Feature Extraction

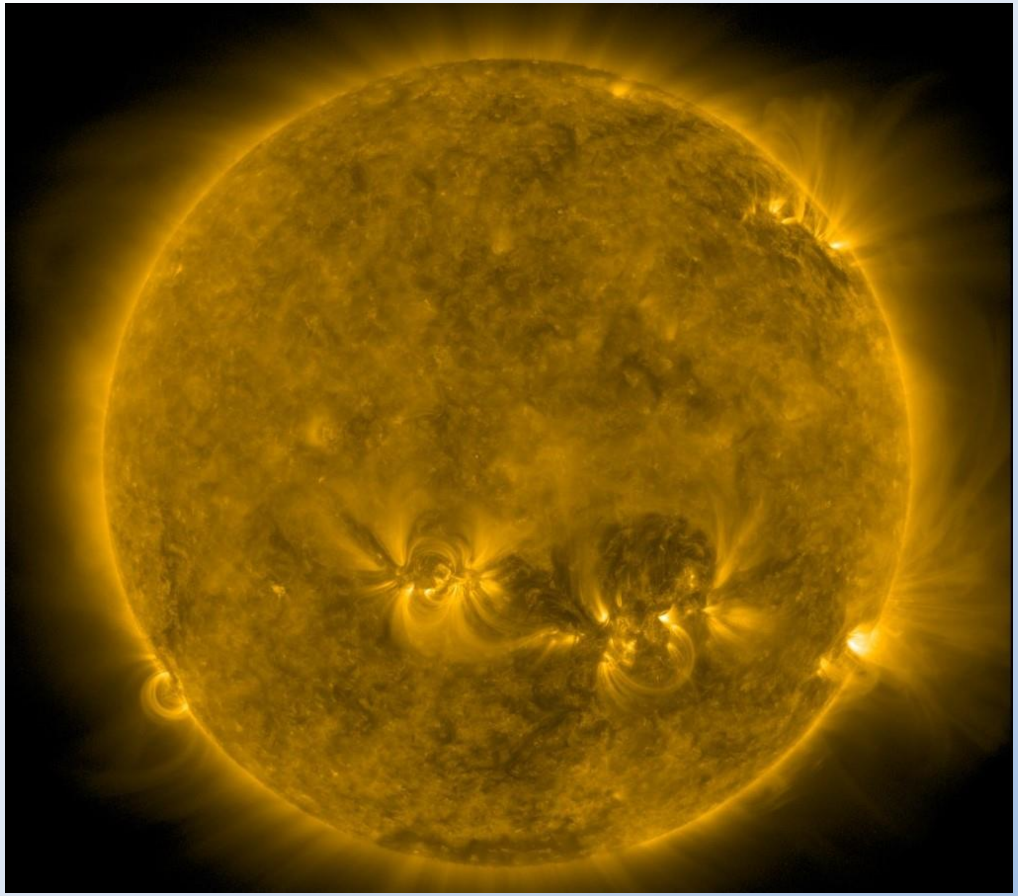
Computer vision tasks:

- Alignment
- Classification
- Recognition
- Detection
- Segmentation
- Prediction
- ...

For all these tasks, we need to determine the similarity among two images or image regions

Image similarity

Are these two images similar?



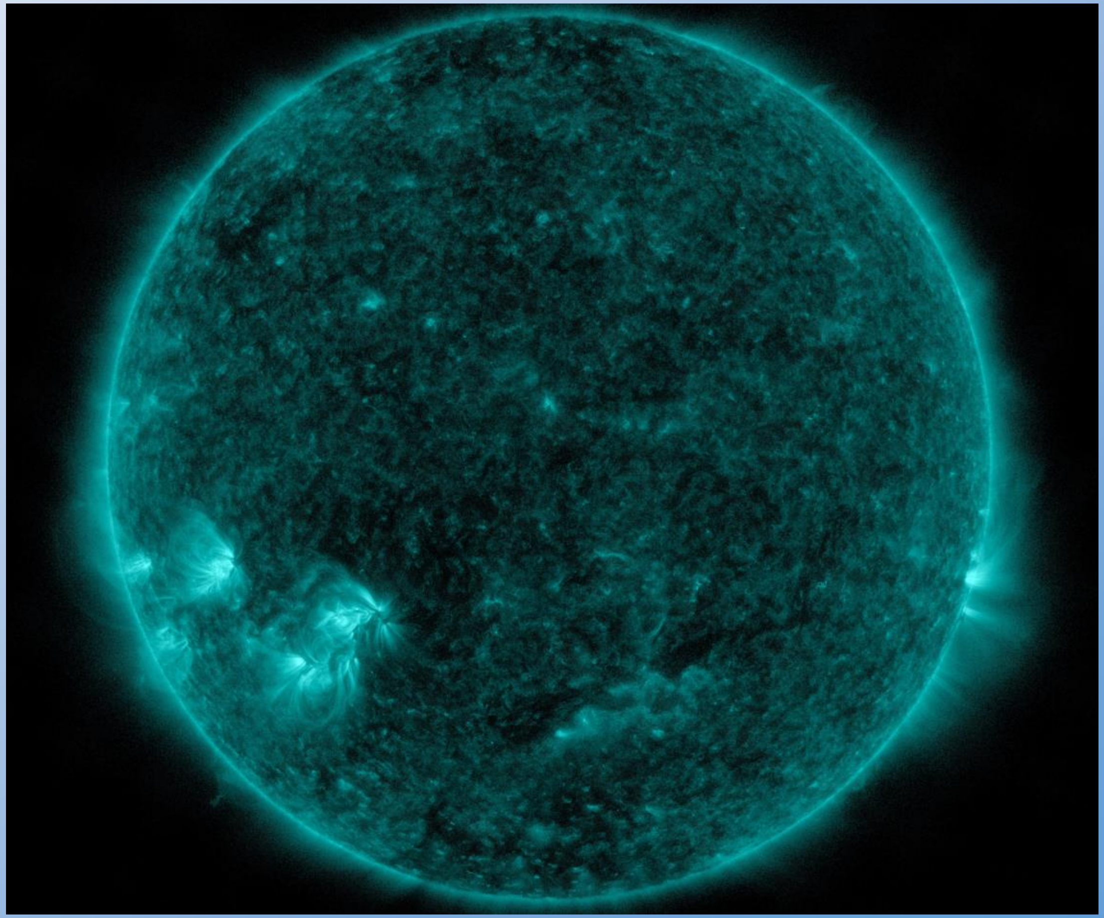


Image similarity

Are these two images similar?

Image similarity

Are these two images similar?

Image similarity

Are these two images similar?

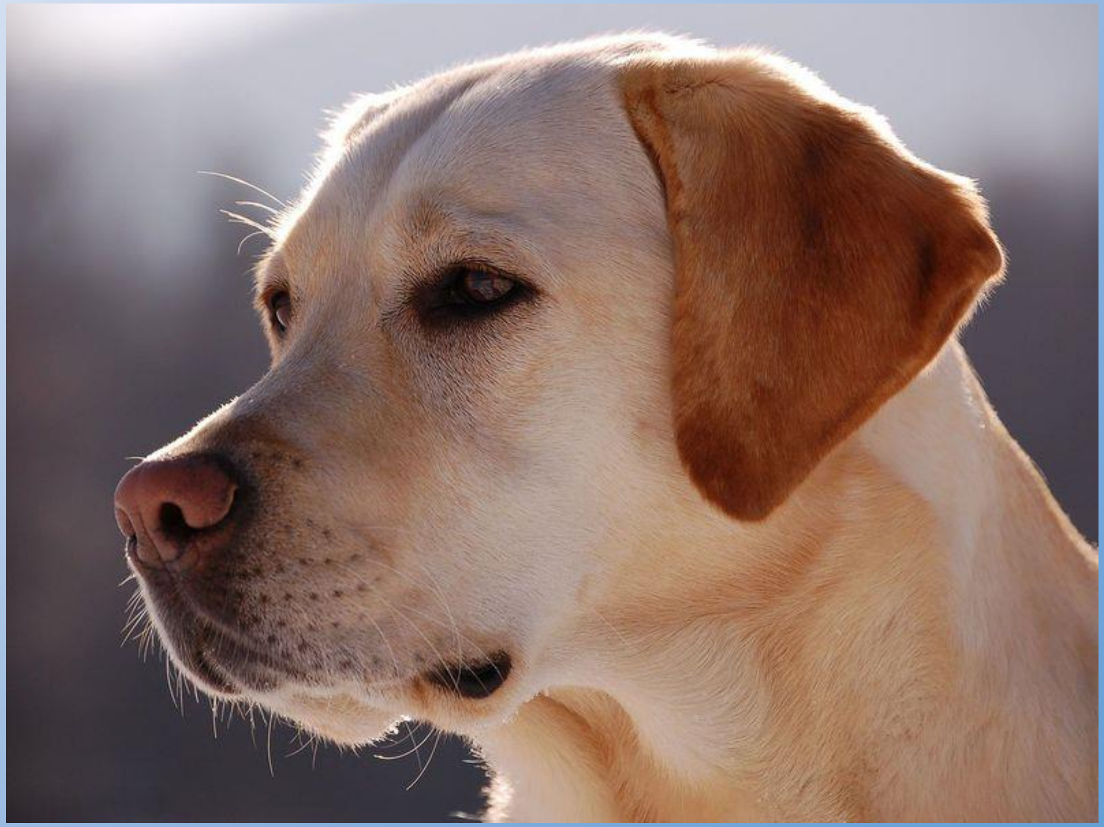


Image similarity

Are these two images similar?

Ideal Features:

- Ideally, we would like features to encode the concept of similarity as a human would evaluate it.
- Of course, this is very difficult.
- The choice of features is application dependent, but in general we want independence to:
    - Intensity transformations
    - Geometric transformations
    - Image cropping

Ideal Features:

- Depending on application, we may want features that either emphasize or deemphasize the following:
  - Texture
  - Lighting
  - Color
  - Shape
  - Higher-level information

Some Feature Sets

- Pixel intensities
- PCA
- Viola-Jones features
- Color histograms
- Histograms of gradients
- SIFT
- Visual words

Pixel intensities

- An m-by-n image is viewed as an object with *m*n* or *m*n*3* features
- Advantages:
    - Simplicity, features are directly available
- Disadvantages:
    - Only images of the same size can be compared (we need to crop or reshape)
    - High dimensional feature space – millions of features to describe a medium-sized image
    - Lack of generalization

Principal Component Analysis

Idea:

- View each image as a point in n dimensions (where n is the number of pixels in the image)
- Project data from n-dimensions (attributes) to m-dimensions (with n>m) while preserving as much information as possible

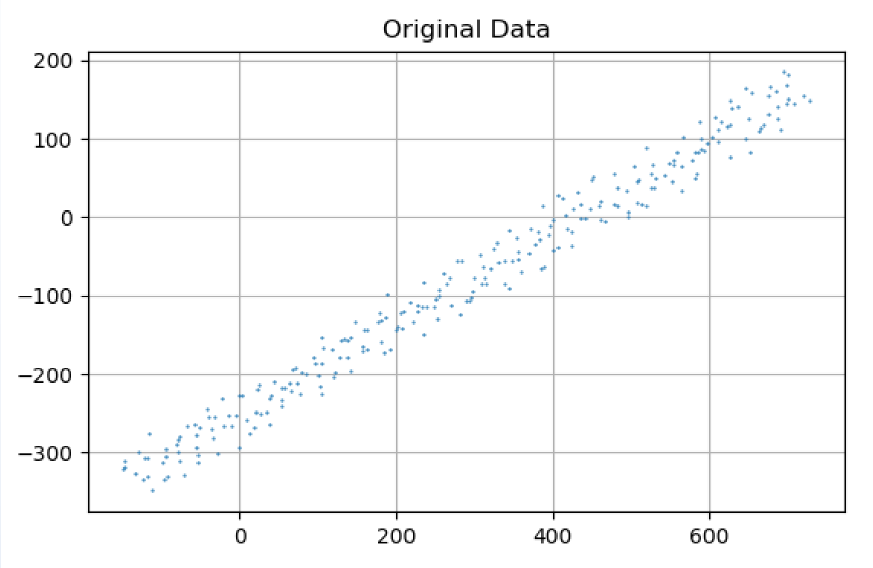
Why: Features tend to be correlated – reduce redundancy

Why: Least important features in projected space approximate noise

Mathematically, the principal components are the eigenvectors of the covariance matrix of the image set

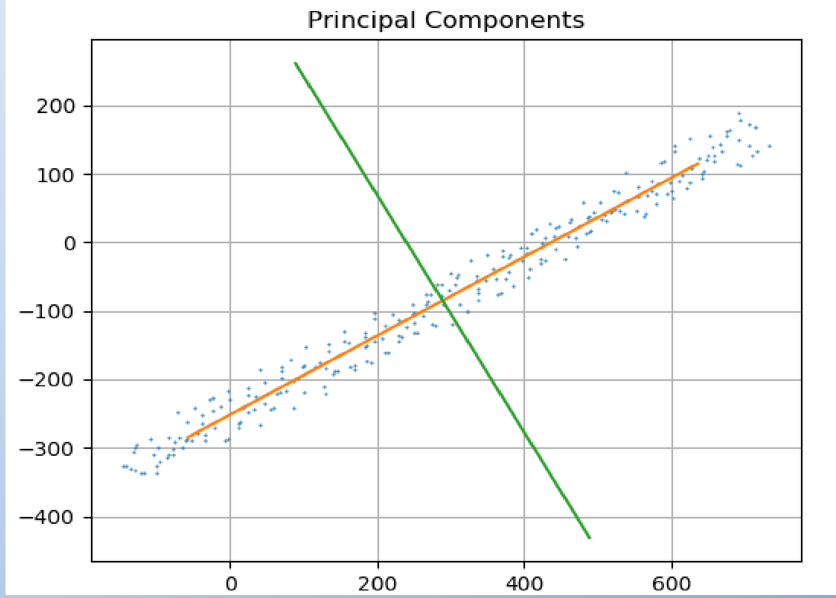
Principal Component Analysis

The principal components of a dataset X are the vectors $v_0, \ldots, v_n$
such that v_i is the vector that best fits X and is perpendicular to each of $v_0,\ldots,v_{i-1}$



