EE 569 Discussion

Zohreh Azizi

03/19/2021 & 03/26/2021

Announcements

- HW4

  Posted: 03/07/21

  Due: 03/28/21 11:59PM

Announcements

Preview of Discussion Sessions:

- …
- Week 10 (Mar. 19): HW4 – part 1
- Week 11 (Mar. 26): HW4 – part 2
- Week 12 (Apr. 2): HW5
- Week 13 (Apr. 9): Midterm2 Review
- …

Announcements

Coding suggestions:

For better code structure, version update, peer evaluation, and OF COURSE EASIER DEBUGGING, try to:

- Design your code structure carefully based on procedure functionality.
- Encapsulate codes into “functions” by procedure functionality, and give independent code tests to all functions, make sure they individually work well (even on marginal or trivial cases)

  → Now, you just need to worry about input and output when you call functions.
- Then, build your main code body using your functions.
- Test!

Overview

- **Texture Classification (Problem 1)**
  - **Feature Extraction (Law's Filter)**
  - Feature Dimension Reduction (PCA)
  - Nearest Neighbor
  - Classifier (K-means, Random Forest, Support Vector Machine)

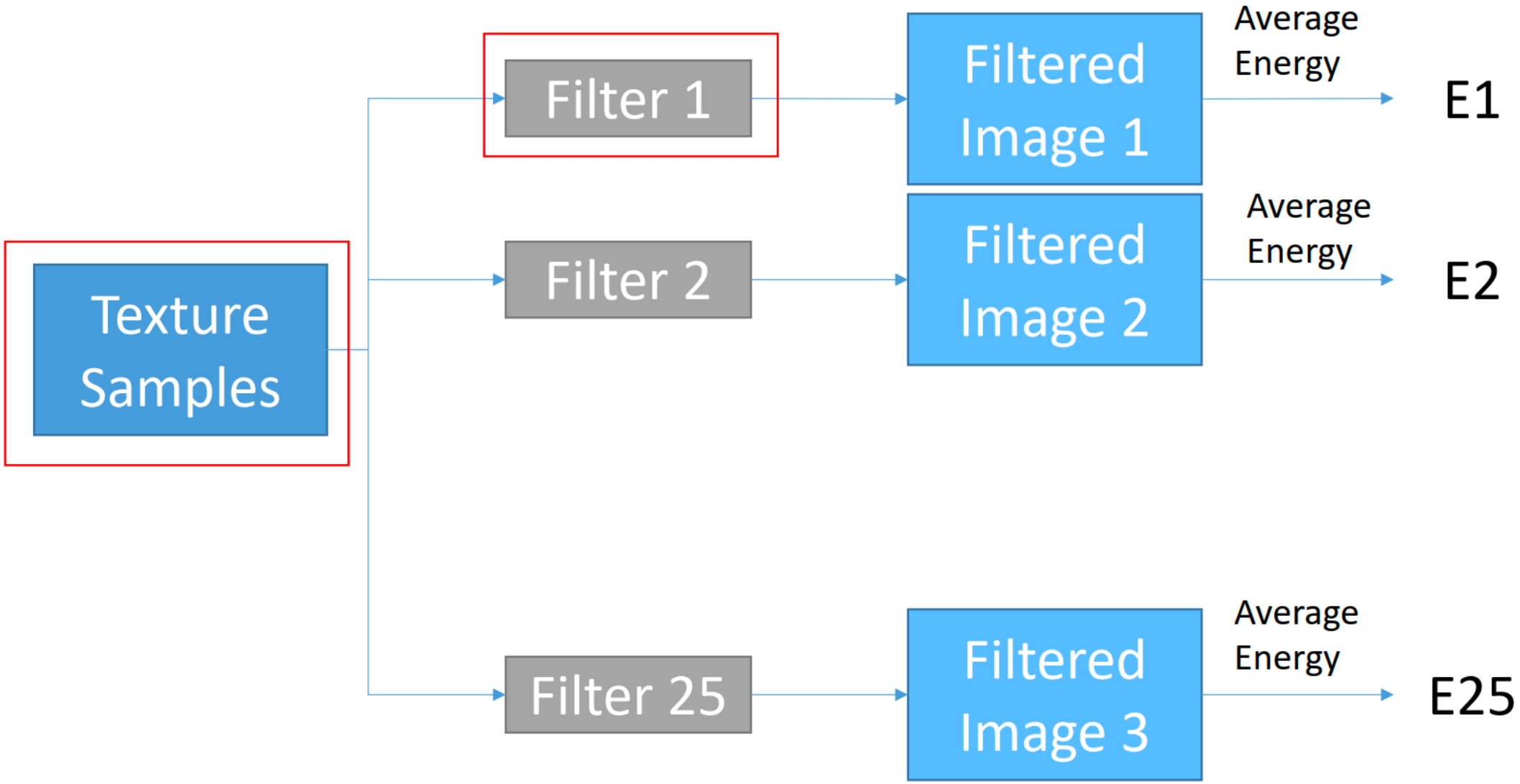
- Texture Segmentation (Problem 2)

- Image Feature Extractor (Problem 3)
  - SIFT
  - Bag of Words (BoW)

Problem 1 – Feature Extraction

What are the texture?

Problem 1 – Feature Extraction



Problem 1 – Feature Extraction

Subtract global mean.

Why? Reduce illumination effects.

Problem 1 – Law's Filters

Table.1 1D Kernel for 5x5 Laws Filters

Name	Kernel
L5 (Level)	[1 4 6 4 1]
E5 (Edge)	[-1 -2 0 2 1]
S5 (Spot)	[-1 0 2 0 -1]
W5 (Wave)	[-1 2 0 -2 1]
R5 (Ripple)	[1 -4 6 -4 1]

Problem 1 – Law's Filters

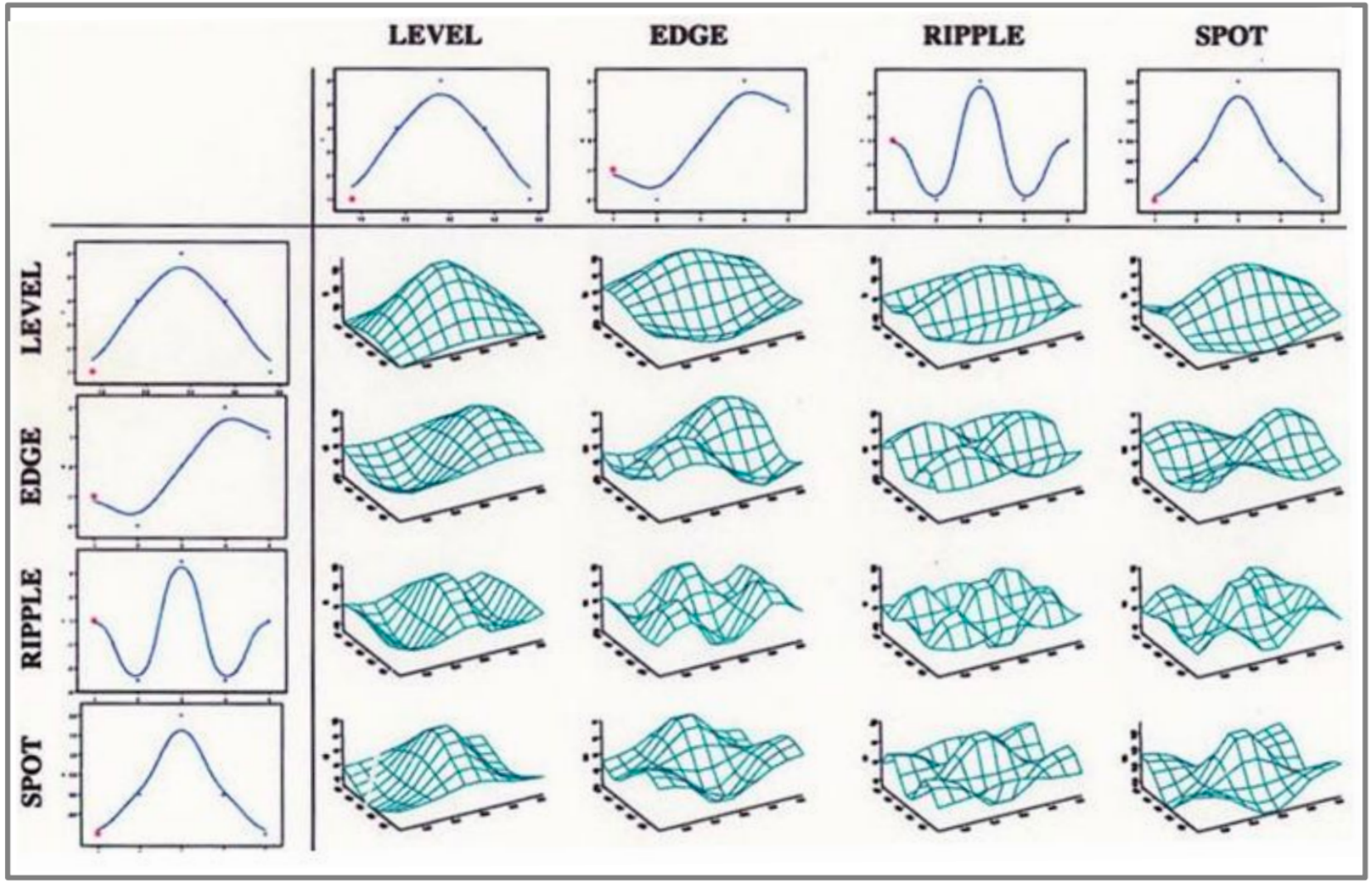
25 2D filters

- $L5L5^T$
- $L5E5^T$
- $L5S5^T$
- …

$$\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix} \times \begin{bmatrix}-1 & -2 & 0 & 2 & 1\end{bmatrix} = \begin{bmatrix}-1 & -2 & 0 & 2 & 1\\-4 & -8 & 0 & 8 & 4\\-6 & 12 & 0 & 12 & 6\\-4 & -8 & 0 & 8 & 4\\-1 & -2 & 0 & 2 & 1\end{bmatrix}$$

