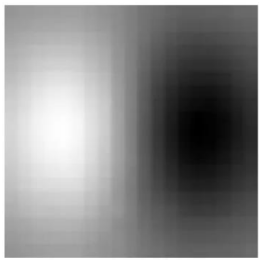
words

$$ab = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 0 \end{bmatrix}$$

Bag-of-Words Representation

- Take a large **corpus of text**:
  - Represent every **letter** by a 26 dimensional vector
  - Represent each **word** by an average of letter representations in it
  - Cluster the words, to get a **"dictionary"** of K representative words. Words that have very similar representations would get clustered together (e.g., smile and smiled)

letter = filter response at image locations

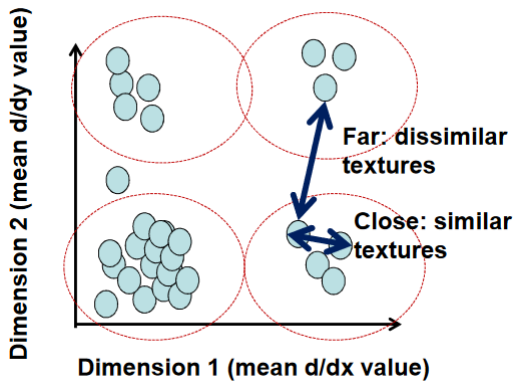


word = patch summary

	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
Win.#9	20	20

words

dictionary = textons



corpus of text = collection of images

Object → Bag of ‘words’

For instance, a face is normally composed of nose, eyes, hair, mouth, eyebrows,... So, if these elements are present in the image, regardless of where they appear, we will interpret that there is a face.

An object is seen as a collection of local features (bag of features)

It makes sense!

- Collection of features/parts reveal the underlying object.