X

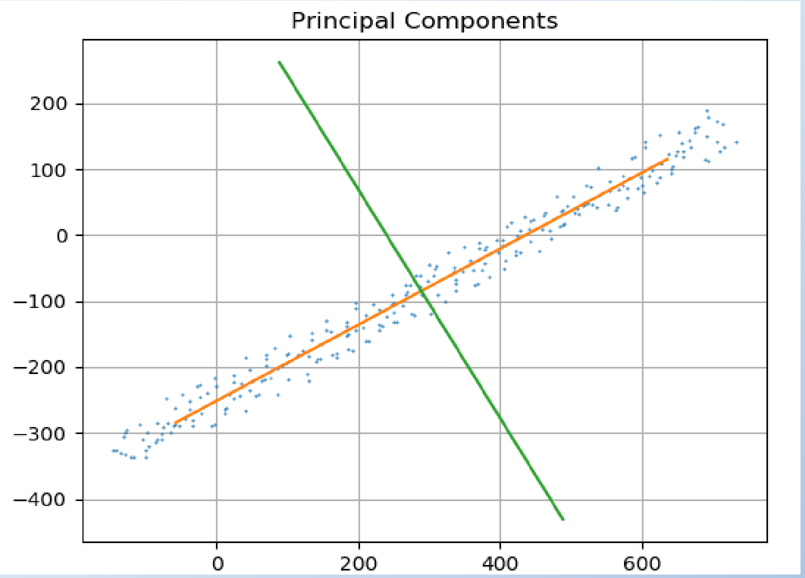
PCA



v_0 – orange, v_1 - green

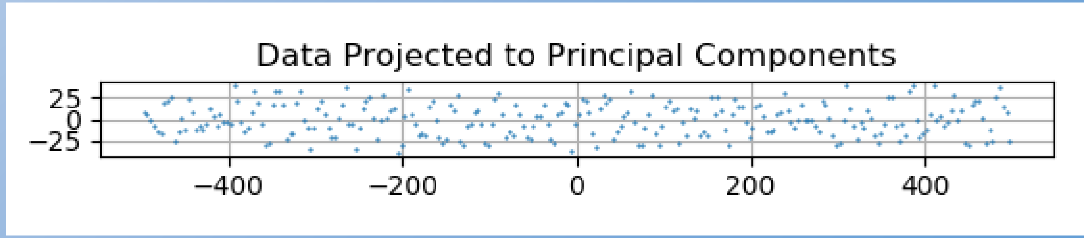
Principal Component Analysis

The principal components of a dataset X are the vectors $v_0, \ldots, v_n$
such that v_i is the vector that best fits X and is perpendicular to each of $v_0, \ldots, v_{i-1}$



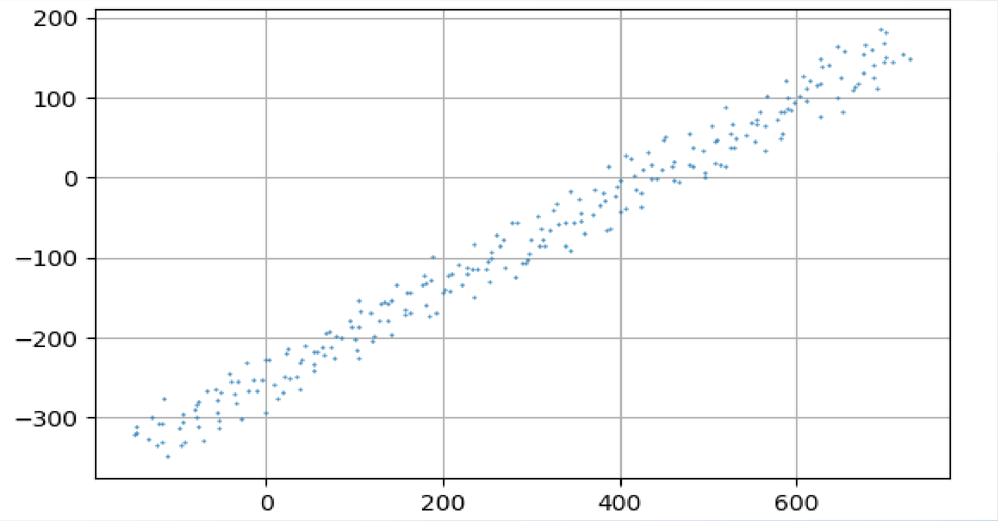
X in original feature space and principal components

Projection

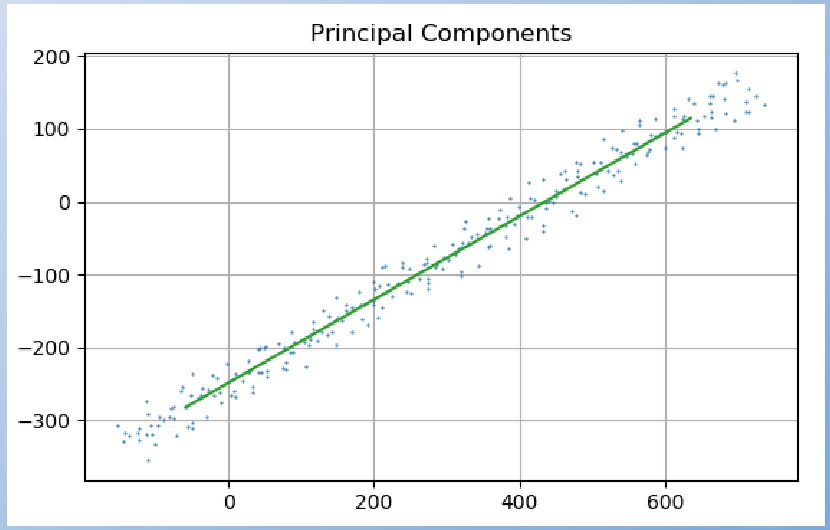


X in new feature space
Features with smallest variance can be discarded

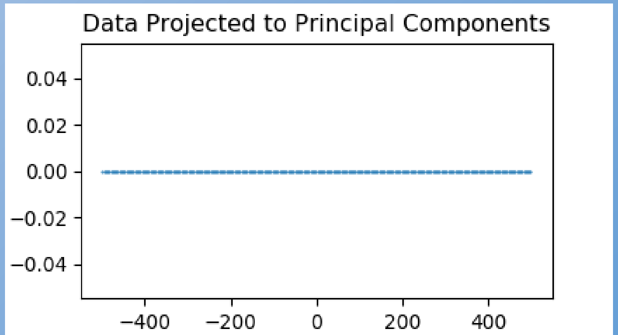
Principal Component Analysis



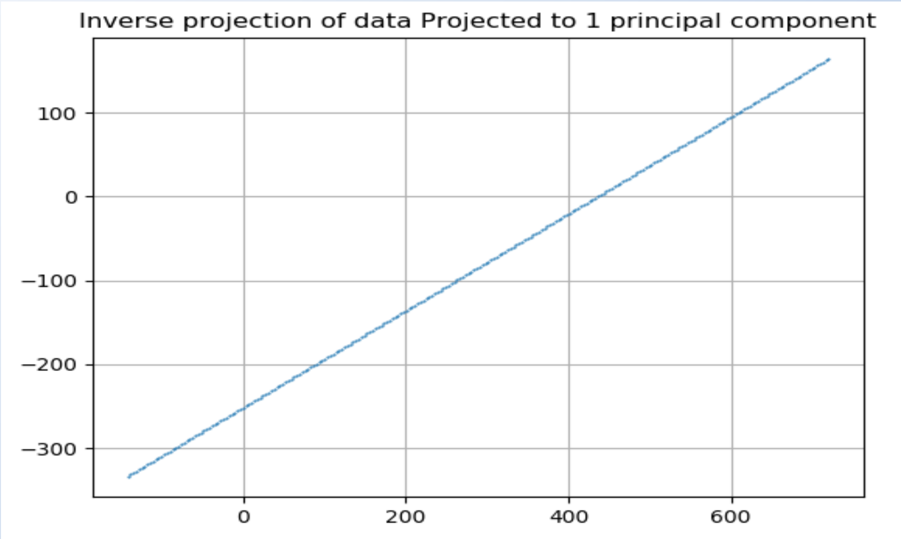
pca.fit(X,n_compone
nts=1)



P = pca.transform(X)



Xf =
pca.inverse_transform(P)



Principal Component Analysis

Let $I = \{I_1,..,I_n\}$ be the image set, where each images is reshaped into a vector

Find the set of eigenvectors of the covariance matrix of I $\{E_1,...,E_m\}$ (with $m<<n$)

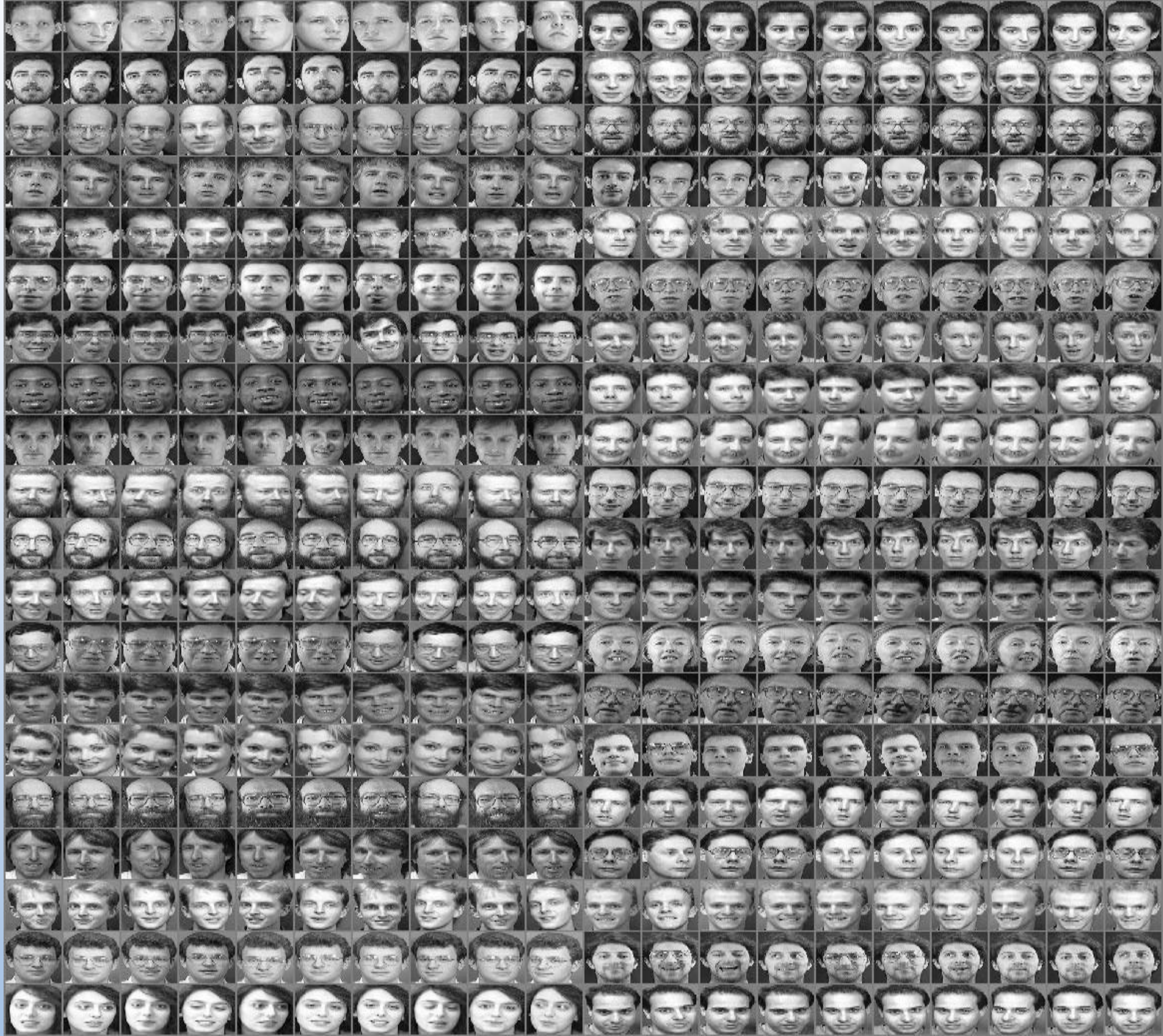
Then each image can be described as:

$$I_j = c_1*E_1 + c_2*E_2 + ... + c_m*E_m + \varepsilon$$

The we use the vector $[c_1,c_2,...,c_m]$ as the features of describing I_j

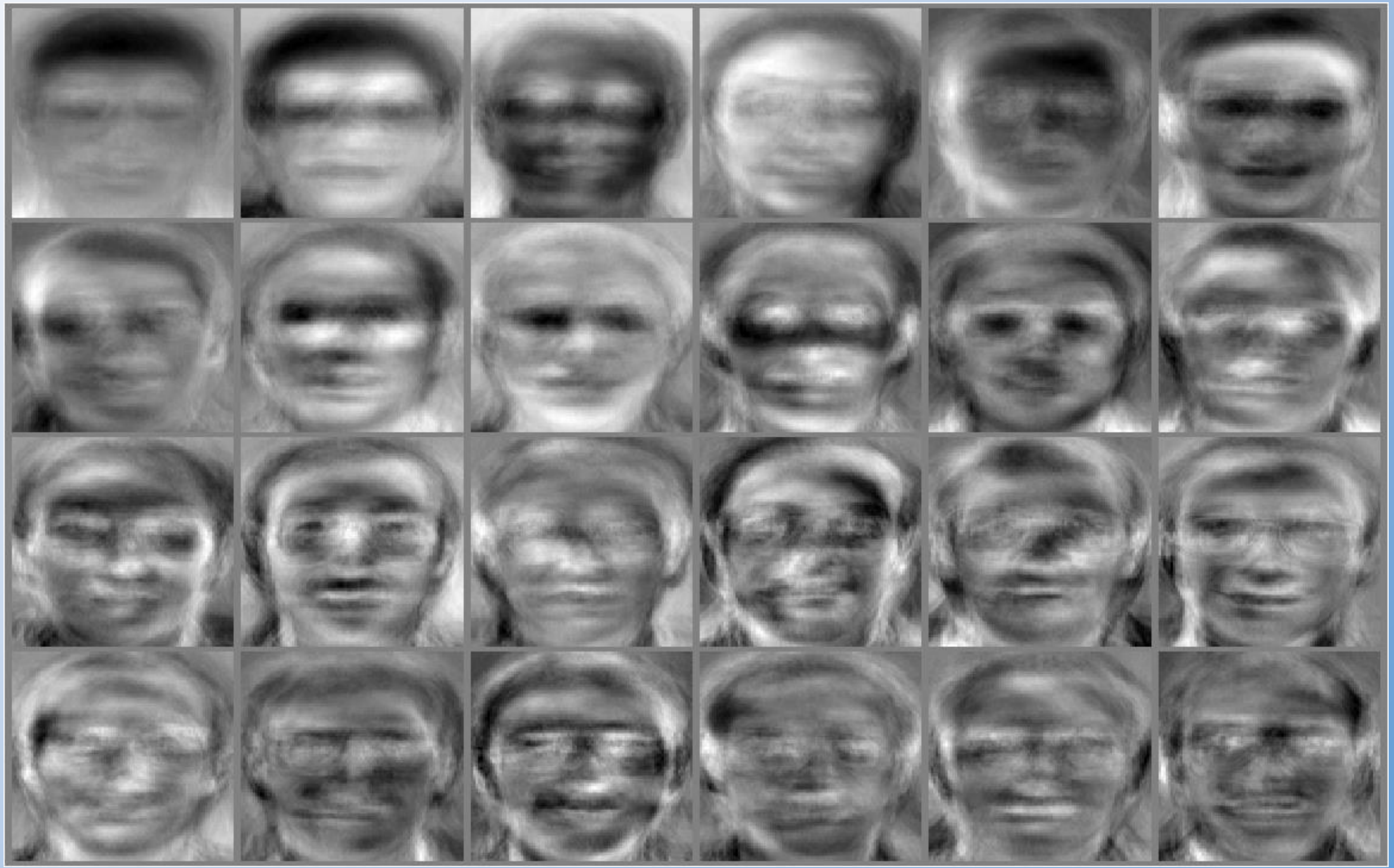
PCA Example: Eigenfaces

Original data



PCA Example: Eigenfaces

Principal components:



Principal Component Analysis

Advantages:

- Global rather than local descriptor
- Representation has well-understood mathematical properties

Disadvantages:

- Computation is time-consuming; unfeasible for large images
- Sensitive to lighting effects

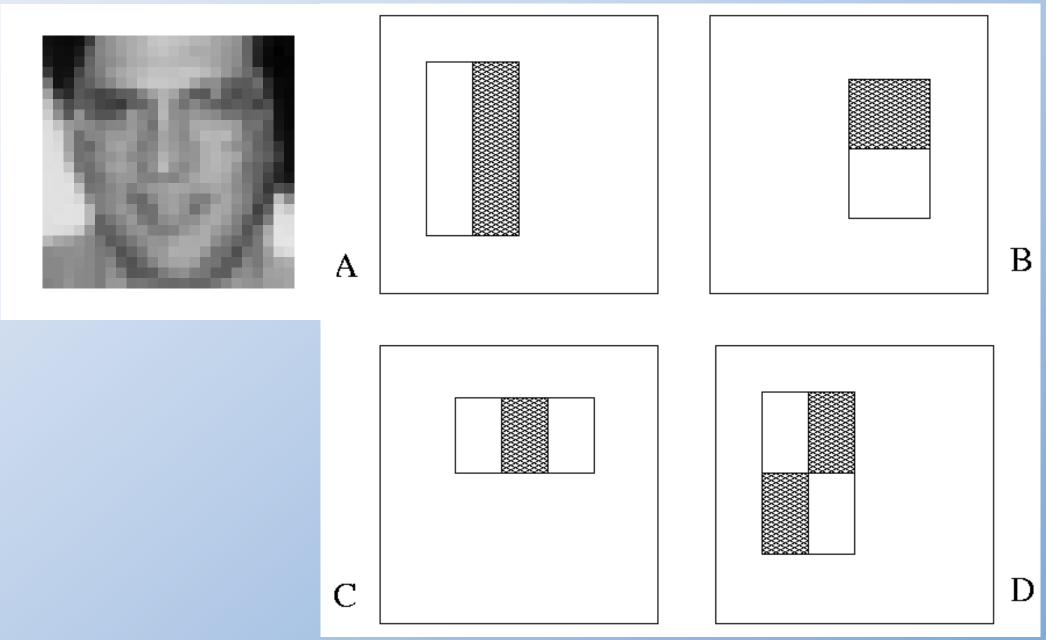
Principal Component Analysis

Final note:

We will give an F in the class to anybody who refers to this algorithm as 'Principle Components'.

Viola-Jones features

"Rectangle filters"



Value =

∑ (pixels in white area) –
∑ (pixels in black area)

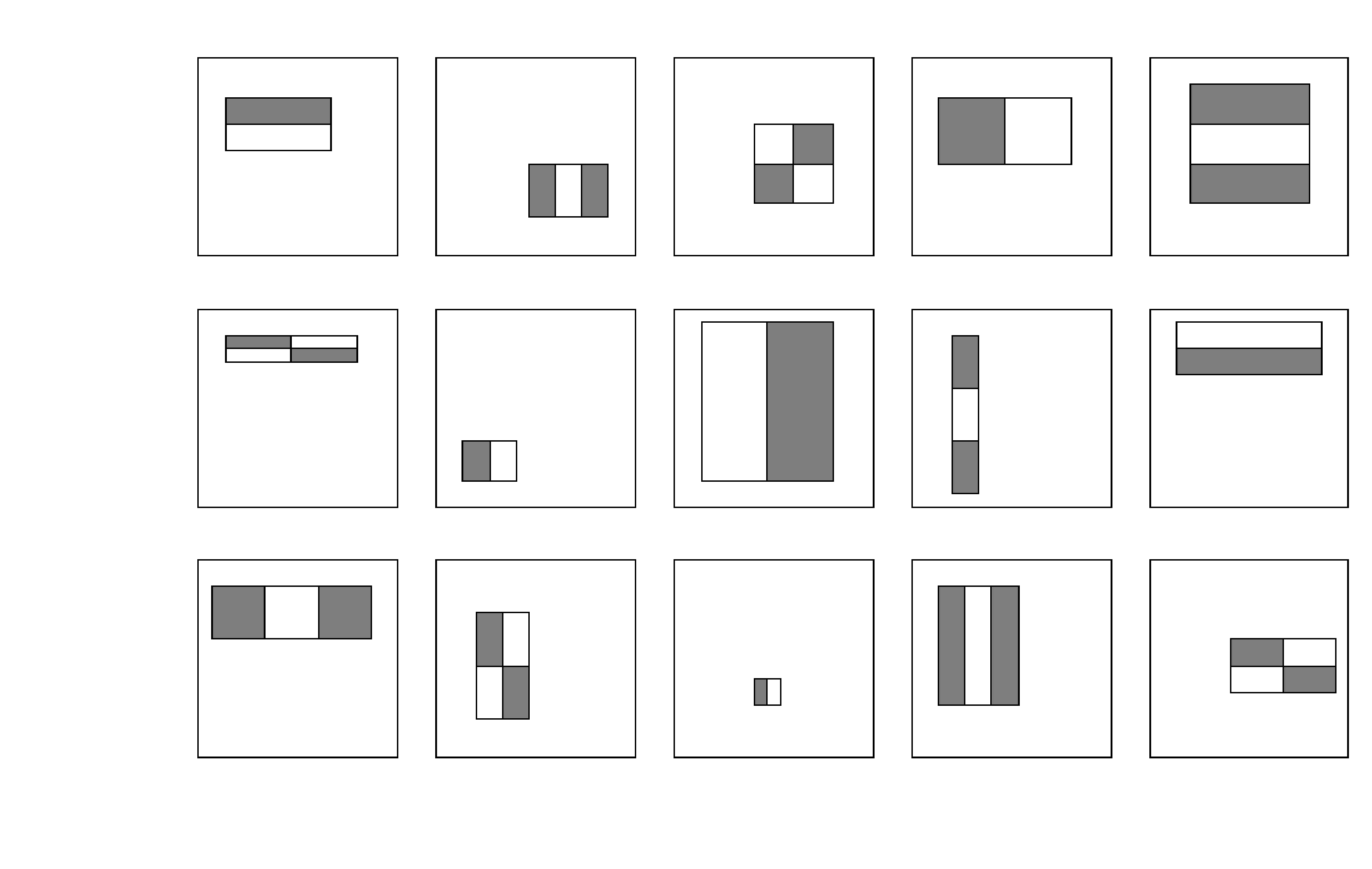
Computed efficiently using the integral image trick

Viola-Jones features

- First successful approach to object detection (Viola and Jones, 2001)
- Designed to work on gray-level images
- Still used today (OpenCV face detector is based on these features)

Viola-Jones features

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!



Viola-Jones features

- For a 24x24 detection region, the number of possible rectangle features is around 160,000
- At test time, it is impractical to evaluate the entire feature set
- We can create a good classifier using just a small subset of all possible features

Viola-Jones features

Advantages:

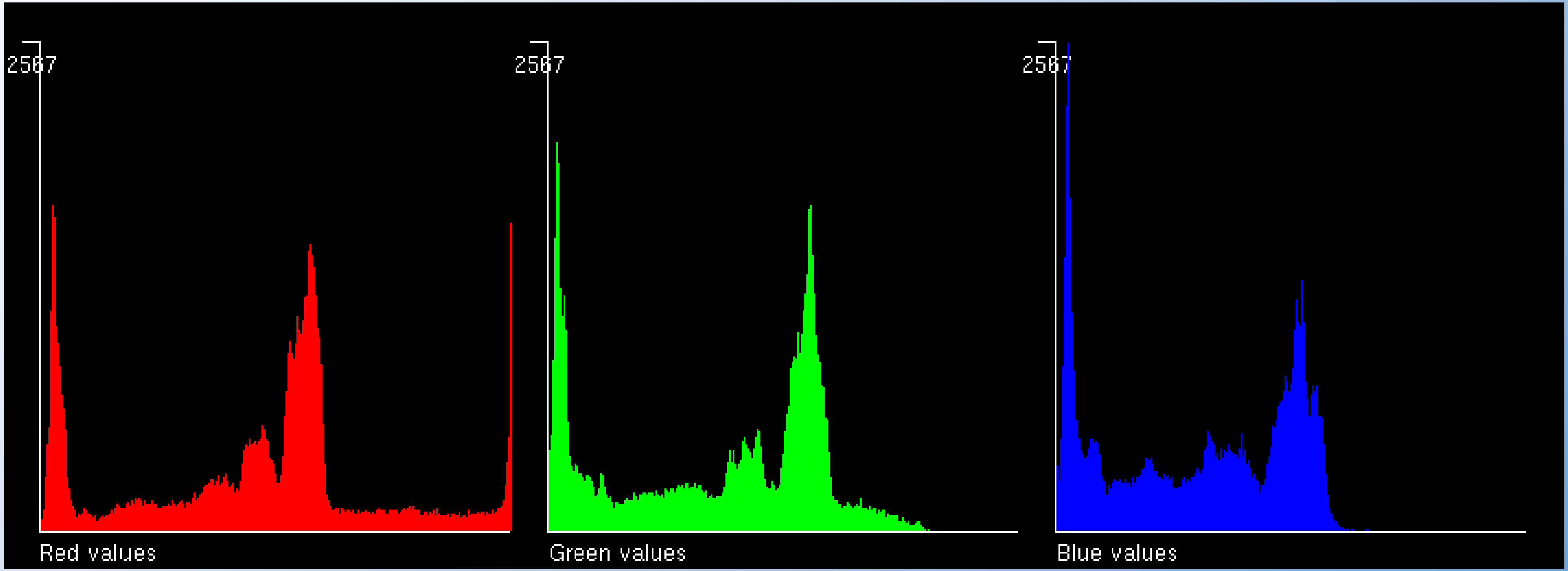
- Can be computed efficiently

Disadvantages:

- Requires a feature selection step, as the number of possible features is huge
- Tailored to grey-level images