An example of $L5E5^T$

Problem 1 – Law's Filters



Problem 1

For each texture image:

- Subtract image mean (reduce illumination effects)
- Apply 25 5x5 Laws filters to get 25 filtered images
- Average energy in each response image to form a 25-D feature vector

→ Feature Set (n samples x 25 feature dimensions)

- Evaluate the discriminant power of each feature dimension

Problem 1 - Discriminant Power

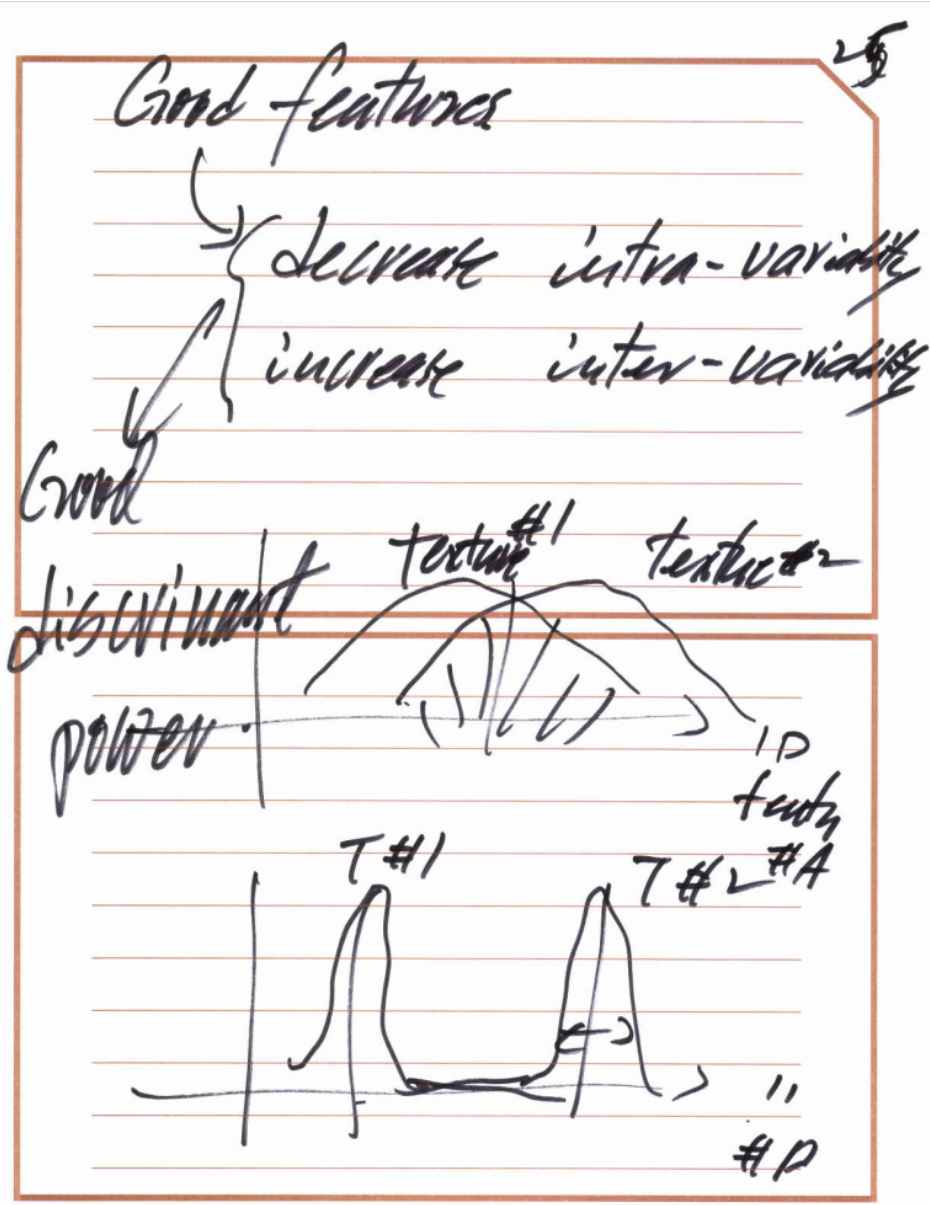
For each dimension:

- Calculate Intra-variety
- Calculate Inter-variety

❖ Discriminant power calculation needs to know the label

❖ Good feature: Low intra-variety & High inter-variety

→ Discriminant power = intra-variety/inter-variety



Problem 1 - Discriminant Power

https://math.stackexchange.com/questions/81024/about-the-total-variance-and-the-inter-and-inter-class-variances

It's the *sums* of squares of deviations, not the *averages* of squares of deviations, for which this identity holds.

Let y_{ij} be the jth observation in the ith class. The overall average is

$$\bar{y}_{\bullet\bullet} = \frac{\sum_i \sum_j y_{ij}}{\sum_i \sum_j 1}.$$

The average within class i is

$$\bar{y}_{i\bullet} = \frac{\sum_j y_{ij}}{\sum_j 1}$$

(where the number of values of j may depend on i, i.e. not all classes must have the same size).

Then the total corrected sum of squares is

$$\sum_i \sum_j (y_{ij} - y_{\bullet\bullet})^2.$$

The intra-class sum of squares is

$$\sum_i \sum_j (y_{ij} - \bar{y}_{i\bullet})^2).$$

The inter-class sum of squares is

$$\sum_i \sum_j (\bar{y}_{i\bullet} - \bar{y}_{\bullet\bullet})^2 = \sum_i \left(n_i (\bar{y}_{i\bullet} - \bar{y}_{\bullet\bullet})^2\right)$$

where n_i is the number of observations in the ith class.

Overview

- **Texture Classification (Problem 1)**
  - Feature Extraction (Law's Filter)
  - **Feature Dimension Reduction (PCA)**
  - Nearest Neighbor
  - Classifier (K-means, Random Forest, Support Vector Machine)

- Texture Segmentation (Problem 2)

- Image Feature Extractor (Problem 3)
  - SIFT
  - Bag of Words (BoW)

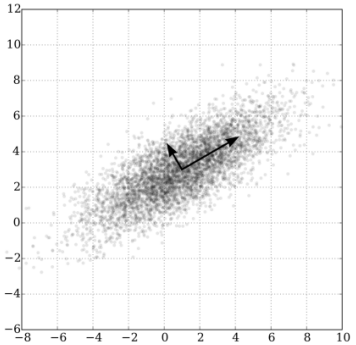
Problem 1 – Principle Component Analysis (PCA)

- Now you do PCA on 25D feature set to reduce its dimension to 3.

