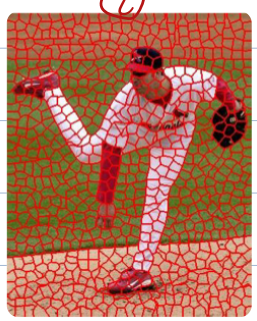
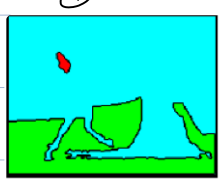
W6 - P1 - Segmentation and clustering

Goals of segmentation:-

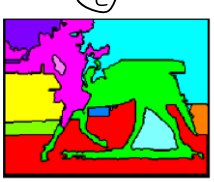
1. Separate image into coherent "objects"
2. Group together similar-looking pixels for efficiency of further processing



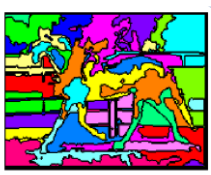
Types of segmentations



Undersegmentation



Multiple Segmentations



Oversegmentation

What is objects?

* Subjective

"what is interesting what is not" → so depends on application

نظرية الغشتالتي (gestalt theory) في علم النفس , تعني ببساطة أنه من الضروري اعتبار الكل، لأن الكل له معنى مختلف عن الأجزاء المكونة له

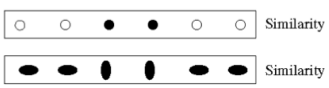
تأثير الغشتلت ويشير إلى قدرة مشكلة للشكل موجودة ضمن حواسنا ، عمليا بالنسبة للتمييز البصري للأشكال والأشخاص وجميع الأشكال بدلا من رؤية مجرد خطوط بسيطة ومنحنيات.

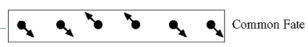
- **Gestalt:** whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/features

Gestalt factors: affect whether elements should be grouped or not

① Proximity (nearby)

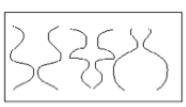
② Similarity





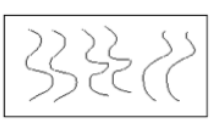
③ common fate (coherent motion)

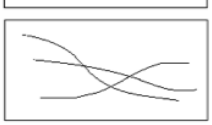
④ Symmetry



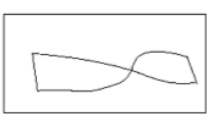
⑤ common region

⑥ Parallelism





⑦ Closure



⑧ continuity

⑨ familiar configuration (if grouped → be familiar object)

Gestalt cues

- Good intuition and basic principles for grouping
- Basis for many ideas in segmentation and occlusion reasoning
- Some (e.g., symmetry) are difficult to implement in practice

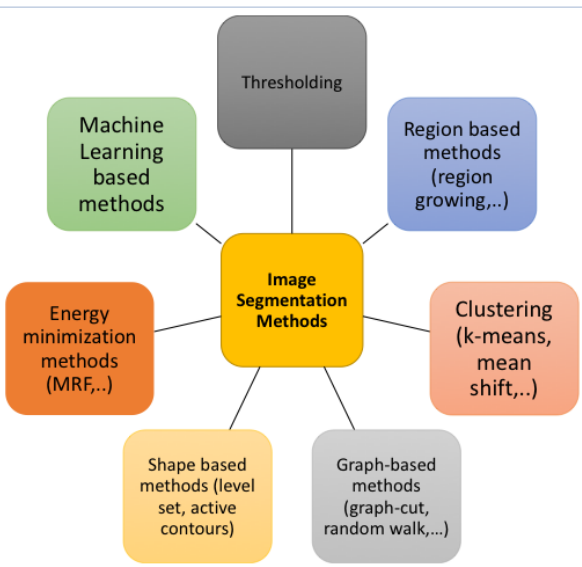
- Clustering: group together similar data points and represent them with a single token
  - Tokens: whatever we need to group (pixels, points, surface elements, etc., etc.)