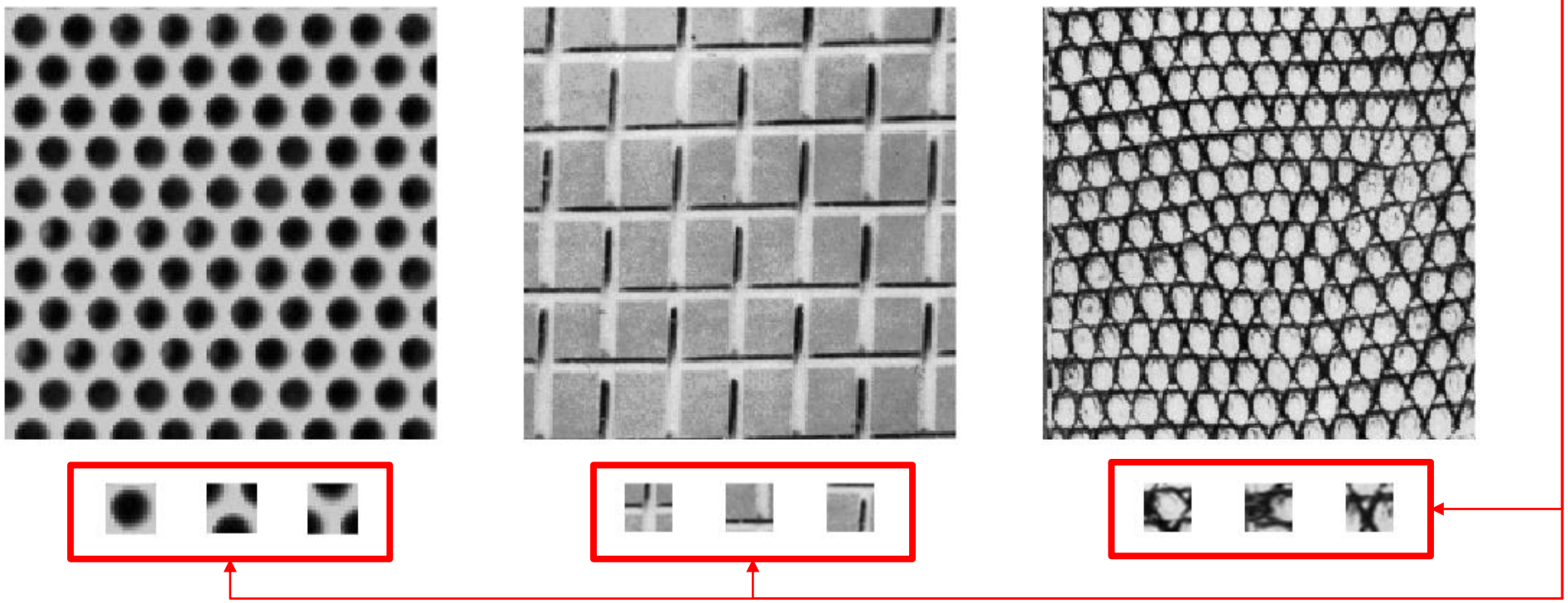






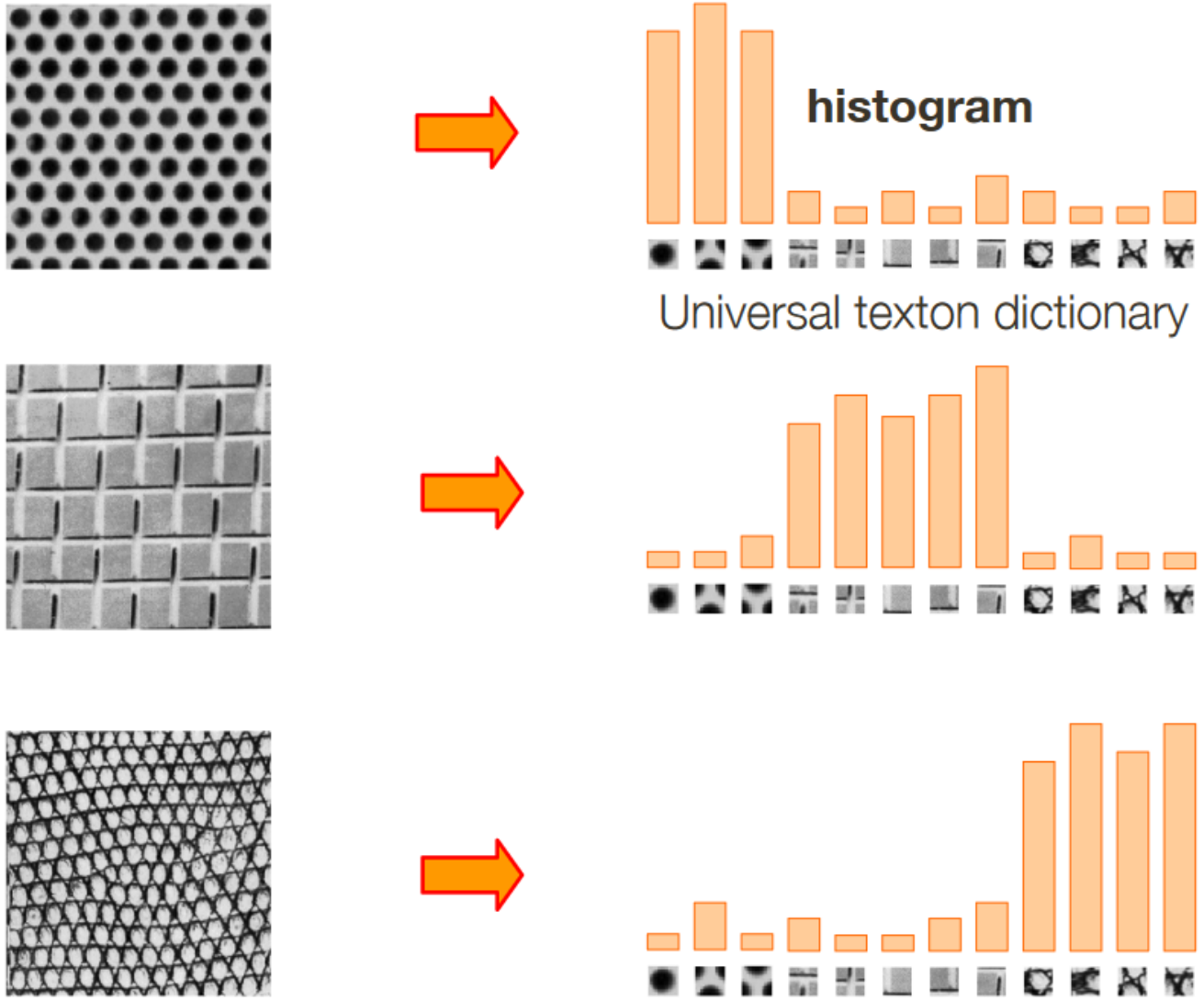
Bag-of-Words for texture recognition

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



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

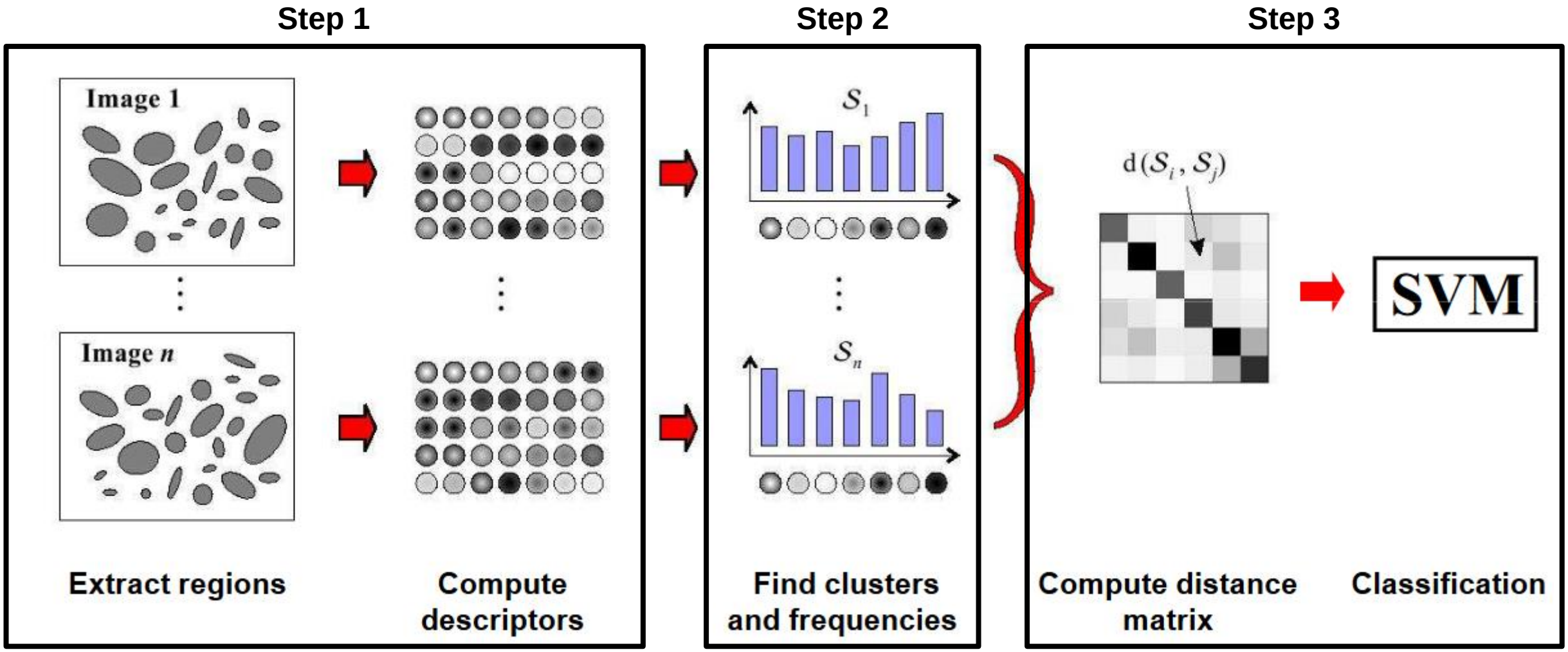
Bag-of-Words for texture recognition



If *"it is generally more important to know that something is present […] than to know where it is. This suggests using a representation that has the form of a histogram"* (Forsyth and Ponce, p.487)

We represent texture images as histograms over a discrete vocabulary of local features

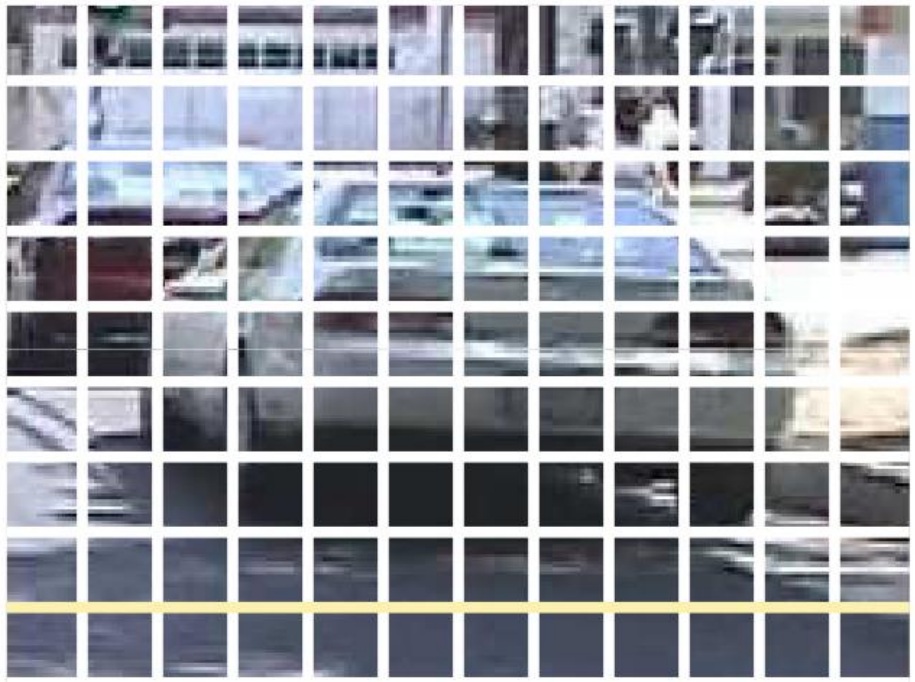
Standard BoW pipeline (for image classification)



https://www.di.ens.fr/willow/events/cvml2011/materials/CVML2011_Cordelia_bof.pdf

Standard BoW pipeline (for image classification)

- Step 1: feature extraction
  - Extract features (edges, SIFT, dense features, color-based descriptors, texture-based descriptors…) from images



Dense features:
- Extraction of small overlapping patches at multiple scales
- Computation of the SIFT descriptor for each grid cell

Dense is on average a bit better!

Interest points and dense are complementary, combination improves results.

https://www.di.ens.fr/willow/events/cvml2011/materials/CVML2011_Cordelia_bof.pdf

Standard BoW pipeline (for image classification)

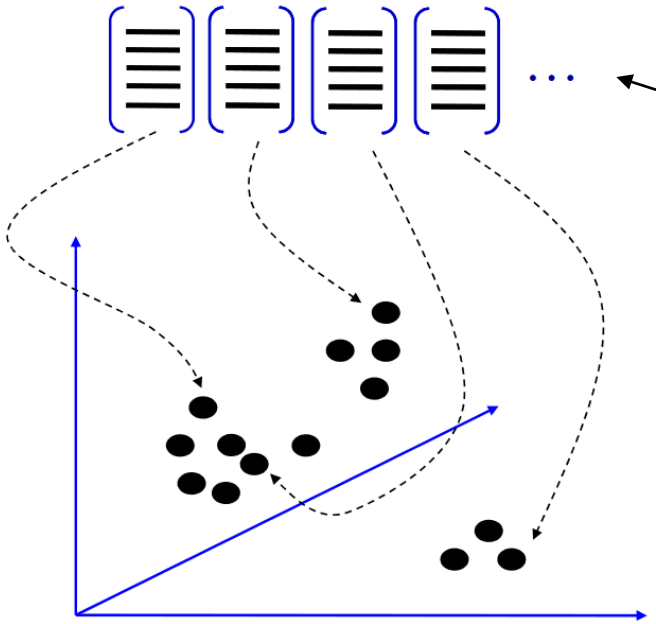
- Step 1: feature extraction
  - Extract features (edges, SIFT, dense features, color-based descriptors, texture-based descriptors…) from images

Interest points (selective sampling)
More useful in geometric problems

Dense features (uniform sampling)
More useful in visual recognition problems

Standard BoW pipeline (for image classification)

- Step 1: feature extraction
  - Extract features (edges, SIFT, dense features, color-based descriptors, texture-based descriptors…) from images
- Step 2: generate visual vocabulary (unsupervised learning process)

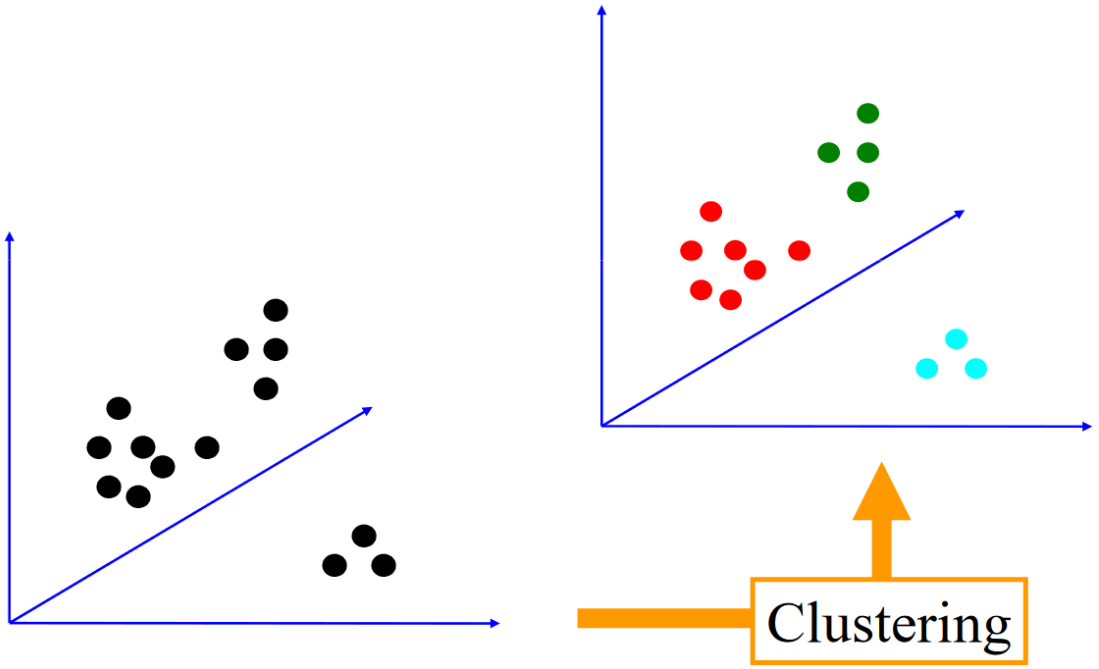


We have a descriptor (feature vector) per window/region/patch. Each image has several of these.

