# Lecture 7:
# Object Recognition

# What we will learn today?

- Introduction to object recognition
- K-nearest neighbor algorithm
- A simple Object Recognition pipeline
- Visual Bag of Words

# Visual Recognition

- Design algorithms that have the capability to:
  - Classify images or videos
  - Detect and localize objects
  - Estimate semantic and geometrical attributes
  - Classify human activities and events

Why is this challenging?

# How many object categories are there?

# Challenges: viewpoint variation

Michelangelo 1475-1564

# Challenges: illumination

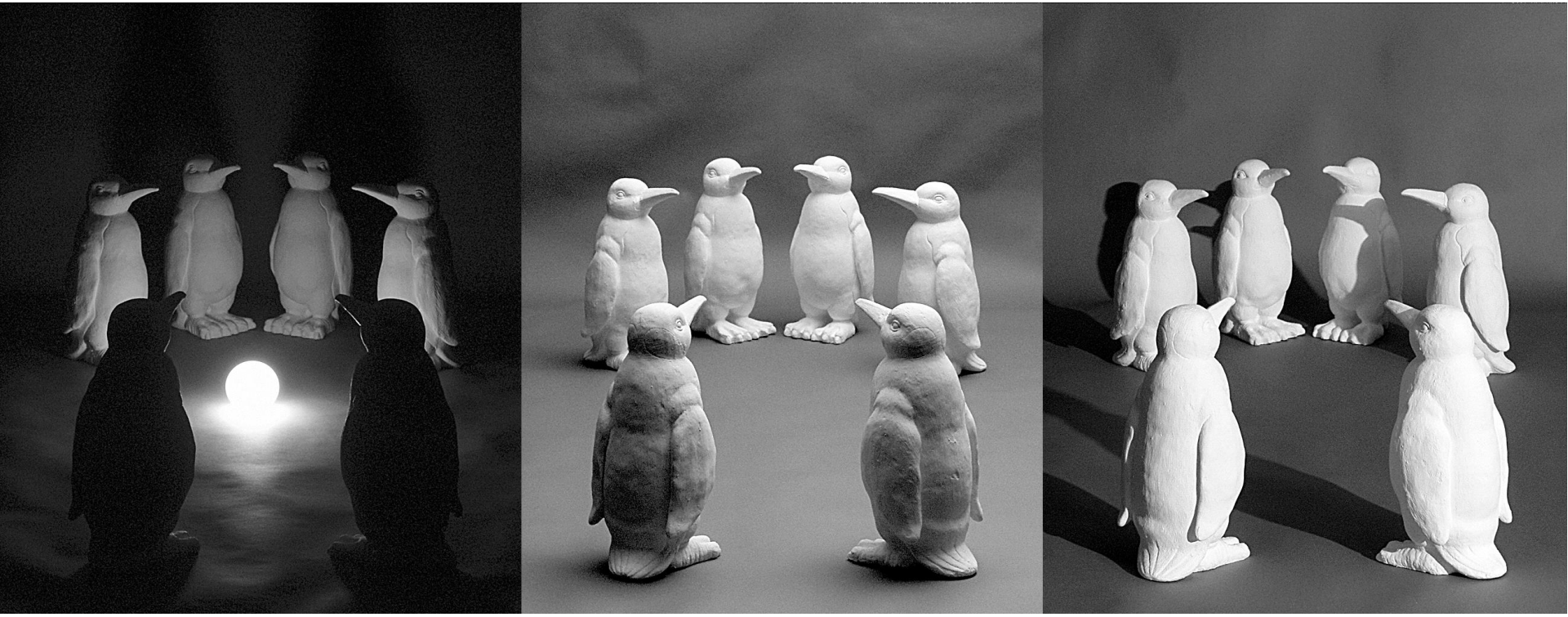
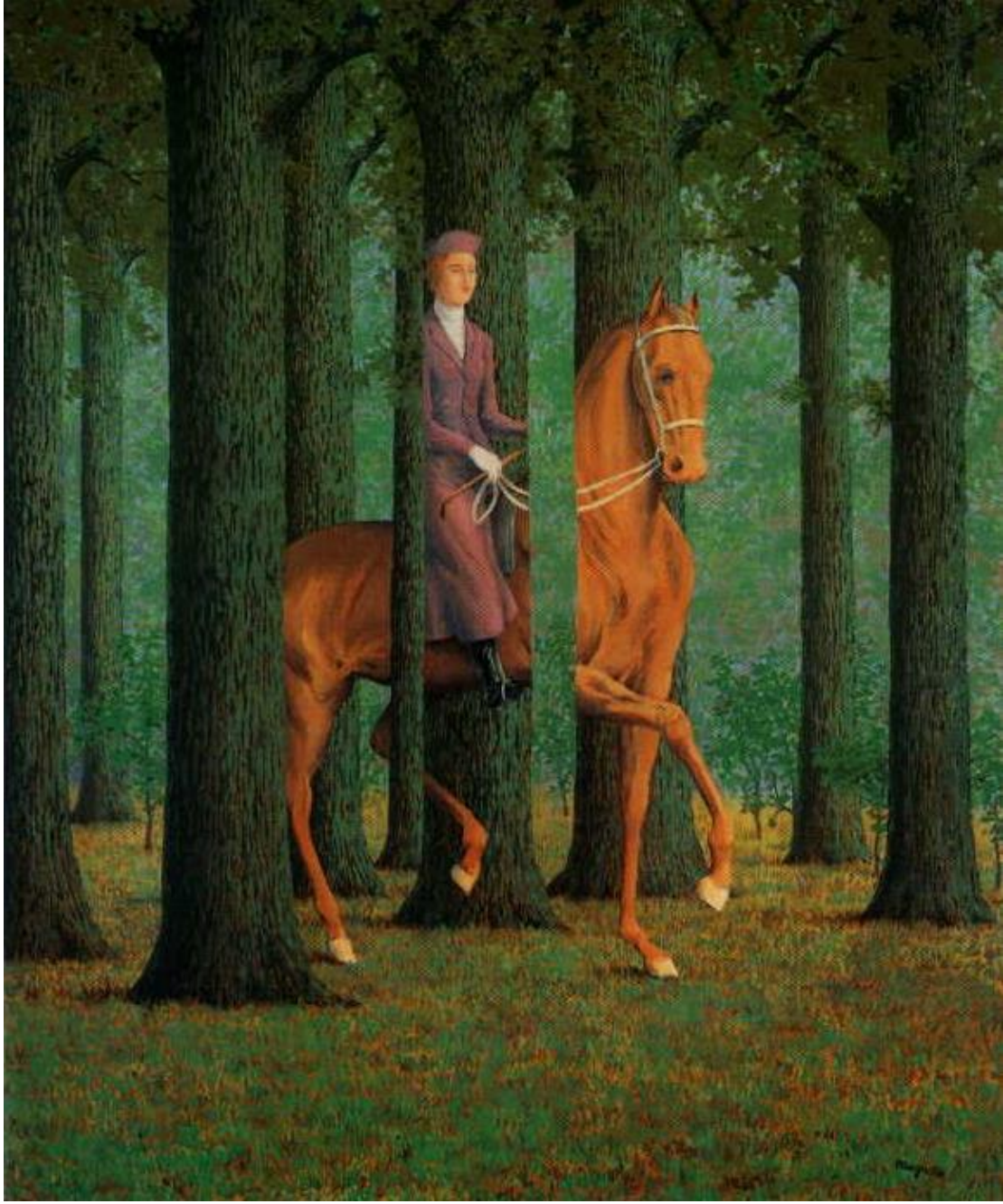


image credit: J. Koenderink

# Challenges: scale

# Challenges: deformation

# Challenges: occlusion

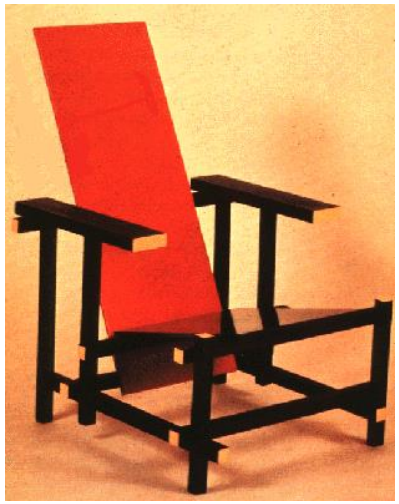


Magritte, 1957

# Challenges: background clutter

Kilmeny Niland. 1995

# Challenges: intra-class variation



# What we will learn today?

- Introduction
- K-nearest neighbor algorithm
- A simple Object Recognition pipeline
- Visual Bag of Words

# Supervised Learning

Functions $\mathcal{F}$

$f : \mathcal{X} \to \mathcal{Y}$

Training data

$\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$

LEARNING

find $\hat{f} \in \mathcal{F}$
s.t. $y_i \approx \hat{f}(x_i)$

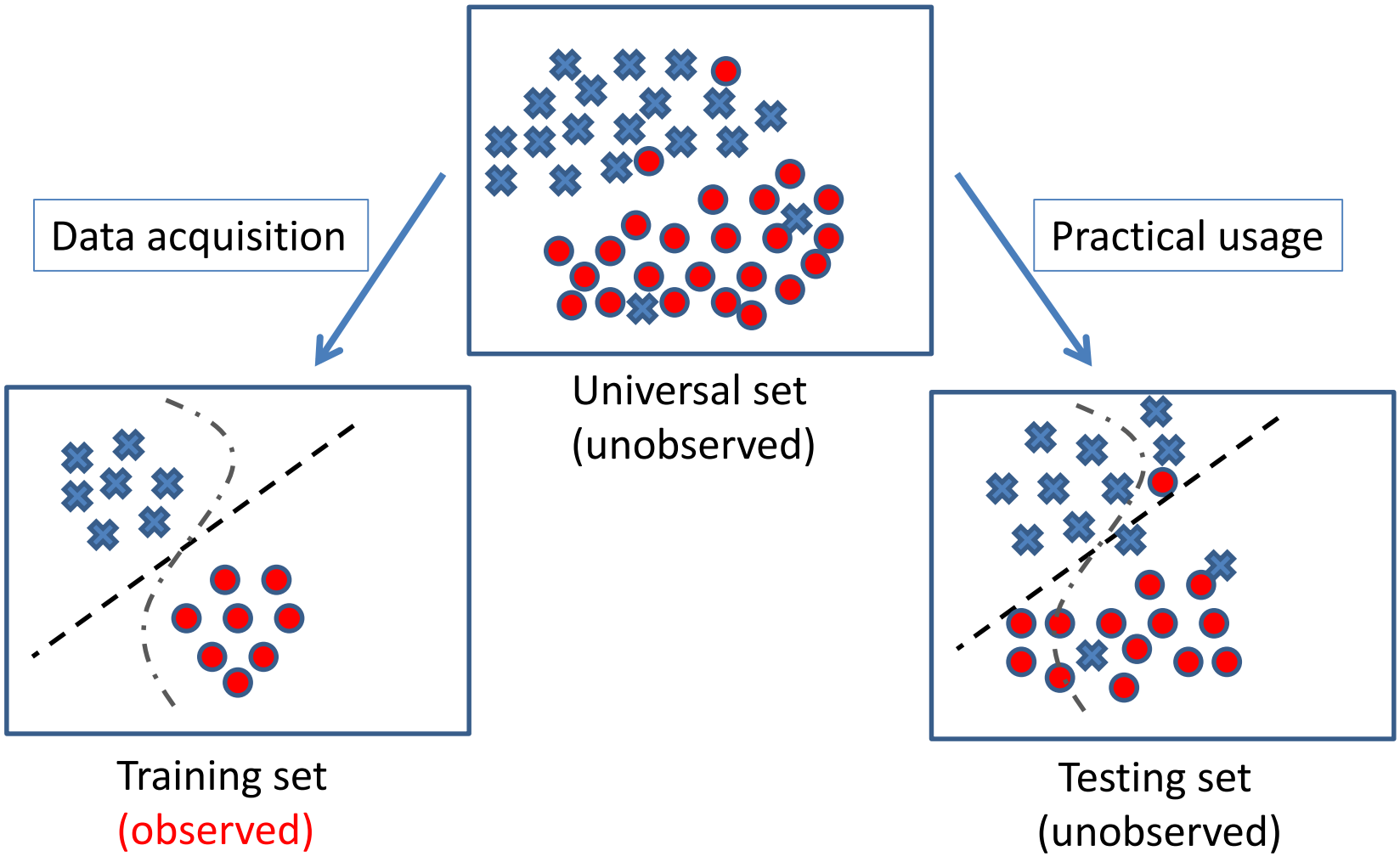
Learning machine

New data

PREDICTION

$y = \hat{f}(x)$ ⟸ $x$

# Training and testing



# The Bias-Variance Decomposition

The *expected squared loss*,

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t$$

where

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x})\,\mathrm{d}t.$$

The second term of E[L] corresponds to the noise inherent in the random variable t.

What about the first term?