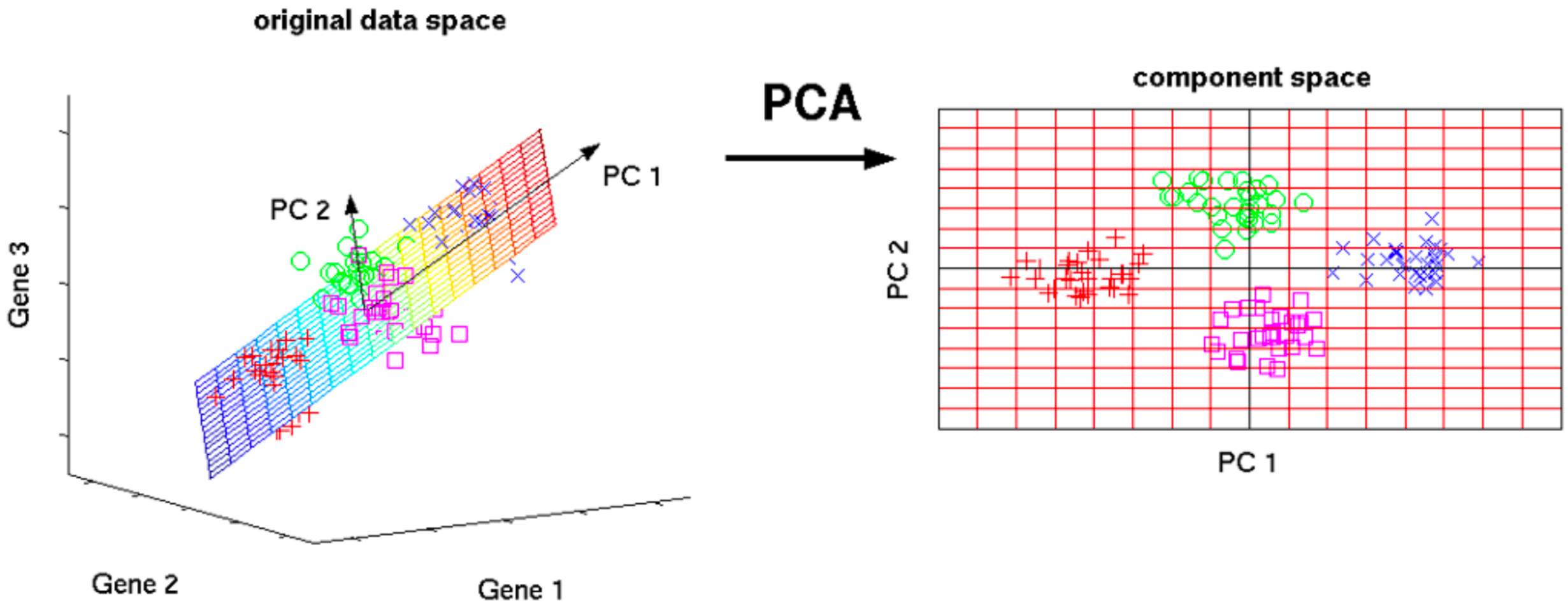
- You can use four different colors to show the 3D feature set of four classes of training images in one 3D plot.

- PCA:
  - Most commonly used method for dimensionality reduction. A.K.A:

    Karhunen–Loève transform, which is also known as Eigenvector transform
  - Orthogonal directions of greatest variance in data



Problem 1 – Principle Component Analysis (PCA)



3D →2D

Problem 1 – Principle Component Analysis (PCA)

Think further…:

- Observe the eigenvalues of the covariance matrix when you implement
  - Does it have many zeros (or close to zero) ?
  - How do those eigenvalues reflect the input feature's quality?
- Since orthogonal transform, resulting transformation gives uncorrelated features
  - Why is uncorrelated features desirable?

Problem 1 – Principle Component Analysis (PCA)

You can use Built-in PCA Object/Function:

- OpenCV: https://docs.opencv.org/ref/2.4/d3/d8d/classcv_1_1PCA.html
- Matlab: https://www.mathworks.com/help/stats/pca.html

For the following allowed built-in functions, pay attention to parameter usage.

Problem 1 – Principle Component Analysis (PCA)

You can use also implement PCA yourself:

1. Consider feature matrix $X_{n\times m}$, n is the number of sample/pixel and m is the number of features
2. Compute mean $(m_X)_j = \sum_{i=1}^{n} \frac{X_{i,j}}{n}, j = 1 \cdots m$
3. Subtract mean to get zero mean data matrix $Y_{n\times m}$
4. Compute SVD of Y: $Y = U \cdot S \cdot V^t$, where $U_{n\times n}, S_{n\times \mathrm{m}}, V_{\mathrm{m}\times m}$

Problem 1 – Principle Component Analysis (PCA)

- Dimensionality Reduction

1. Sort singular values in descending order of magnitude
2. Retain first k singular values if we want to reduce dimension from n to k, keep top k components
   - ✓ $V_R = \mathrm{V}[:,:\mathrm{k}]$
3. Form dimension reduced matrix $Y_R = X * V_R$

Overview

- **Texture Classification (Problem 1)**
  - Feature Extraction (Law's Filter)
  - Feature Dimension Reduction (PCA)
  - **Nearest Neighbor**
  - Classifier (K-means, Random Forest, Support Vector Machine)

- Texture Segmentation (Problem 2)

- Image Feature Extractor (Problem 3)
  - SIFT
  - Bag of Words (BoW)

Problem 1 – Nearest Neighbor

Now that you have the 3D representation of the train features, you need to classify test images.

For each test image:

1. Do the same analysis to get 3D feature.
2. Calculate the Mahalanobis Distance of the 3D feature to the 3D feature set of each class (grass, brick, blanket, and rice).
3. Predict label. (The class with the least distance would be predicted.)

Please include the error rate of predictions in your report.

Problem 1 – Mahalanobis Distance

The Mahalanobis Distance of an observation $\vec{x} = (x_1, x_2, x_3, \ldots, x_N)^T$ from a set of observations with mean $\vec{\mu} = (\mu_1, \mu_2, \mu_3, \ldots, \mu_N)^T$ and a covariance matrix S is defined as:

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}.$$

Reference: https://en.wikipedia.org/wiki/Mahalanobis_distance

Problem 1 – Covariance Matrix

Covariance Matrix of two matrices X and Y:

The formula for computing the covariance of the variables X and Y is

$$\mathrm{COV} = \frac{\sum_{i=1}^{n}(X_i - \bar{x})(Y_i - \bar{y})}{n-1},$$

with $\bar{x}$ and $\bar{y}$ denoting the means of X and Y, respectively.

Covariance Matrix of X:

$$S = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})'(X_i - \bar{X}) \ ,$$

n is the number of observations.

Example:

$$\mathbf{X} = \begin{bmatrix} 4.0 & 2.0 & 0.60 \\ 4.2 & 2.1 & 0.59 \\ 3.9 & 2.0 & 0.58 \\ 4.3 & 2.1 & 0.62 \\ 4.1 & 2.2 & 0.63 \end{bmatrix}$$

$$\bar{\mathbf{x}} = \begin{bmatrix} 4.10 & 2.08 & 0.604 \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} 0.025 & 0.0075 & 0.00175 \\ 0.0075 & 0.0070 & 0.00135 \\ 0.00175 & 0.00135 & 0.00043 \end{bmatrix}$$

Reference: https://www.itl.nist.gov/div898/handbook/pmc/section5/pmc541.htm

Overview

- **Texture Classification (Problem 1)**
  - Feature Extraction (Law's Filter)
  - Feature Dimension Reduction (PCA)
  - Nearest Neighbor
  - **Classifier (K-means, Random Forest, Support Vector Machine)**

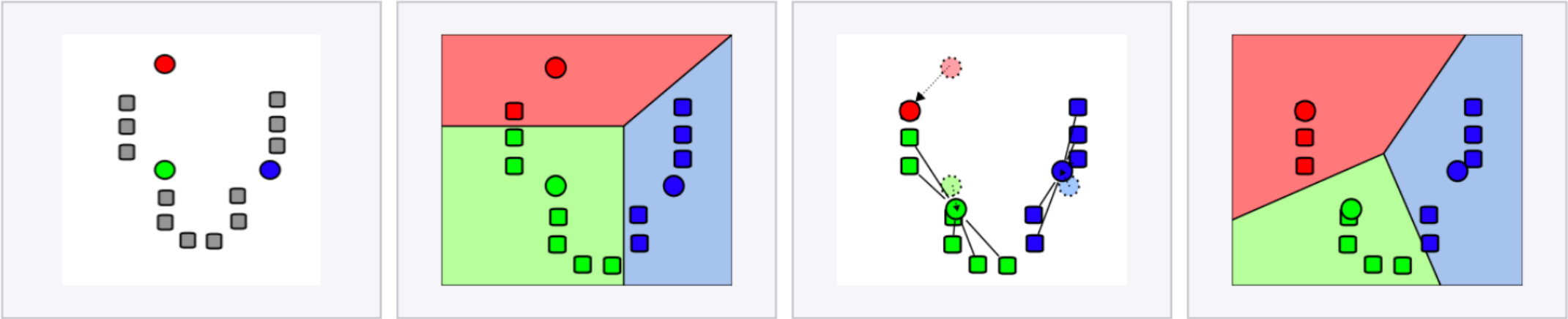
- Texture Segmentation (Problem 2)

- Image Feature Extractor (Problem 3)
  - SIFT
  - Bag of Words (BoW)

Problem 1 – K-means

- K-means: a type of unsupervised data analysis algorithm

Demonstration of the standard algorithm



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

Problem 1 – K-means

- Animation:

  http://shabal.in/visuals/kmeans/2.html

Problem 1 – K-means

You can use Built-in K-means Object/Function:

- OpenCV: https://docs.opencv.org/2.4/modules/core/doc/clustering.html?highlight=kmeans
- Matlab: https://www.mathworks.com/help/stats/kmeans.html

Problem 1 – K-means

You may also implement K-means yourself:

1. Initialize the cluster centroids by randomly select K samples from the data
2. Calculate the distance between samples and centroids to check their similarity

   ✓ $d(E_i, C_k) = \sqrt[2]{\sum_{j=1}^{m}\left(E_{i,j} - C_{k,j}\right)^2}, where\ i = 1, \cdots, n\ and\ k = 1, \cdots, K$

   ✓ Except Euclidean distance, you can try Mahalanobis distance, etc.