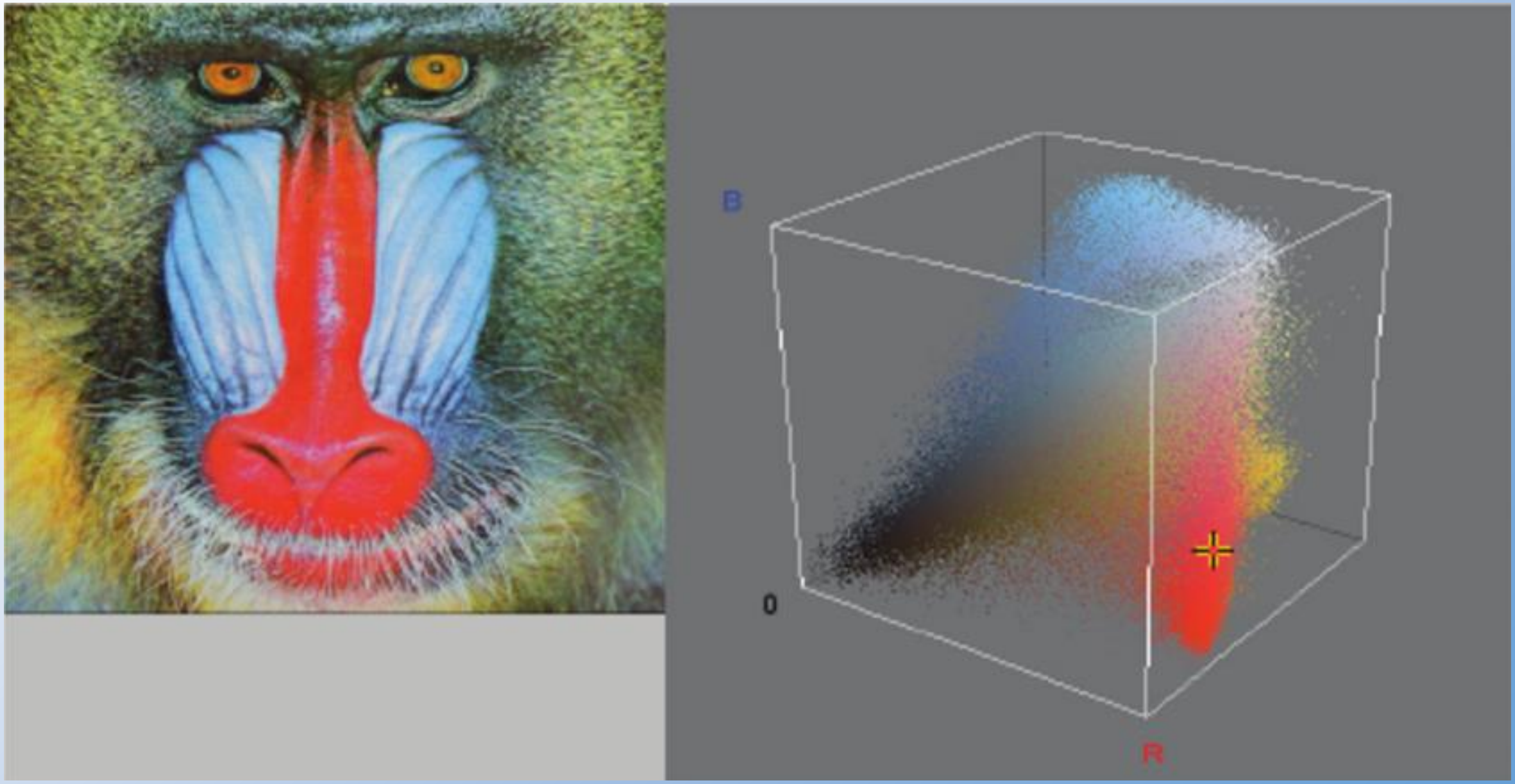
Color Histograms

- Color histograms – Three 1-D descriptors

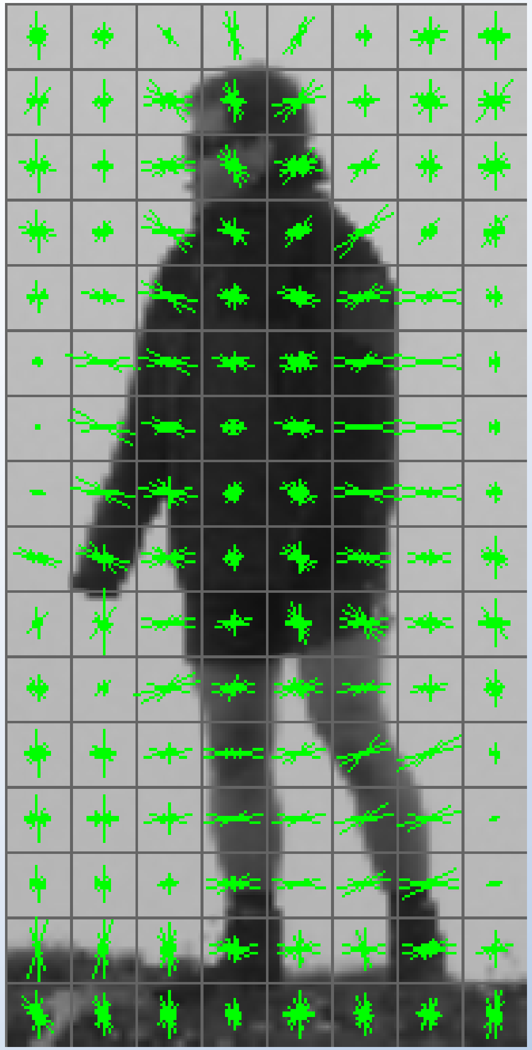


Color Histograms

- Color histograms – 3-D descriptor



Histograms of gradients (HOGs)



Find vertical and horizontal gradients in image g_v and g_h

Compute gradient magnitude

$g_m = \sqrt{(g_v)^2 + (g_h)^2}$

Compute gradient direction

$g_d = \mathrm{atan2}(g_v, g_h)$

Split the [-180,180º] range of possible gradient directions into *n* equally size intervals $[i_0, i_{n-1}]$

For every region, generate a descriptor vector $[h_0, h_{n-1}]$ where:

$h_i = \sum g_m[r,c]$ if $g_d[r,c]$ in i_i for all pixels $[r,c]$ in region

Visual words

Visual words

Extracting visual words:

Let $I = \{I_1,..,I_n\}$ be the image set

Find the set of patches (small subimages) $P = \{p_1,...,p_m\}$ (with $m >> n$) – we may use all patches from I, or select according to SIFT-like interestingness criterion.

Cluster P using k-means

Let $M=\{m_1,...,m_k\}$ be the means obtained by clustering

Describe each patch p' in test image I' by the feature vector

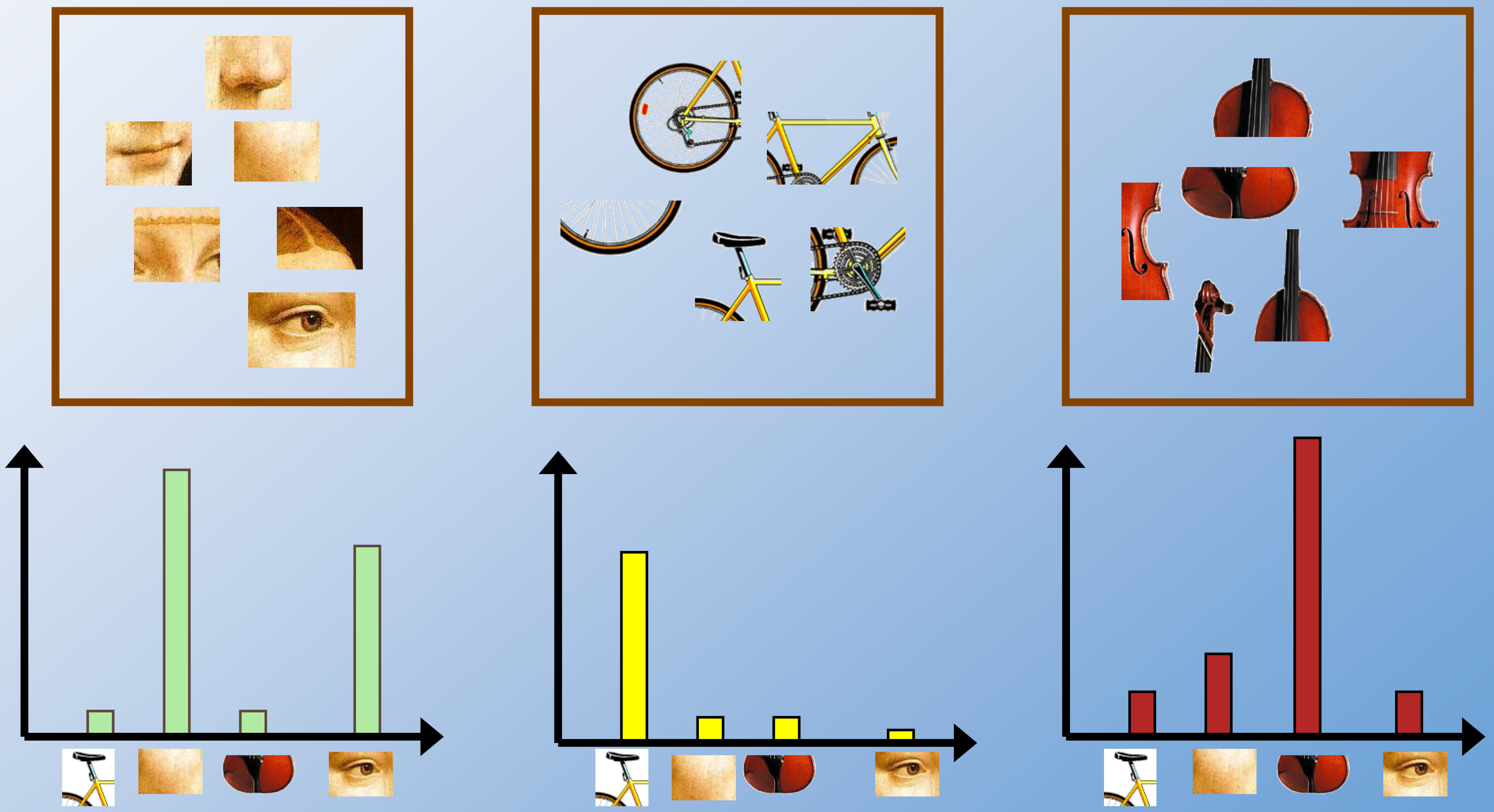
$f(p') = \{1/|m_1-p'|, 1/|m_2-p'|, ..., 1/|m_k-p'|\}$

(thus the descriptor measures similarity to the cluster means)

Describe larger regions by either average or maximum values of f(p')

Visual words

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



K-means Algorithm

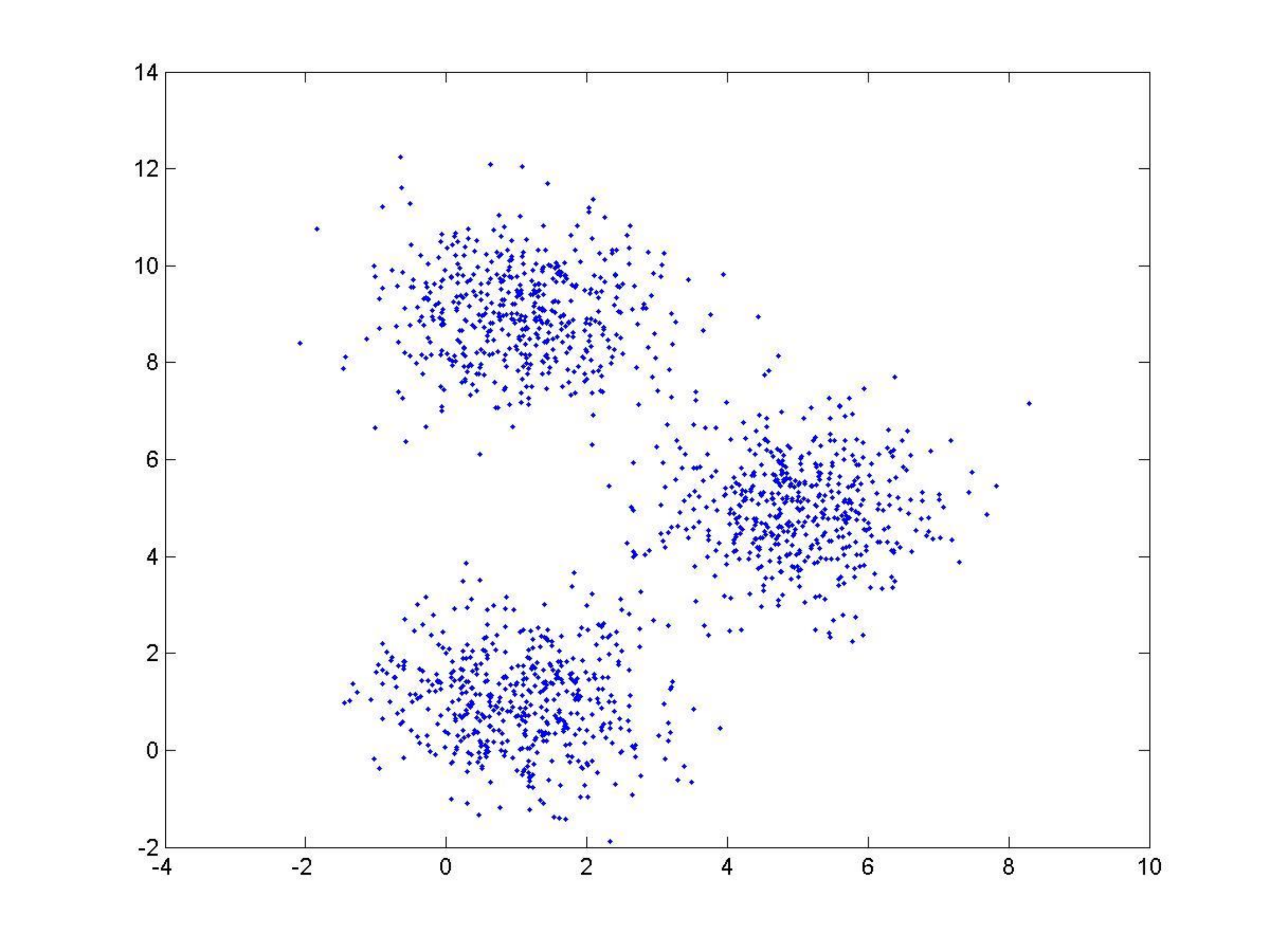
- K-means Algorithm
  - Randomly generate k cluster means m1,...,mk
  - Repeat until convergence
    - Partition data into clusters
      - For each data point x, find the cluster mean that is closest to x (that is, find argmin |x-mi|), assign x to cluster ci
    - Recompute cluster means
      - For i=1 to k, let mi be the mean of cluster ci