- Key Challenges:
  1) What makes two points/images/patches similar?
  2) How do we compute an overall grouping from pairwise similarities?
  3) • Hard to measure success
     • What is interesting depends on the app.

Ways of clustering:

1. Top down: pixels belong together because they are from the same object
2. Bottom up: pixels belong together because they look similar

- Image Segmentation
  - histogram based
  - cluster based — K-means, mean-Shift



- histogram based:
  - each color is segment
    - not good with real images

Clustering

- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i: $\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$

sum squared distance

What is Clustering?

- Organizing data into classes such that:
  - High intra-class similarity
  - Low inter-class similarity

Cluster by features

- **Color**
- **Intensity**
- **Location**
- **Texture**
- ....

• intensity + position

1 K-means

- Clusters don't have to be spatially coherent

if we use color

K-means clustering: Algorithm

1. Randomly initialize the cluster centers, $c_1, \ldots, c_K$
2. Given cluster centers, determine points in each cluster
   - For each point p, find the closest c_i. Put p into cluster i
3. Given points in each cluster, solve for c_i
   - Set c_i to be the mean of points in cluster i
4. If c_i have changed, repeat Step 2

Properties

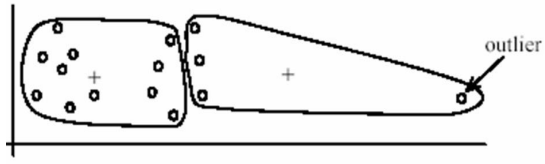
- Will always converge to *some* solution
- Can be a "local minimum"
  - does not always find the global minimum of objective function

How to find a good "K"?

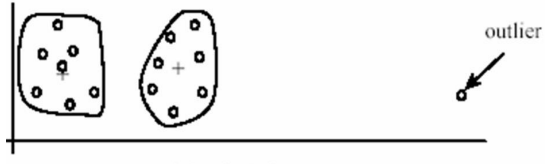
1. Use prior knowledge about image.
2. Apply the algorithm for different values of K and test for goodness of clusters.
3. Analyze Image Histograms.

K-Means for Segmentation

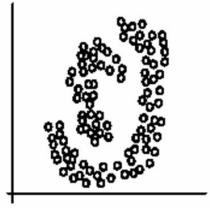
- Pros
  - Very simple method
  - Converges to a local minimum of the error function
- Cons
  - Memory-intensive
  - Need to pick K
  - Sensitive to initialization
  - Sensitive to outliers
  - Only finds "spherical" clusters
  - Assuming means can be computed
  - hard clustering