# The Bias-Variance Decomposition

Suppose we were given multiple data sets, each of size N. Any particular data set, D, will give a particular function y(x;D). We then have

$$
\begin{aligned}
&\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2 \\
&\quad = \quad \{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 \\
&\quad = \quad \{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 \\
&\qquad +2\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}.
\end{aligned}
$$

# The Bias-Variance Decomposition

Taking the expectation over D yields

$$\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2\right]$$
$$= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right]}_{\text{variance}}.$$

# The Bias-Variance Decomposition

Thus we can write

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

where

$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

$$\text{noise} = \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t$$

# The Bias-Variance Decomposition

- Components of expected loss
  - Noise in our observations: unavoidable
  - Bias: how much the average model over all training sets differs from the true model
    - Error due to inaccurate assumptions/simplifications made by the model
  - Variance: how much models estimated from different training sets differ from each other



Adapted from L. Lazebnik

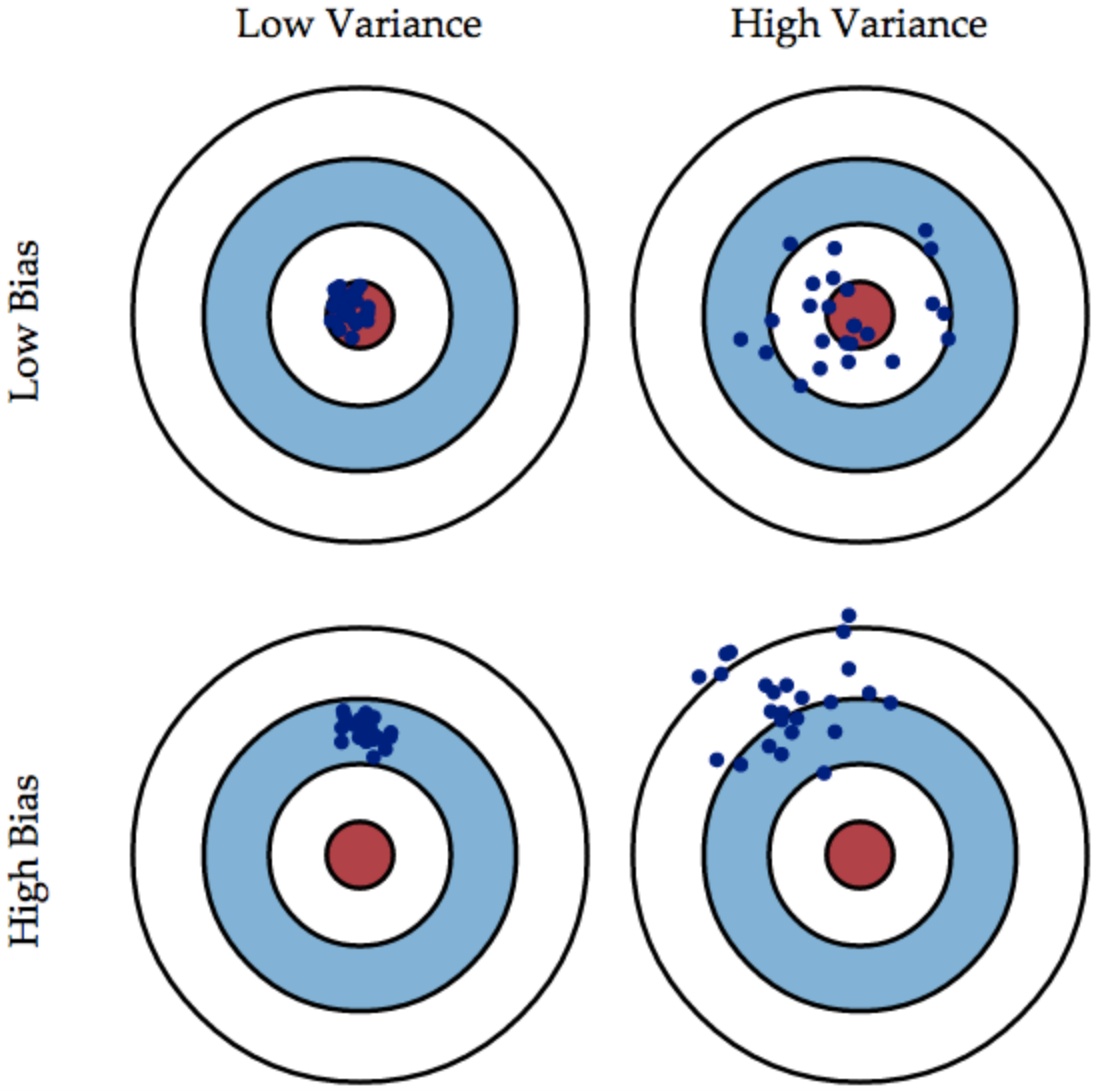
# The Bias-Variance Decomposition

- Underfitting: model is too “simple” to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- Overfitting: model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error



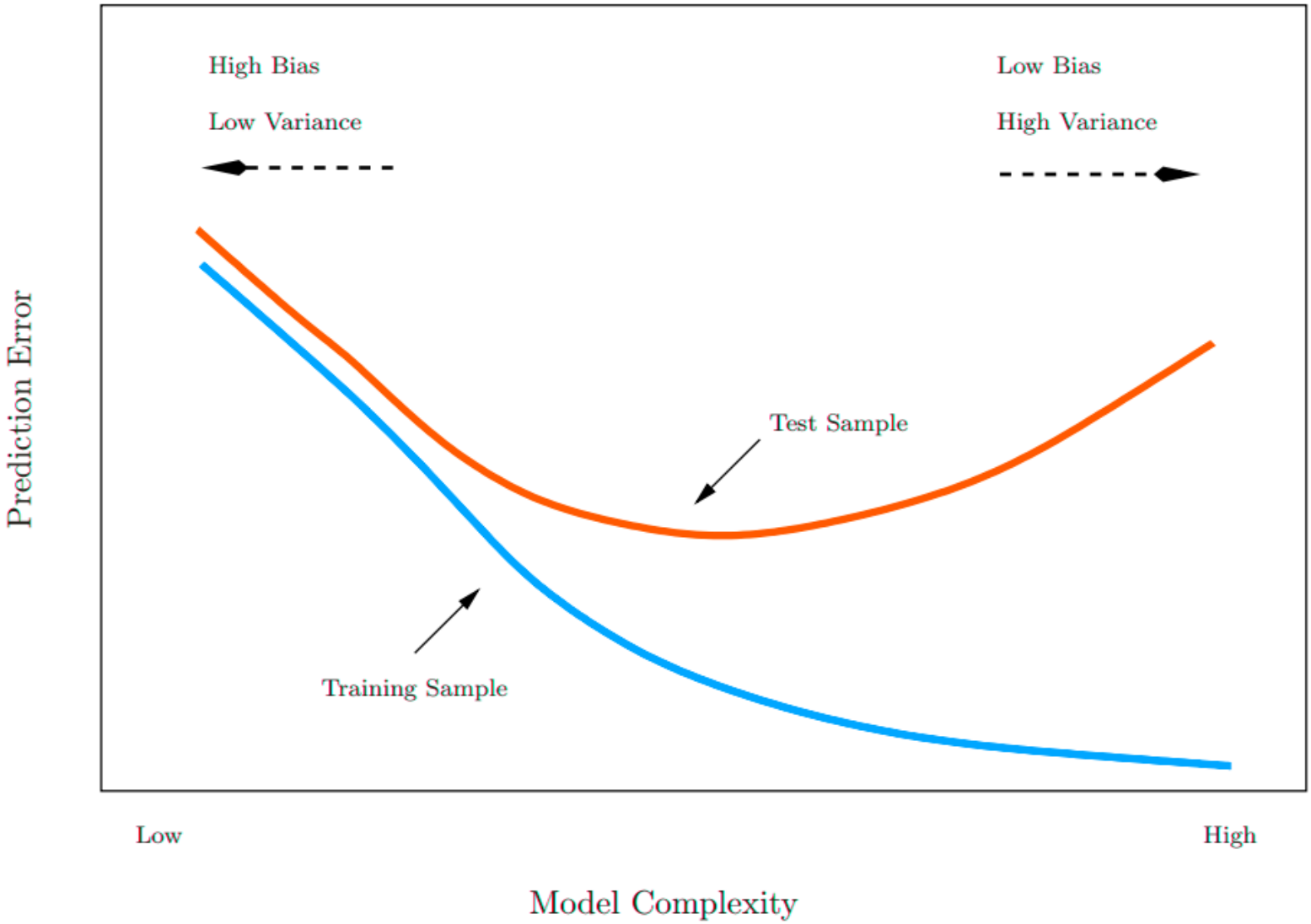
Adapted from L. Lazebnik

# The Bias-Variance Decomposition

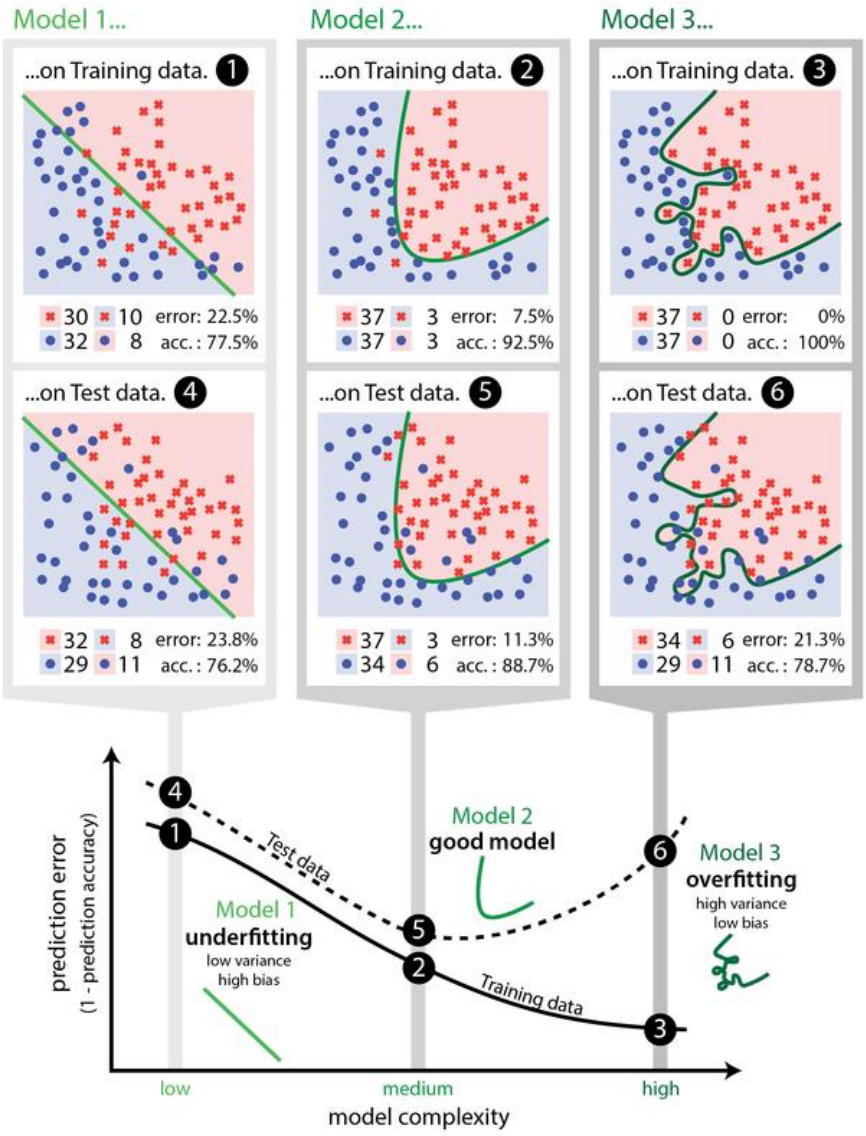


# The Bias-Variance Decomposition

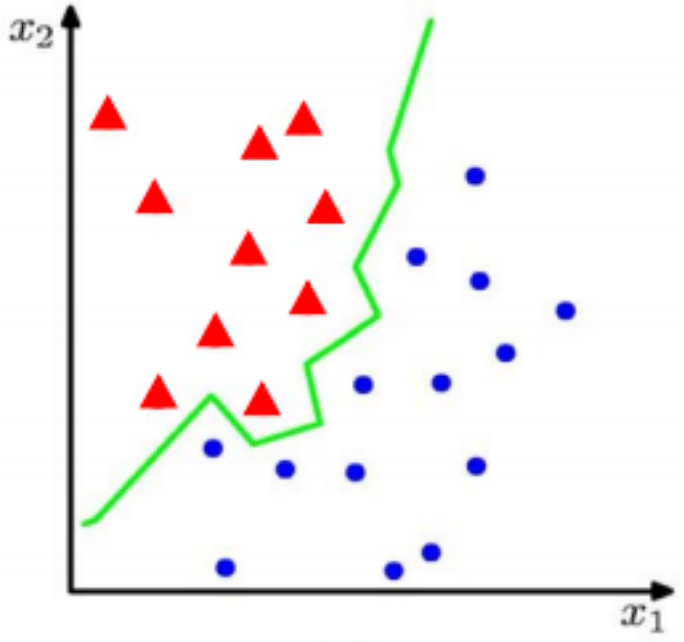
- (bias$^2$+variance) is what counts for prediction
- Often:
  - low bias => high variance
  - low variance => high bias
- Tradeoff:
  - bias$^2$ vs. variance


High Bias
Low Variance
Low Bias
High Variance
Test Sample
Training Sample
Prediction Error
Low
High
Model Complexity