For instance, if you use SIFT, after feature detection and description, your image will be a collection of vectors of the same dimension (128 for SIFT).

Standard BoW pipeline (for image classification)

- Step 1: feature extraction
  - Extract features (edges, SIFT, dense features, color-based descriptors, texture-based descriptors…) from images
- Step 2: generate visual vocabulary (unsupervised learning process)

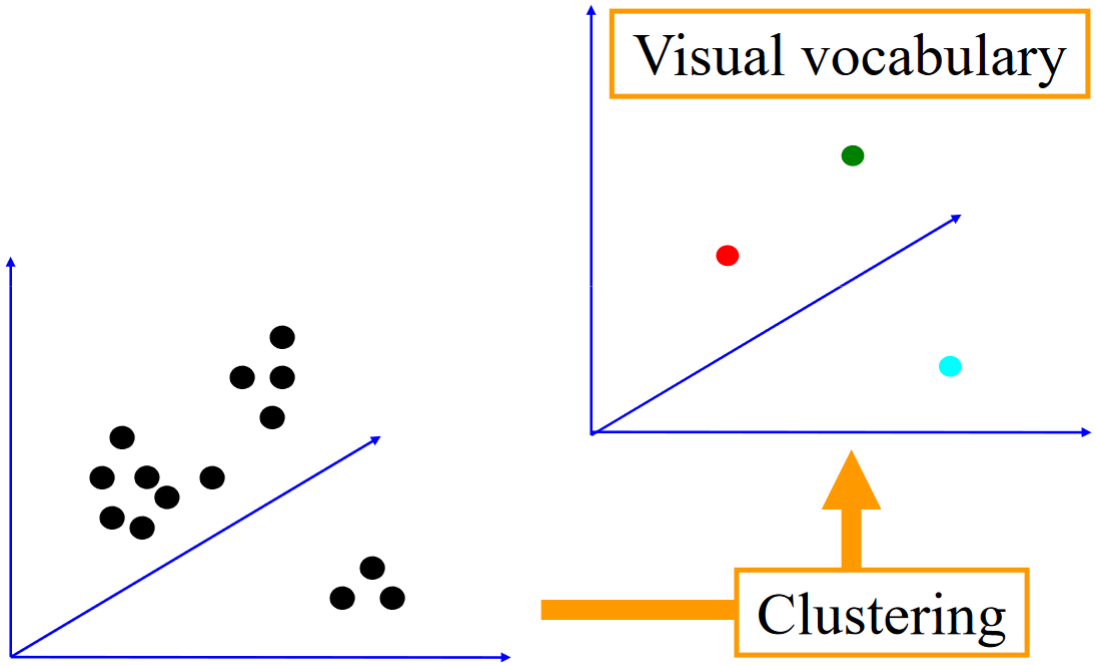


We apply a clustering algorithm (like k-means) to group similar features.

Note: nice reminder of k-means clustering at https://www.cs.cmu.edu/~16385/s17/Slides/8.2_Bag_of_Visual_Words.pdf (slides 30-35)

Standard BoW pipeline (for image classification)

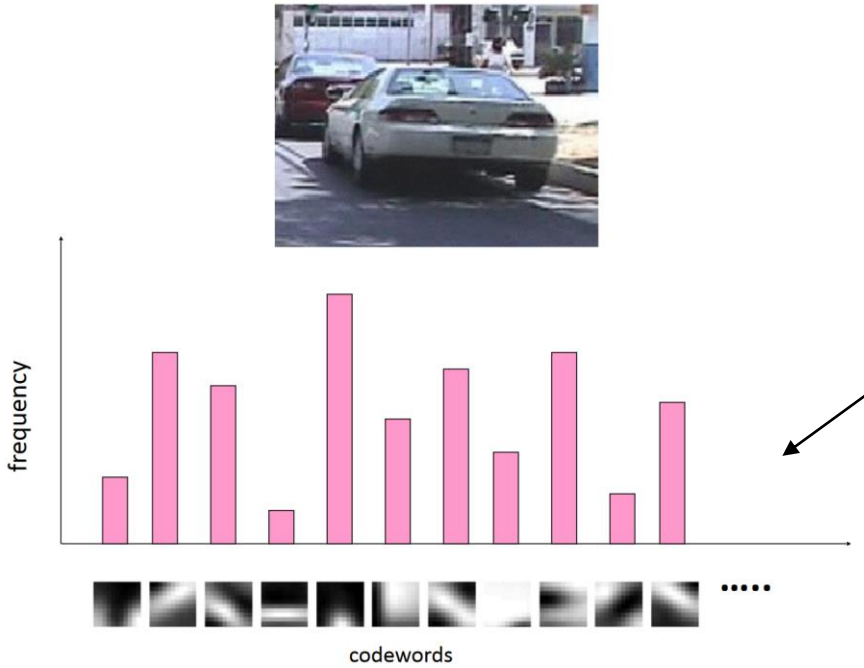
- Step 1: feature extraction
  - Extract features (edges, SIFT, dense features, color-based descriptors, texture-based descriptors…) from images
- Step 2: generate visual vocabulary (unsupervised learning process)



A codeword (cluster center) can be considered as a representative of similar patches (a prototype “word”). We learn our visual vocabulary or **codebook**. The number of clusters is the codebook size.

Standard BoW pipeline (for image classification)

- Step 1: feature extraction
  - Extract features (edges, SIFT, dense features, color-based descriptors, texture-based descriptors…) from images
- Step 2: generate visual vocabulary (unsupervised learning process)



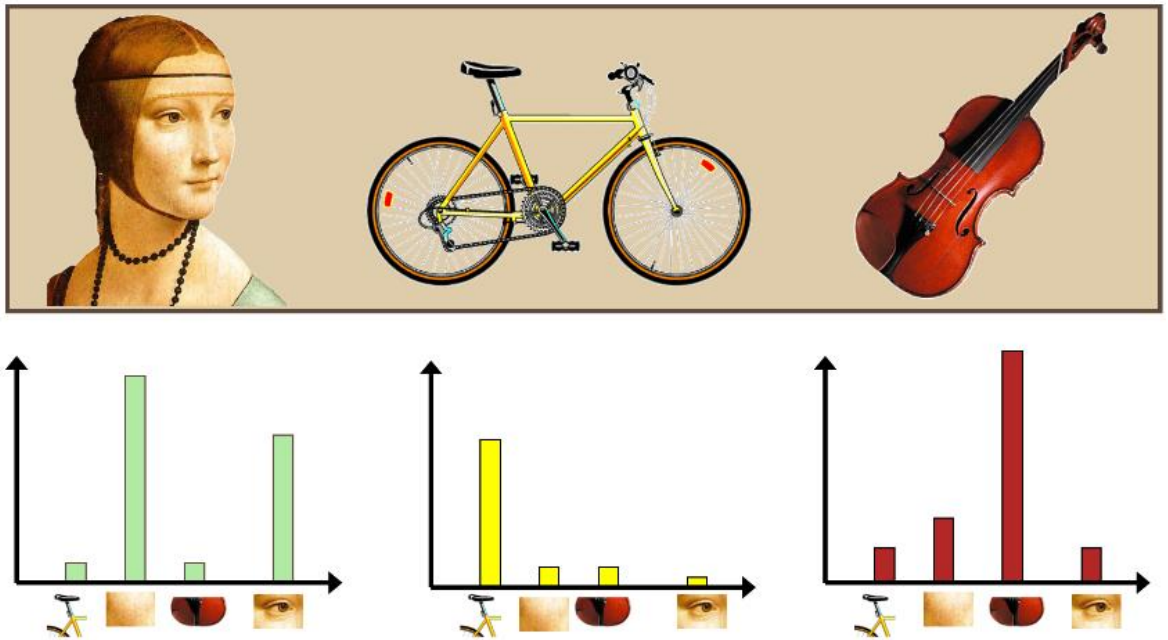
Now, each patch in an image can be mapped to the closest codeword and the image can be represented by the **histogram of the codewords**.

