Spring 2019: Venu: Haag 315, Time: M/W 4-5:15pm

ECE 5582 Computer Vision
Lec 12: Part I Review

Zhu Li
Dept of CSEE, UMKC
Office: FH560E, Email: lizhu@umkc.edu, Ph: x 2346.
http://l.web.umkc.edu/lizhu
Quiz-1

☐ Covers what we reviewed today

☐ Format:
  - True or False
  - Multiple Choices
  - Problem Solving
  - Extra Credit (25%)

  - cheating sheet need to be turned in with name and id number
  - cheating sheet will also be graded for up to 10 pts

☐ Relax 😊
  - Quiz is actually on me.
Cheating sheet is also graded.

- Up to 10 pts extra
- A nice way to summarize and organize/indexing your knowledge base.
- Cheating sheet + HW = what you really get from the class.
Quiz-1

- Review part I: Image Features & Classifiers
  - Image Formation: Homography
  - Color Space & Color Features
  - Filtering and Edge Features
  - Harris Detector
  - SIFT
  - Aggregation: BoW, VLAD, Fisher, and Supervector
  - Image Retrieval System Metrics: TPR/Precision, FPR, Recall, mAP
  - Classification: kNN, GMM Bayesian, SVM
Classification in Image Retrieval

- An image retrieval pipeline (hand crafted features)

- Image Formation
  - Homography, Color space

- Feature Computing
  - Color histogram, Filtering, Edge Detection, HoG, Harris Detector, SIFT

- Feature Aggregation
  - Bow, VLAD, Fisher Vector Supervector

- Classification
  - Knowledge / Data Base

Image Formation - Geometry

Intrinsic Assumptions
• Unit aspect ratio
• Optical center at (0,0)
• No skew of pixels

Extrinsic Assumptions
• No rotation
• Camera at (0,0,0)

\[
\begin{align*}
\mathbf{x} &= \mathbf{K} \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{X} \\
\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} &= \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
\end{align*}
\]
In general, homography $H$ maps 2d points according to, $x' = Hx$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Up to a scale, as $[x, y, w] = [sx, sy, sw]$, so $H$ has 8 DoF

**Affine Transform:**

- 6 DoF: Contains a translation $[t_1, t_2]$, and invertable affine matrix $A_{2x2}$

$$H = \begin{bmatrix} a_1 & a_2 & t_1 \\ a_3 & a_4 & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

**Similarity Transform,**

- 4 DoF: a rigid transform that preserves distance if $s=1$:

$$H = \begin{bmatrix} s \cos(\theta) & -s \sin(\theta) & t_1 \\ s \sin(\theta) & s \cos(\theta) & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$
VL_FEAT Implementation

VL_FEAT has an implementation:

```matlab
[f1,d1] = vl_sift(im1g);  
[f2,d2] = vl_sift(im2g);  

[matches, scores] = vl_ubcmatch(d1,d2);  
umMatches = size(matches,2);  

X1 = f1(1:2,matches(:,1)); X1(3,:) = 1;  
X2 = f2(1:2,matches(:,2)); X2(3,:) = 1;  

% RANSAC with homography model  

clear H score ok;  
for t = 1:100  
    % estimate homography: select a random subset of matches  
    subset = vl_colsubset(1:numMatches, 4);  
    A = [] ;  
    for i = subset  
        A = cat(1, A, kron(X1(:,i)', vl_hat(X2(:,i))));  
    end  
    [U,S,V] = svd(A);  
    H(t) = reshape(V(:,9),3,3);  

    % score homography  
    X2_ = H(t) * X1;  
    du = X2_(1,:)/X2_(3,:); du = X2(1,:)/X2(3,:);  
    dv = X2_(2,:)/X2_(3,:); dv = X2(2,:)/X2(3,:);  
    ok(t) = (du+.2*du + dv+.2*dv) < 6*6;  
    score(t) = sum(ok(t));  
end  

[score, best] = max(score);  
H = H(best);  
ok = ok{best};
```

627 tentative matches

580 (92.5%) inliner matches out of 627

Mosaic
Image Formation – Color Space

- **RGB Color Space:**
  - 8 bit Red, Green, Blue Channels
  - Mixed together

- **YUV/YCbCr Color Space**

  ![YUV/YCbCr Formula](image)

- **HSV Color Space**
  - Hue: color in polar coord
  - Saturation
  - Value
Color Histogram – Fixed Codebook