# The Bias-Variance Trade-off



# Classification




- Suppose we are given a training set of N observations

$$(x_1, \ldots, x_N) \text{ and } (y_1, \ldots, y_N), x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}$$

- Classification problem is to estimate f(x) from this data such that

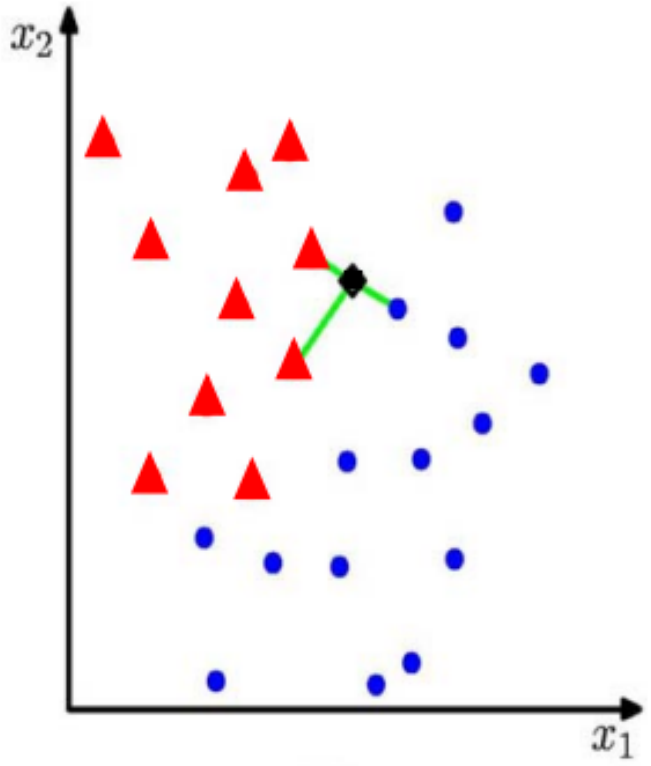
$$f(x_i) = y_i$$

# K Nearest Neighbour (K-NN) Classifier

## Algorithm

- For each test point, x, to be classified, find the K nearest samples in the training data
- Classify the point, x, according to the majority vote of their class labels

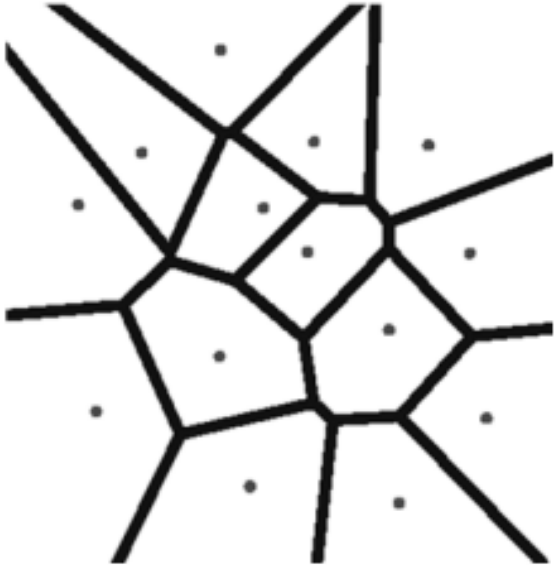
e.g. K = 3




• applicable to multi-class case

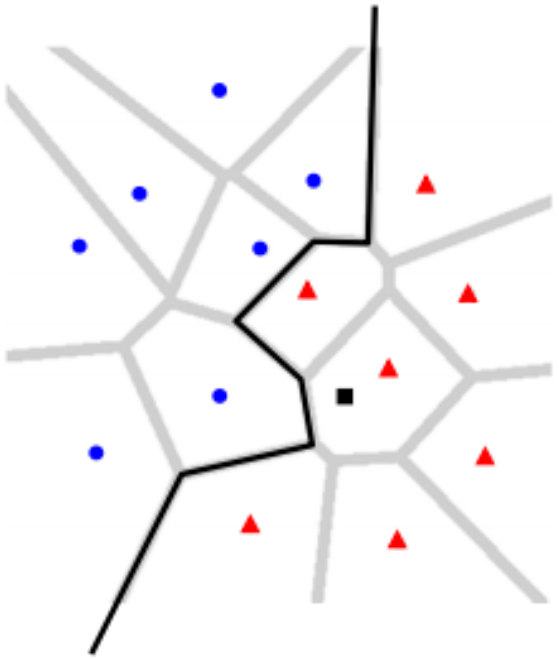
# K Nearest Neighbour (K-NN) Classifier

K = 1



Voronoi diagram:

- partitions the space into regions
- boundaries are equal distance from training points



Classification boundary:

- non-linear

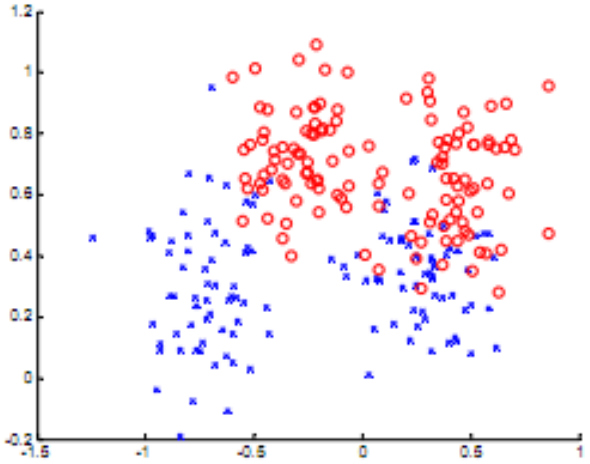
# K Nearest Neighbour (K-NN) Classifier

## A sampling assumption: training and test data

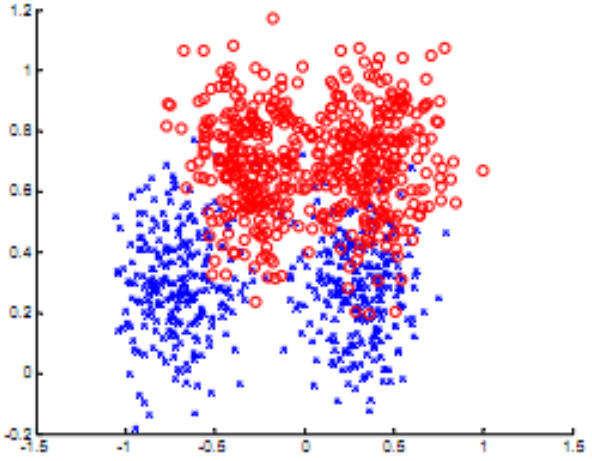
- Assume that the training examples are drawn independently from the set of all possible examples.

- This makes it very unlikely that a strong regularity in the training data will be absent in the test data.

- Measure classification error as $= \frac{1}{N} \sum_{i=1}^{N} \underbrace{[\mathbf{y}_i \neq f(\mathbf{x}_i)]}_{\text{loss function}}$ The "risk"



Training data

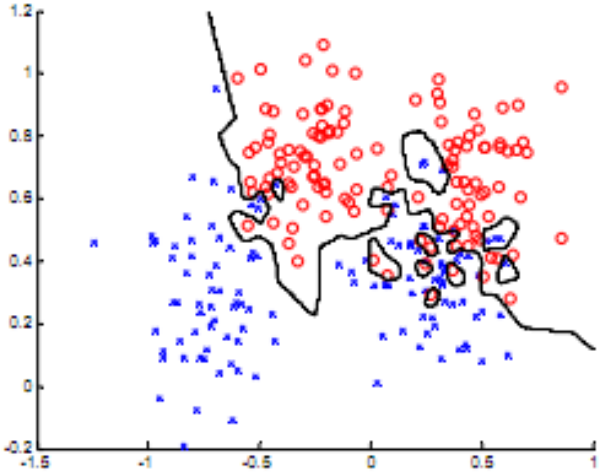


Testing data

# K Nearest Neighbour (K-NN) Classifier

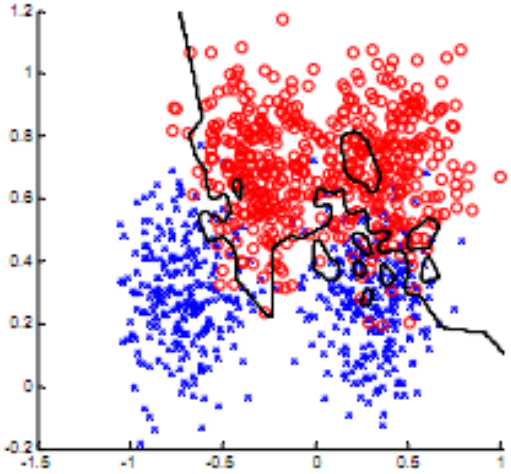
K = 1

Training data




error = 0.0

Testing data

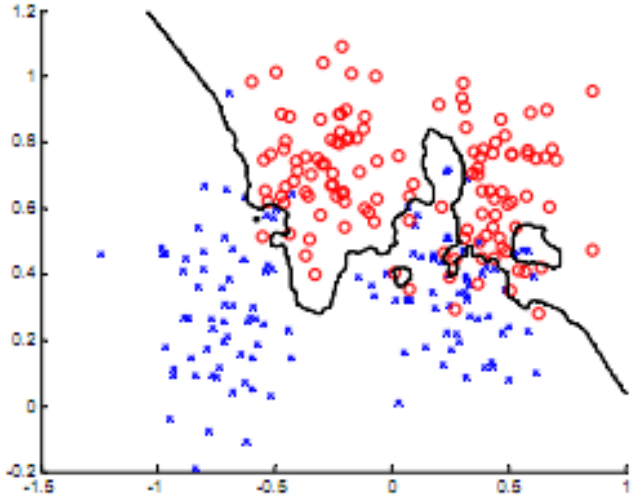



error = 0.15

# K Nearest Neighbour (K-NN) Classifier

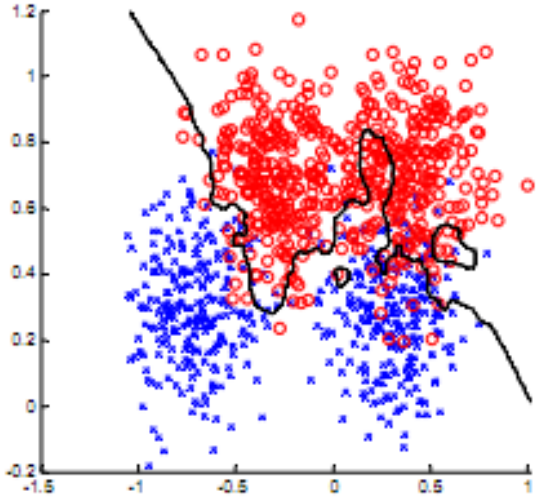
K = 3

Training data




error = 0.0760

Testing data

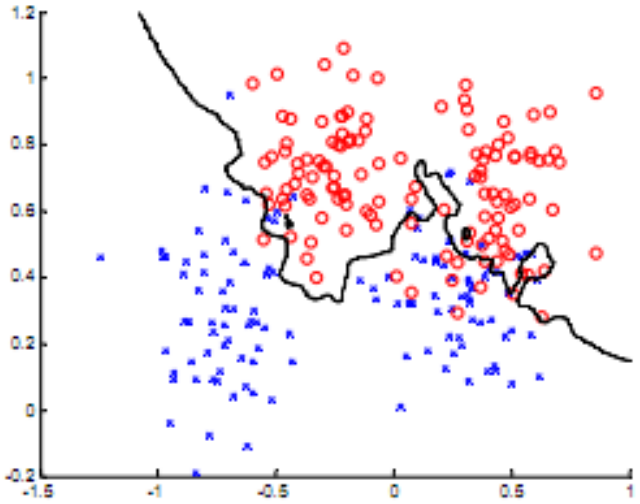



error = 0.1340

# K Nearest Neighbour (K-NN) Classifier

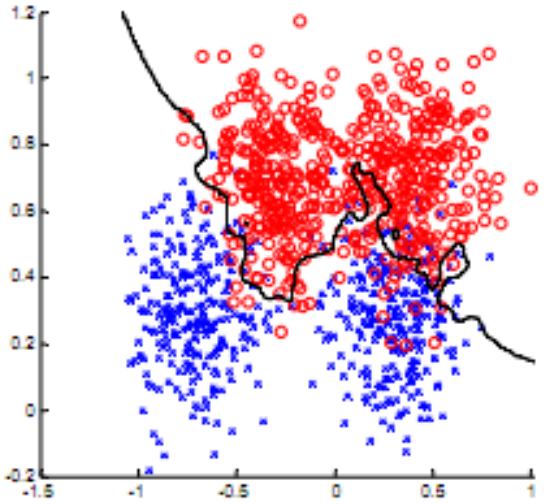
K = 7

Training data




error = 0.1320

Testing data

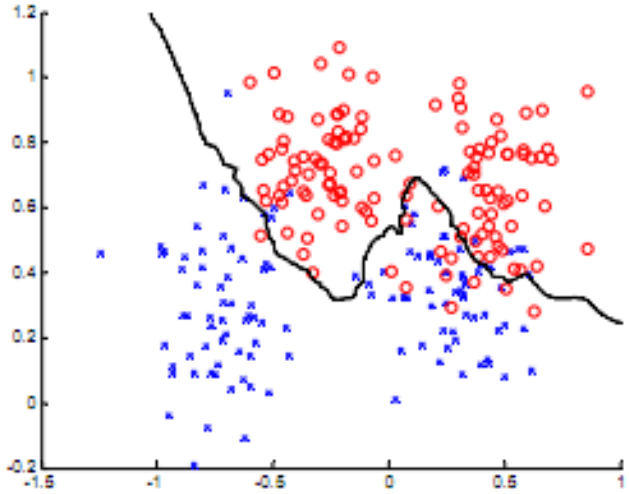



error = 0.1110

# K Nearest Neighbour (K-NN) Classifier

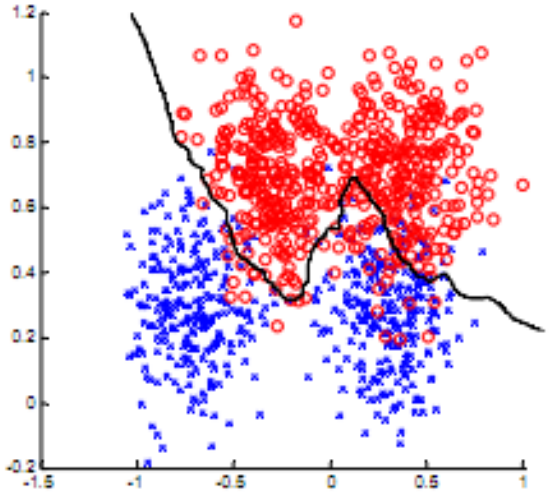
K = 21

Training data




error = 0.1120

Testing data




error = 0.0920

# What we will learn today?

- Introduction
- K-nearest neighbor algorithm
- A simple Object Recognition pipeline
- Visual Bag of Words