In this example, we show the BoW representation of a car.

We represent images by frequencies of "visual words" (but downweighting words that appear very often, i.e. a word (like "the", "and", "or") that appears in all documents is not helping us; see TF-IDF weighting)

Standard BoW pipeline (for image classification)

- Step 1: feature extraction
  - Extract features (edges, SIFT, dense features, color-based descriptors, texture-based descriptors…) from images
- Step 2: generate visual vocabulary (unsupervised learning process)



Now, each patch in an image can be mapped to the closest codeword and the image can be represented by the **histogram of the codewords**.

Source: Kristen Grauman

Standard BoW pipeline (for image classification)

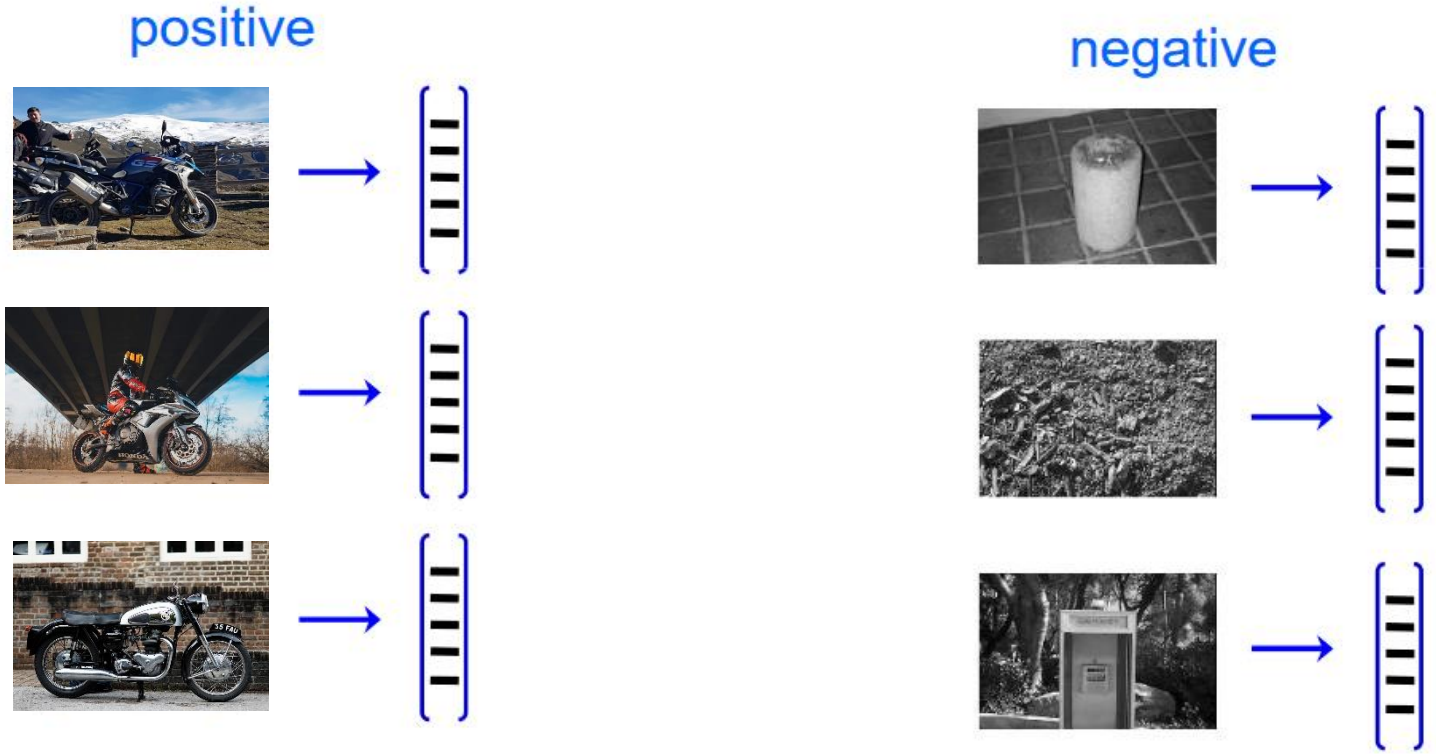
- Step 1: feature extraction
  - Extract features (edges, SIFT, dense features, color-based descriptors, texture-based descriptors…) from images
- Step 2: generate visual vocabulary (unsupervised learning process)
  - From what data should I learn the codebook?
    - Provided the training set is sufficiently representative, the codebook will be “universal”
  - How to choose vocabulary size?
    - Too small: visual words not representative of all patches
    - Too large: overfitting, higher computational cost

Standard BoW pipeline (for image classification)

- Step 1: feature extraction
  - Extract features (edges, SIFT, dense features, color-based descriptors, texture-based descriptors…) from images
- Step 2: generate visual vocabulary (unsupervised learning process)
- Step 3: classification
  - Given the bag-of-features representation of images from different classes, learn a classifier using machine learning
  - Typically, each image is represented by a 1000-5000 dimensional vector.

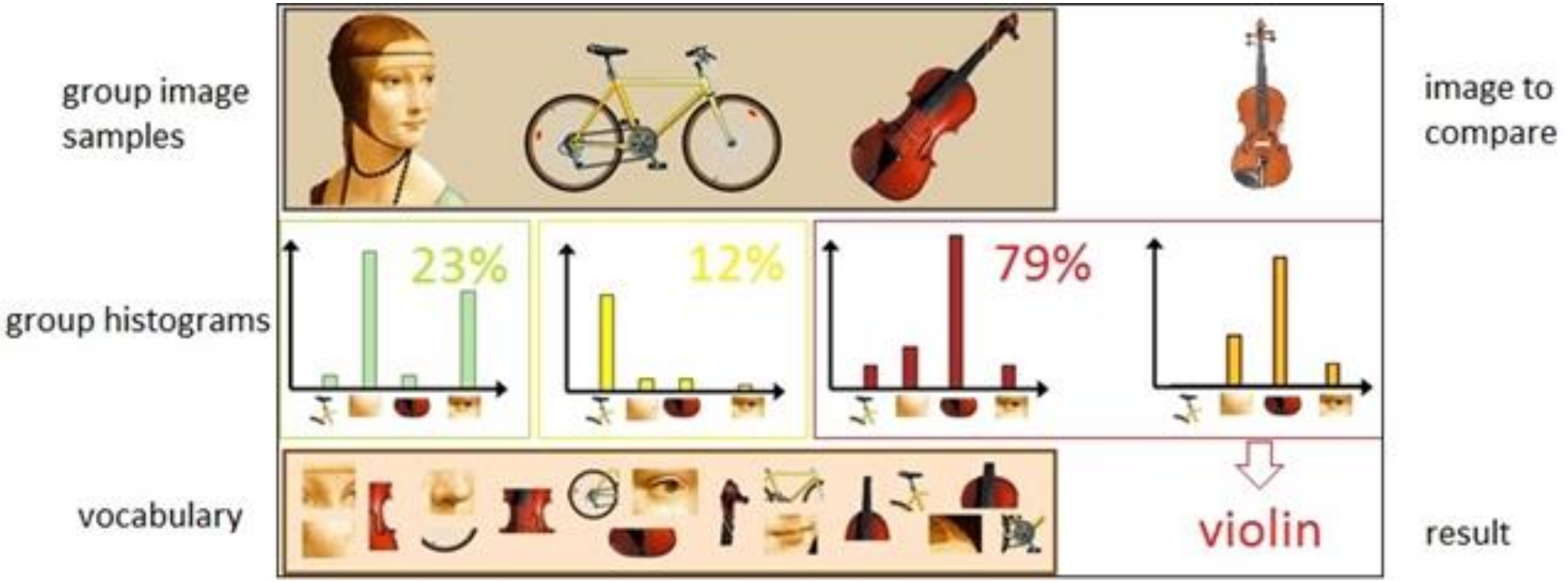
Standard BoW pipeline (for image classification)

- Step 3: classification
  - Vectors are histograms, one from each training image
  - Train classifier, e.g. SVM



Standard BoW pipeline (for image classification)

- Outline



https://vgg.fiit.stuba.sk/2015-02/bag-of-visual-words-in-opencv/

The histogram of the new input image should be more similar to the one representing the violin than to the other ones.