Color Histogram:
- **Fixed code book** to generate histogram
- Example: MPEG 7 Scalable Color Descriptor
- HSV Color Space, 16x4x4 partition for bins
- Scalability thru quantization and Haar Transform
Adaptive Color Histogram

- **No fixed code book**
- Example: MPEG 7 Dominant Color Descriptor
- Distance Metric
- Not index-able

\[
D^2(F_1, F_2) = \sum_{i=1}^{N_1} p_{1,i}^2 + \sum_{j=1}^{N_2} p_{2,j}^2 - \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} 2a_{1,i,2,j}p_{1,i}p_{2,j}
\]

\[a_{k,l} : \text{similarity coefficient between two colors } c_k \text{ and } c_l\]

\[
a_{k,l} = \begin{cases} 
1 - d_{k,l} / d_{\text{max}} & d_{k,l} \leq T_d \\
0 & d_{k,l} > T_d
\end{cases}
\]

\[d_{k,l} : \text{Euclidean distance between two color } c_k \text{ and } c_l\]

\[d_{k,l} = ||c_k - c_l||\]

\[T_d : \text{maximum distance for two colors to be considered similar,}\]

\[d_{\text{max}} = \alpha T_d, \quad \alpha \text{ values 1.0-1.5, } T_d \text{ values 10-20}\]
Image Filters

- **Basic Filter Operations**

  \[ H \ast F = G \]

  - \( H \) is a 3x3 kernel with values 1/9.
  - \( F \) is the input image.
  - \( G \) is the output image.

- **Region of Support**

  \[ N_1 \times M_1 \ast N_2 \times M_2 = (N_1 + N_2 - 1) \times (M_1 + M_2 - 1) \]
Linear Filters are:

- **Shift-Invariant:**
  \[ f(m-k, n-j)*h = g(m-k, n-j), \text{ if } f* h = g \]

- **Associative:**
  \[ f*h_1*h_2 = f*(h_1*h_2) \]
  this can save a lot of complexity

- **Distributive:**
  \[ f*h_1 + f*h_2 = f*(h_1+h_2) \]
  useful in SIFT’s DoG filtering.
The need of smoothing

- Gaussian smoothing
- Combine differentiation and Gaussian

\[
\frac{d}{dx}(f * h) = f * \frac{d}{dx}h
\]

Sigma = 50

derivative of Gaussian \((x)\)

\[
\frac{\partial}{\partial x} h_\sigma(u, v)
\]

\[
\frac{d}{dx} h
\]

\[
f * \frac{d}{dx} h
\]
Image Gradient from DoG Filtering

- **Image Gradient (at a scale):**
  \[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]
  The gradient points in the direction of most rapid increase in intensity

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right] \]
\[ \nabla f = \left[ 0, \frac{\partial f}{\partial y} \right] \]

The **edge strength** is given by the gradient magnitude:

\[ ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \]

The gradient direction is given by:

\[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]
HOG – Histogram of Oriented Gradients

- Generate local histogram per block
  - Size of HoG: \( n \times m \times 4 \times 9 = 36\text{nm} \), for \( n \times m \) cells.

\[
f = (h_1^1, ..., h_9^1, \quad h_1^2, ..., h_9^2, \quad h_1^3, ..., h_9^3, \quad h_1^4, ..., h_9^4)
\]

- Cell size

- Binary voting

- Magnitude voting

**Harris Detector**

**Smoothed 2\textsuperscript{nd} moment:**

\[
\mu(\sigma_1, \sigma_D) = g(\sigma_1) * \begin{bmatrix}
I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\
I_x I_y(\sigma_D) & I_y^2(\sigma_D)
\end{bmatrix}
\]

1. Image derivatives (optionally blur first)

\[
det M = \lambda_1 \lambda_2
\]

trace \( M \) = \( \lambda_1 + \lambda_2 \)

2. Square of derivatives

3. Gaussian filter \( g(\sigma) \)

4. Harris Function – detect corners, no need to compute the eigen values

\[
R_{\text{harris}} = det[\mu(\sigma_1, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_1, \sigma_D))^2] = g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2
\]
Harris Detector Steps

- Response function -> Threshold -> Local Maxima
  - Harris Detector NOT scale invariant

