16720J: Homework 3 - Object Detection

Wenbo Zhao
(wzhao1@andrew.cmu.edu, NetID: zhaowb7)

October 25, 2015

Collaboration declaration: *This homework is done in partial collaboration with Wenbo Liu() and Yan Xu(). Specifically, for Q. 3, the author discussed the methods of finding nearest examplars to cluster centers with them and adopted the one with counting pixel label membership. The author also discussed with them on selecting new features for clustering. The author thanks them for their contribution to this work.*

1 Warming up with some theory (9pts)

Question 1.1 (2pts, 1 line)

$(M - h + 1) \times (N - w + 1)$ windows.

Question 1.2 (5pts, 2-3 lines)

Algorithms that optimize the area under the ROC curve are not guaranteed to optimize the area under the PR curve [1]. E.g. if the number of negative examples outnumbers positive examples, ROC curve can't capture the effect of large number change of false positives, since it leads to small variation in false positive rate, while precision captures this change.

Question 1.3 (2 pts, 1 line)

1000. 1.

2 Object Detection via DPMs and Non-Maximum Suppression (40 pts)

2.1 Mean-Shift Clustering (20 pts)

Question 2.1.1 `MeanShift.m`

```
% Created by zhaowb7 on 2015-10-20.

function [CCenters,CMemberships] = MeanShift(data,bandwidth,stopThresh)
% This func: performs mean shift clustering given:
% - INPUTS: * data: N(umber of points) * (F(eature dimension) +
%           1(score))
%           * bandwidth: window size
%           * stopThresh: check convergence
% and
% - OUPUTS: * CCenters: M(cluster)*F cluster center
```

```
%           * CMemberships: N*1 membership
%
% Author: WENBO ZHAO (wzhao1@andrew.cmu.edu)
% Date: Oct 20, 2015
% Log: (v0.1)-(first draft, written all the functions)-(Oct 20, 2015)
%      (v0.2)-(modified: fixed bug: improved: )
%
if nargin < 2
    error('Please define bandwidth!\n');
end
if nargin < 3
    stopThresh = 1e-3*bandwidth; % default
end
% initialize useful variables
numPoint = size(data,1);
dimFeat = size(data,2)-1;
CCenters = [];
CMemberships = zeros(numPoint, 1);

% initialize cluster setups
numClus = 0; % initial number of clusters ?? 1 ??
voteClus = [];
pointLooked = zeros(numPoint,1); % store points that have been looked
numInitPoint = numPoint;

while numInitPoint
    initPoint = datasample(find(pointLooked==0),1); % init random point
    center = data(initPoint, :); % initial center
    member = []; % points fall into the same cluster
    vote = zeros(numPoint, 1); % store votes for members
% start cluster
while 1
    distCenter2Point = pdist2(center, data); % distance from center to ...
       all active data points
    inPoint = find(distCenter2Point<bandwidth); % find data points within ...
       bandwidth
    vote(inPoint) = vote(inPoint)+1; % add votes
    oldCenter  = center;
    center = sum(bsxfun(@times, data(inPoint, 1:end), ...
       data(inPoint, end)), 1)./sum(data(inPoint, end), 1);
    member = [member inPoint];
    pointLooked(member,:) = 1;

    %% plot in progress
    plotFlag = 0;
    if plotFlag
        if dimFeat == 2
            figure(157),clf,hold on
            plot(data(:, 1),data(:, 2),'.')
            plot(data(member, 1),data(member, 2),'ys')
            plot(center(1),center(2),'go')
            plot(oldCenter(1),oldCenter(2),'rd')
            pause(0.1)
        end
    end
    %%
    if norm((center-oldCenter),2) < stopThresh
        merge = 0; % clusters to merge
        for i = 1:numClus
            dist2NewC  = norm((center-CCenters(i,:)),2);
```

```
            if dist2NewC < bandwidth/2
                merge = i;
                break
            end
        end

        if merge>0 % merge clusters if too close
            CCenters(merge,:) = mean((center+CCenters(merge,:)),1);
            voteClus(merge,:) = voteClus(merge,:)+vote';
        else
            numClus = numClus + 1; % found new cluster
            CCenters(numClus,:) = center;
            voteClus(numClus,:) = vote';
        end

        break
    end
end
numInitPoint = length(find(pointLooked==0));
end
[¬, CMemberships] = max(voteClus, [], 1);
CMemberships = CMemberships';
numClus
fprintf('saving CCenters and CMemberships\n');
% save('q21_result', 'CCenters', 'CMemberships');
end
```

Question 2.1.2

Save your `CCenters` and `CMemberships` into `q21_result.mat`. Also save the visualization result from `q21_test.m`, as `q21_clustering.jpg` and include here.