K-means Algorithm

- Example. Consider a two-dimensional data set to be partitioned into 3 clusters (that is, k=3)

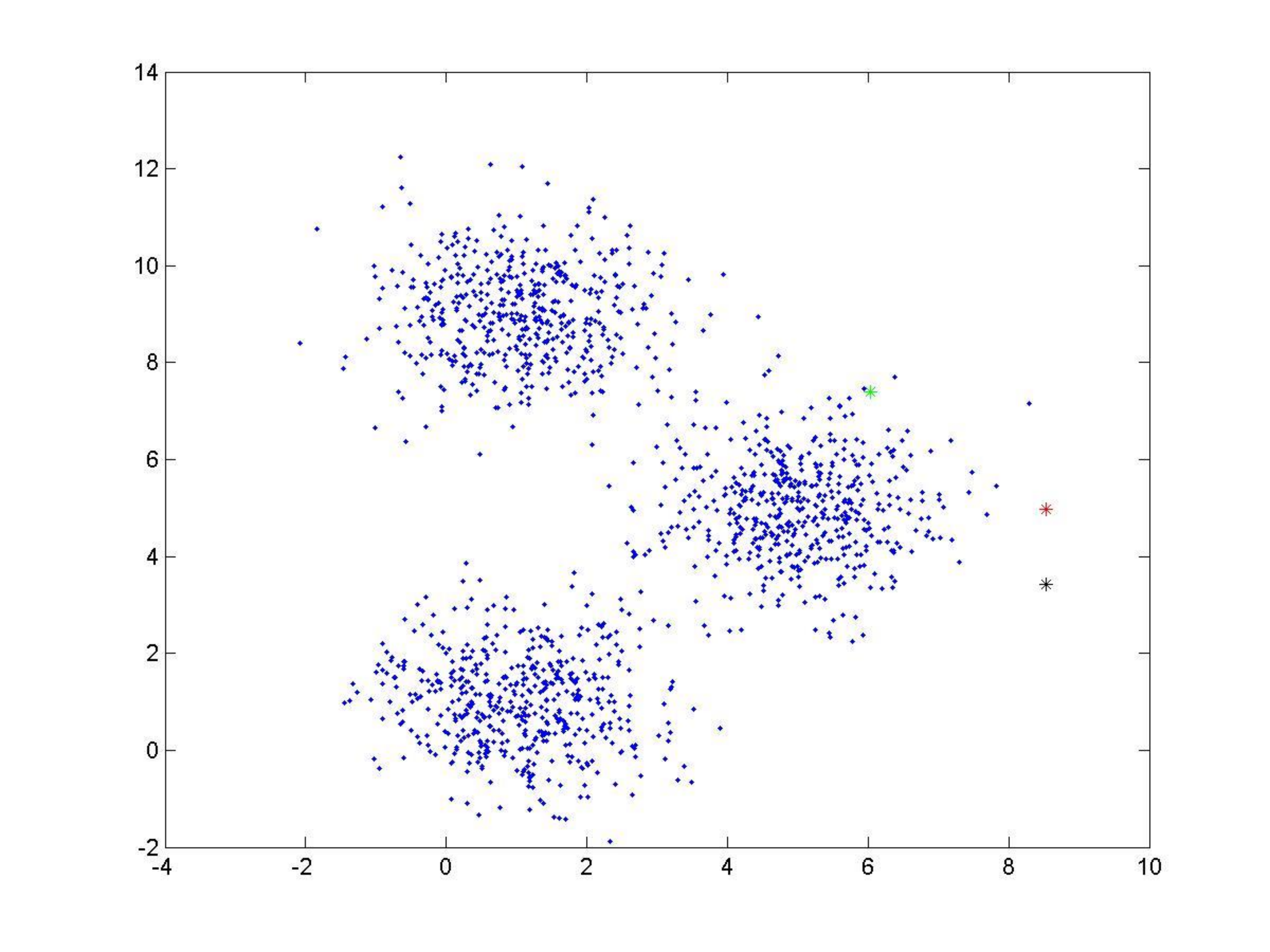
K-means Algorithm

- Original Data



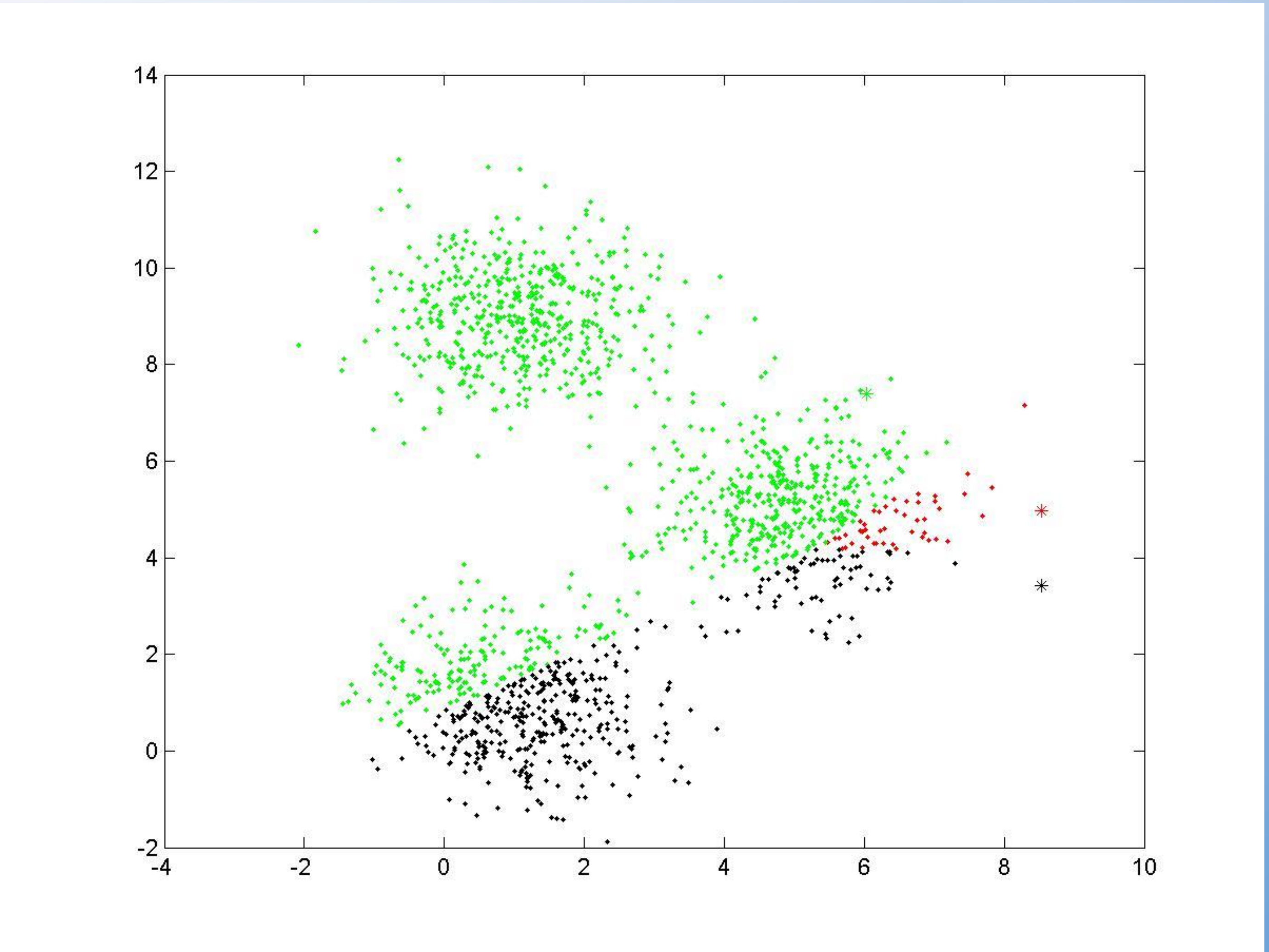
K-means Algorithm

- Original (random) Means



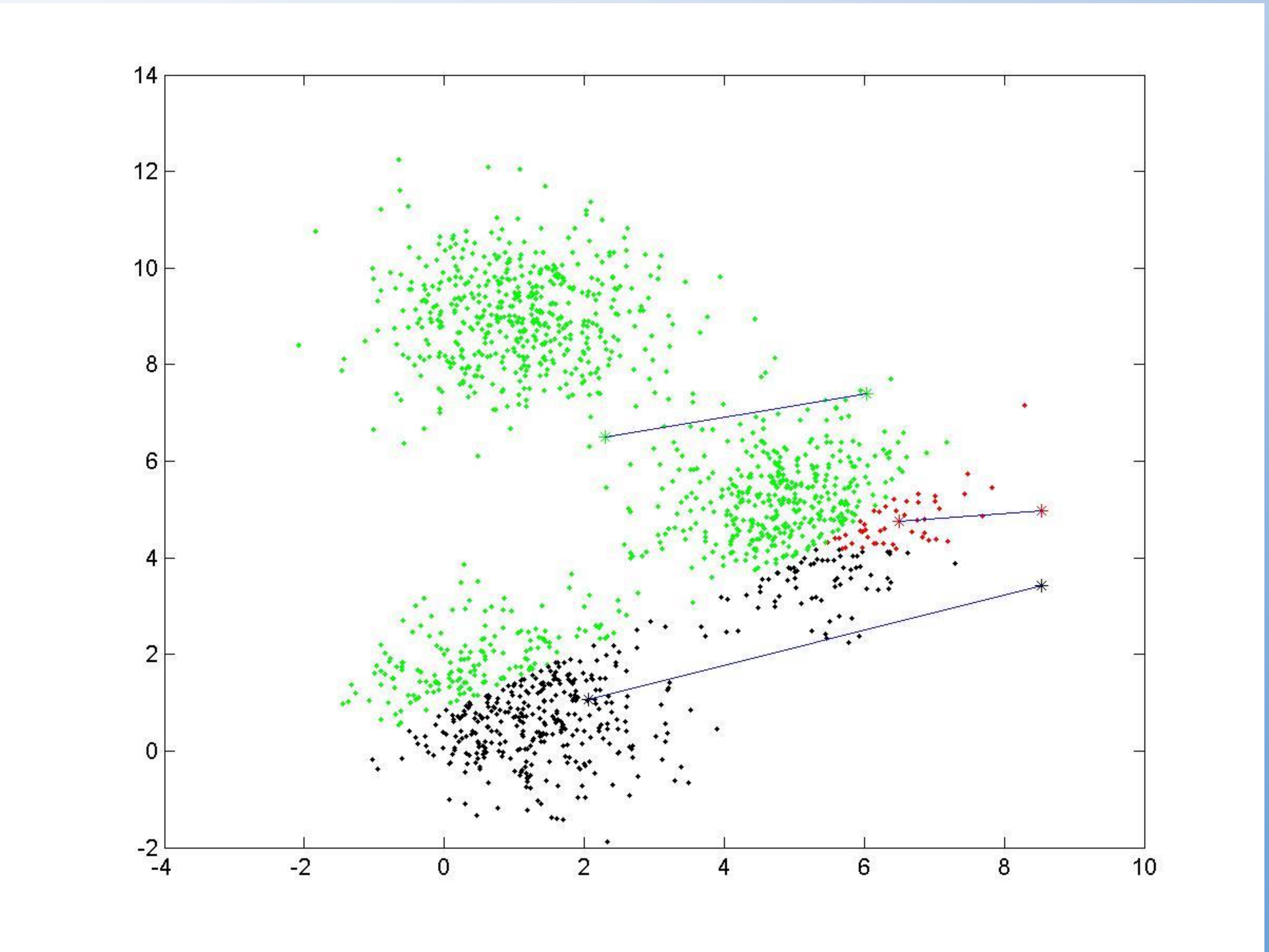
K-means Algorithm

- Iteration 1: Assigning points to clusters



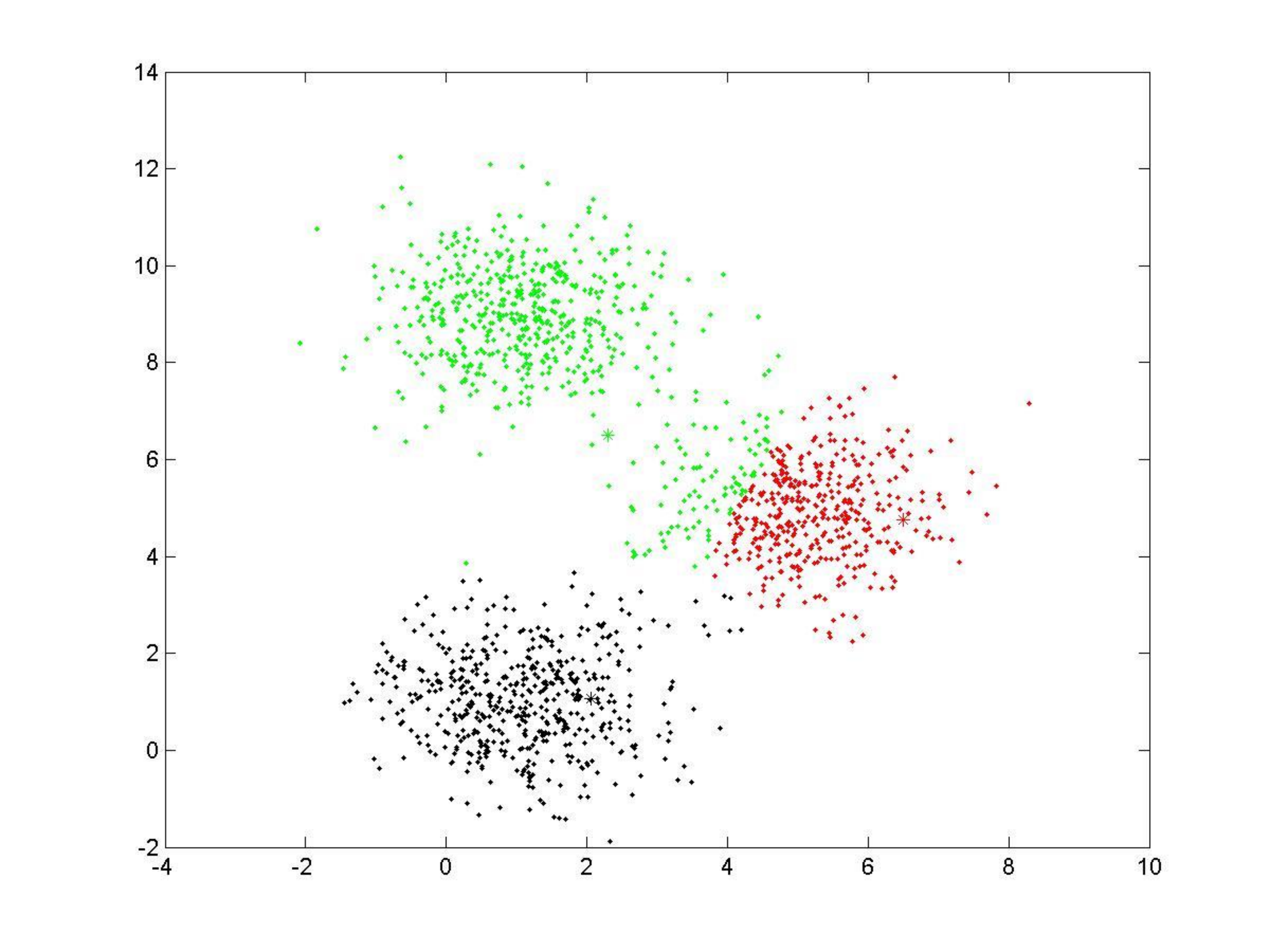
K-means Algorithm

- Iteration 1: Recomputing means