Commonly, we compare bags of words using cosine similarity or χ^2 distance

Standard BoW pipeline (for image classification)

- Advantages:
  - largely unaffected by position and orientation of objects in the image
  - fixed length vector irrespective of number of detections
  - successful in classifying images according to the objects they contain
- Disadvantages:
  - no explicit use of configuration of visual word positions
  - poor at localizing objects within an image

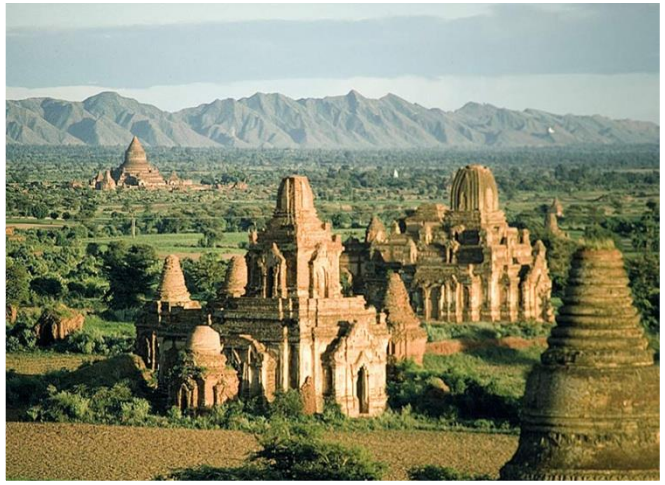
Extension to bag-of-words models

- BoW does not manage any spatial information → it does not discriminate if a patch was obtained from the top or bottom of the image

- Spatial Pyramid representation.
  - Add spatial information to the BoW representation
  - Representation at several levels of spatial resolution

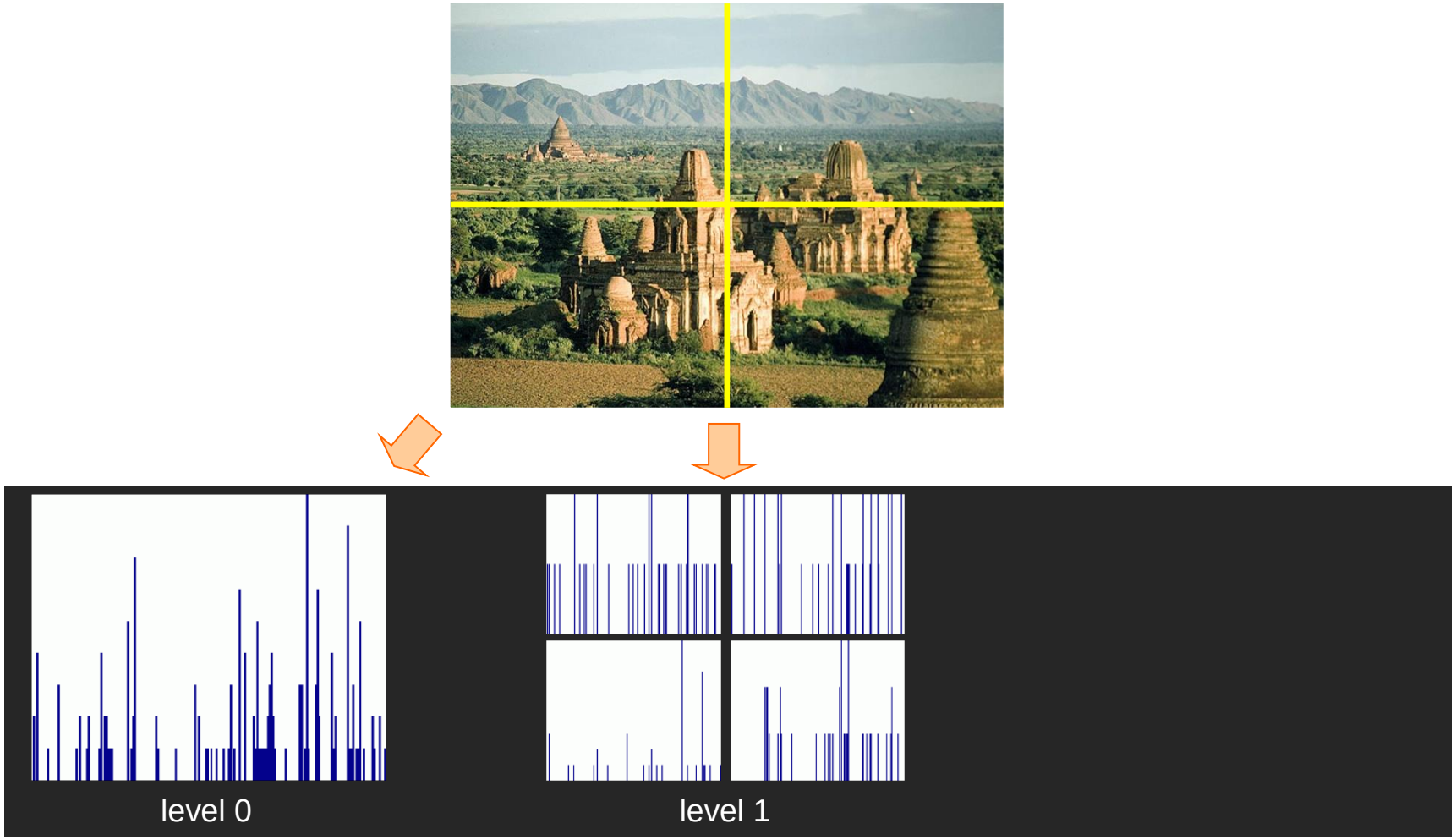
Lazebnik, S., Schmid, C., & Ponce, J. “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories”. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)* (Vol. 2, pp. 2169-2178).

Spatial pyramids



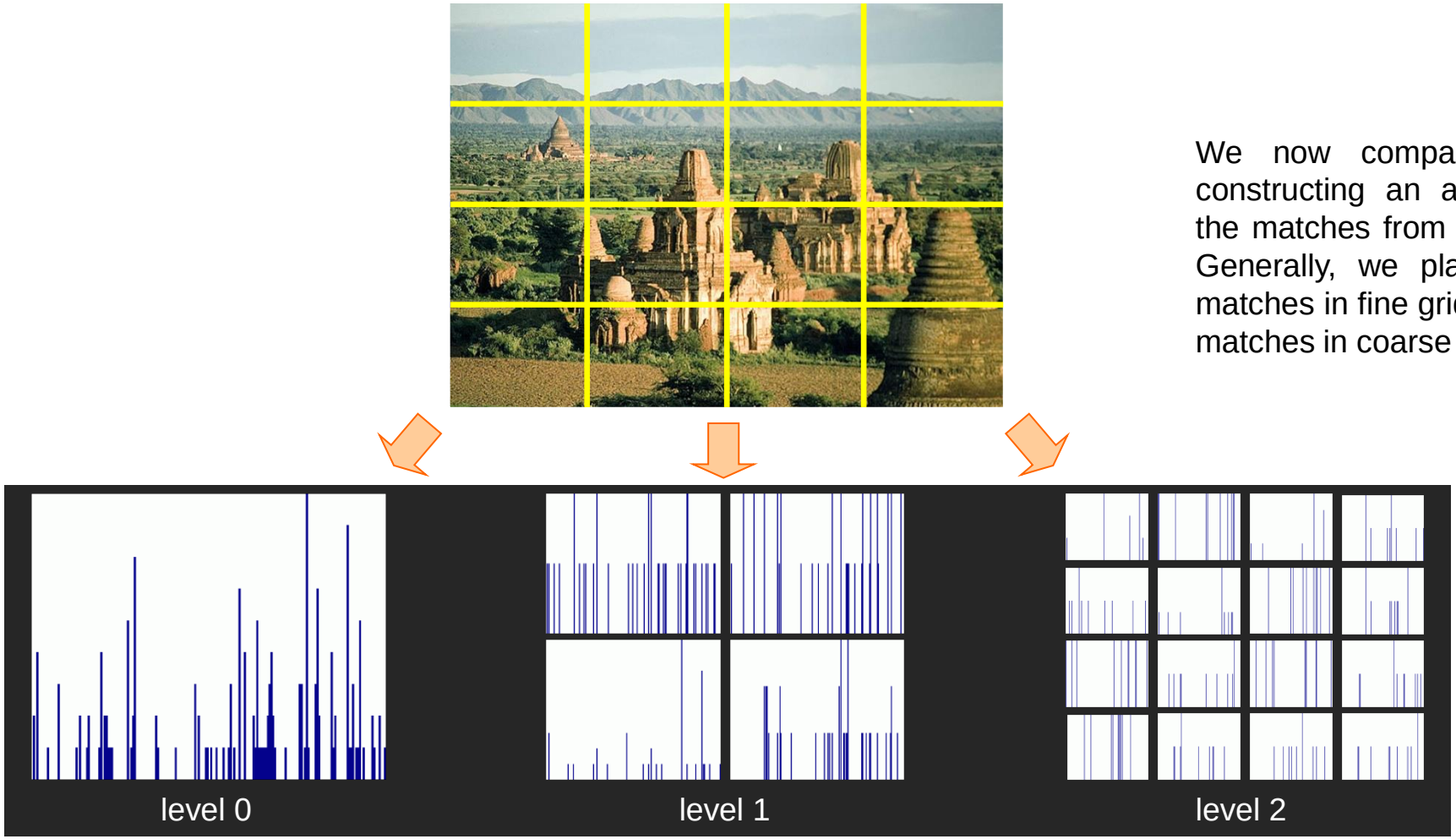
Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramids



Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramids



We now compare two images by constructing an approximate score of the matches from these 21 histograms. Generally, we place more weight on matches in fine grids and less weight on matches in coarse grids.