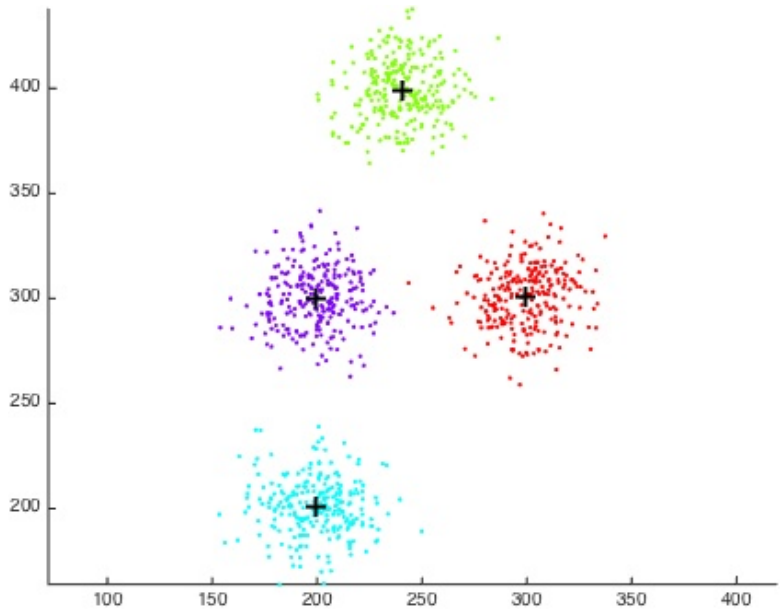


Figure 1: Mean Shift Clusters

```
clear all, close all
load('q21_data.mat');

```

```
bandwidth = 55  % This is an example. You may need to adjust this value
threshold = bandwidth*0.01; % This is an example. You may need to adjust ...
    this value
[clusterCenters,clusterMemberships] = MeanShift(data,bandwidth,threshold);

%% Draw
clusterNum  = size(clusterCenters,1);
figure; hold on; axis equal
set(gcf,'color','w');
cc=hsv(clusterNum);
for cIdx = 1:clusterNum
    tempMembership = find(clusterMemberships == cIdx);
    plot(data(tempMembership,1),data(tempMembership,2),'.','color',cc(cIdx,:));

    tempCenter = clusterCenters(cIdx,:);
    plot(clusterCenters(cIdx,1),clusterCenters(cIdx,2),'k+','MarkerSize',10,'lineWidth',
end
```

Question 2.1.3 (at most 3 lines in your write-up)

For low bandwidth, too many number of clusters are found, this is not reasonable when a group of points have large inter-cluster distance but slightly large in-cluster distance and would fall into different clusters. So tune up the bandwidth until the cluster is reasonably placed.

2.2 Detecting using Deformable Part Models (DPMs) (20 pts)

Question 2.2.1

Submit your `nms` function and include here

```
% Created by zhaowb7 on 2015-10-20.

function [refinedBBoxes] = nms(bboxes, bandwidth,K)
% set useful variables
numBox = size(bboxes,1);
dimFeat = size(bboxes, 2)-1;
stopThres = bandwidth*0.01;

% positive scores
bboxes(:,end) = bboxes(:,end)+1; % ?? normalize
% bboxes(:,end) = abs(bboxes(:,end));
% refine boxes via Non-Maximum Suppression using mean-shift cluster
[refinedBBoxes, boxTags] = MeanShift(bboxes, bandwidth, stopThres);
refinedBBoxes = refinedBBoxes(:,1:dimFeat);
end
```

Question 2.2.2 (at most 3 lines in your write-up)

Given input detection boxes, what nms does is treating the boxes as feature points and cluster them. So picking top-K candidates is equivalent to, as in Q. 2.1.3, tuning bandwidth, and K is not necessarily to be significantly vary – they basically yield the same cluster numbers.