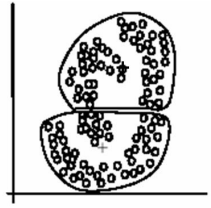
(A): Undesirable clusters



(B): Ideal clusters



(A): Two natural clusters

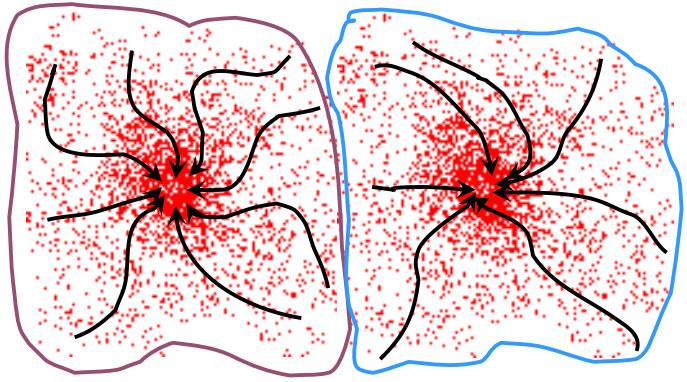


(B): k-means clusters

Slide by Svetlana Lazebnik

Mean-shift Clustering

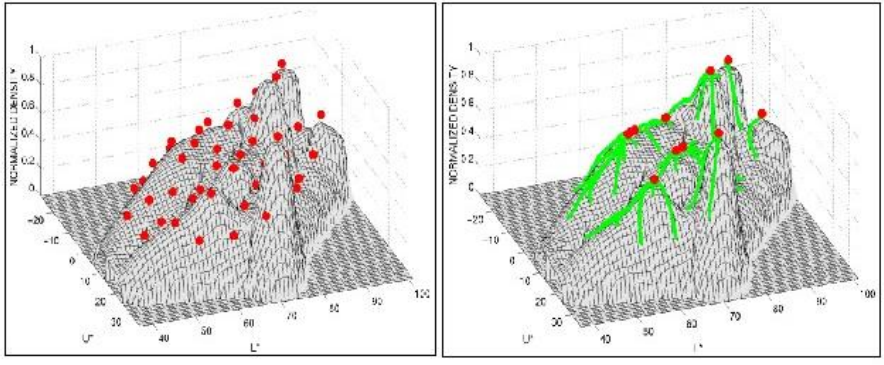
- Cluster all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



Slide by Y. Ukrainitz & B. Sarel

Mean-shift Approach

- Initialize a window around each point
- See where it shifts—this determines which segment it's in
- Multiple points will shift to the same segment

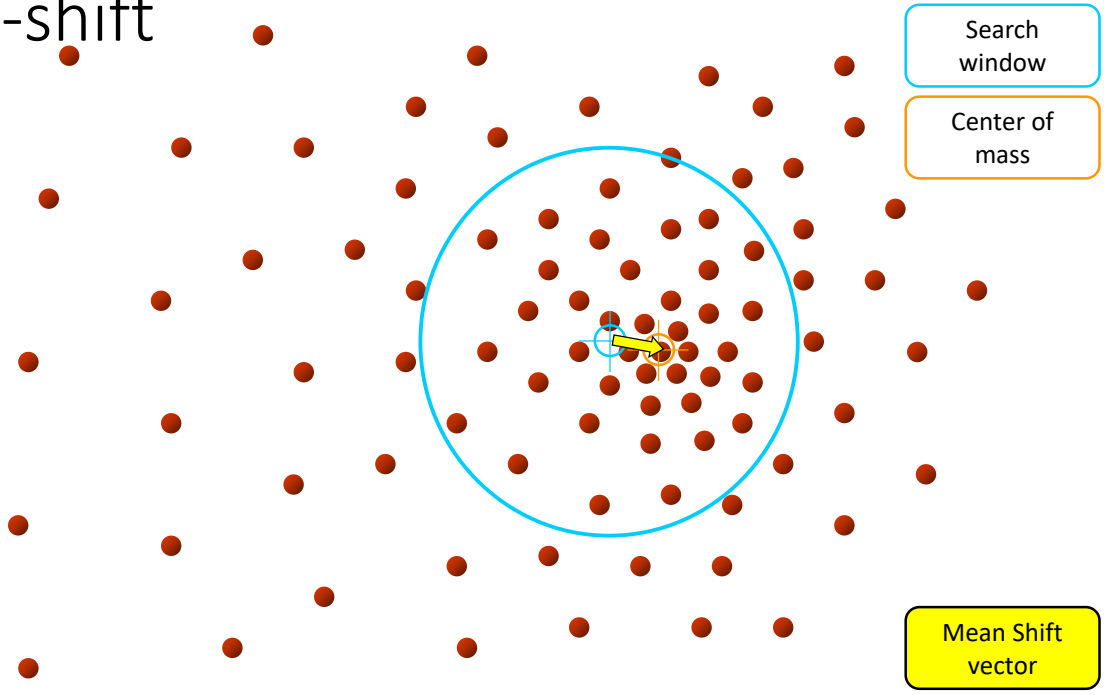


Mean shift trajectories

Mean-shift Algorithm

- First run the mean shift procedure for each data point x and store its convergence point z.
- Link together all the z's that are closer than .5 from each other to form clusters
- Assign each point to its cluster
- Eliminate small regions