Lazebnik, Schmid & Ponce (CVPR 2006)

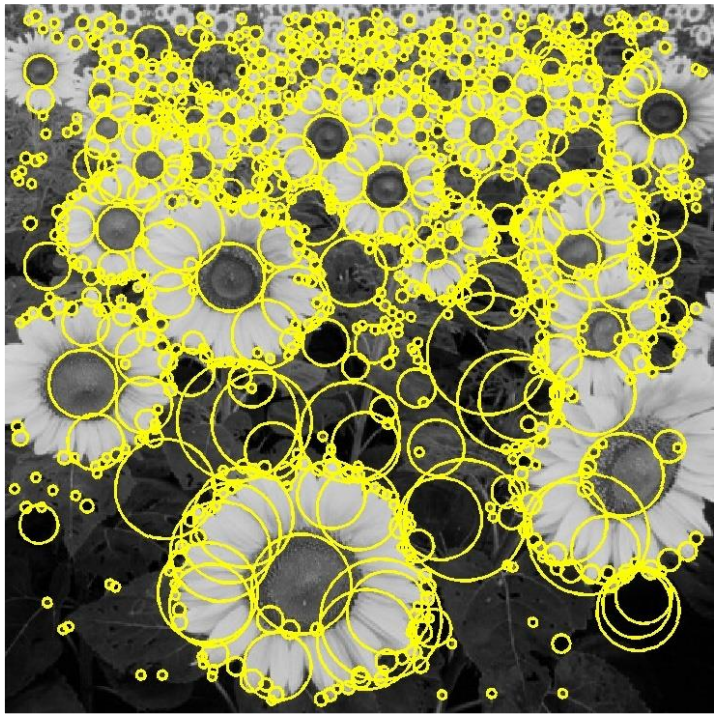
Blob Detection and Automatic Scale Selection

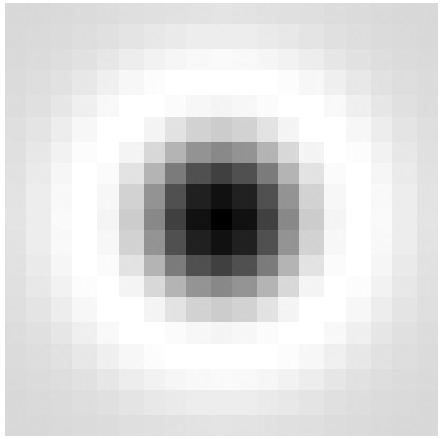
Blob detection

- Remember that “filters intended to give a strong response to a pattern look like that pattern” (Forsyth & Ponce, 2012)
- We can convolve the image with a “**blob filter**” at multiple scales and look for extrema of filter response in the resulting scale space.

All these are blobs. In Spanish, “masa informe”, “mancha”, o “borrón”

We are detecting regions that contrast with what is around them.



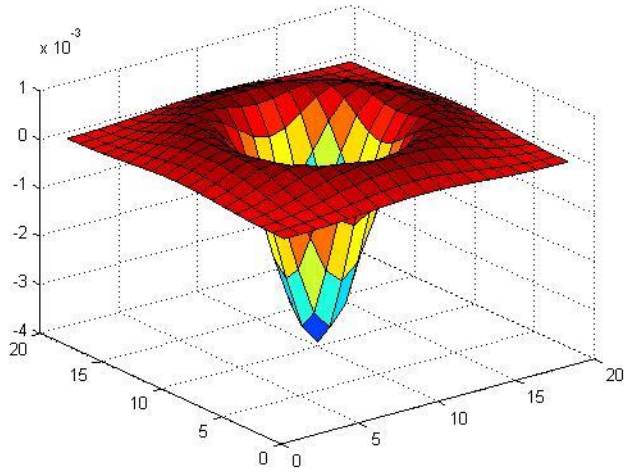


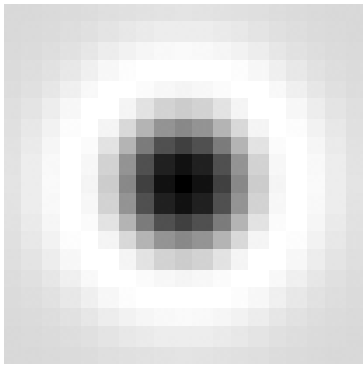
What type of filter is this? Is it similar to something we have seen before?

🤔

T. Lindeberg, Feature detection with automatic scale selection, *IJCV* 30(2), pp 77-116, 1998

The *Laplacian of Gaussian* (*LoG*)



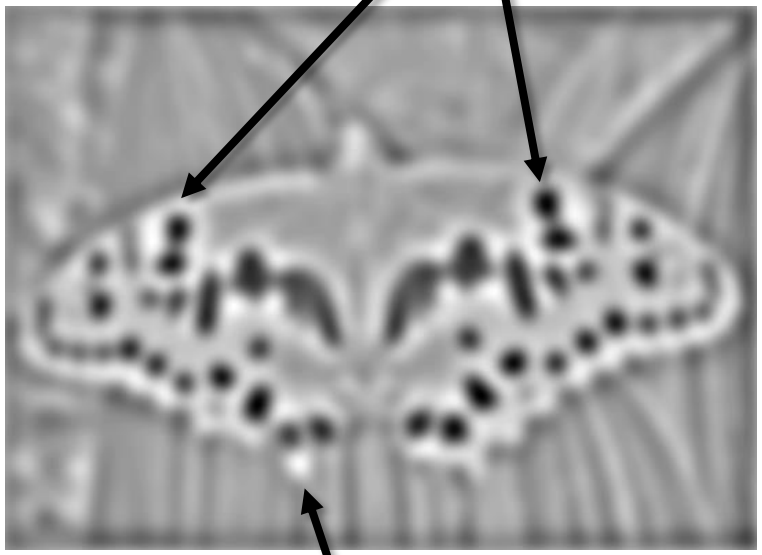


$$\nabla^2_{\text{norm}} g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

(very similar to a Difference of Gaussians (DoG) – i.e. a Gaussian minus a slightly smaller Gaussian)

Laplacian of Gaussian

- “Blob” detector



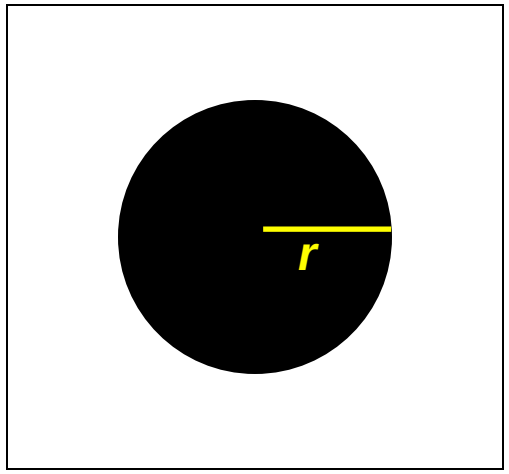
minima (light blobs on dark background)

maxima (dark blobs on light background)

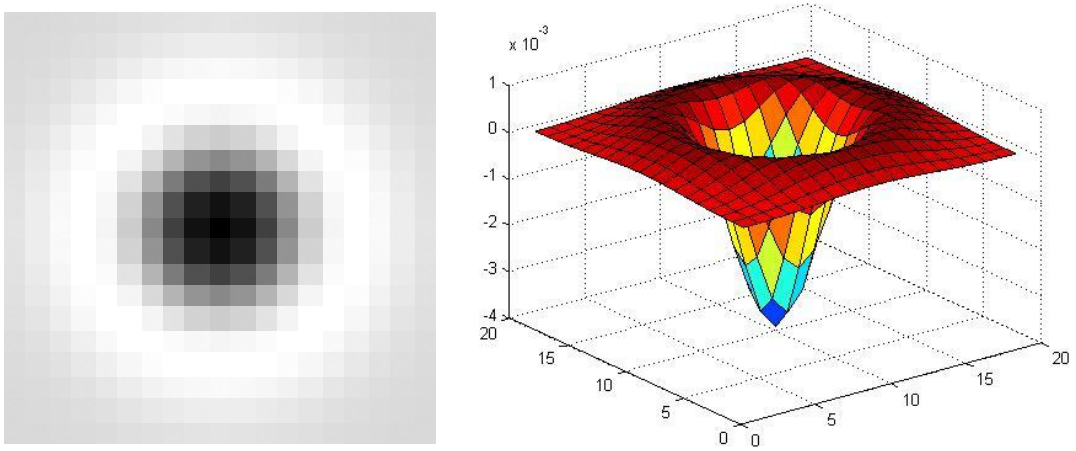
- Find maxima *and minima* of LoG operator in space and scale

Scale selection

- At what scale does the Laplacian achieve a maximum response for a binary circle of radius r?