Question 2.2.3

Your result images here:

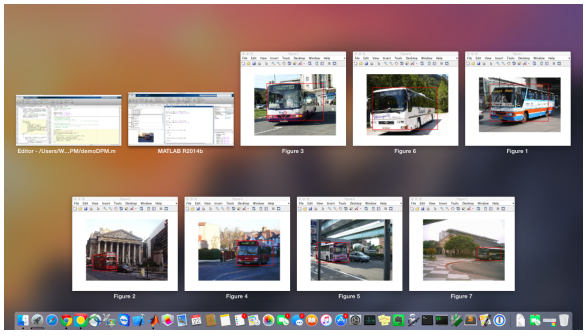
(a) NMS Result 1

(b) NMS Result 2

(c)NMS Result 3

(d)NMS Result 4

(e)NMS Result 5



(f)NMS Result 6

Figure 2: NMS results

```
clear all, close all
%% setting path and load model
addpath ../export_fig/

addpath(genpath('../utils'));
addpath(genpath('../lib/dpm'));
load('../../data/bus_dpm.mat');

%% Object detection via DPMs
```

```
I = imread('q42_test.jpg');
detectionBoxes = imgdetect(I,model);
figure; showboxes(I,  detectionBoxes);      %% show detected bounding boxes.

%% Non-Maximum suppression
bestBBox = nms(detectionBoxes,200,5); % K varies, but still got 1 detection
figure; hold on; image(I); axis ij; hold on;
showboxes(I,  bestBBox);

%% Find buses!
busStation = '../../data/voc2007/';
busNum = dir(fullfile(busStation,'*.jpg'));
ind = datasample(1:length(busNum), 15);
for i = ind
    bus = imread(fullfile(busStation, busNum(i).name))
    detectionBoxes = imgdetect(bus,model);
    bestBBox = nms(detectionBoxes,200,5);
    figure; hold on; image(bus); axis ij; hold on;
    showboxes(bus,  bestBBox);
end
```

3 Reducing Exemplar Detectors (55 pts)

3.1 Detecting using Exemplar Detectors (10 pts)

Question 3.1.1 (10 pts)

```
% Created by zhaowb7 on 2015-10-23.

function [boundingBoxes] = batchDetectImageESVM(imageNames, models, params)
%% Set par pool
% if nargin < 4
%     %default to 2 cores
%     numCores = 2;
% end
% % Close the pools, if any
% try
%     fprintf('Closing any pools...\n');
% %     matlabpool close
%     delete(gcp('nocreate'))
% catch ME
%     disp(ME.message);
% end
% fprintf('Will process %d files in parallel to compute visual words ...
    ...\n',length(imageNames));
% fprintf('Starting a pool of workers with %d cores\n', numCores);
% myPool = parpool(numCores);

%% Get bounding boxes
fprintf('Start taking in images and models, return their bounding ...
   boxes.\n ');
numImg = length(imageNames);
boundingBoxes = cell(1,numImg);
imgDir = '../../data/voc2007'; % image directory
for i = 1:numImg
    fprintf('get bounding box for %s\n', imageNames{i});
```

```
    image = imread(fullfile(imgDir, imageNames{i}));
    boundingBoxes{i} = esvm_detect(image,models,params);
end
% save boundingBoxes
fprintf('Save bounding boxes...\n');
% save('boundingBoxes', 'boundingBoxes');
fprintf('Done.\n');

%close the pool
% fprintf('Closing the pool.\n');
% delete(myPool)

end
```

3.2 Evaluating Detection Performance (15 pts)

Question 3.2.1 Theory (5 pts, 2 lines)

Average precision (AP) is the average value of precision over recall $p(r)$ in the precision-recall curve, and is calculated by $AP = \int_0^1 p(r)dr$.

Question 3.2.2 (10 pts)

Submit your script as `q3_2_2.m` and include here. Include an interpretation of your graph here.