$R(x,y)$
Scale Space Theory - Lindeberg

Scale Space Response via Laplacian of Gaussian

- The scale is controlled by $\sigma$

$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

$$g = e^{\frac{-(x+y)^2}{2\sigma}}$$

- Characteristic Scale:

image

$\sigma = 0.8r$  $\sigma = 1.2r$  $\sigma = 2r$

characteristic scale
ALP Scale Space Extrema Detection

ALP filtering scale space response as polynomials

- Scale Space Extrema: \((I * h)[430, 122, \sigma] \approx 2.85 \sigma^3 - 39.51 \sigma^2 + 172.65 \sigma - 172.61\)

- No Extrema (saddle point): \((I * h)[153, 356, \sigma] \approx 0.12 \sigma^3 - 1.06 \sigma^2 + 3.15 \sigma - 2.5\)
However, LoG computing is expensive – non-separable filter

SIFT: uses Difference of Gaussian filters to approximate LoG

- Separate-able, much faster
- Box Filter Approximation

Scale space construction
By Gaussian Filtering, and Image Difference
Peak Strength & Edge Removal

- **Peak Strength:**
  - Interpolate true DoG response and pixel location by Taylor expansion

\[
D(\hat{x}) = D + \frac{1}{2} \frac{\partial D^T}{\partial x} \hat{x}.
\]

- **Edge Removal:**
  - Re-do Harris type detection to remove edge on much reduced pixel set

\[
\frac{\text{Tr}(H)^2}{\text{Det}(H)} < \frac{(r + 1)^2}{r}.
\]
Voting for the dominant orientation

- Weighted by a Gaussian window to give more emphasis to the gradients closer to the center

Orientation of keypoint is approximately 25 degrees
SIFT Description

- Rotate to dominant direction, divide Image Patch (16x16 pixels) around SIFT point into 4x4 Cells
- For each cell, compute an 8-bin edge histogram
- Overall, we have 4x4x8=128 dimension
SIFT Matching and Repeatability Prediction

- **SIFT Distance – True Love Test**
  \[
  \frac{d(s^1_1, s^2_k)}{d(s^1_1, s^2_k)} \leq \theta
  \]

- **Not all SIFT are created equal…**
  - Peak strength (DoG response at interpolated position)

Combined scale/peak strength pmf
Outline

- HW-2
- Quiz-1

Review part I: Image Features & Classifiers

- Image Formation: Homography
- Color Space & Color Features
- Filtering and Edge Features
- Harris Detector
- SIFT

- Aggregation: BoW, VLAD, Fisher, and Supervector
- Image Retrieval System Metrics: TPR/Precision, FPR, Recall, mAP
- Classification: kNN, GMM Bayesian, SVM
Why Aggregation?

- Curse of Dimensionality

- Decision Boundary / Indexing
Histogram Aggregation: Bag-of-Words

- **Codebook:**
  - Feature space: $R^d$, k-means to get $k$ centroids, $\{\mu_1, \mu_2, \ldots, \mu_k\}$

- **BoW Hard Encoding:**
  - For $n$ feature points, $\{x_1, x_2, \ldots, x_n\}$ assignment matrix: $k \times n$, with column only 1-non zero entry
  - Aggregated dimension: $k$
Kernel Code Book Soft Encoding

- Kernel Affinity: \( K(x_j, \mu_k) = e^{-k|x_j - \mu_k|^2} \)
- Assignment Matrix: \( A_{j,k} = \frac{K(x_j, \mu_k)}{\sum_k K(x_j, \mu_k)} \)
- Encoding: \( k\)-dimensional: \( X(k) = \left( \frac{1}{n} \right) \sum_j A_{j,k} \)
VLAD- Vector of Locally Aggregated Descriptors

- Aggregate feature difference from the codebook
  - Hard assignment by finding the NN of feature \( \{x_k\} \) to \( \{\mu_k\} \)
  - Compute aggregated differences

- L2 normalize

- Final feature: \( k \times d \)
Fisher Vector Aggregation

- **FV encoding**
  - Gradient on the mean, for GMM component $k$, $j=1..D$

\[
\begin{align*}
    u_{jk} &= \frac{1}{N \sqrt{\pi_k}} \sum_{i=1}^{N} q_{ik} \frac{x_{ji} - \mu_{jk}}{\sigma_{jk}}, \\
    v_{jk} &= \frac{1}{N \sqrt{2\pi_k}} \sum_{i=1}^{N} q_{ik} \left[ \left( \frac{x_{ji} - \mu_{jk}}{\sigma_{jk}} \right)^2 - 1 \right]
\end{align*}
\]