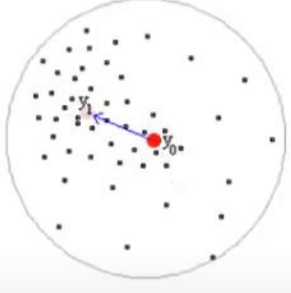
Mean-shift



Slide by Y. Ukrainitz & B. Sarel

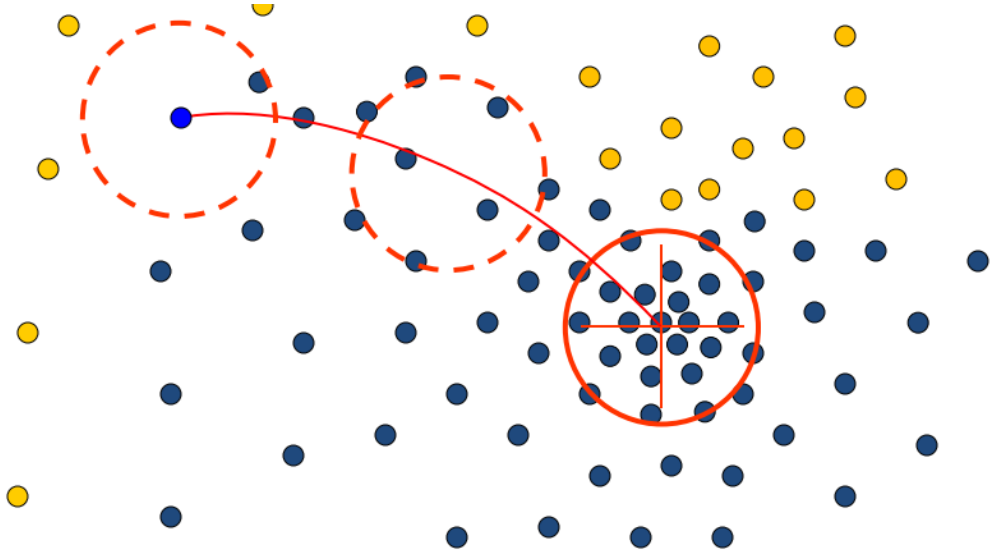
Mean-shift Vector Example

- Mean-shift vector always points towards the direction of the maximum increase in the density.



$$M_h(\mathbf{y}) = \left[\frac{1}{n_x} \sum_{i=1}^{n_x} \mathbf{x}_i \right] - \mathbf{y}_0$$

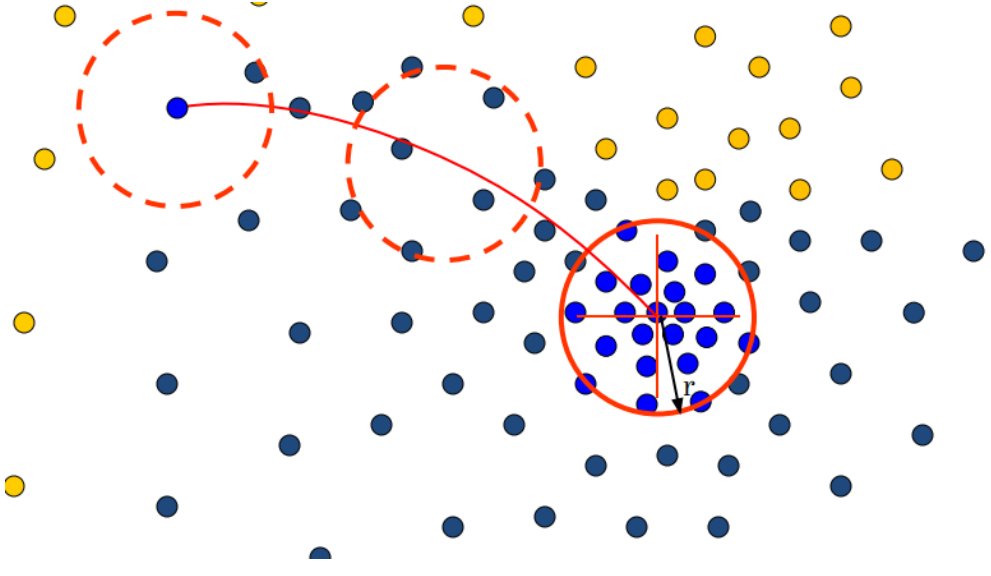
Problem: Computational Complexity



- Need to shift many windows…
- Many computations will be redundant.