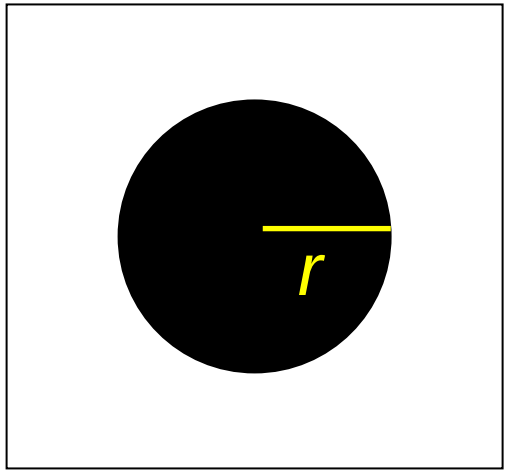
image



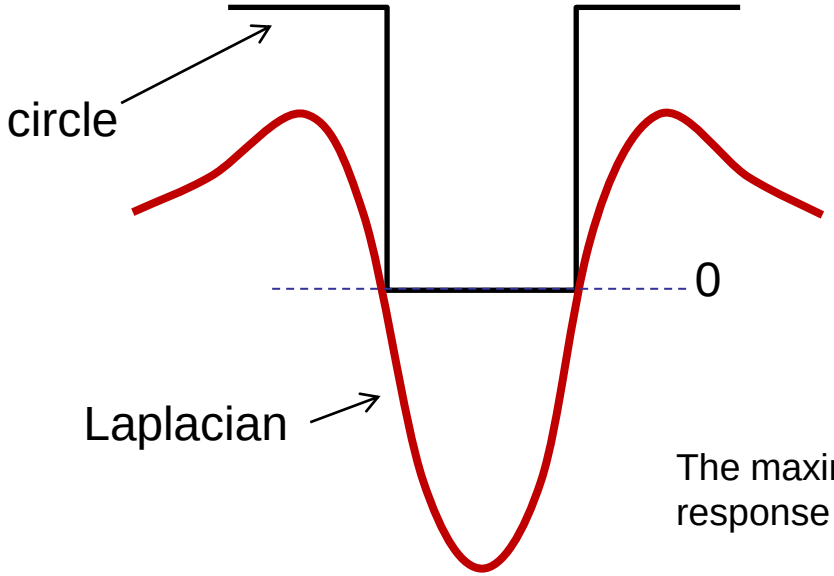
Laplacian

Scale selection

- At what scale does the Laplacian achieve a maximum response for a binary circle of radius r?
  - **To get maximum response, the zeros of the Laplacian have to be aligned with the circle**



image



$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} = \frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

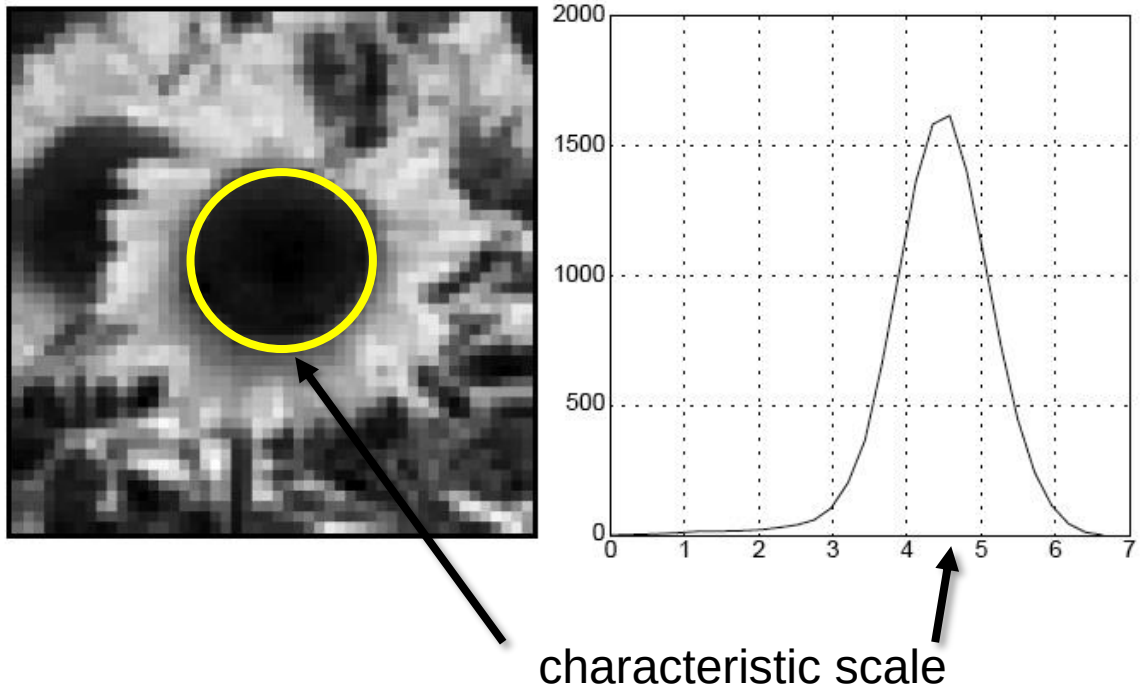
$$\nabla^2_{\text{norm}} g = \sigma^2\left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}\right)$$

$$\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) = 0 \quad \xleftarrow{x^2+y^2=r^2} \quad \sigma = r/\sqrt{2}$$

The maximum response occurs at

Characteristic scale

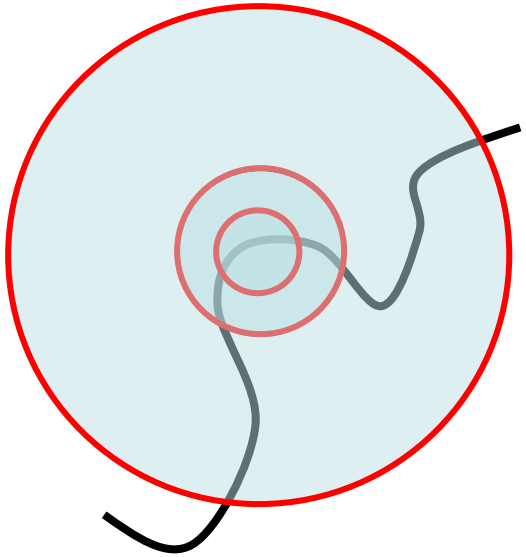
- We define the characteristic scale as the scale that produces peak of Laplacian response



T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* 30 (2): pp 77--116.

Scale invariant detection

Suppose you're looking for potentially interesting structures (like blobs)

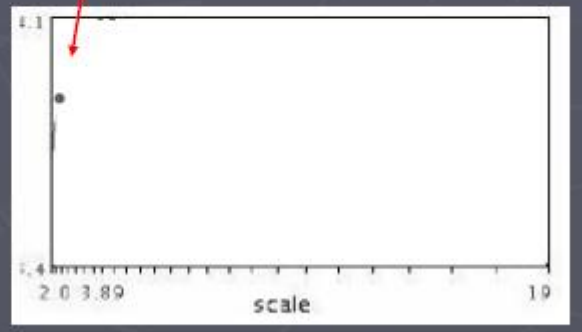


Key idea: find scale that gives local maximum of *f*

- in both position and scale
- One definition of *f*: the Laplacian of Gaussian (to detect corners, we could employ the Harris operator)

Automatic scale selection

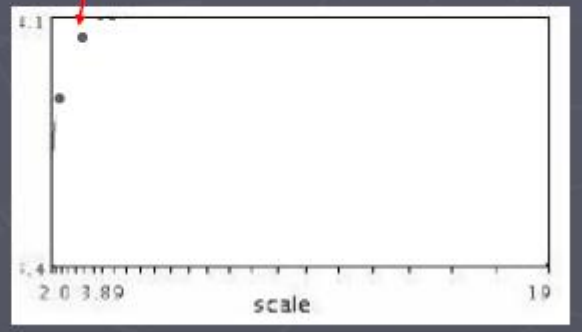
Lindeberg et al., 1996



$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

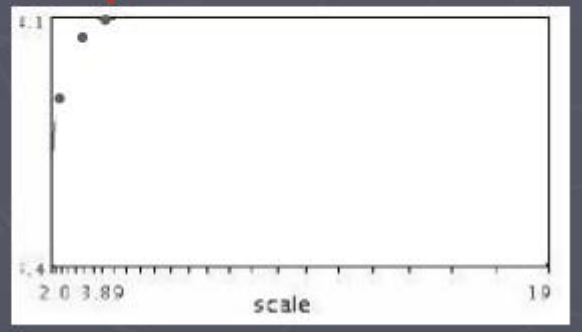
Slides from Tinne Tuytelaars

Automatic scale selection



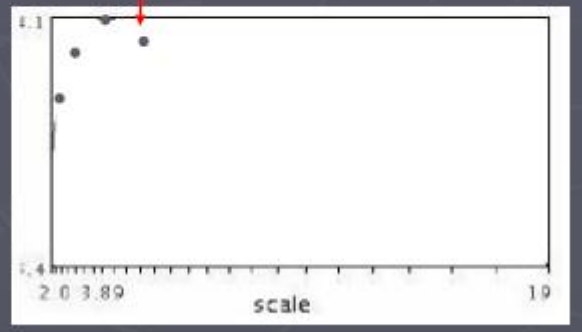
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Automatic scale selection