3. Compare the distance of the sample to K centroids to see which centroid has the minimum distance to it, label the sample to that cluster

   ✓ $L_i = \underset{k \in \{1,2,\ldots,K\}}{\operatorname{argmin}}\ d(E_i, C_k)$

Problem 1 – K-means

4. Obtain the new cluster centers

✓ $C_k = \frac{\sum_{i=1}^{n} l(L_i=k)E_i}{\sum_{i=1}^{n} l(L_i=k)}, where\ l(L_i = k) = \begin{cases} 1 & if\ L_i = k \\ 0 & if\ L_i \neq k \end{cases}$

5. Repeat step 2-5 until change in cluster centroids are effectively negligible

Depends on initialization very much!

Problem 1 – K-means

How to select initial cluster centers?

- Randomly
- Use mean-std and mean+std as first two clusters

Problem 1 – Supervised Classifier

- Random Forest (RF)
- Support Vector Machine (SVM)

1. First, label training images by yourself.
   Bark: 0
   Brick: 1
   Knit: 2
   Stone: 3
2. Train RF and SVM using training images and labels.
3. Predict labels for test images and calculate error rate.

Problem 1 – Random Forest

You can use Built-in Random Forest Object/Function:

- OpenCV: https://docs.opencv.org/2.4/modules/ml/doc/random_trees.html
- Matlab: https://www.mathworks.com/help/stats/treebagger.html
- Python (Just for your reference and understanding):

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Problem 1 – SVM

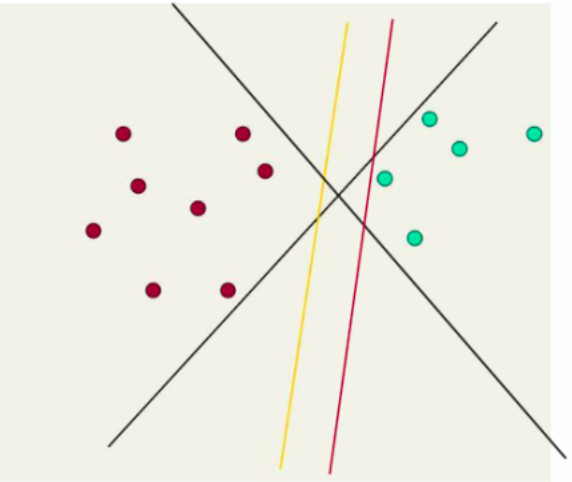
You can use Built-in SVM Object/Function:

- OpenCV: https://docs.opencv.org/2.4/modules/ml/doc/support_vector_machines.html
- Matlab: https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html#bsr5b6n
- Other resources:
  https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
  https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html
  https://www.csie.ntu.edu.tw/~cjlin/libsvm/
  https://en.wikipedia.org/wiki/Support-vector_machine
  http://web.stanford.edu/class/cs276/

Problem 1 - SVM

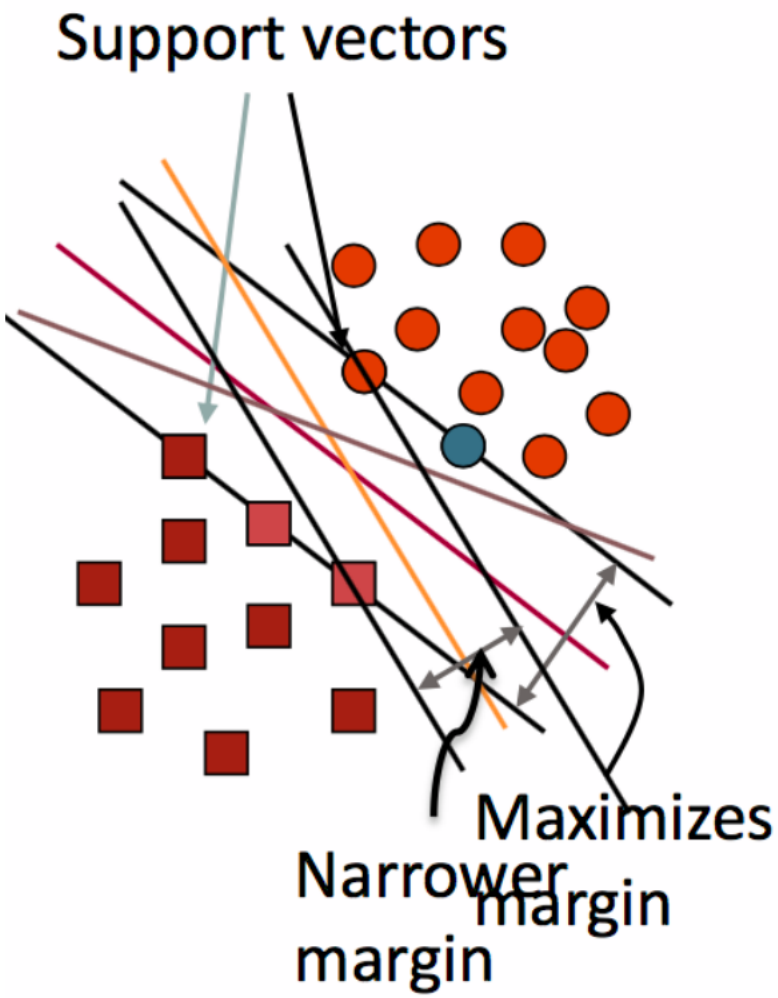
- **Linear Classifiers**
  - Lots of possible solutions for a, b, c, some methods find a separating hyperplane, but not the optimal one
  - SVM finds an optimal solution
    - ✓ Maximize the distance between the hyperplane and the "difficult points" close to decision boundary
    - ✓ If there are no points near the decision surface, then there are no very uncertain classification decisions

These lines represent the decision boundary:
$ax + by - c = 0$



Problem 1 - SVM

- SVMs maximize the margin around the separating hyperplane, A.K.A. large margin classifiers
- The decision function is fully specified by a subset of training samples, the support vectors
- Solving SVMs is a quadratic programming problem



Problem 1 - SVM

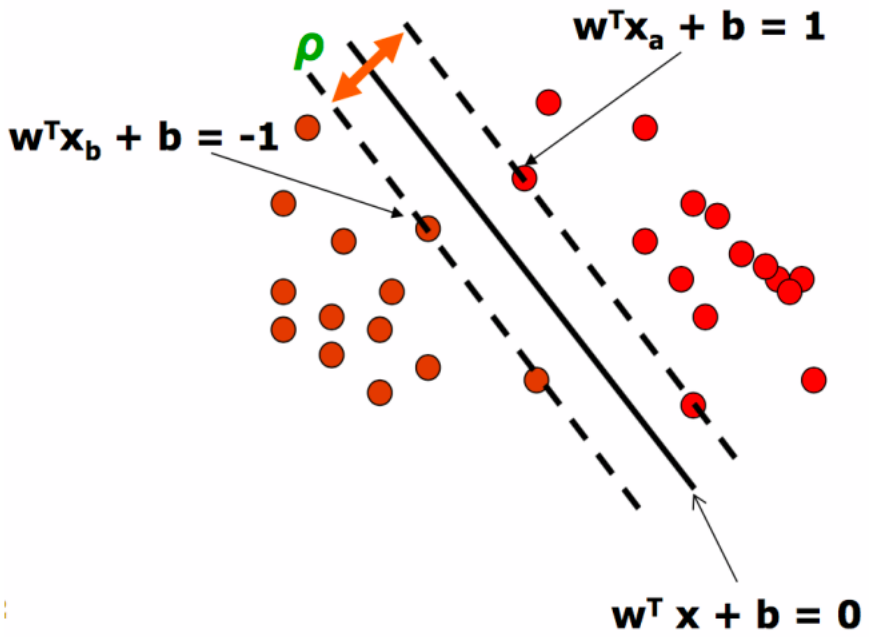
Formalization

- w: decision hyperplane normal vector
- x_i: data point i
- y_i: class of data point i (+1 or -1) **Not 1/0**
- Classifier is: $f(x_i) = sign(w^T x_i + b)$
- Functional margin of x_i is: $y_i(w^T x_i + b)$

Problem 1 - SVM

Geometric margin

- Distance from example to the separator is $r = y\frac{w^T x+b}{\|w\|}$
- Examples closest to the hyperplane are support vectors
- Margin ρ of the separator is the width of separation between support vectors of classes



Problem 1 - SVM

Loss function

- Hinge loss $\left[\frac{1}{n}\sum_{i=1}^{n} max\left(0, 1 - y_i(\vec{w} \cdot \vec{x_i} + b)\right)\right] + \lambda\|\vec{w}\|^2$

Where the parameter λ determines the tradeoff between increasing the margin-size and ensuring that the samples lie on the correct side of the margin.