# Object recognition: a classification framework

- Apply a prediction function to a feature representation of the image to get the desired output:

f(  ) = “apple”

f(  ) = “tomato”

f(  ) = “cow”

Dataset: ETH-80, by B. Leibe  Slide credit: L. Lazebnik

# A simple pipeline - Training

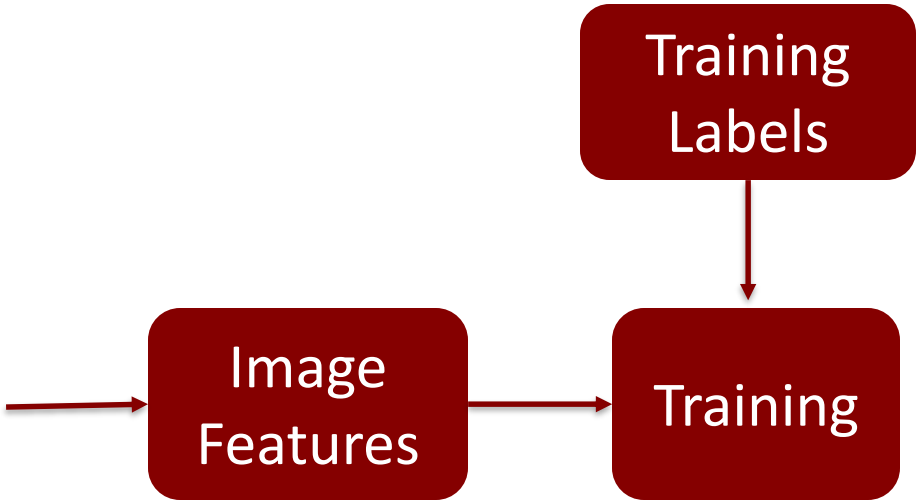
Training Images



Image Features

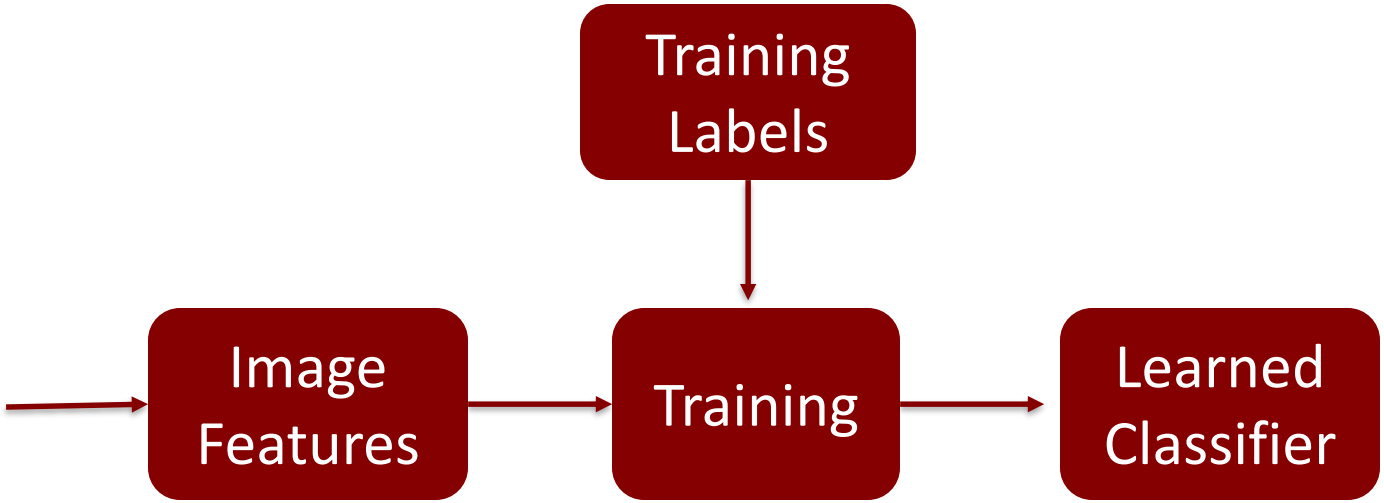
# A simple pipeline - Training

Training Images



# A simple pipeline - Training

Training Images



# A simple pipeline - Training

Training Images

Training Labels

Image Features

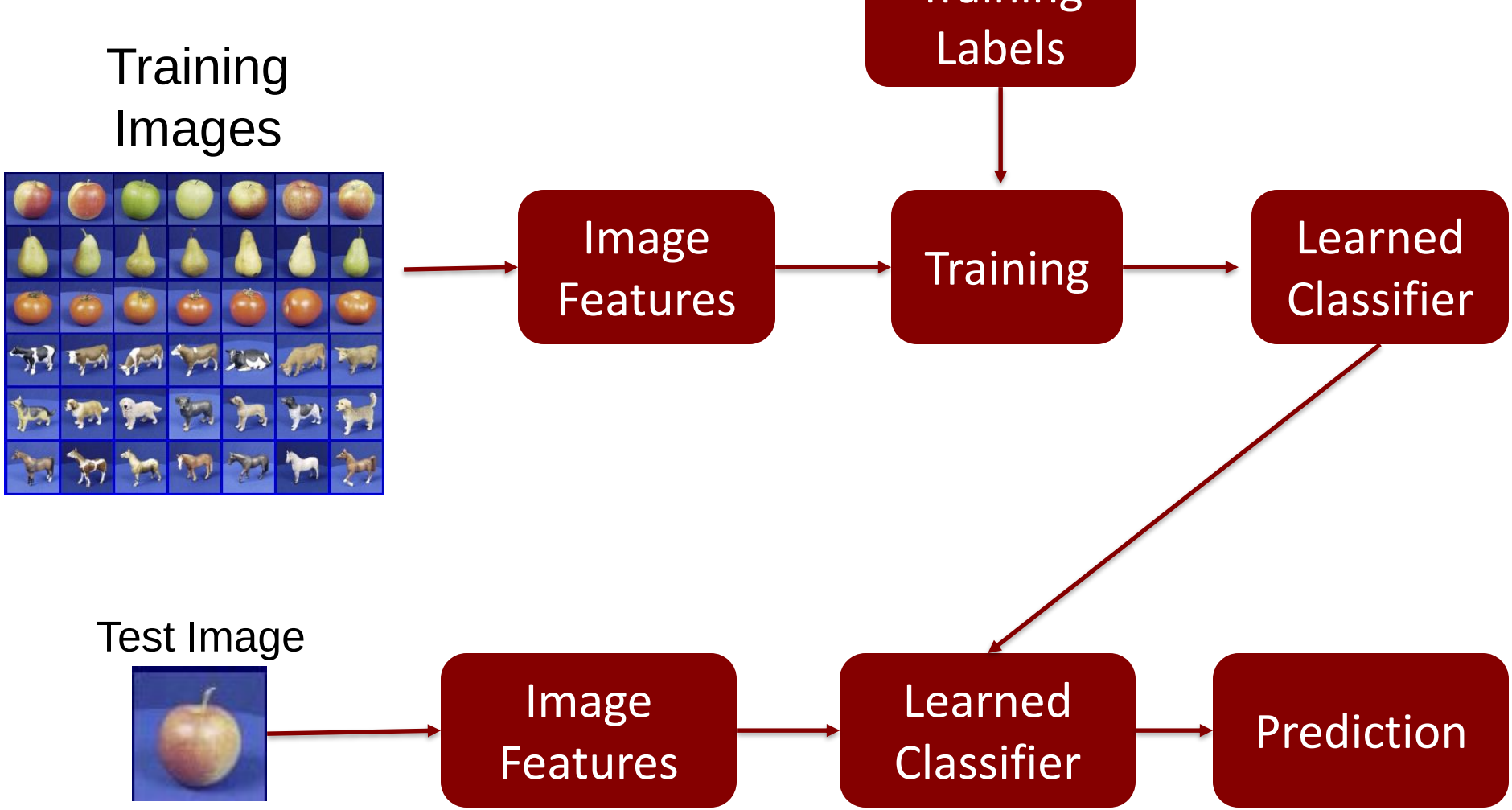
Training

Learned Classifier

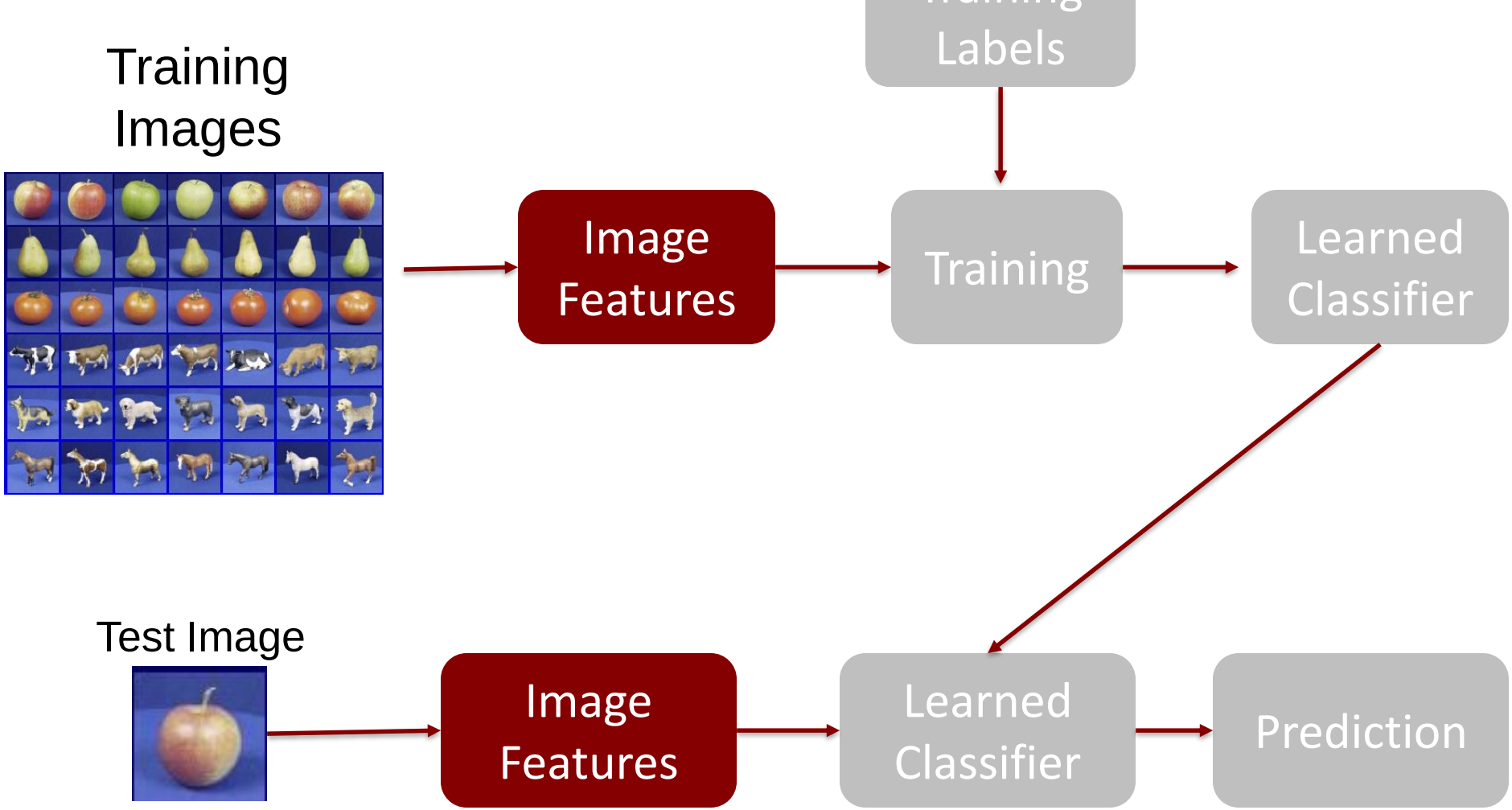
Test Image

Image Features

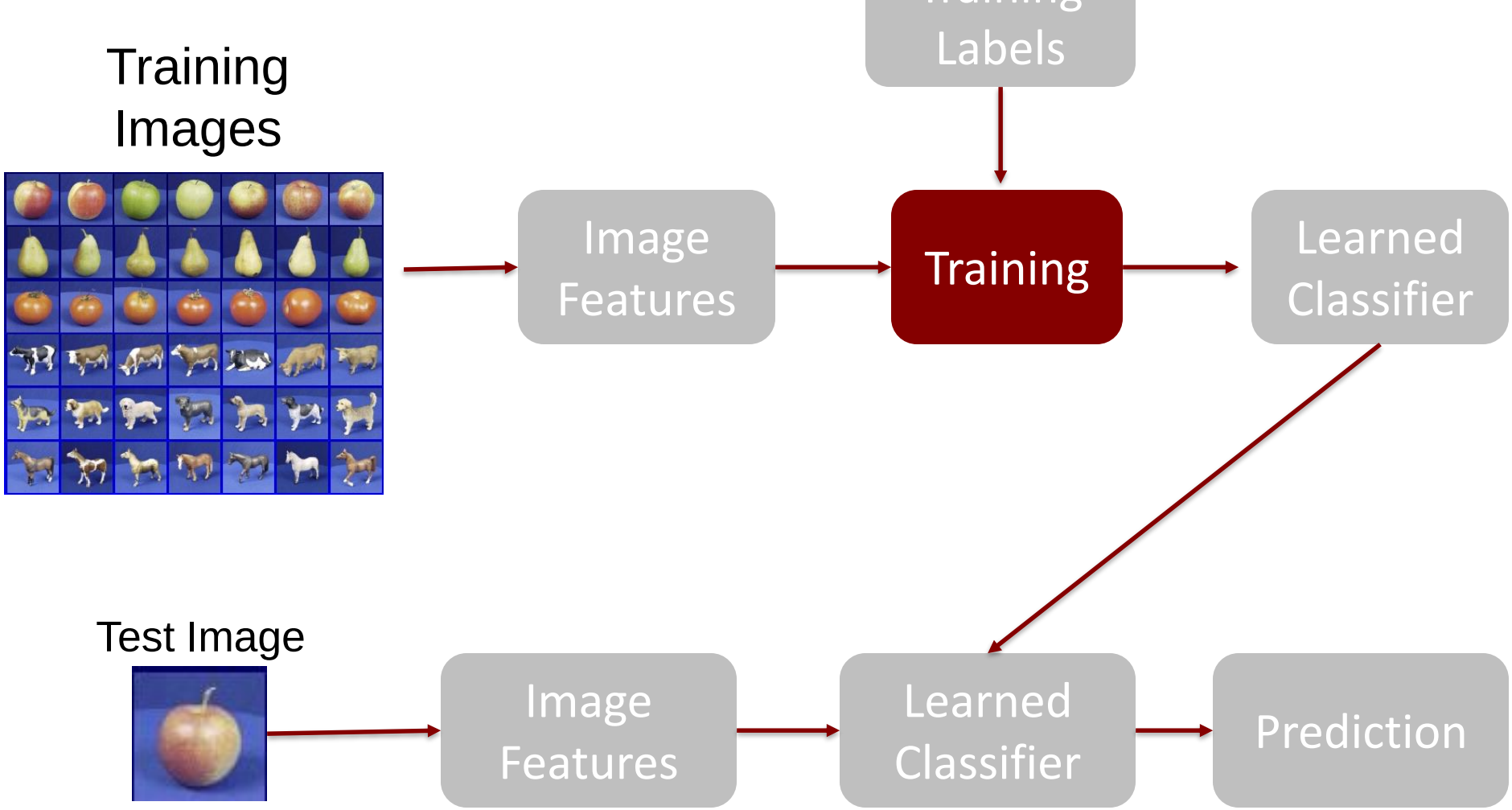
# A simple pipeline - Training



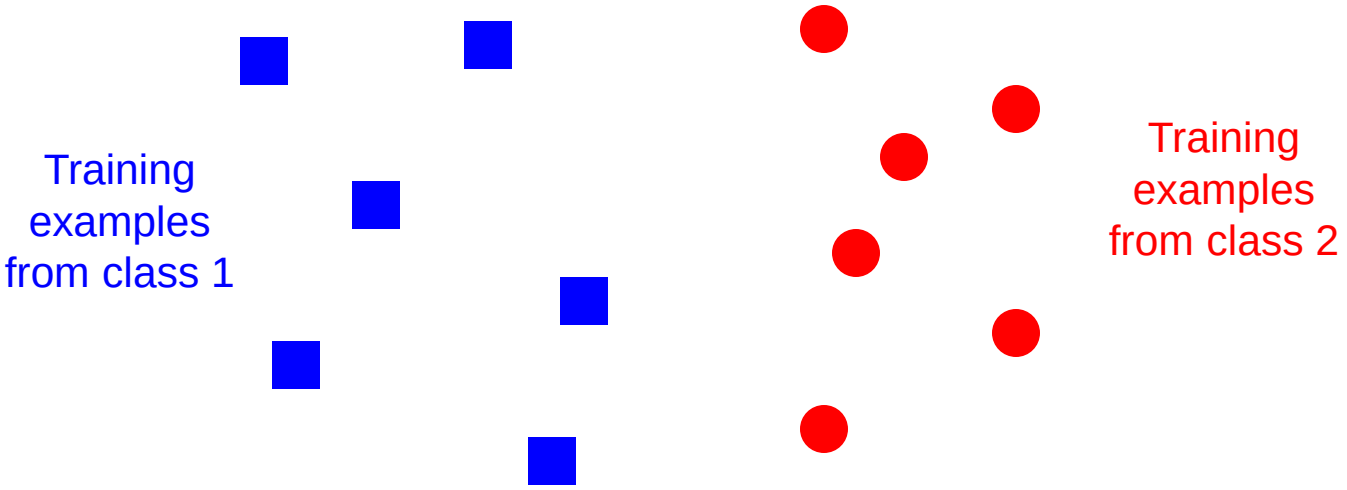
# A simple pipeline - Training



# A simple pipeline - Training



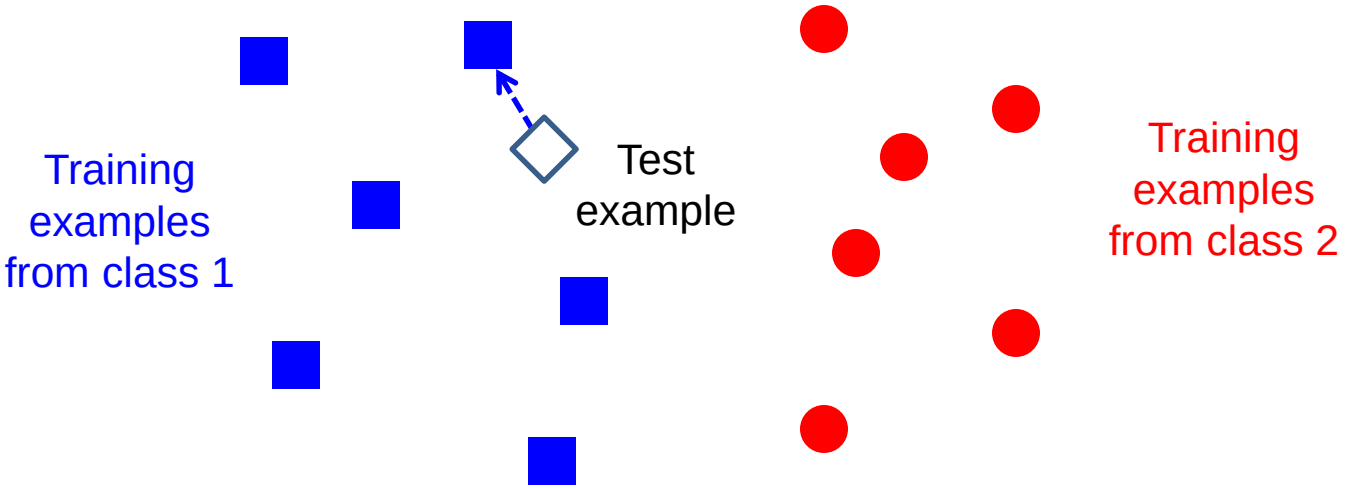
# Classifiers: Nearest neighbor




Slide credit: L. Lazebnik