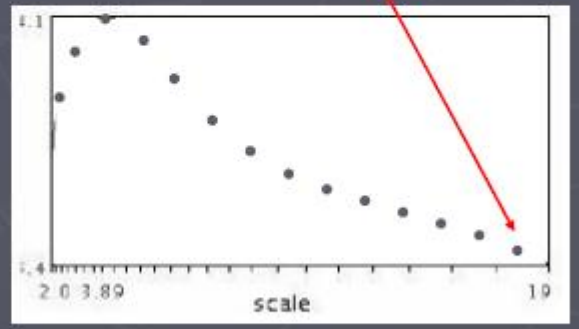
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Automatic scale selection



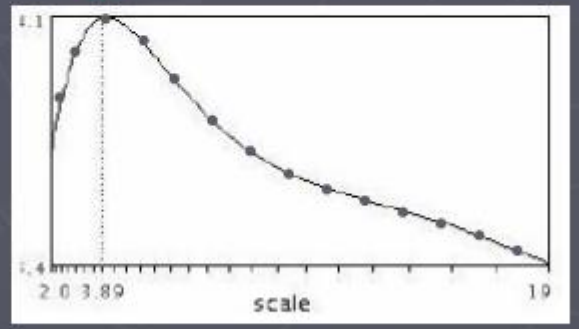
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Automatic scale selection



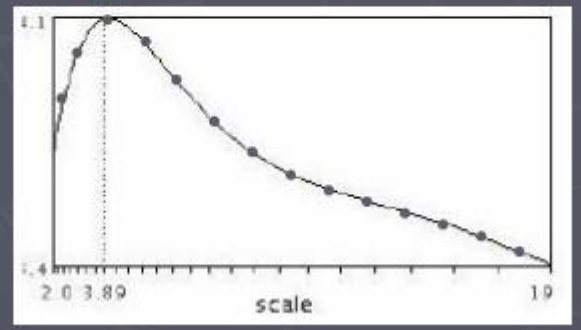
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Automatic scale selection

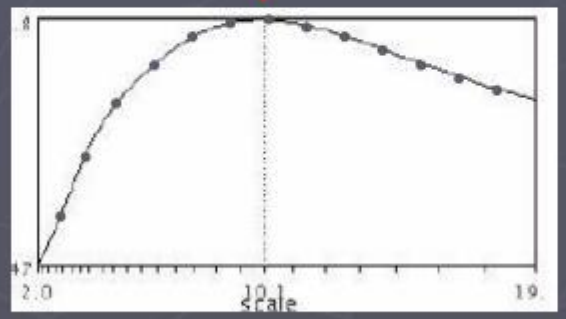


$$f(I_{i_1 \cdots i_m}(x,\sigma))$$

Automatic scale selection

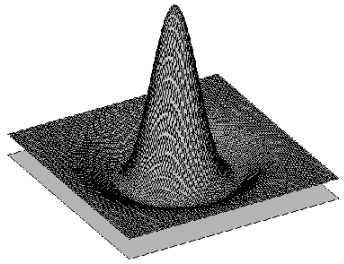


$f(I_{i_1 \ldots i_m}(x,\sigma))$

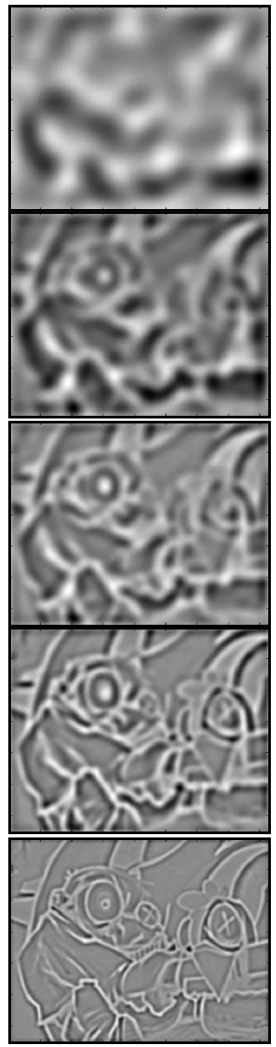


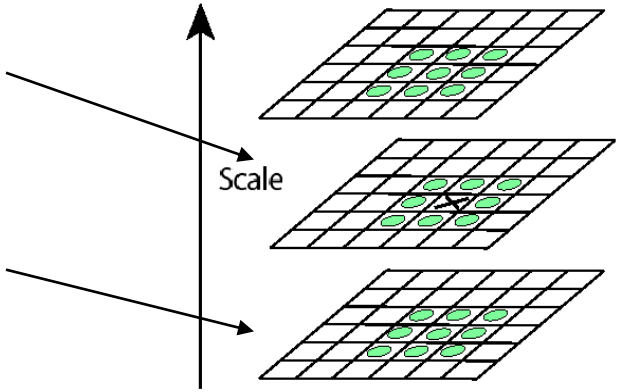
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Find local maxima in 3D position-scale space



$$L_{xx}(\sigma) + L_{yy}(\sigma) \rightarrow$$





1. Convolve image with scale-normalized Laplacian at several scales
2. Find maxima of squared Laplacian response in scale-space

⇒ **List of (x, y, s)**

Could we also introduce here the orientation of the detected blob?

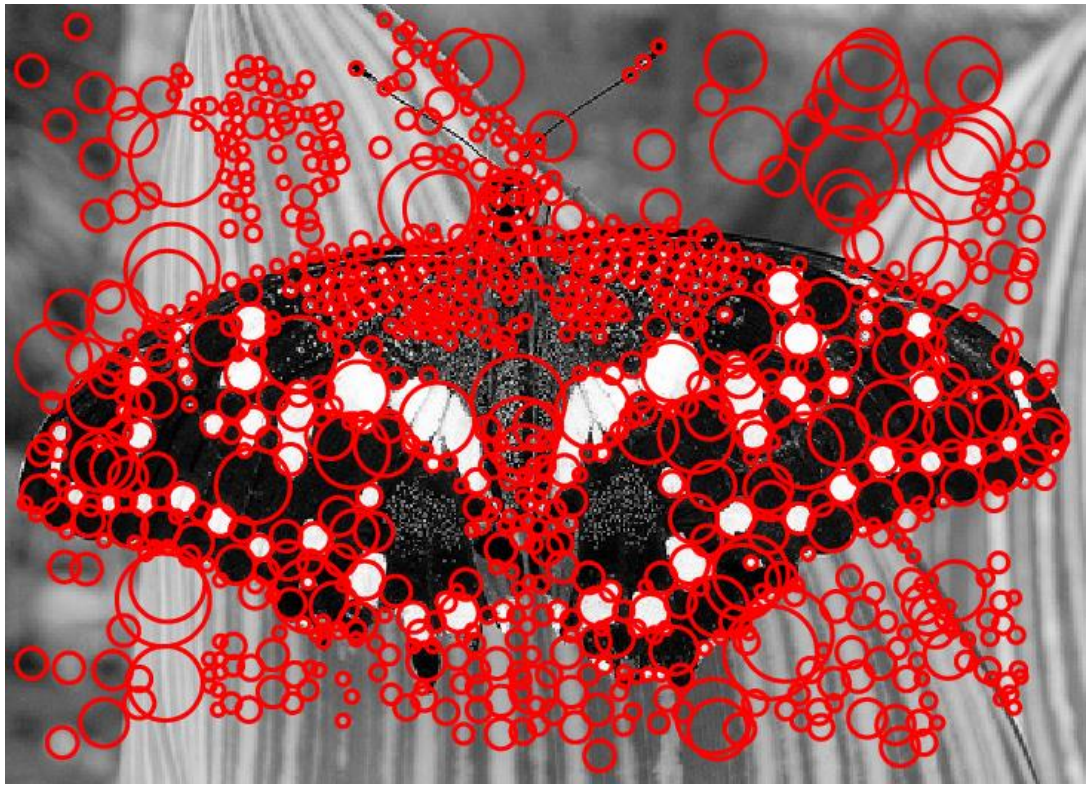
Why squared?

Scale-space blob detector: Example

Scale-space blob detector: Example

sigma = 11.9912

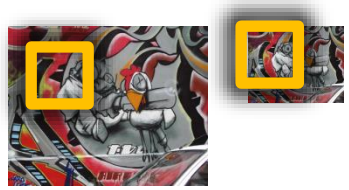
Scale-space blob detector: Example



The larger the sigma (s), the larger the radius of the detected “blob”

Implementation

- Instead of computing *f* for larger and larger windows, we can implement it using a fixed window size with a Gaussian pyramid



(sometimes need to create in-between levels, e.g. a ¾-size image)

Handcrafted Feature Detection and Extraction

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE GRANADA

DaSCI

Instituto Andaluz de Investigación en **Data Science** and **Computational Intelligence**