Overview

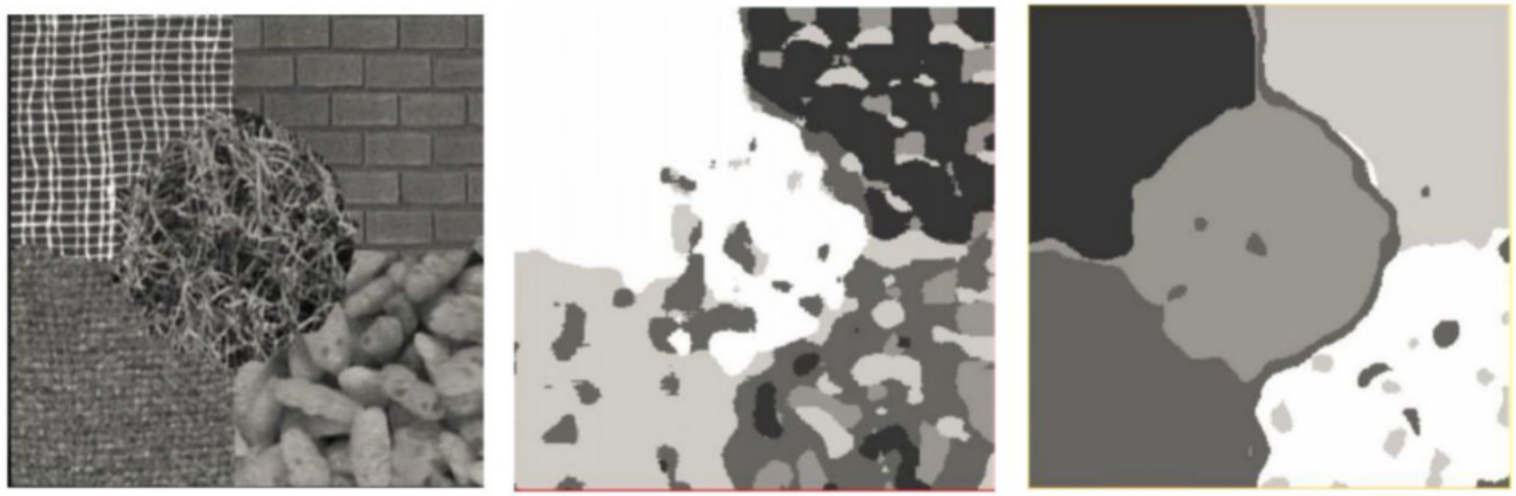
- Texture Classification (Problem 1)
  - Feature Extraction (Law's Filter)
  - Feature Dimension Reduction (PCA)
  - Nearest Neighbor
  - Classifier (K-means, Random Forest, Support Vector Machine)

- **Texture Segmentation (Problem 2)**

- Image Feature Extractor (Problem 3)
  - SIFT
  - Bag of Words (BoW)

Problem 2

Texture segmentation example:



Window size selection?

Problem 2

- You extract features same as problem 1. But you should keep in mind:

  1. After applying law filters and finding 25 image responses, the average energy for each pixel in each image response is calculated within a window (of your selected size), not within the whole image.

  2. After obtaining the 25D feature vector for each pixel, you normalize it to $L_5^T L_5$ and obtain the 24D feature vector.

- Apply K-means clustering to the feature set of size (number of pixels, 24D feature) to divide the pixels into 5 clusters.

- You can use values 0, 63, 127, 191, and 255 in order to show segmentation results.

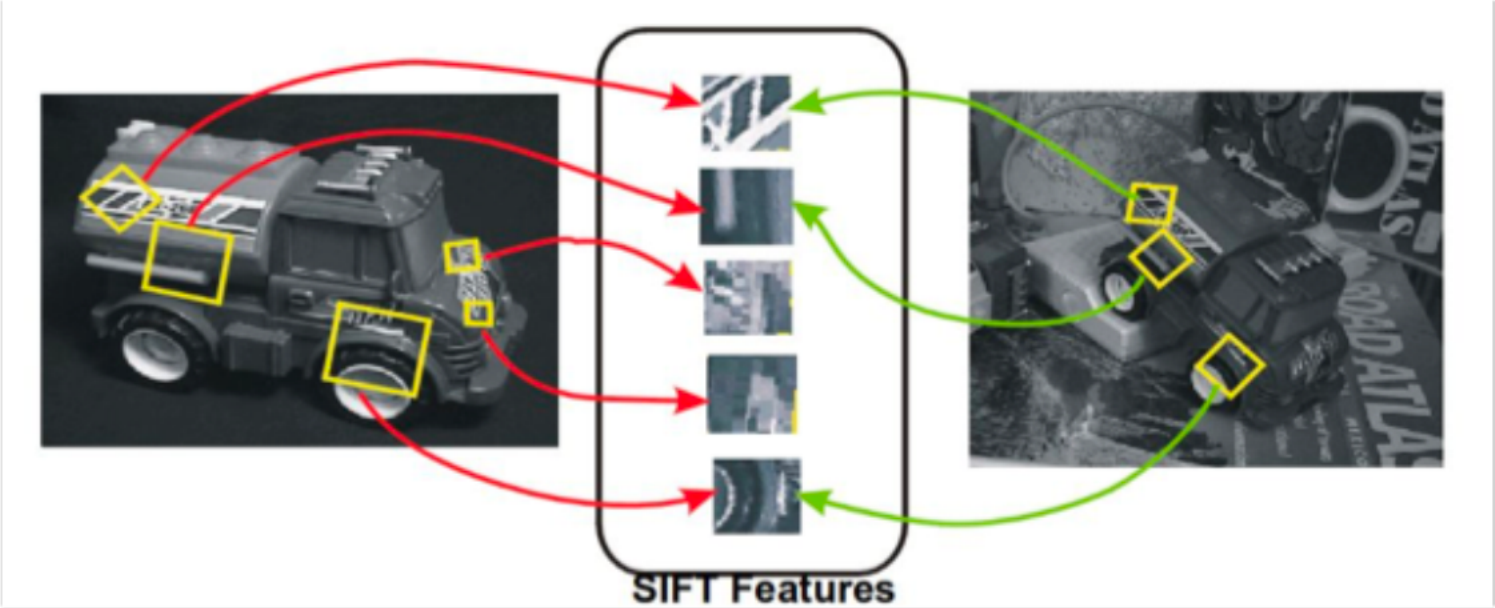
Overview

- Texture Classification (Problem 1)
  - Feature Extraction (Law's Filter)
  - Feature Dimension Reduction (PCA)
  - Nearest Neighbor
  - Classifier (K-means, Random Forest, Support Vector Machine)
- Texture Segmentation (Problem 2)
- **Image Feature Extractor (Problem 3)**
  - **SIFT**
  - Bag of Words (BoW)

Problem3 - SIFT

Idea:

Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale and other imaging parameters



Problem3 - SIFT

Procedure:

- **Scale-space extrema detection**
- Keypoint localization
- Orientation assignment
- Keypoint descriptor

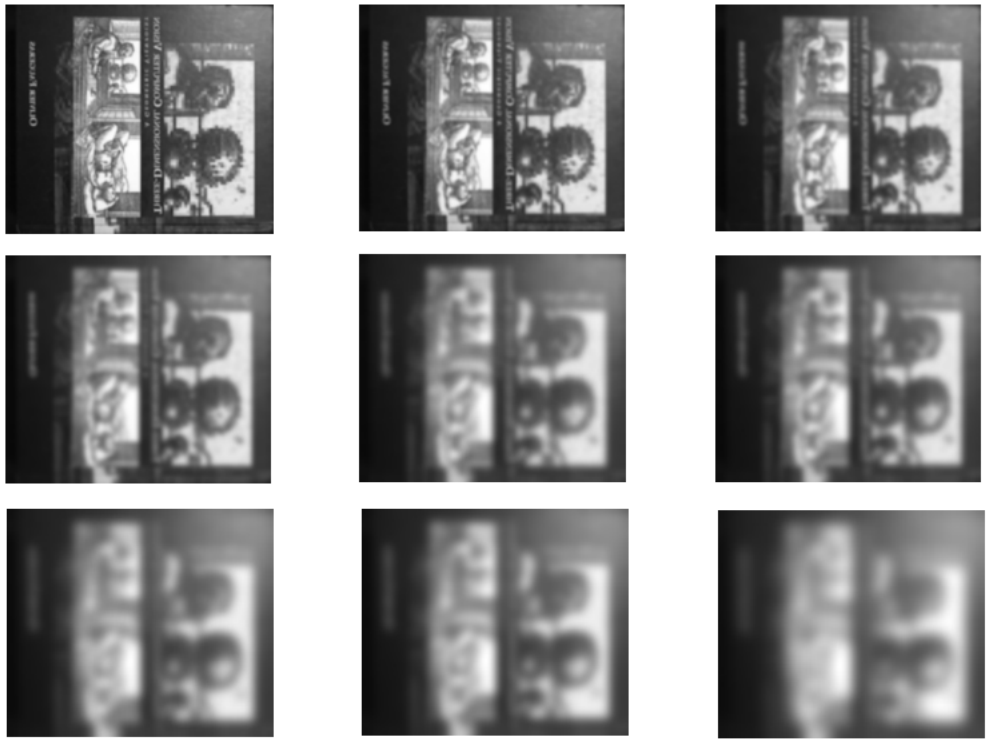
Problem3 – Scale-space extrema detection