# A simple pipeline - Training

# Classifiers: Nearest neighbor




Slide credit: L. Lazebnik

# Lecture:
# Visual Bag of Words

# What we will learn today?

- Introduction
- K-nearest neighbor algorithm
- A simple Object Recognition pipeline
- Visual Bag of Words

Object → Bag of ‘words’

# Bags of features for object recognition

face, flowers, building

- Works pretty well for image-level classification and for recognizing object *instances*

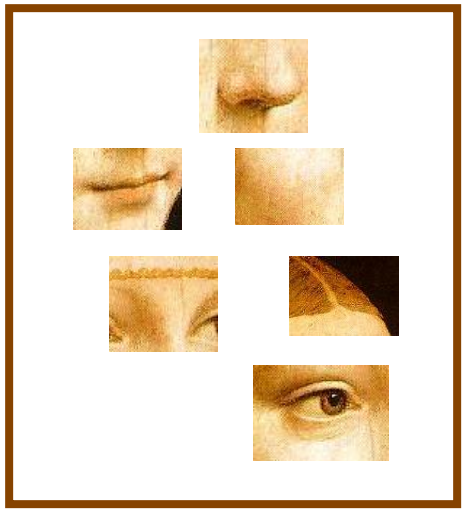
# Bag of features

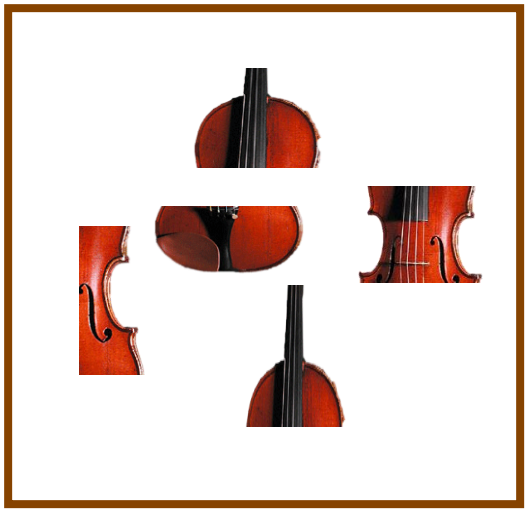
- First, take a bunch of images, extract features, and build up a “dictionary” or “visual vocabulary” – a list of common features

- Given a new image, extract features and build a histogram – for each feature, find the closest visual word in the dictionary

# Bag of features: outline

1. Extract features





# Bag of features: outline

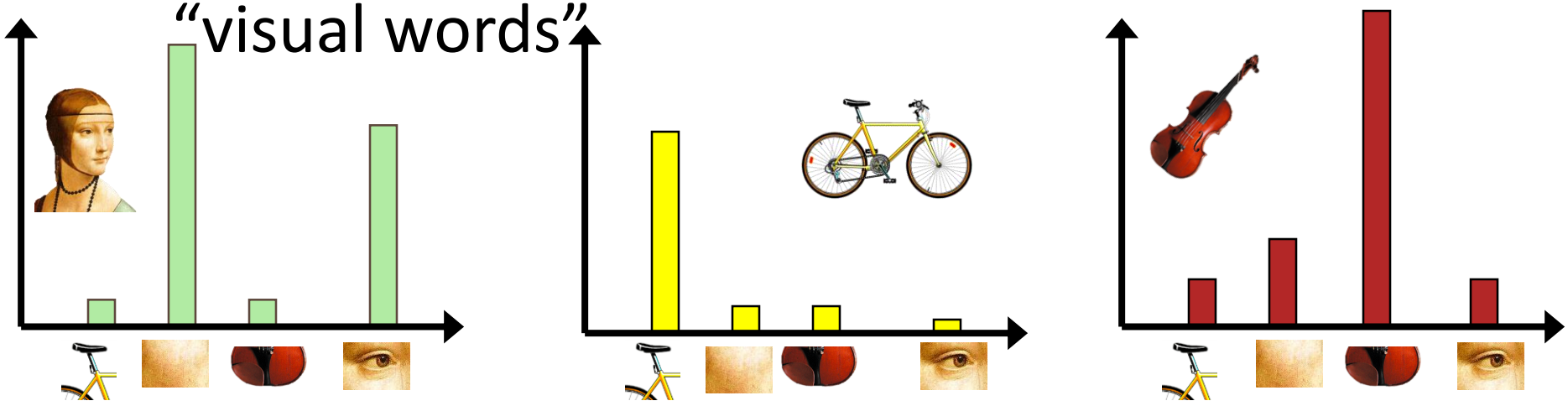
1. Extract features
2. Learn “visual vocabulary”