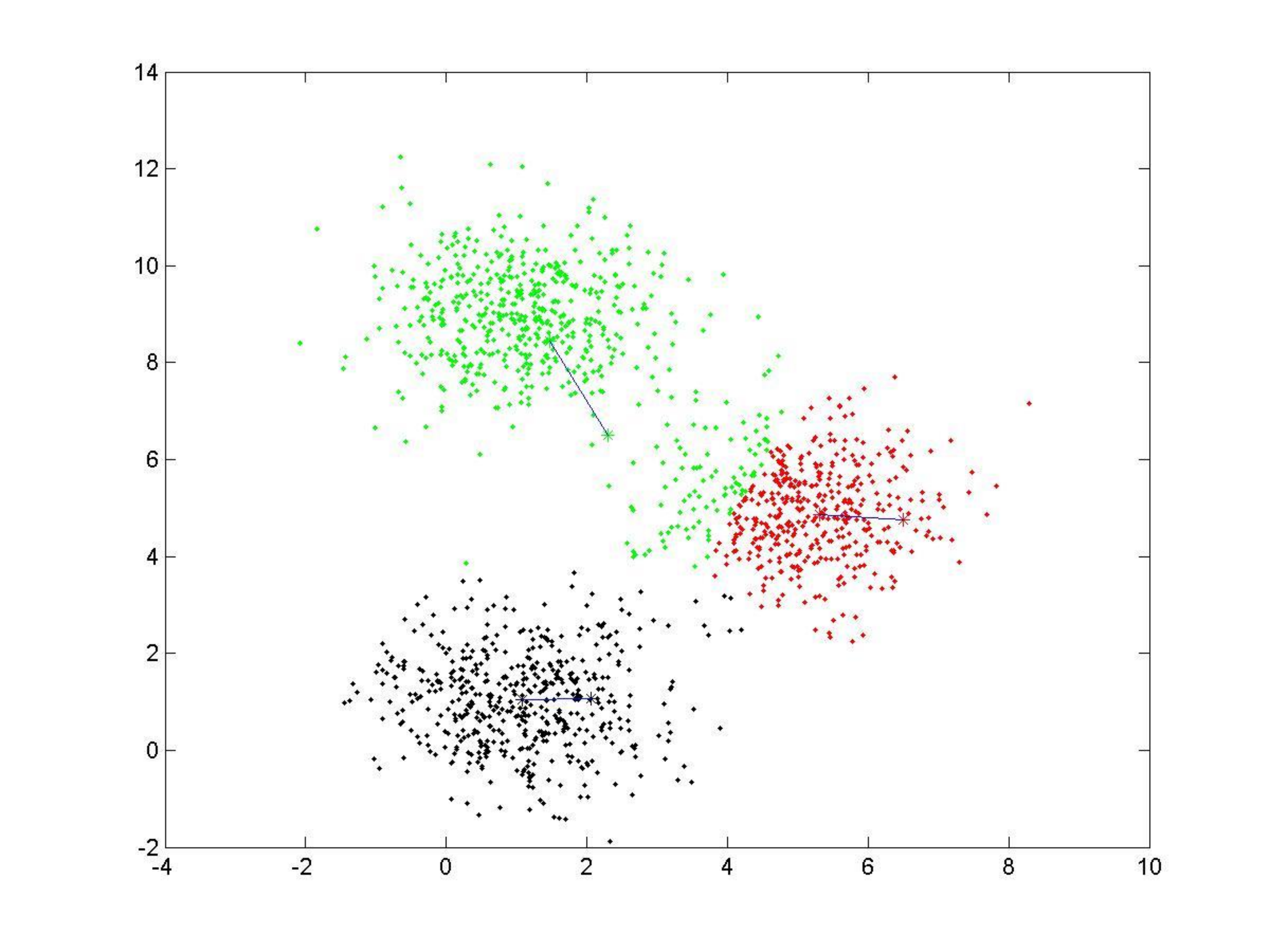
K-means Algorithm

- Iteration 2: Assigning points to clusters



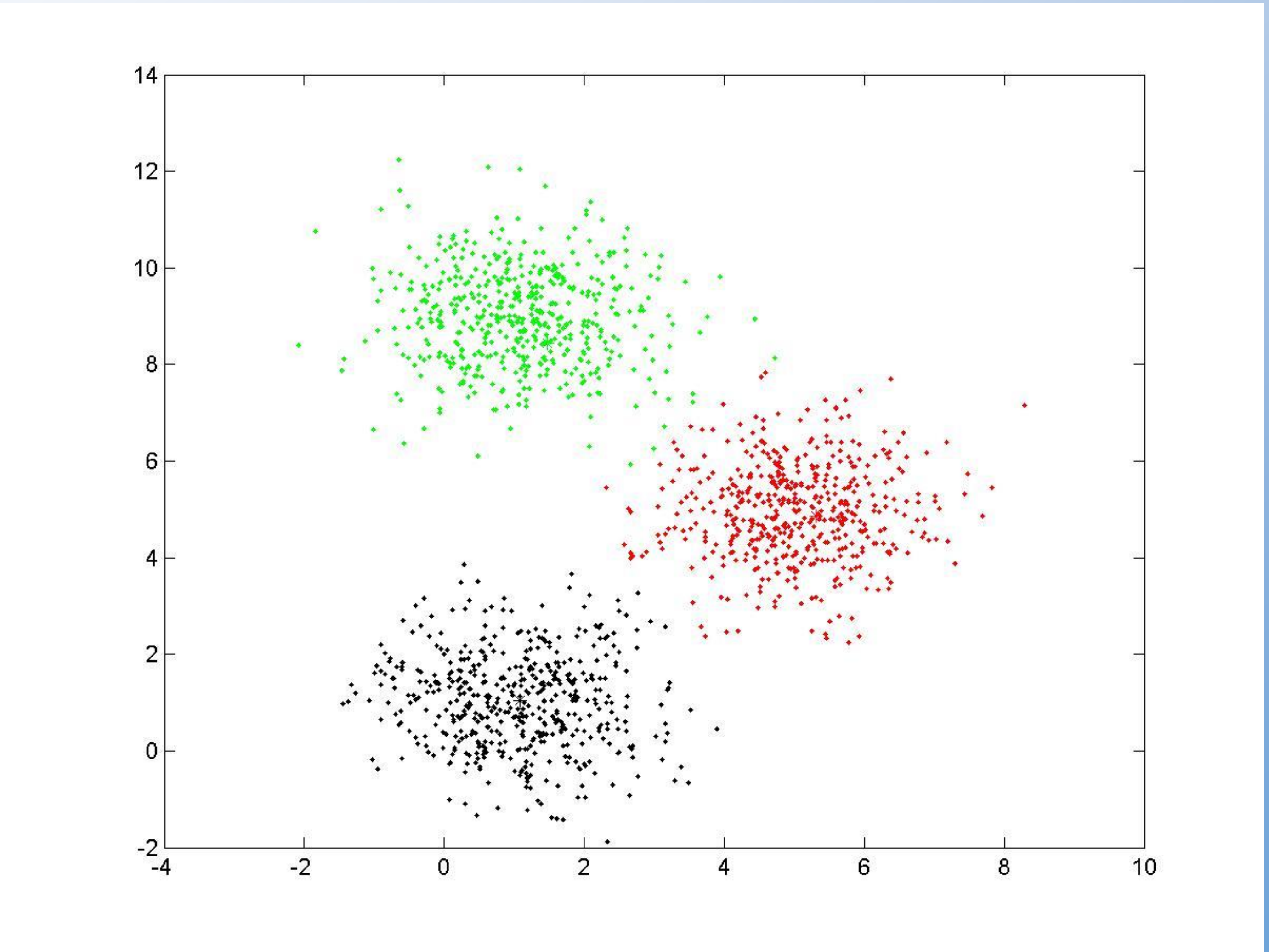
K-means Algorithm

- Iteration 2: Recomputing means



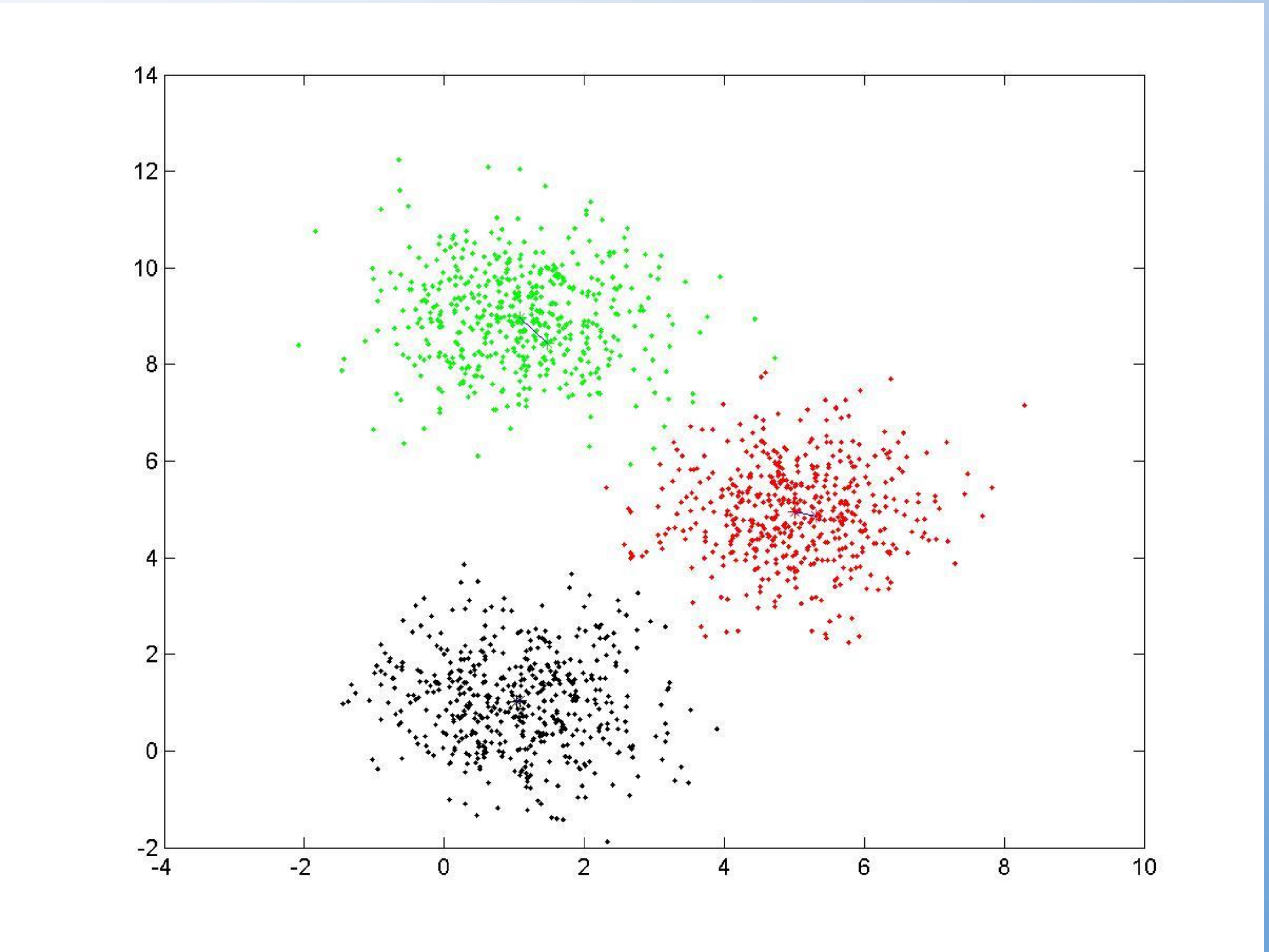
K-means Algorithm

- Iteration 3: Assigning points to clusters



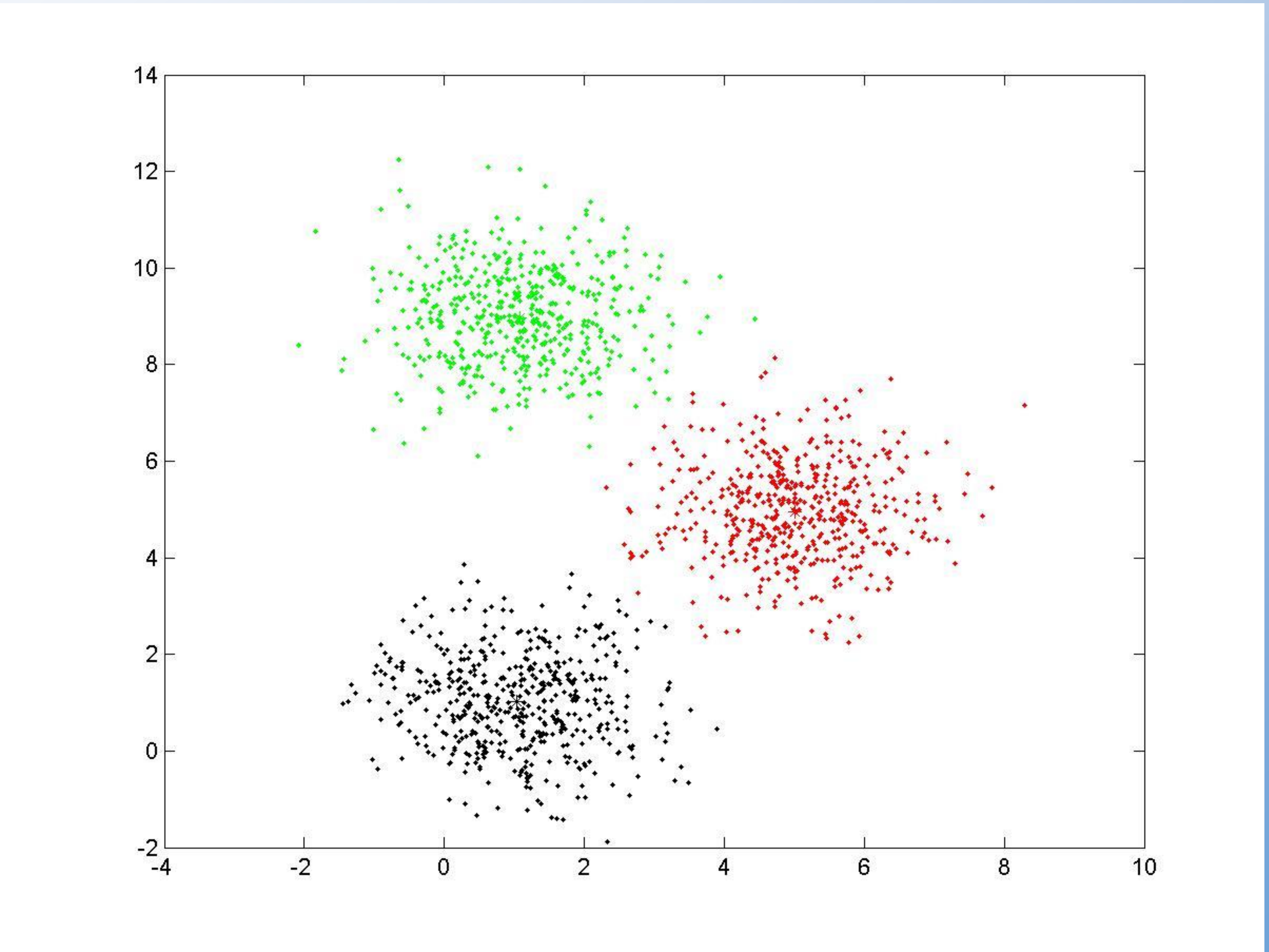
K-means Algorithm

- Iteration 3: Recomputing means



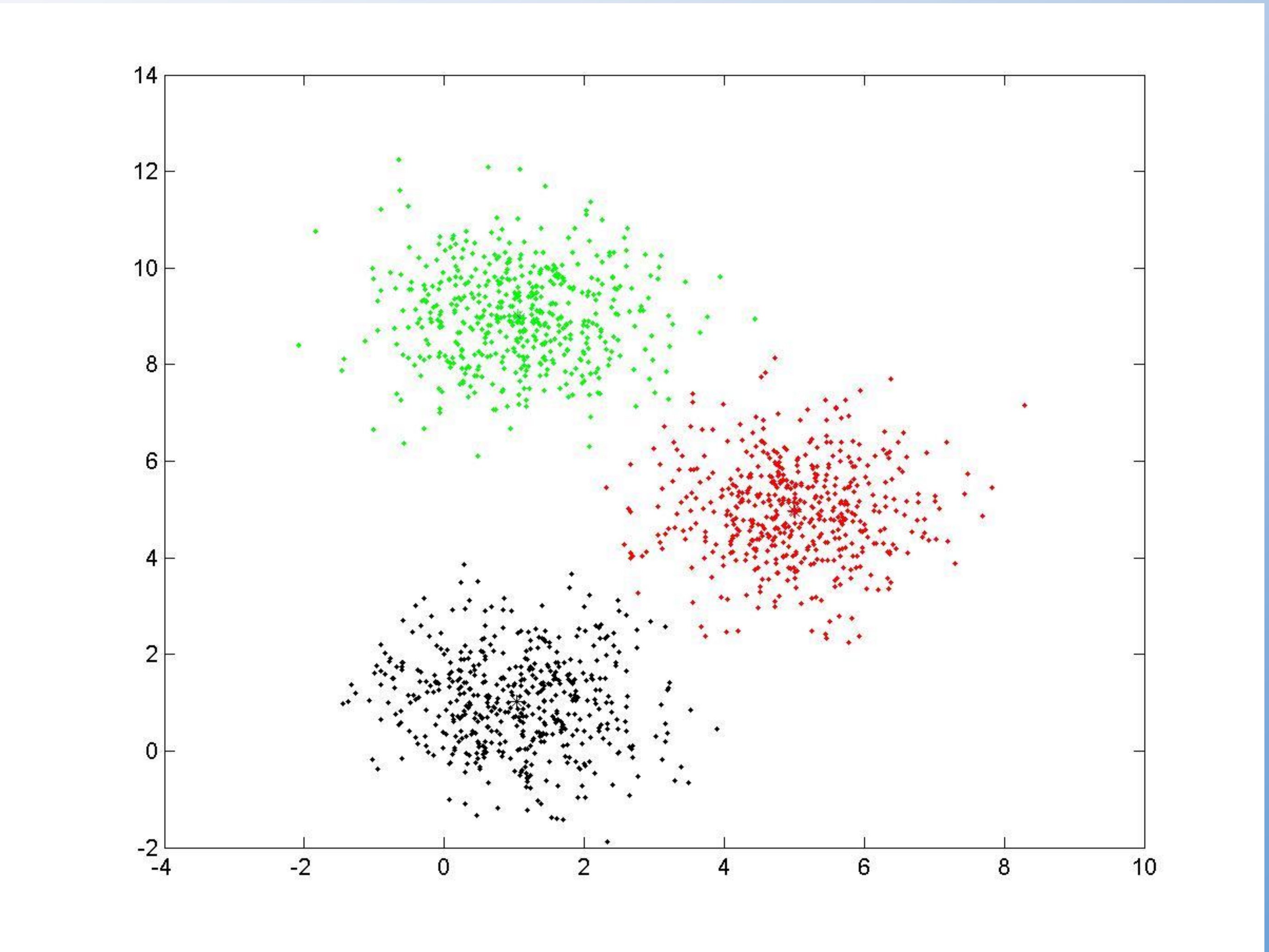
K-means Algorithm

- Iteration 4: Assigning points to clusters