- Goal: potential locations for finding features
- Interest points: local extrema in scale-space of the image
- Scale: sigma value in Gaussian function

Problem3 – Scale-space extrema detection

Smoothed image: $\mathrm{L}(x, y, \sigma) = G(x, y, \sigma) * I(x, y, \sigma)$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$



Problem3 – Scale-space extrema detection

An approximation: $\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x,y,k\sigma)-G(x,y,\sigma)}{k\sigma-\sigma}$

→ $G(x,y,k\sigma) - G(x,y,\sigma) \approx (\mathrm{k}-1)\sigma^2\nabla^2 G$

→ extrema location depends on $\nabla^2 G = 0$, k is constant, σ^2 is scale normalization (Typical values: $\sigma = 1.6; k = \sqrt{2}$)

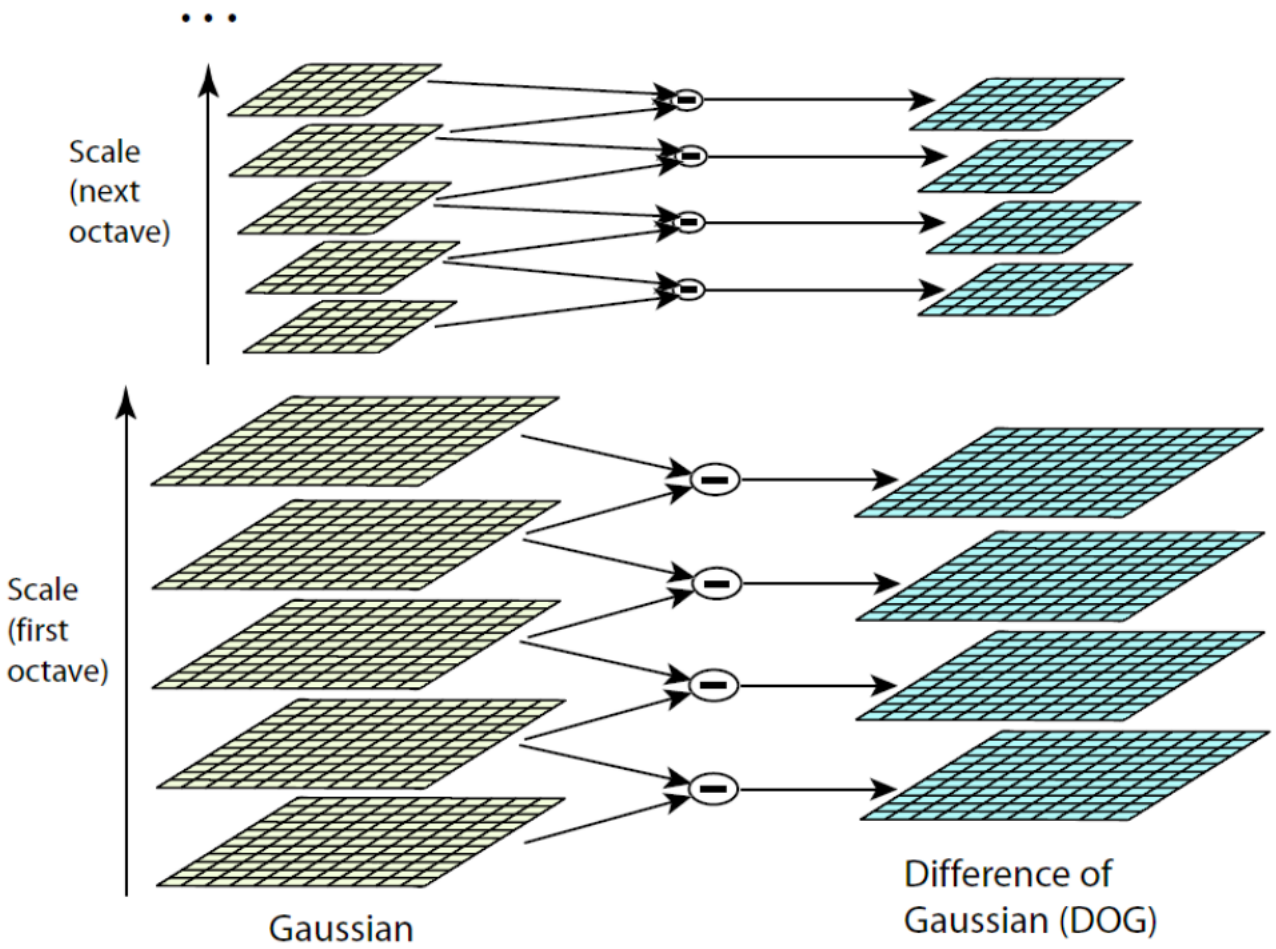
→ scale-invariant

Problem3 – Scale-space extrema detection

Difference-of-Gaussian: $D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$

Problem3 – Scale-space extrema detection

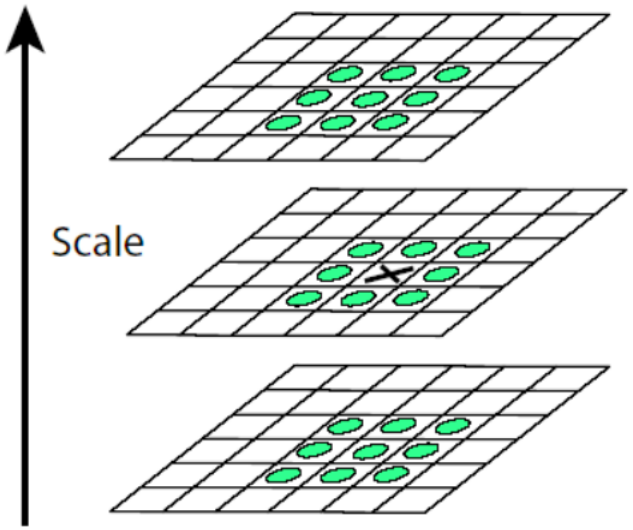
Scale space construction:



Problem3 – Scale-space extrema detection

Find extrema:

Sample point is selected only if it is a minimum or a maximum of these points



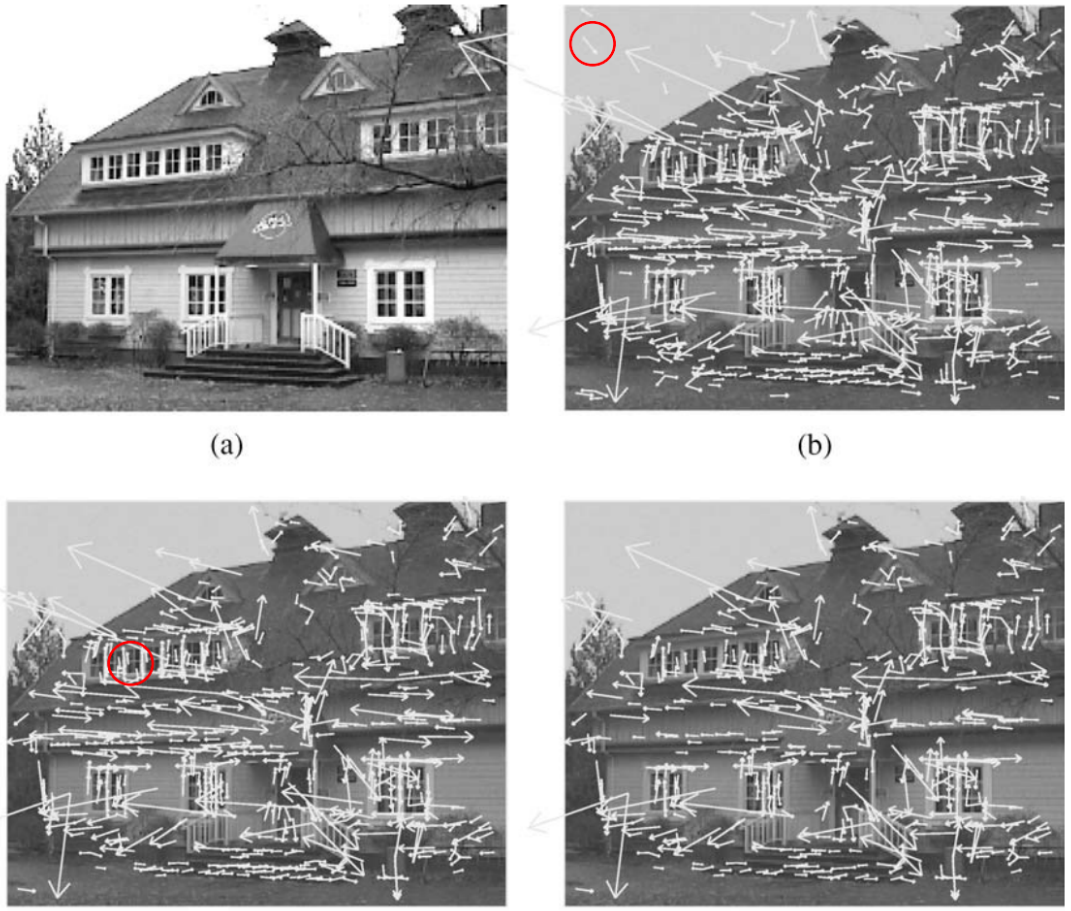
Problem3 – SIFT

Procedure:

- Scale-space extrema detection
- **Keypoint localization**
- Orientation assignment
- Keypoint descriptor

Problem3 – Keypoint localization

- Eliminate low contrast candidates which are chosen from extrema detection
- Eliminate edge locations



(a) (b) (c) (d)

Problem3 – Keypoint localization

- Eliminate low contrast candidates
  - Talyor expansion: $D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$, where $\mathbf{x} = (x, y, \sigma)^T$
  - Taking derivative, the location of extrema is $\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}}$
  - $D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}}$
  - Discard extrema with $|D(\hat{\mathbf{x}})| < 0.03$

Problem3 – Keypoint localization

- Eliminate edge locations because DoG has a strong response along edges
  - Use ratio of principle curvatures to throw out poorly defined peaks because poorly defined peaks has a large principal curvature across the edge but a small one in the perpendicular direction.
  - The principle curvatures comes from Hessian matrix: $\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$
  - Compute eigenvalues of H: $\alpha \; and \; \beta$, where $\alpha > \beta$
  - Let $\alpha = r\beta$, then $\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r+1)^2}{r}$ depends only on r
  - Throw keypoints that have a ratio greater than $r = 10$

Problem3 – SIFT

Procedure:

- Scale-space extrema detection
- Keypoint localization
- **Orientation assignment**
- Keypoint descriptor

Problem3 – Orientation Assignment

- Goal: to achieve rotation invariance
  - Compute central derivatives, gradient magnitude and direction of L (smooth image) at the scale of key point (x, y)
  - $m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$
  - $\theta(x,y) = tan^{-1}((L(x+1,y) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$

Problem3 – Orientation Assignment

- A histogram is formed by quantizing the orientations into 36 bins
- Peaks in the histogram correspond to the orientations of the patch
- Compute relative orientation and magnitude in a 16×16 neighborhood at key point

Problem3 – SIFT

Procedure:

- Scale-space extrema detection
- Keypoint localization
- Orientation assignment
- **Keypoint descriptor**

Problem3 – Keypoint descriptor

- The magnitude is weighted by a Gaussian window
- Form weighted histogram (8 bin) for 4×4 regions by summing the gradient magnitudes near that direction within the region
- Distribute each sample to adjacent bins by trilinear interpolation
- Concatenate 16 histograms in one long vector of 128 dimensions
- The feature vector is normalized to unit length to reduce effect of illumination change
- Cap each element to be 0.2, normalize again to reduce non-linear illumination change

Problem3 – Keypoint descriptor

Example: 8×8 neighborhood and 4 bins

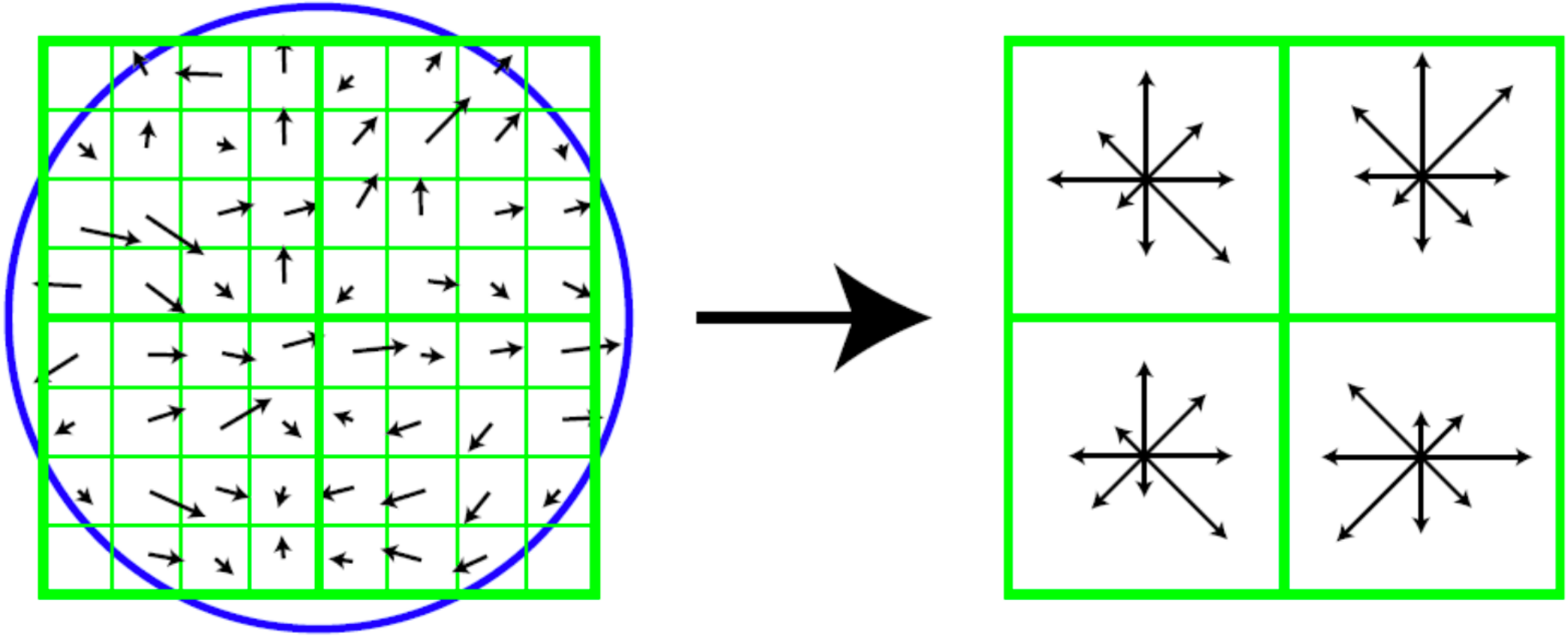


Image gradients

Keypoint descriptor

Problem3 – SIFT

- Locality: features are local, so robust to occlusion and clutter
- Distinctiveness: individual features can be matched to a large database of objects
- Quantity: many features can be generated for even small objects
- Efficiency: close to real-time performance

Problem3 – SIFT

Applications: object recognition

- The SIFT features of training images are extracted and stored
- For the query image, extract SIFT feature and do nearest neighbor search
- Key points, geometry verification

Problem3 – SIFT

Example:

Problem3 – SIFT

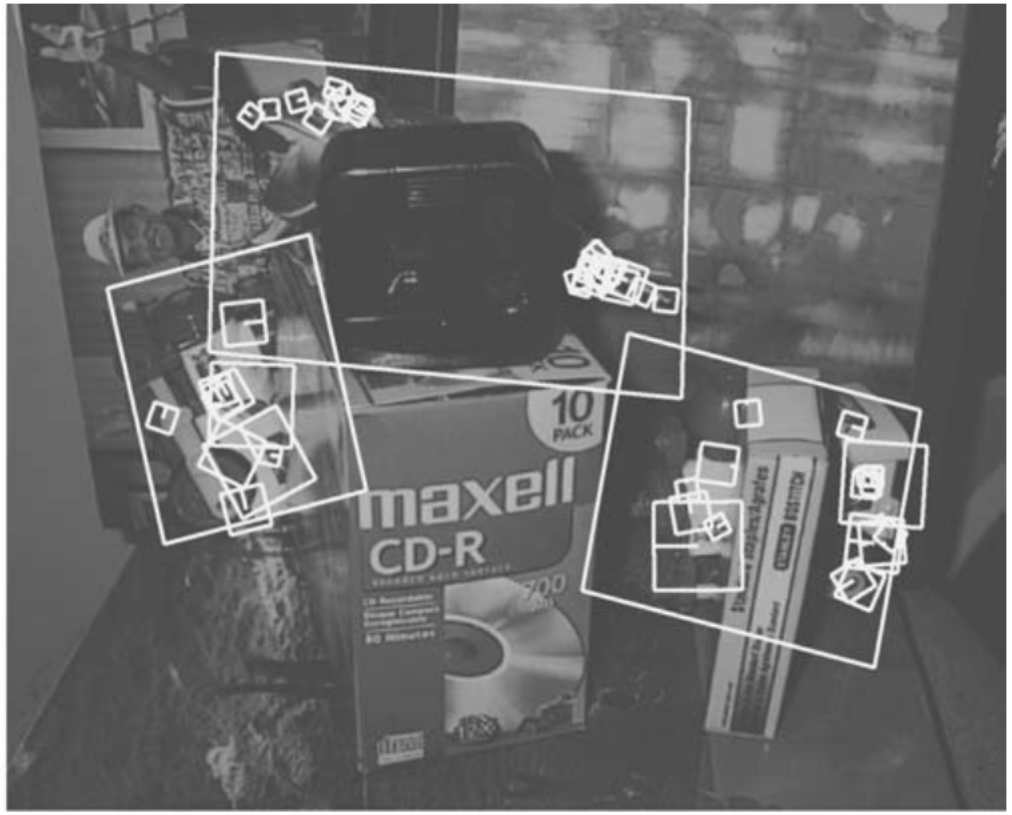
Example:

Problem3 – SIFT

Example:

Problem3 – SIFT

Example:



Problem3 – SIFT

Extensions:

- PCA-SIFT
  - Working on 41*41 patches
  - Using PCA to project it to 20 dimensions
- SURF
  - Approximate SIFT
  - Works almost equally well – Very fast

Problem3 – SIFT

- Matlab: https://www.vlfeat.org/index.html
- OpenCV:
  - https://docs.opencv.org/3.4/da/df5/tutorial_py_sift_intro.html
  - https://docs.opencv.org/3.4/d5/d3c/classcv_1_1xfeatures2d_1_1SIFT.html

Overview

- Texture Classification (Problem 1)
  - Feature Extraction (Law's Filter)
  - Feature Dimension Reduction (PCA)
  - Nearest Neighbor
  - Classifier (K-means, Random Forest, Support Vector Machine)

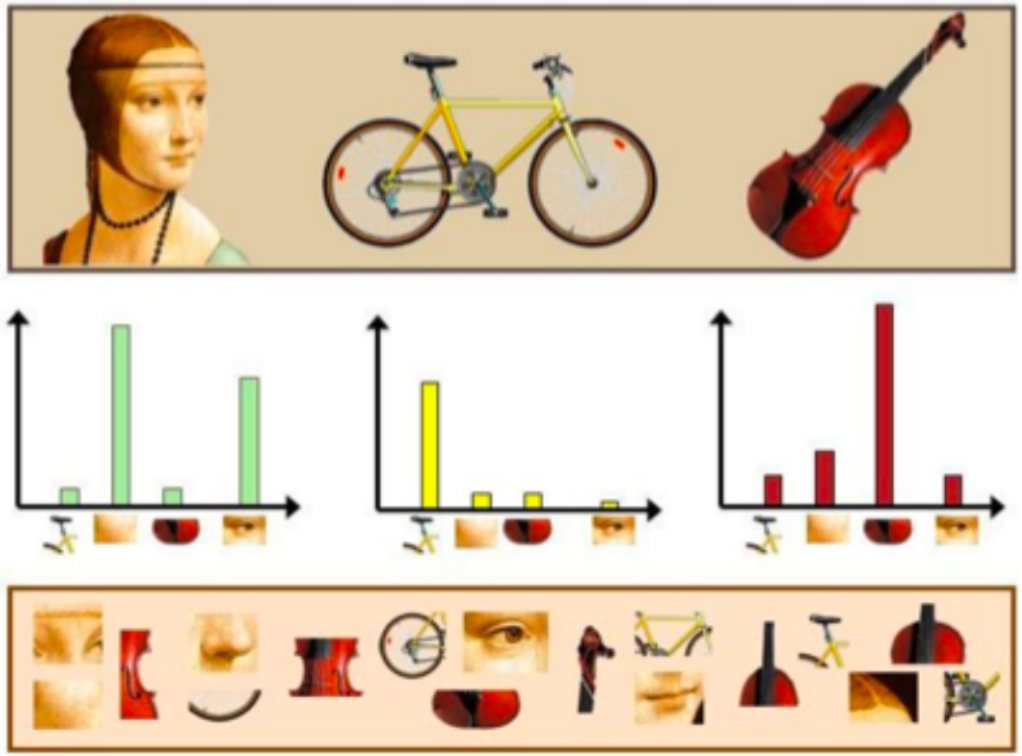
- Texture Segmentation (Problem 2)

- **Image Feature Extractor (Problem 3)**
  - SIFT
  - **Bag of Words (BoW)**

Problem3 - BoW

Idea:

Each image can be represented as a histogram of SIFT feature vectors

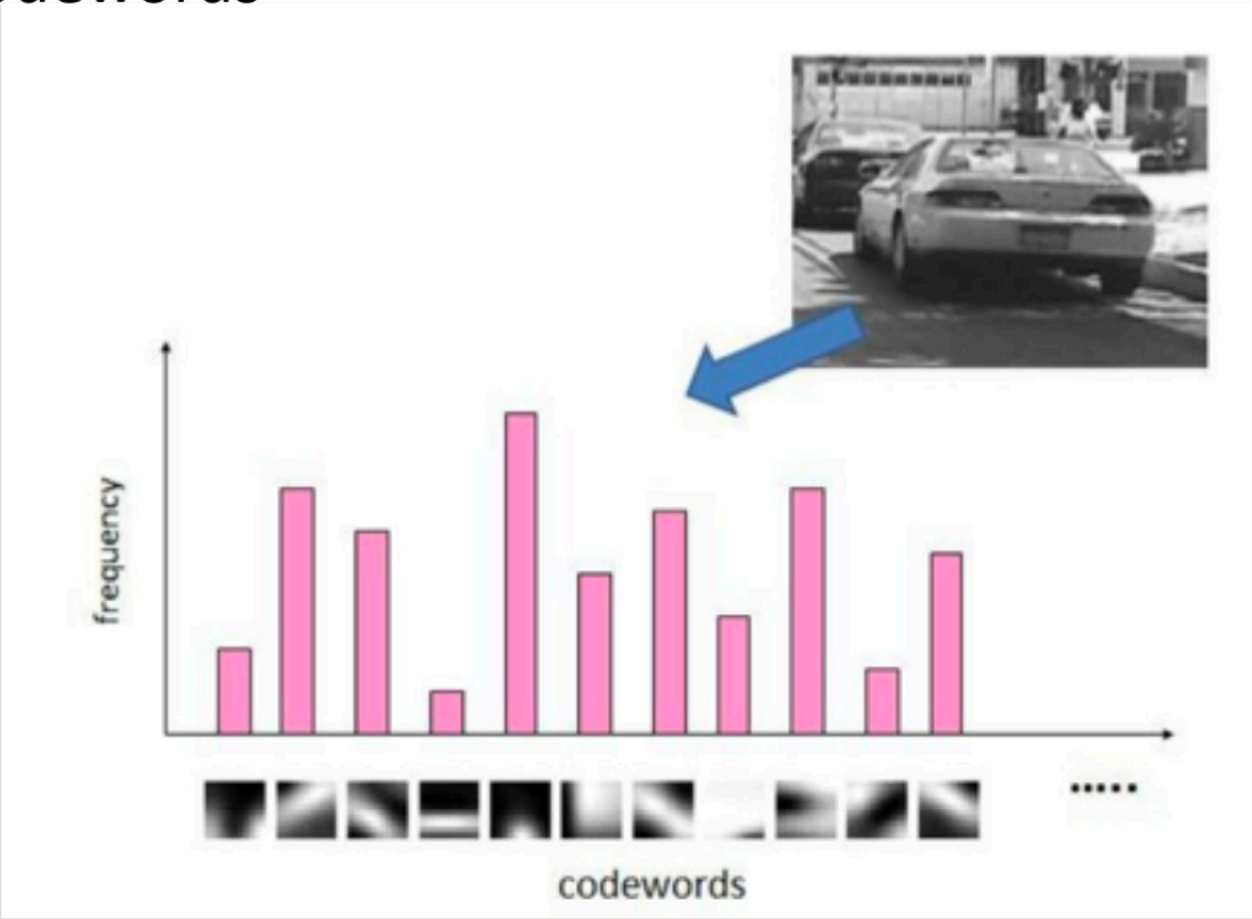


Vocabulary of visual words

Problem3 - BoW

Features:

Histogram of codewords



Problem3 - BoW

Procedures:

- Extract SIFT features from all four image.
- Do PCA to reduce feature dimension from 128 to 20.
- Apply K-means clustering to create dictionary's visual words (k=8)
- Cluster centers are the words (k=8 words)
- For each feature in each of four images, calculate the nearest neighbor in the dictionary.
- For each of four images, build a histogram of length k (frequency of words in that image).
- Compare codeword histogram intersection of the three other images with Dog_3 image codeword histogram

Problem3 – Histogram Intersection

In order to match the three images' histograms with Dog_3 histogram, you need to use “Histogram Intersection” method:

$$Similarity\ Index = \frac{\sum_{j=1}^{k} \min(I_j, M_j)}{\sum_{j=1}^{k} max(I_j, M_j)}$$

Where I_j is one of the three image’s codewords histograms, M_j is the Dog_3 codeword histogram, and k is 8.

References

- Min Zhang, EE569 Spring 2020 discussion 8 & 9
- https://en.wikipedia.org/wiki/Mahalanobis_distance
- https://www.itl.nist.gov/div898/handbook/pmc/section5/pmc541.htm
- https://mpatacchiola.github.io/blog/2016/11/12/the-simplest-classifier-histogram-intersection.html
- https://robo.fish/wiki/images/5/58/Image_Features_From_Scale_Invariant_Keypoints_Lowe_2004.pdf
- http://www.cse.psu.edu/~rtc12/CSE486/lecture11.pdf
- https://courses.cs.washington.edu/courses/cse455/10au/notes/SIFT.ppt
- https://en.wikipedia.org/wiki/Scale-invariant_feature_transform#Scale-invariant_feature_detection

Q
A