# Bag of features: outline

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary

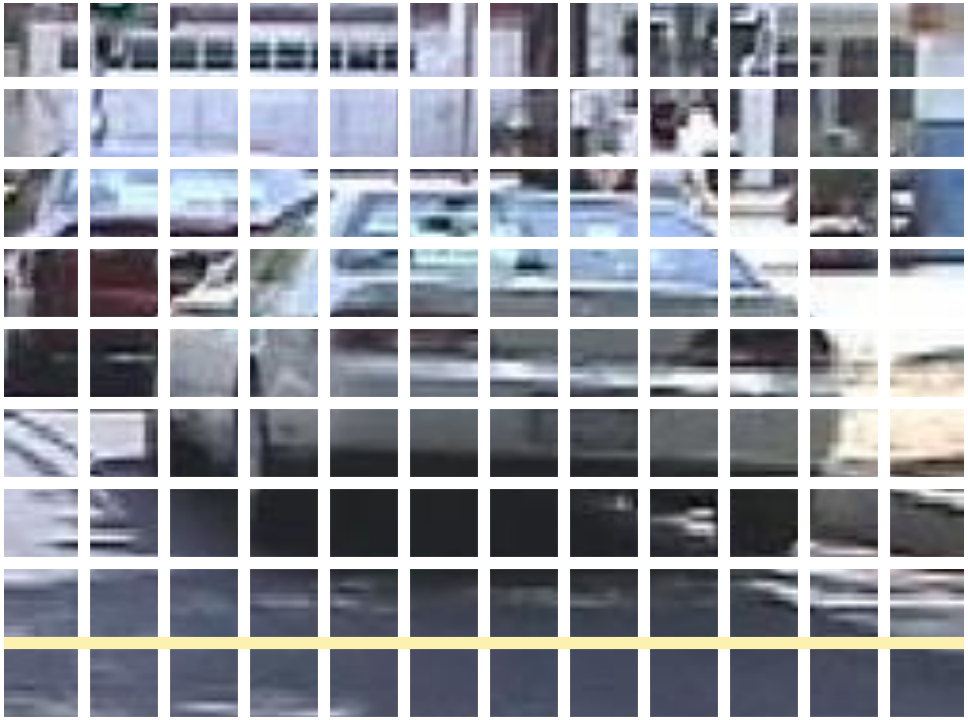
# Bag of features: outline

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images by frequencies of “visual words”



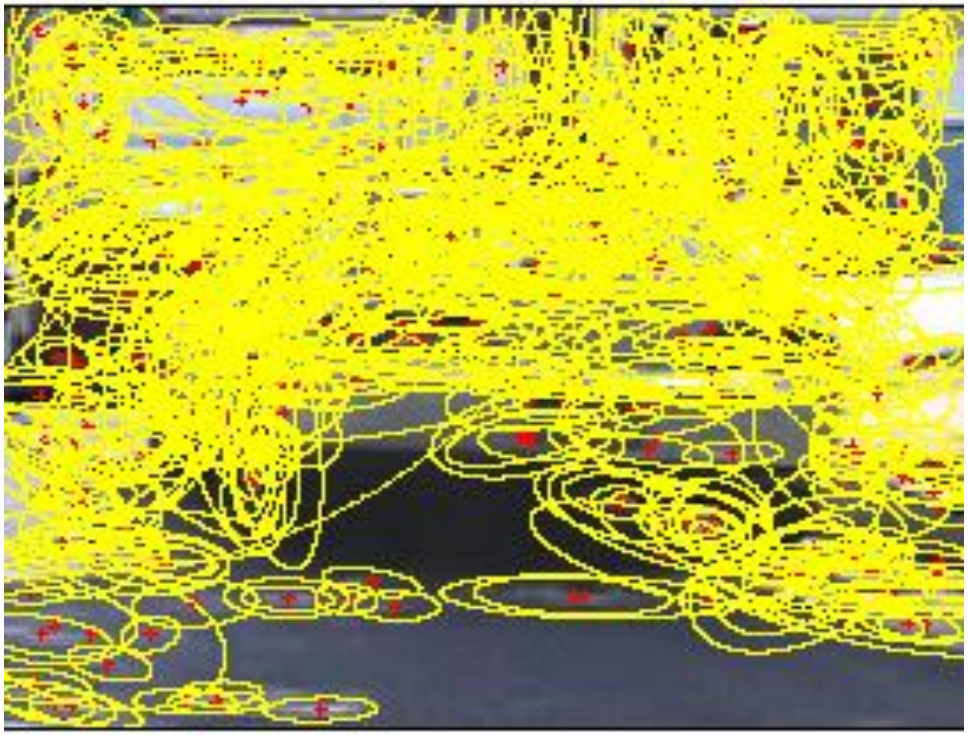
# 1. Feature extraction

- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005



# 1. Feature extraction

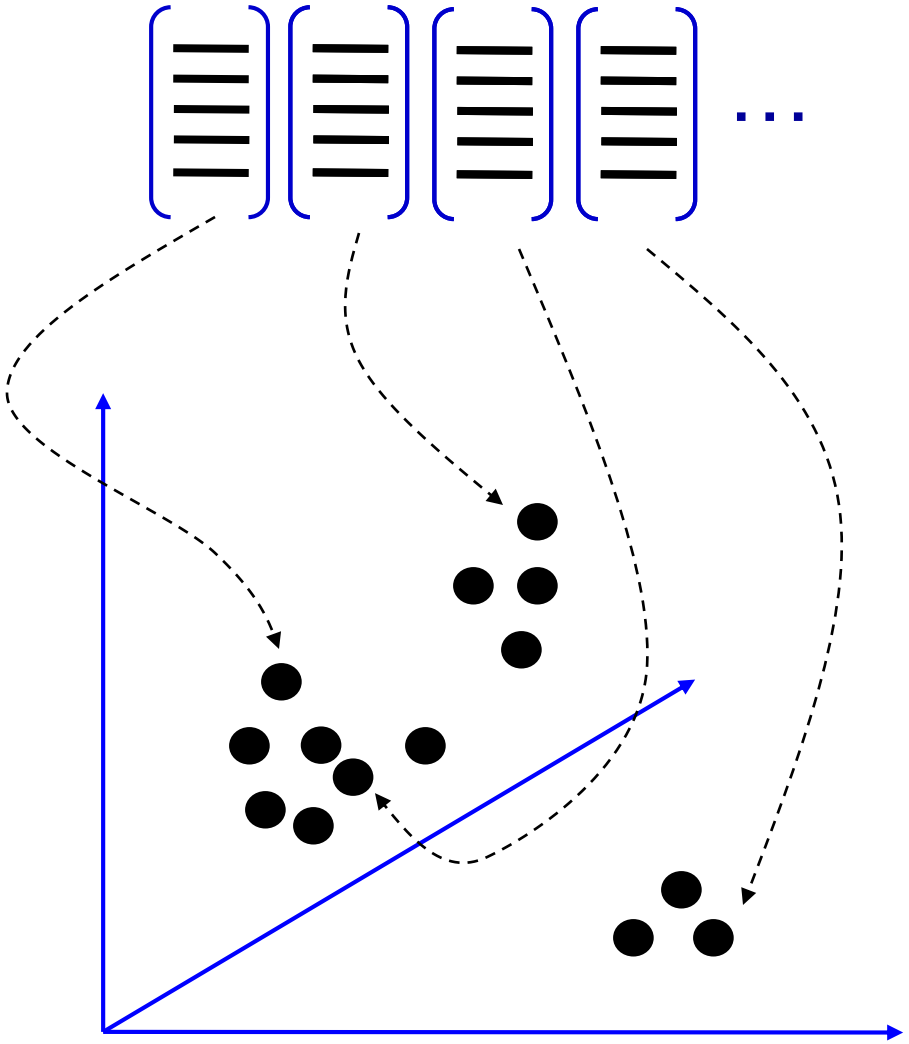
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005



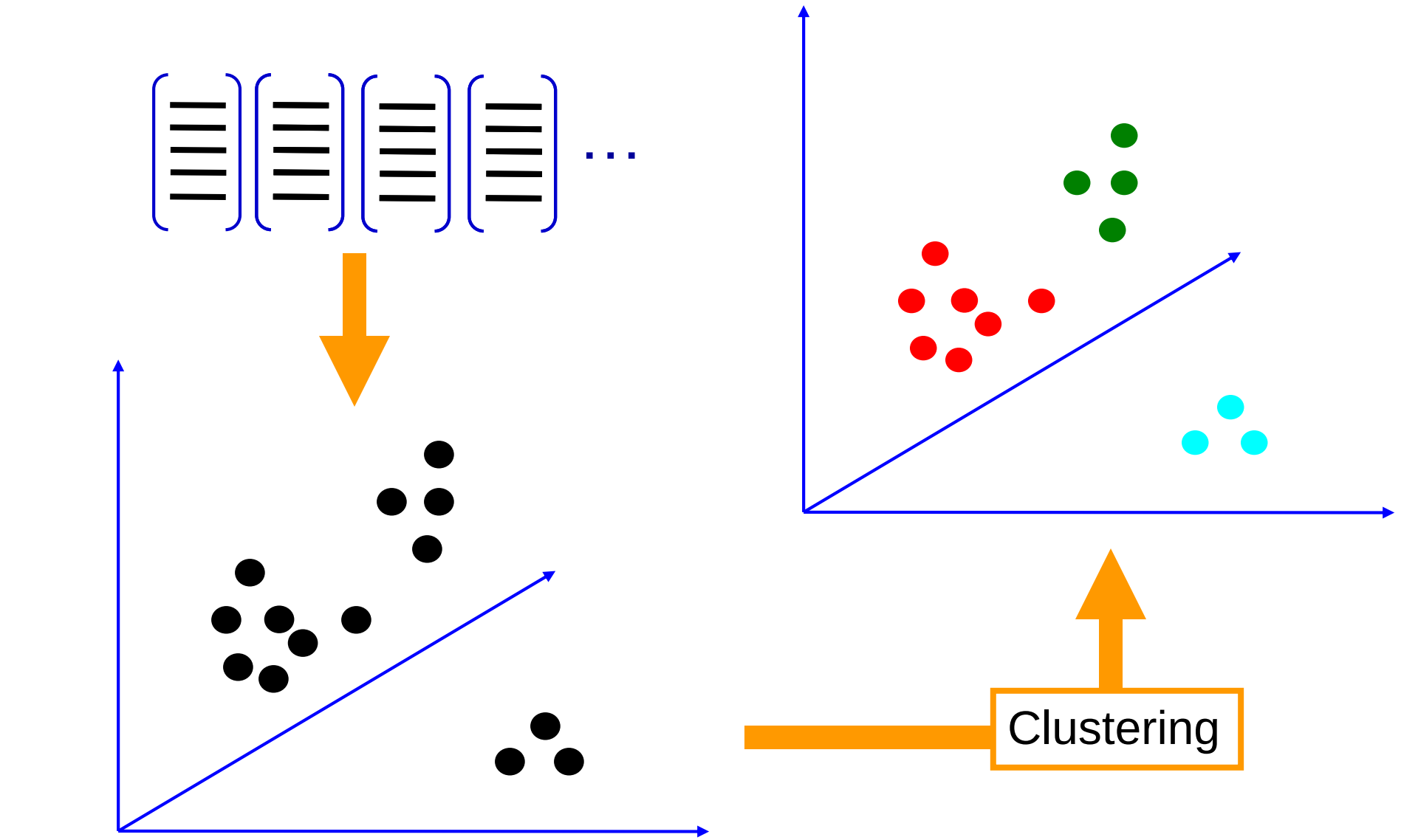
# 1. Feature extraction

- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005
- Other methods
  - Random sampling (Vidal-Naquet & Ullman, 2002)
  - Segmentation-based patches (Barnard et al. 2003)