- In the end, we have $2K \times D$ aggregation on the derivation w.r.t. the means and variances

\[
FV = [u_1, u_2, ..., u_K, v_1, v_2, ..., v_K]^T
\]

Normalize by Fisher Kernel

\[
\sigma = 0.8r
\]
Supervector Aggregation

Assuming we have UBM GMM model

\[ \lambda_{UBM} = \{P_k, \mu_k, \Sigma_k\}, \]

with identical prior and covariance

Then for two utterance samples a and b, with GMM models

- \[ \lambda_a = \{P_k, \mu_k^a, \Sigma_k\}, \]
- \[ \lambda_b = \{P_k, \mu_k^b, \Sigma_k\}, \]

The SV distance is,

\[ K(\lambda_a, \lambda_b) = \sum_k \left( \sqrt{P_k \Sigma_k^{-\frac{1}{2}}} \mu_k^a \right)^T \left( \sqrt{P_k \Sigma_k^{-\frac{1}{2}}} \mu_k^b \right) \]

It means the means of two models need to be normalized by the UBM covariance induced Mahalanobis distance metric

This is also a linear kernel function scaled by the UBM covariances
AKULA – Adaptive KLuster Aggregation

**AKULA Descriptor:**
- No fixed codebook!
- Outer loop: subspace optimization
- Inner loop: cluster centroids + SIFT count

\[ A_1 = \{ yc_1^{1,1}, yc_1^{1,2}, \ldots, yc_1^{1,k} ; pc_1^{1,1}, pc_1^{1,2}, \ldots, pc_1^{1,k} \} \]
\[ A_2 = \{ yc_2^{1,1}, yc_2^{1,2}, \ldots, yc_2^{1,k} ; pc_2^{1,1}, pc_2^{1,2}, \ldots, pc_2^{1,k} \} \]

**Distance metric:**
- Min centroids distance, weighted by SIFT count (kind of manifolds tangent distance)

\[
d(A_1, A_2) = \frac{1}{k} \sum_{j=0}^{k} d_{\text{min}}^1(j)w_{\text{min}}^1(j) + \frac{1}{k} \sum_{i=0}^{k} d_{\text{min}}^2(i)w_{\text{min}}^2(i)
\]

\[
d_{\text{min}}^1(j) = \min_i d_{j,i} \]
\[ d_{\text{min}}^2(i) = \min_j d_{j,i} \]

\[
w_{\text{min}}^1(j) = w_{j,i^*}, \quad i^* = \arg \min_i d_{j,i} \]
\[
w_{\text{min}}^2(i) = w_{j^*,i}, \quad j^* = \arg \min_j d_{j,i} \]
Image Retrieval Performance Metrics

- **True Positive Rate**
  \[ TPR = \frac{TP}{TP + FP} \]

- **False Positive Rate**
  \[ FPR = \frac{FP}{TP + FP} \]

- **Precision**
  \[ \text{Precision} = \frac{TP}{TP + FP} \]

- **Recall**
  \[ \text{Recall} = \frac{TP}{TP + FN} \]

- **F-Measure**
  \[ \text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]
Retrieval Performance: Mean Average Precision

- mAP measures the retrieval performance across all queries
  - Also called “average precision at seen relevant documents”
  - Determine precision at each point when a new relevant document gets retrieved
  - Use P=0 for each relevant document that was not retrieved
  - Determine average for each query, then average over queries

\[
MAP = \frac{1}{N} \sum_{j=1}^{N} \frac{1}{Q_j} \sum_{i=1}^{Q_j} P(doc_i)
\]

with:
- \( Q_j \) number of relevant documents for query \( j \)
- \( N \) number of queries
- \( P(doc_i) \) precision at \( i \)th relevant document
mAP example

- MAP is computed across all query results as the average precision over the recall

<table>
<thead>
<tr>
<th>Rank</th>
<th>Relev.</th>
<th>$P(doc_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>0.67</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td>0.50</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>0.40</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>X</td>
<td>0.25</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Query 1 AVG:** 0.564

<table>
<thead>
<tr>
<th>Rank</th>
<th>Relev.</th>
<th>$P(doc_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>0.67</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>X</td>
<td>0.2</td>
</tr>
</tbody>
</table>

**Query 2 AVG:** 0.623

- MAP favours systems which return relevant documents fast
- Precision-biased

$$MAP = \frac{0.564 + 0.623}{2} = 0.594$$
ROC Curve (Receiver Operating Characteristics)