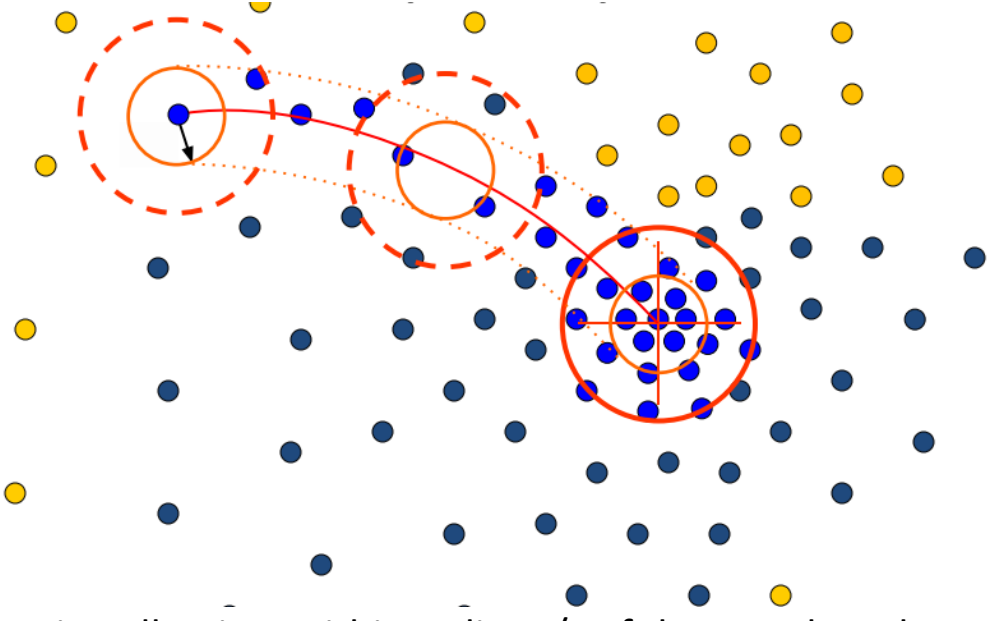
solution

Speedups: Basin of Attraction



1. Assign all points within radius r of end point to the mode.

Speedups



2. Assign all points within radius r/c of the search path to the mode

Mean-shift for Segmentation

- Pros:
  - General, application-independent tool
  - Model-free, does not assume any prior shape (spherical, elliptical, etc.) on data clusters
  - Just a single parameter (window size h)
    - h has a physical meaning (unlike k-means) – Finds variable number of modes
  - Robust to outliers
- Cons:
  - Output depends on window size
  - Window size (bandwidth) selection is not trivial
  - Computationally (relatively) expensive (~2s/image)
  - Does not scale well with dimension of feature space

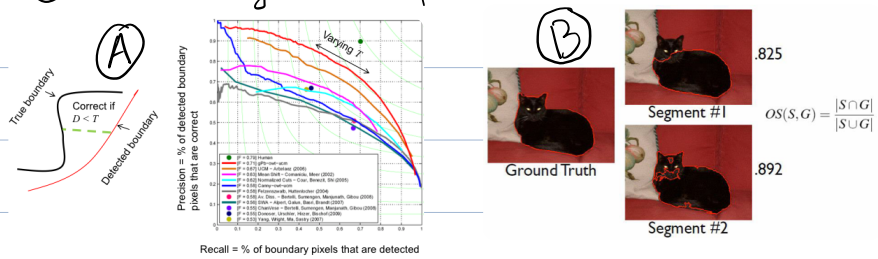
W6 - P2 - Segmentation and clustering

What is good segmentation?