# 2. Learning the visual vocabulary

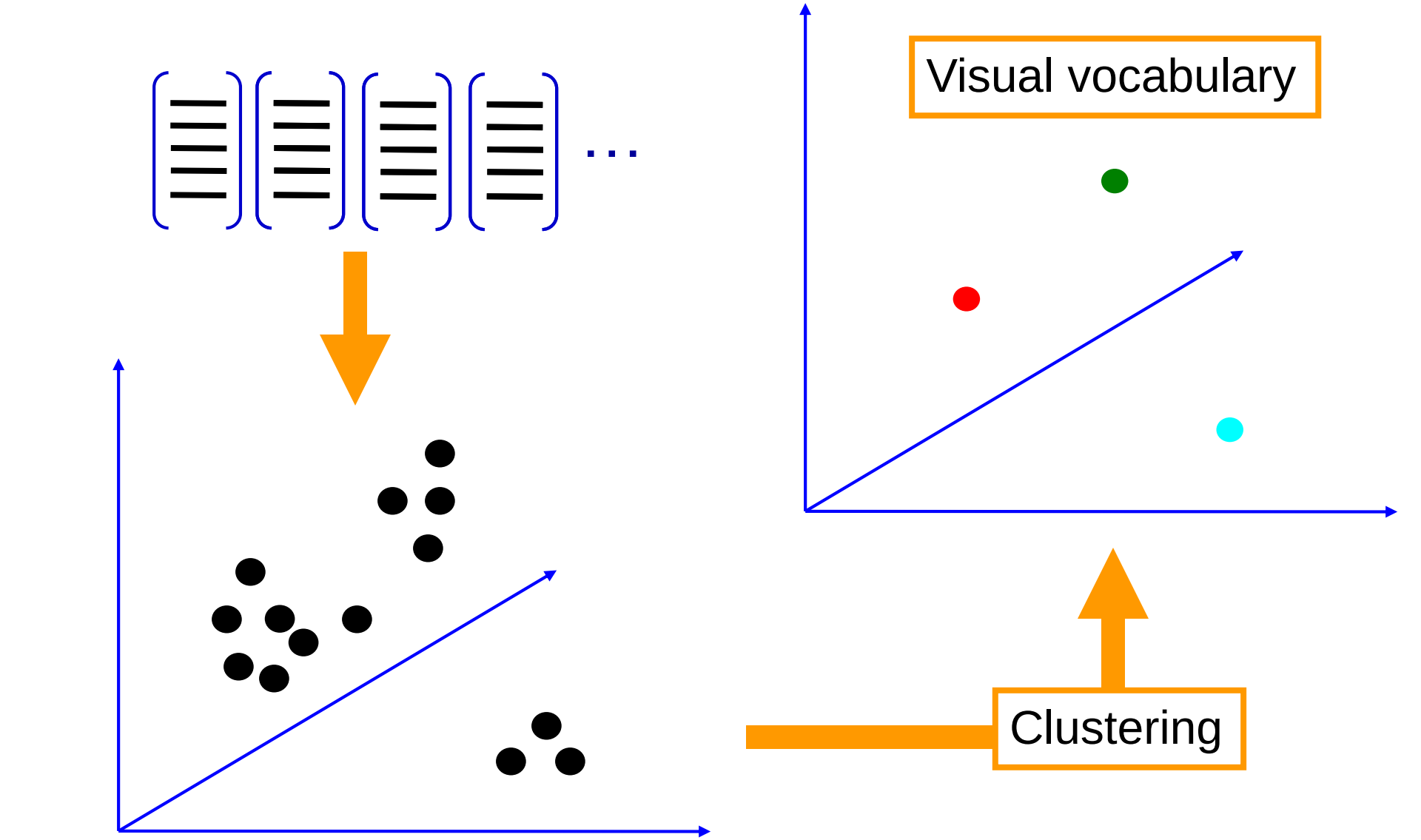


# 2. Learning the visual vocabulary




Slide credit: Josef Sivic

# 2. Learning the visual vocabulary




Slide credit: Josef Sivic

# K-means clustering recap

- Want to minimize sum of squared Euclidean distances between points $x_i$ and their nearest cluster centers $m_k$

$$D(X, M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (x_i - m_k)^2$$

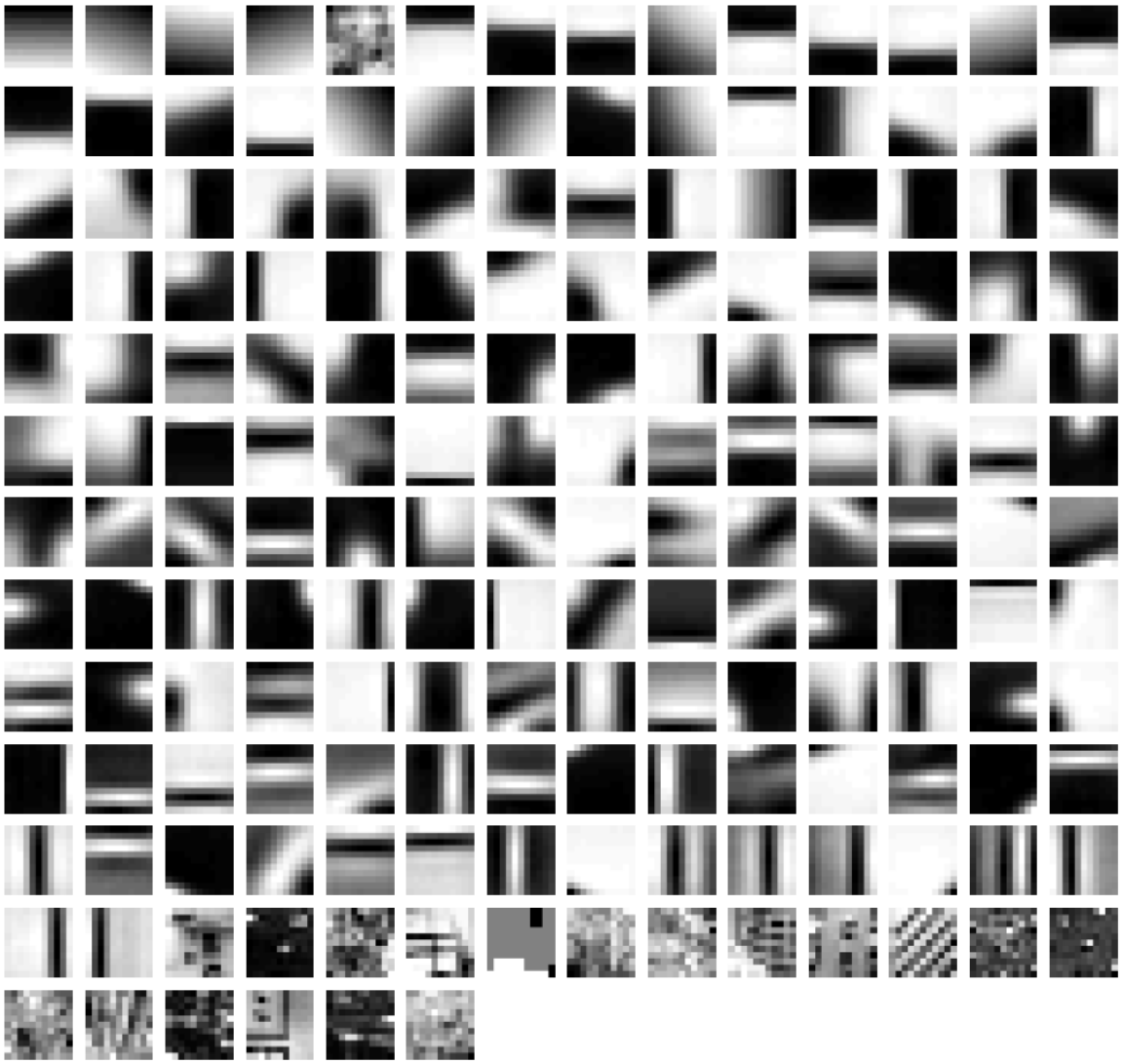
- Algorithm:
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

# From clustering to vector quantization

- Clustering is a common method for learning a visual vocabulary or codebook
  - Unsupervised learning process
  - Each cluster center produced by k-means becomes a codevector
  - Codebook can be learned on separate training set
  - Provided the training set is sufficiently representative, the codebook will be “universal”

- The codebook is used for quantizing features
  - A *vector quantizer* takes a feature vector and maps it to the index of the nearest codevector in a codebook
  - Codebook = visual vocabulary
  - Codevector = visual word

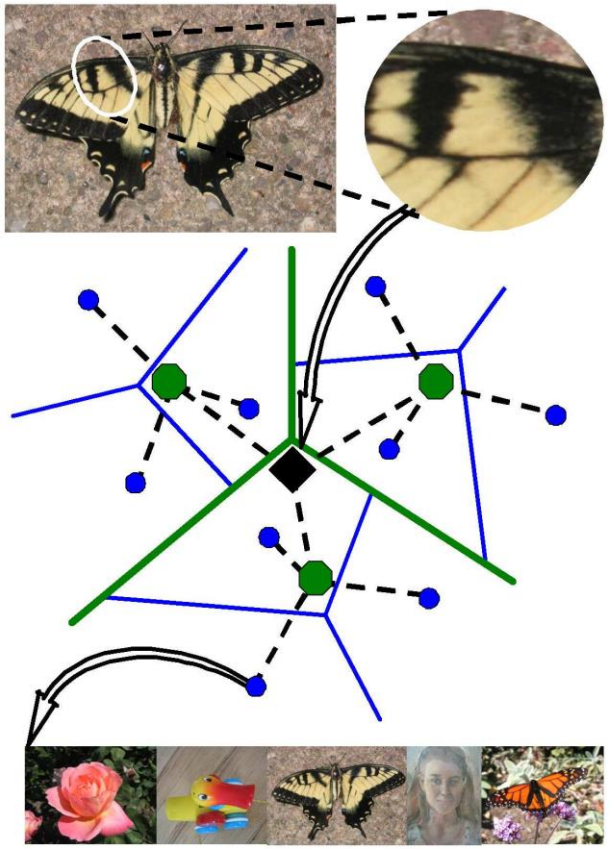
# Example visual vocabulary



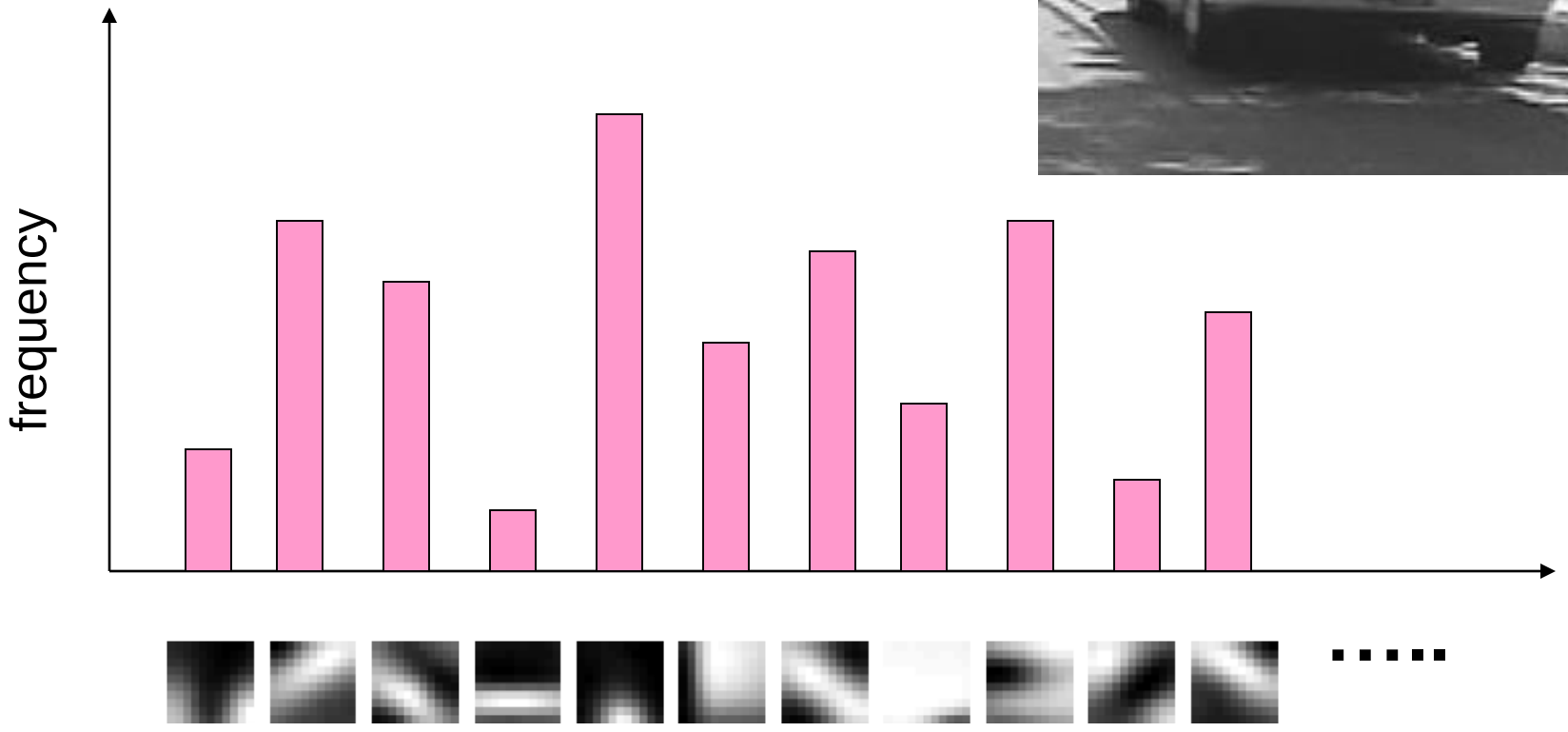
Fei-Fei et al. 2005

# Visual vocabularies: Issues

- How to choose vocabulary size?
  - Too small: visual words not representative of all patches
  - Too large: quantization artifacts, overfitting

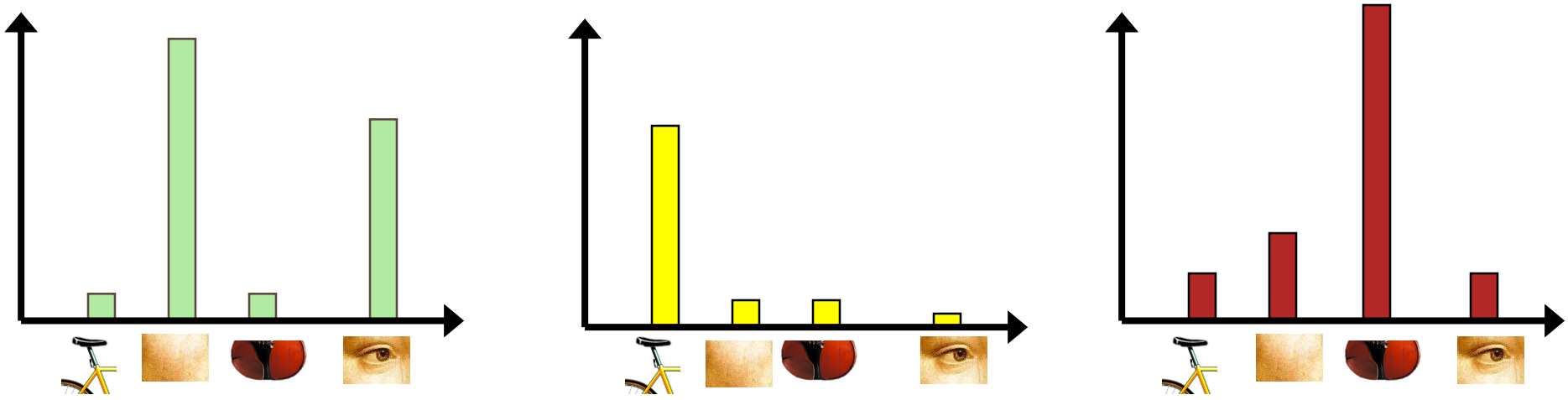


# 3. Image representation



# Image classification

- Given the bag-of-features representations of images from different classes, how do we learn a model for distinguishing them?