Confusion/distance matrix: $d(j,k)$

Ground Truth: $m(j,k)$

$$m(j, k) = \begin{cases} 
1, & \text{doc } j \text{ and } k \text{ are match} \\
0, & \text{not match} 
\end{cases}$$
Classification in Image Retrieval

A typical image retrieval pipeline

- Image Formation
- Feature Computing
- Feature Aggregation
- Classification

Knowledge/Data Base
kNN Classifier

- kNN is Nonlinear Classifier
  - Operations: majority vote

- Performance bounds on $k$:
  - not worse off by 2 times the error rate of Bayesian error rate
  - As $k$ increases, it improves, but not much

- Accelerate NN retrieval operation
  - Kd-tree:
    - a space partition scheme, to limit the number of data points comparison operation
  - Kd-tree supported kNN search:
    - Approx KNN search
    - Kd-Forest supported Approx Search
Gaussian Generative Model Classifier

- Shaped by the relative shape of Covariance Matrix

\[ g_i(x) = x^T A_i x + W_i x + W_{i0}, \]

where

\[ A_i = -\frac{1}{2} \Sigma_i^{-1} \]
\[ W_i = \Sigma_i^{-1} \mu_i \]
\[ W_{i0} = -\frac{1}{2} \mu_i^T \Sigma_i^{-1} \mu_i - \frac{1}{2} \ln |\Sigma_i| + \ln \rho(\omega_i) \]

Note: The decision boundary is no longer linear! It is hyperquadrics.

- Matlab:
  - `[rec_label, err]=classify(q, x, y, method`)
  - Method = ‘quadratic’
Logistic Regression

- Logistic Function:
  - Mapping linear function to $[0 \ 1]$.
  - Give a prob of observed $X$ is has label 0 or 1 via logistic mapping

$$z = \sum_{i} w_{i}x_{i} + w_{0} = WX$$

$$g(z) = \frac{1}{1 - e^{z}}$$
Gradient Search Solution

- With Log loss function the problem is convex, and a gradient search solution can reach optimal.

  - for each weights $w_j$, we have,
    \[
    \frac{\partial}{\partial w_j} L(w) = -(y \frac{1}{g(w^T x)} - (1 - y) \frac{1}{1 - g(w^T x)}) \frac{\partial}{\partial w_j} g(w^T x)
    \]
    \[
    = -(y \frac{1}{g(w^T x)} - (1 - y) \frac{1}{1 - g(w^T x)}) g(w^T x)(1 - g(w^T x)) \frac{\partial}{\partial w_j} w^T x
    \]
    \[
    = -(y(1 - g(w^T x)) - (1 - y)g(w^T x))x_j
    \]
    \[
    = (h(x) - y)x_j
    \]

  - For a batch of $m$ oberved \( \{x^{(i)}, y^{(i)}\} \), we can do batch descent, or stochastic gradient descent (SGD)
    \[
    \frac{\partial}{\partial w_j} L(w) = \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})x_j
    \]

  - matlab: `mnrfit()`
Why called SVM?

Let’s write down the discriminant function

\[ f(x) = w \cdot x + b = \sum_{i=1}^{n} \alpha_i y_i (x_i \cdot x) + b \]

Notice that \( \alpha_i = 0 \) for non-support vectors, so

\[ f(x) = w \cdot x + b = \sum_{x_i \in \{S.V.\}} \alpha_i y_i (x_i \cdot x) + b \]
Matlab:

```matlab
% train svm
svm = svmtrain(x, y, 'showplot', true);

% svm classification
for k=1:5
    figure(41); hold on; grid on; axis([0 1 0 1]);
    [u,v]=ginput(1);
    q=[u,v];
    plot(u, v, '*b');
    rec = svmclassify(svm, q);
    fprintf('
 recognized as %c', rec);
end
```

Summary

- **Image Analysis & Retrieval Part I**
  - Image Formation
  - Color and Texture Features
  - Interest Points Detection – Harris Detector
  - Invariant Interest Point Detection – SIFT
  - Aggregation
  - Classification tools: knn, gmm, svm
  - Performance metrics:
    - tp, fp, tn, fn, tpr/precision, fpr, recall, mAP

- **Part II: Holistic approach**
  - Just throw the pixels to the algorithm to figure out what are the good features.
  - Subspace approach: Eigenface, Fisherface, Laplacian Embedding
  - Deep Learning: Classification, Identification, and Segmentation
  - Hashing