① Compare to human segmentation or "ground truth"

evaluations: (A) boundary agreement

(B) Calculate region overlap with ground truth



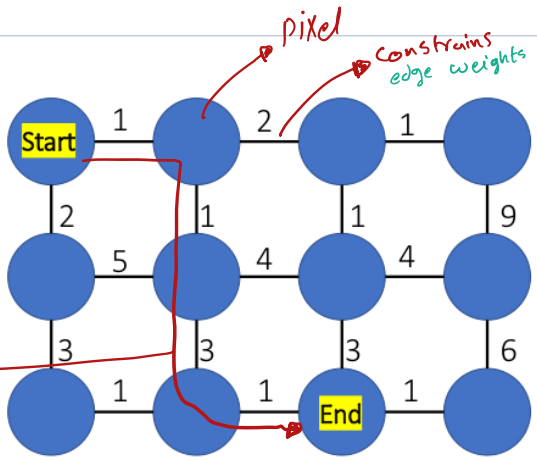
② Superpixels (over segmentation) → مزايا وعيوب

③ Multiple Segmentation then take AND (consensus)

* Cut and Past procedure:-

① Extract Sprites (animation 2D photo)

by finding Seams (find where to cut) or Segmentation (split into foreground background)



Shortest Path by Dijkstra's Algo.

Define boundary cost between neighboring pixels:

1. Lower if an image edge is present (e.g., as found by Sobel filtering).
2. Lower if the gradient magnitude at that point is strong.
3. Lower if gradient is similar in boundary direction.

Good places to cut:

- similar color in both images
- high gradient in both images

② Blend them into the composite

* GrabCut:

GrabCut is a mixture of two components

1. Segmentation using graph cuts
2. Foreground-background modeling using unsupervised clustering

user input the box

W7-P1 - Image classification

Object recognition

Introduction

KNN

A simple object recognition pipeline

Challenges of object recognitions:-

1. many object categories
2. viewpoint variation
3. illumination
4. Scale
5. deformation
6. occlusion
7. Background clutter
8. intra-class variation

KNN :-

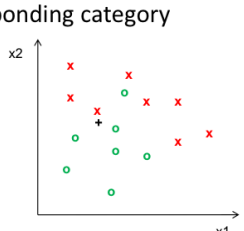
- Assign label of nearest training data point to each test data point

Training:

1. Store all training data points $\boldsymbol{xi}$ with their corresponding category labels $\boldsymbol{yi}$

Testing:

1. We are given a new test point $\boldsymbol{x}$
2. Compute distance to all training data points
3. Select k training points closest to $\boldsymbol{x}$
4. Assign $\boldsymbol{x}$ to label $\boldsymbol{y}$ that is most common among the k nearest neighbors.



Distance Metrics

$$D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{(x_1-y_1)^2+\cdots+(x_N-y_N)^2}$$ Euclidean

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\boldsymbol{x}\cdot\boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|} = \frac{x_1y_1+\cdots+x_Ny_N}{\sqrt{\sum_n x_n^2}\sqrt{\sum_n y_n^2}}$$ Cosine

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2}\sum_n \frac{(x_n-y_n)^2}{(x_n+y_n)}$$ Chi-squared

W7-P2 - Image classification

Of KNN

Pros:

① simple

② flexible decision boundaries

Cons:

① which K?

Small K ⇒ sensitive to noise point

Larg K ⇒ underfitting (include from other classes)

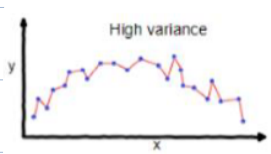
Solutions: cross-validation

② different measurment scales:

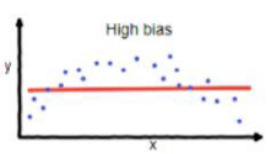
- E.g., patient weight in kg (range [50,200]) vs. blood protein values in ng/dL (range [-3,3])