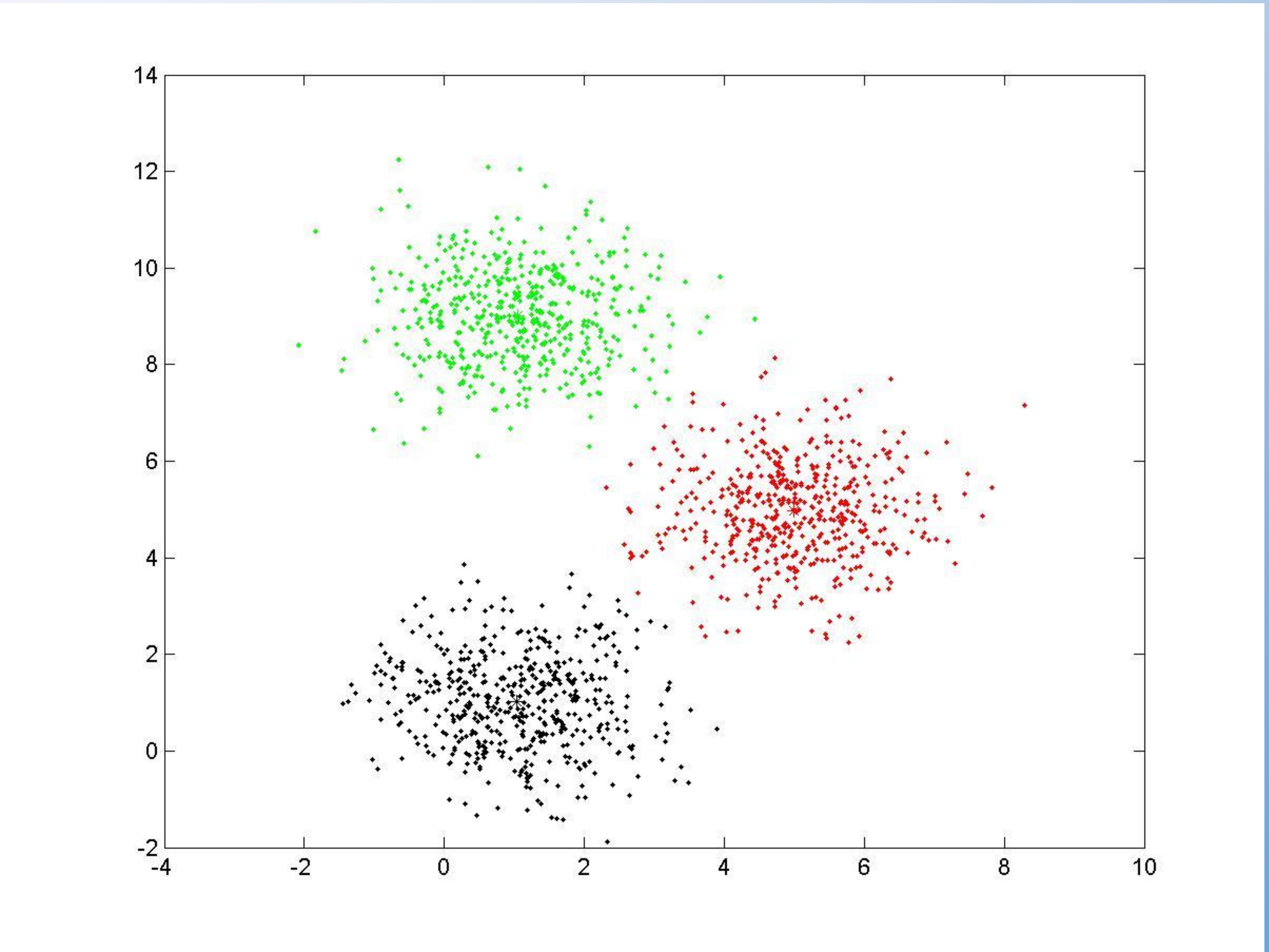
K-means Algorithm

- Iteration 4: Recomputing means



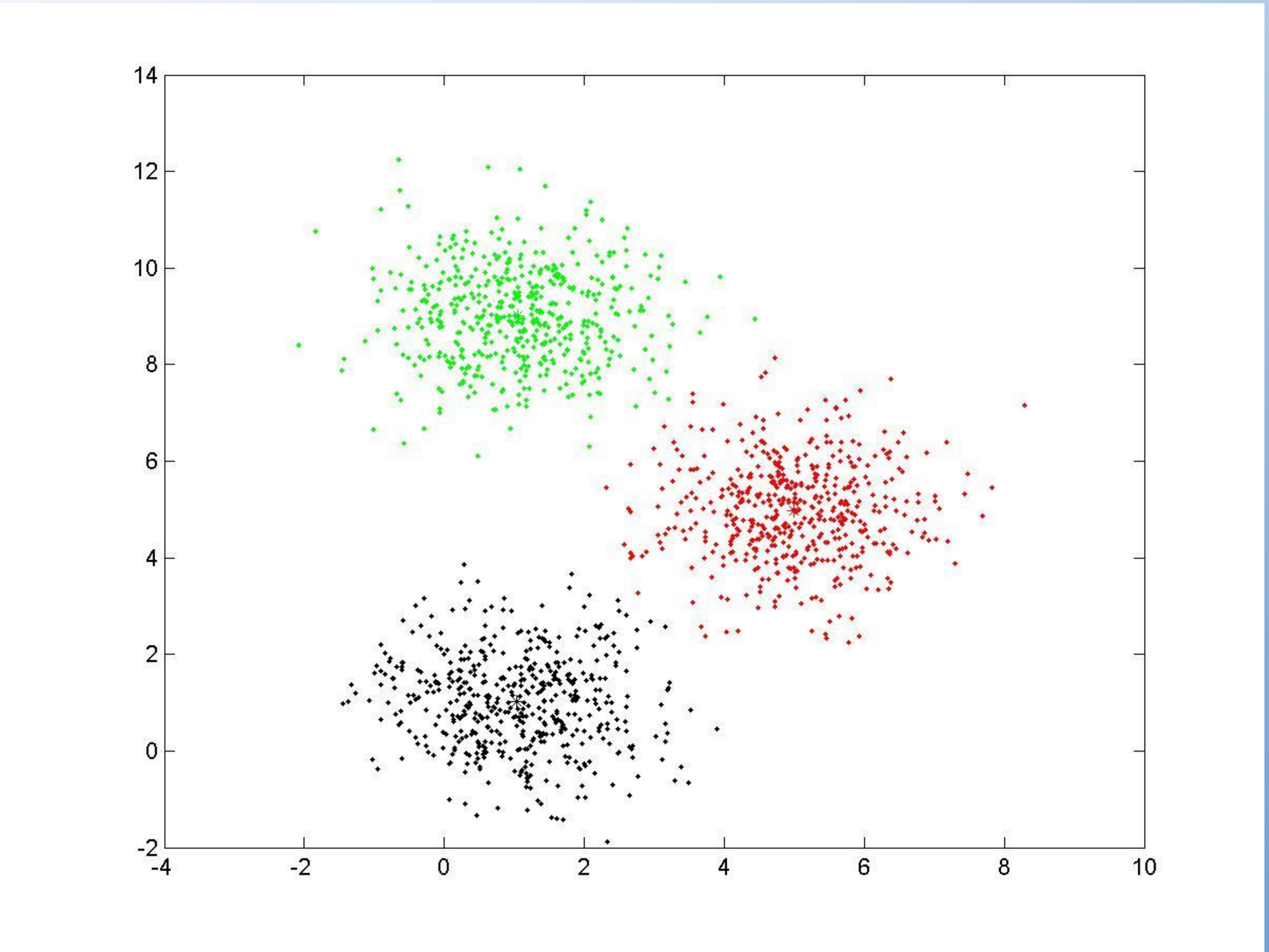
K-means Algorithm

- Iteration 5: Assigning points to clusters

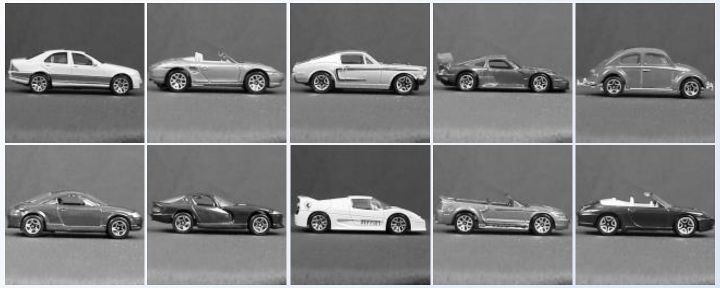


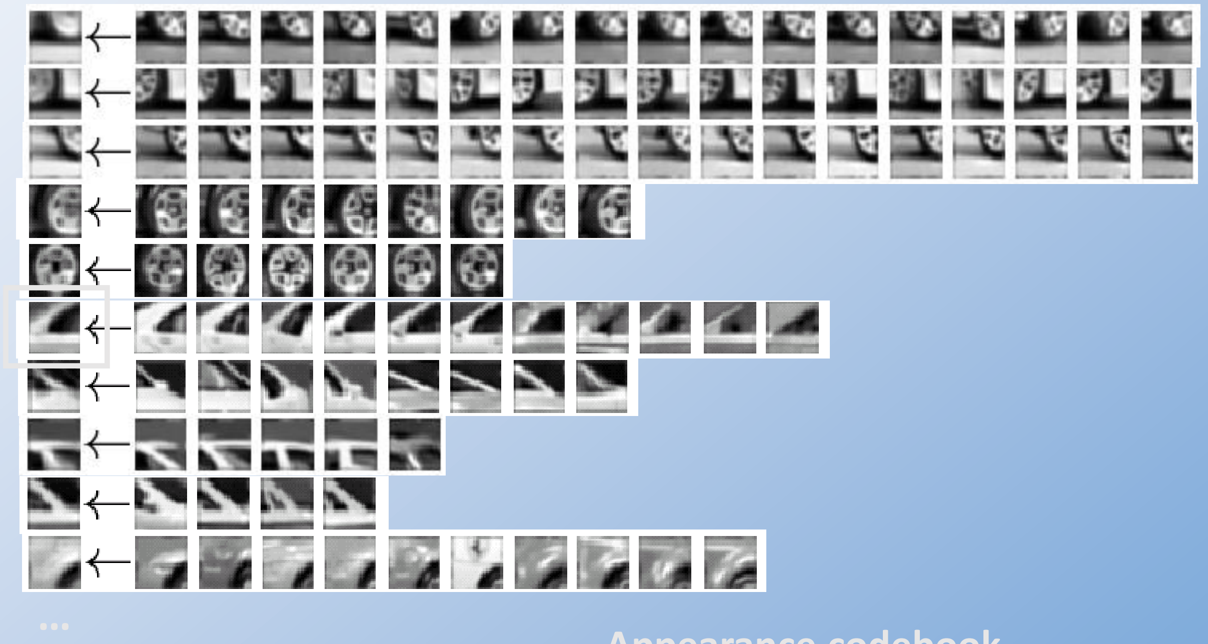
K-means Algorithm

- Iteration 5: Recomputing means



Example visual vocabulary

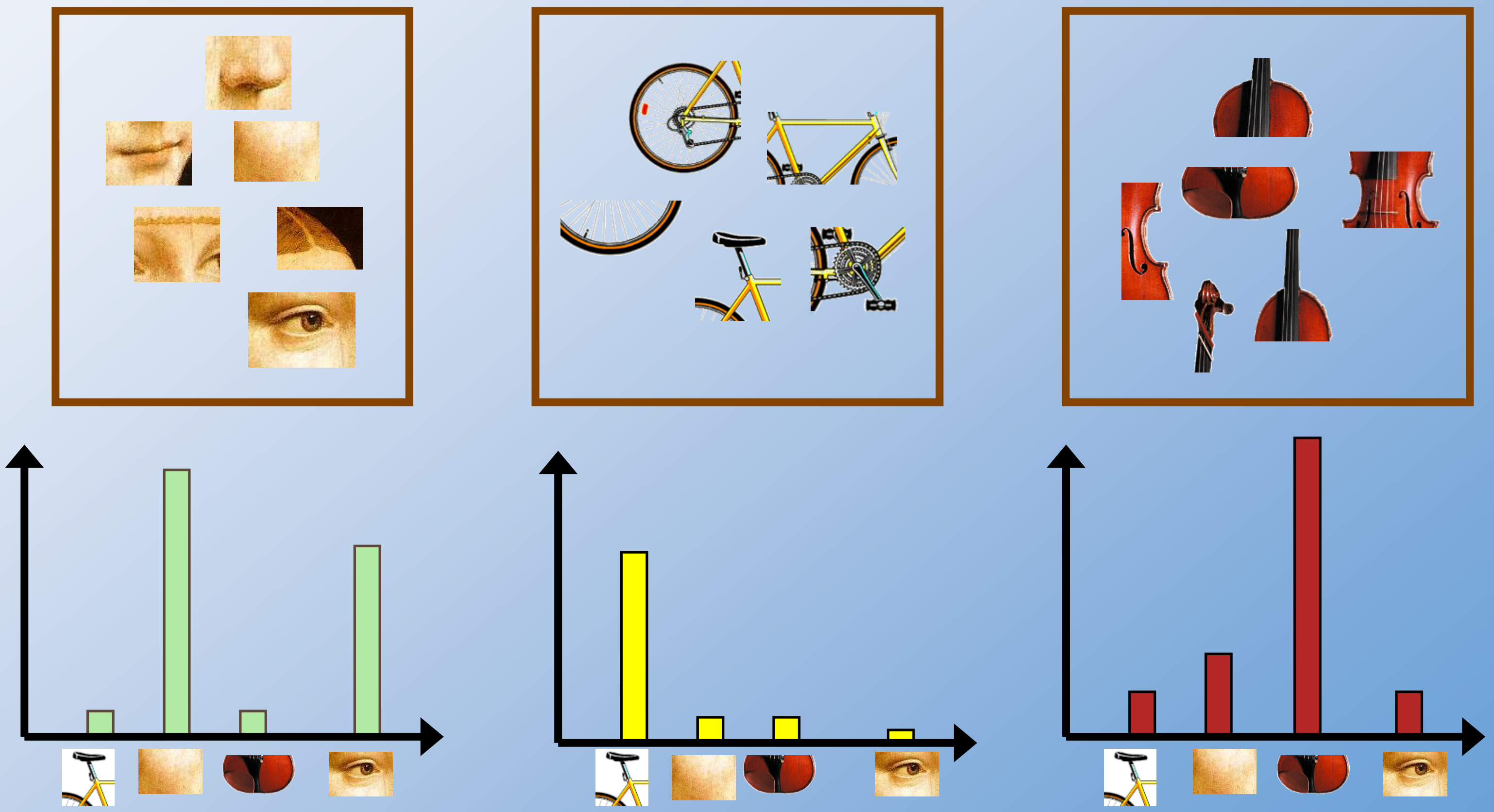




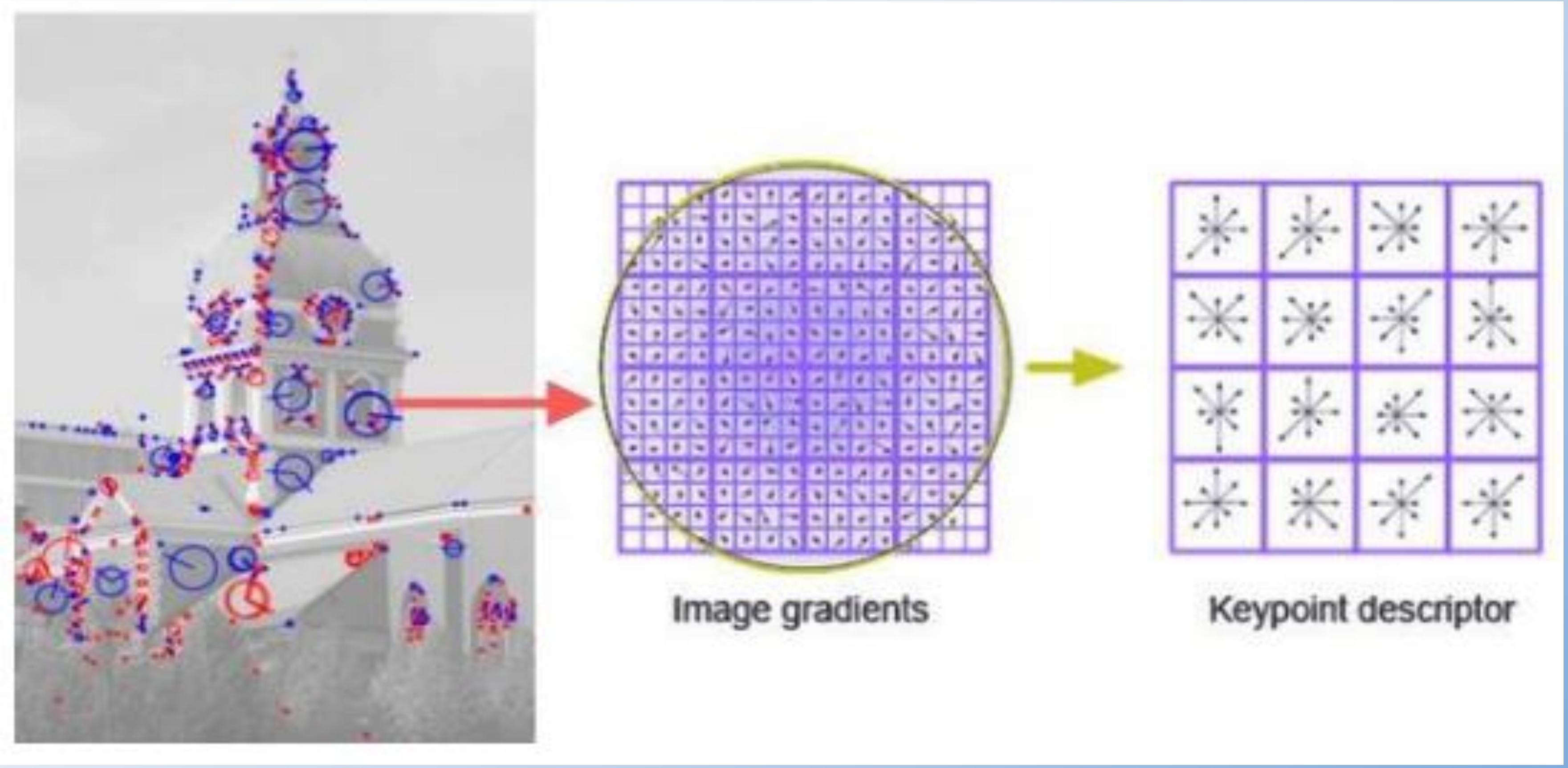
Appearance codebook

Bag-of-features steps

1. Extract local features
2. Learn "visual vocabulary"
3. **Quantize local features using visual vocabulary**
4. **Represent images by frequencies of "visual words"**



Scale Invariant Feature Transform (SIFT)



Find small region with high gradient magnitude in two directions, split into 4x4 subregions and compute 8-bar gradient histogram for each subregion – this yields a 128-dimensional descriptor. It exploits the property that rotations are equivalent to histogram shifts.