# Uses of BoW representation

- Treat as feature vector for standard classifier
  - e.g k-nearest neighbors, support vector machine

- Cluster BoW vectors over image collection
  - Discover visual themes

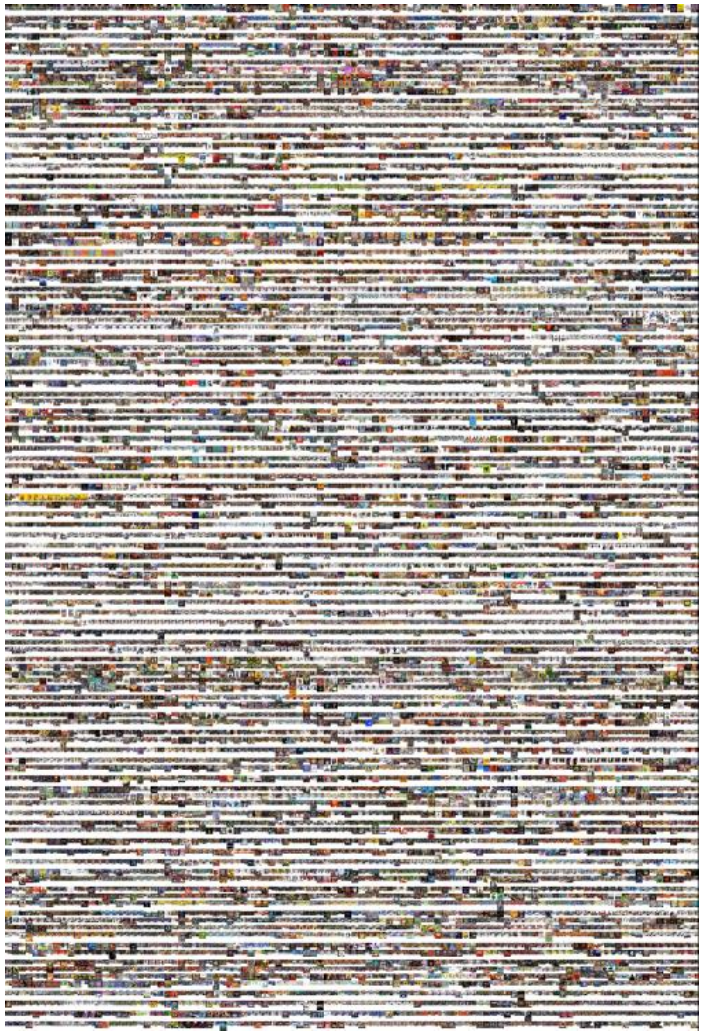
# Large-scale image matching

11,400 images of game covers
(Caltech games dataset)

- Bag-of-words models have been useful in matching an image to a large database of object *instances*


how do I find this image in the database?

# Large-scale image search



Build the database:

- Extract features from the database images
- Learn a vocabulary using k-means (typical k: 100,000)
- Compute *weights* for each word
- Create an inverted file mapping words → images

# Large-scale image search

query image

top 6 results

- Cons:
  - performance degrades as the database grows

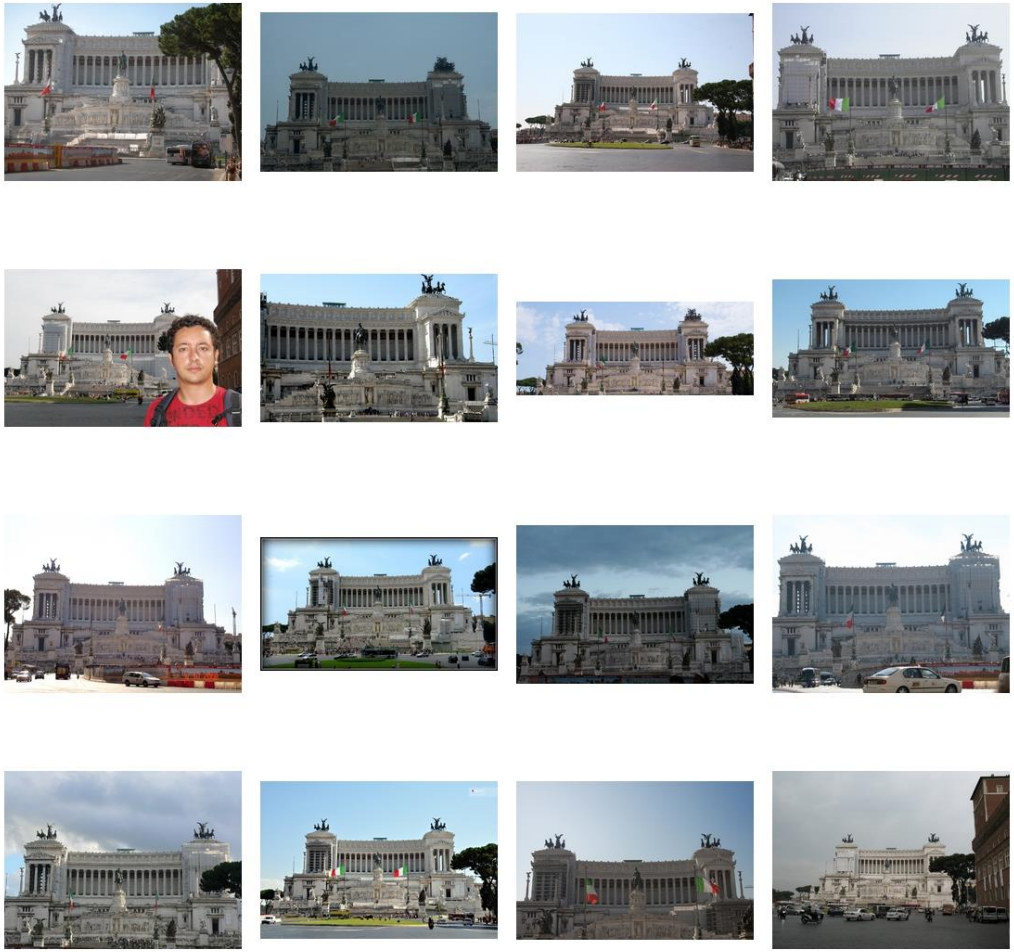
# Large-scale image search

- Pros:
  - Works well for CD covers, movie posters
  - Real-time performance possible

real-time retrieval from a database of 40,000 CD covers

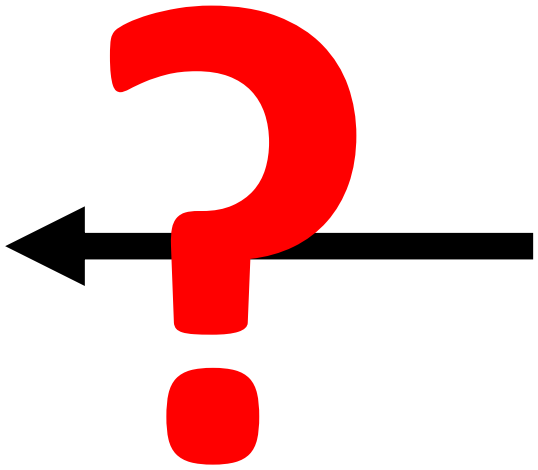
Nister & Stewenius, **Scalable Recognition with a Vocabulary Tree**

# Example bag-of-words matches



# Example bag-of-words matches

# What about spatial info?



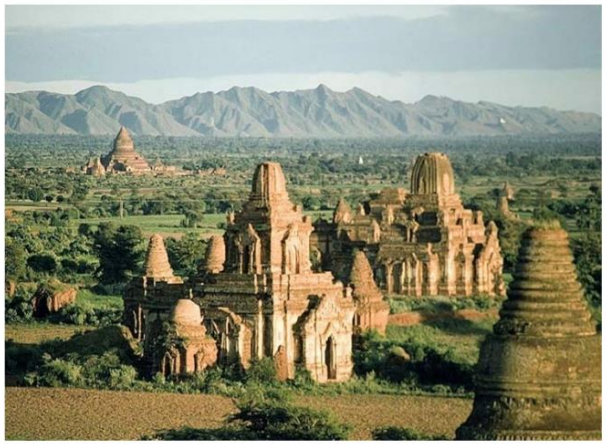
# What we will learn today

- Visual bag of words (BoW)
- Spatial Pyramid Matching

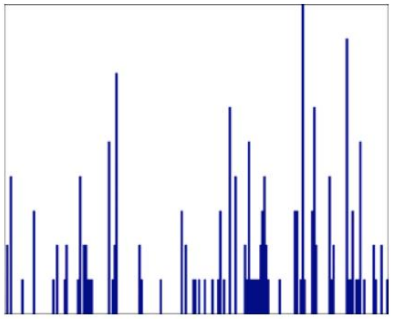
# Pyramids

- Very useful for representing images.
- Pyramid is built by using multiple copies of image.
- Each level in the pyramid is 1/4 of the size of previous level.
- The lowest level is of the highest resolution.
- The highest level is of the lowest resolution.

# Bag of words + pyramids

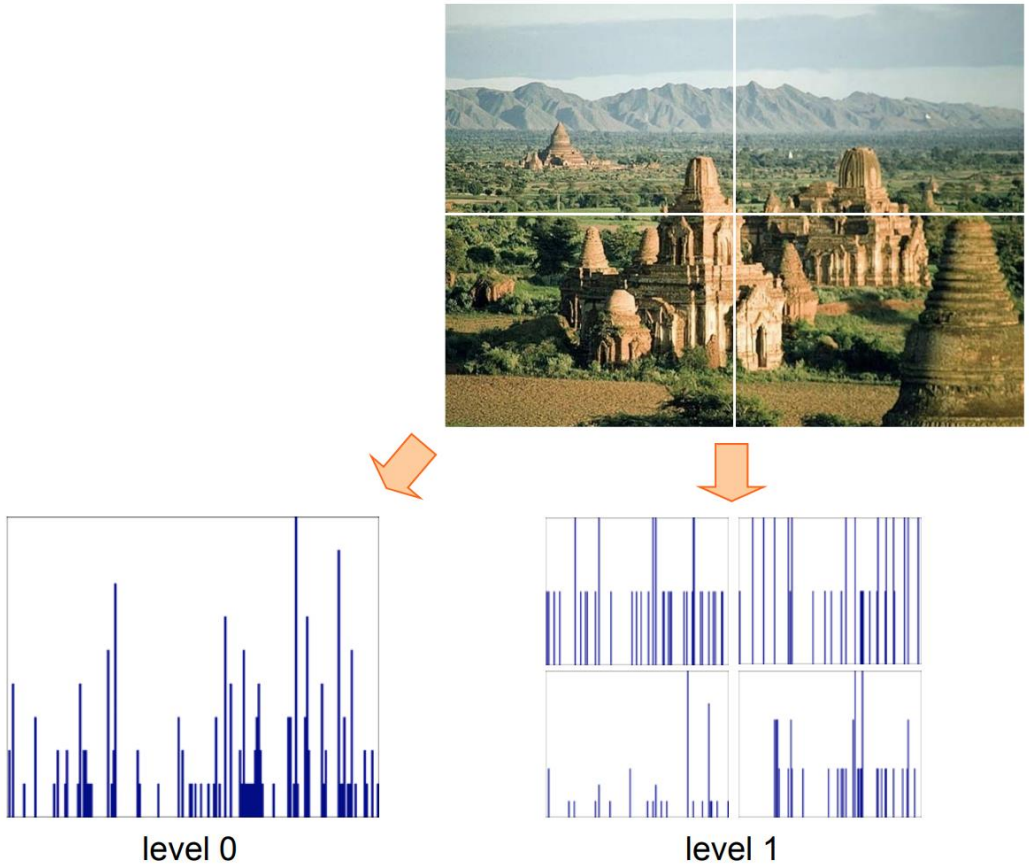


Locally orderless representation at several levels of spatial resolution



level 0

# Bag of words + pyramids




Locally orderless representation at several levels of spatial resolution

# Bag of words + pyramids