⇒ patient weight will have greater influence

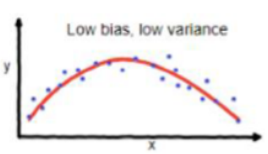
Solutions: Normalization $z_{ij} = \frac{x_{ij} - m_j}{s_j}$ (m_j: mean, s_j: standard deviation)



Overfitting



Underfitting

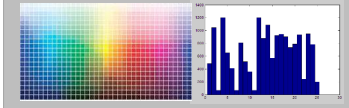


Good balance

Image features

Input image

Color: Quantize RGB values

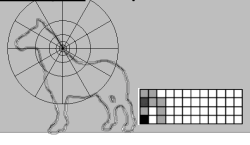


Invariance?
- Translation
- ? Scale
- Rotation
- Occlusion

Global shape: PCA space

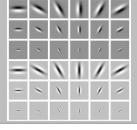
Invariance?
- Translation
- ? Scale
- Rotation
- Occlusion

Local shape: shape context

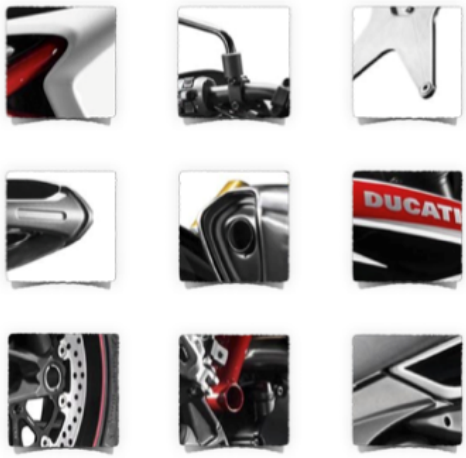


Invariance?
- Translation
- Scale
- Rotation (in-planar)
- Occlusion

Texture: Filter banks



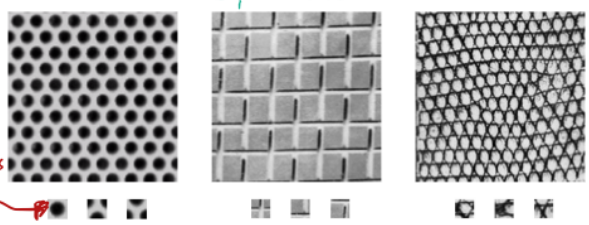
Invariance?
- Translation
- ? Scale
- ? Rotation (in-planar)
- Occlusion



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

Bag of words

Bag of textons



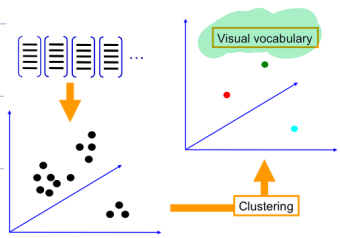
textons

a collection of local features

(bag-of-features)

→ enough information about the object

- deals well with occlusion
- scale invariant
- rotation invariant



1

Extract features

① by SIFT or other methods

② apply TF-IDF

2

Learn visual dictionary / Vocabulary

use clustring

K-means or mean shift

We then use the centers of each cluster as the textons.

3

from clustring to vector quantization

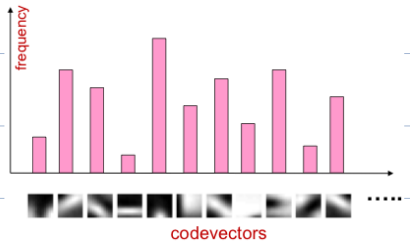
Example visual dictionary

Appearance codebook

- Each cluster center produced by k-means becomes a codevector
- A vector quantizer takes a feature vector and maps it to the index of the nearest codevector in a codebook

- Codebook = visual vocabulary = dictionary
- Codevector = center of each feature cluster

Classification (5) ← Histogram of codevector freq. (4)



TF - IDF
- TF → term freq.
- IDF → inverse document freq.

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

- n_{id}: Number of occurrences of word i in document d
- n_d: Number of words in document d
- N: Total number of documents in database
- n_i: Number of occurrences of word i in whole database

$t_i = \frac{1}{6} \quad \frac{3}{2}$