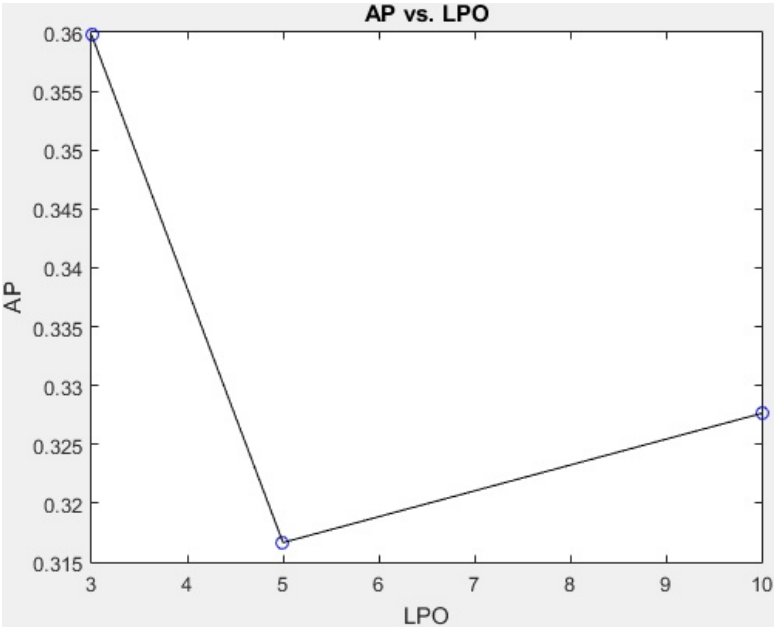


Figure 3: **AP vs LPO (test set)** A higher lpo implies more levels in the HOG feature pyramid. It shows that with 3 layers we get the best AP result, and as the layers increase, the AP value decreases might due to too many layers of scaling.

```
% Created by zhaowb7 on 2015-10-23.

% Q3.2.2
close all, clear all
%% Set path
addpath(genpath('../utils'));
```

```
addpath(genpath('../lib/esvm'));
load('../../data/bus_esvm.mat');
load('../../data/bus_data.mat');

%% get bounding boxes from all images
% params = esvm_get_default_params();
% boundingBoxes = batchDetectImageESVM(modelImageNames, models, params);

%% set variables
% - detect
imgDir = '../../data/voc2007';
numTestImg = length(gtImages); % # of test images
% - AP
IOU_ratio = 0.5;
draw = true

%% Detect and compute AP
% - detect
params = esvm_get_default_params();
lpo = [3 5 10];
detectBoxes = cell(length(lpo),numTestImg);
ap = zeros(1,length(lpo));
for i = 1:length(lpo)
    params.detect_levels_per_octave = lpo(i)
    for j = 1:numTestImg
        fprintf('get bounding box for %s\n', gtImages{j});
        image = imread(fullfile(imgDir, gtImages{j}));
        detectBoxes{i,j} = esvm_detect(image,models,params);
    end
% - evaluate AP
[¬,¬,ap(i)] = evalAP(gtBoxes, detectBoxes(i,:),IOU_ratio,draw);
end
fprintf('Save bounding boxes...\n');
% save('detectBoxes', 'detectBoxes');
fprintf('Done.\n');

%% Plot
% ap = [0.3598    0.3167    0.3276]
figure
plot(lpo, ap, 'bo');
hold on, plot(lpo, ap, 'k-');
title('AP vs. LPO')
xlabel('LPO'), ylabel('AP')

% addpath ../export_fig
% export_fig('AP_LPO', '-jpg')
```

3.3 Compacting the set of exemplar detectors

Question 3.3.1 (20 pts)

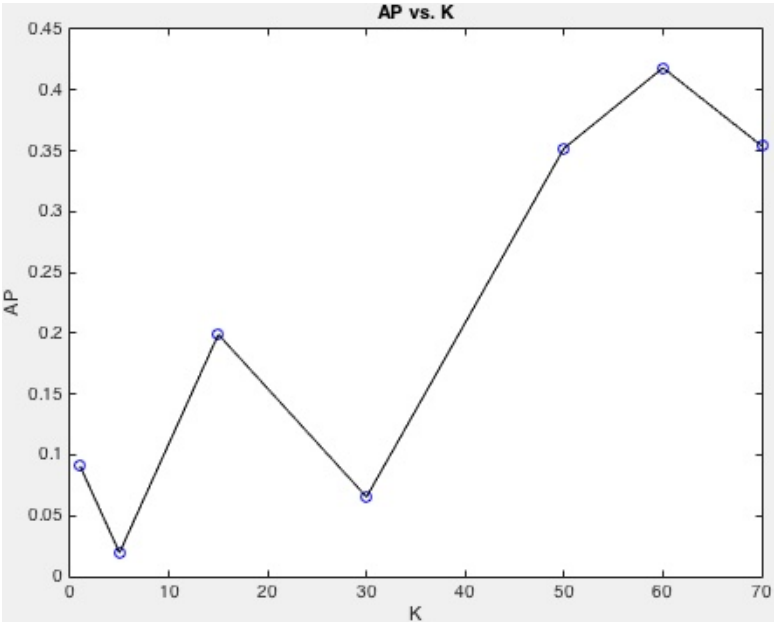


Figure 4: **K vs AP**
NOTE: AP values vary each run

Figure 5: Average Images (K=60)

```
% This script takes the output bounding boxes from E-SVM detector, filters
% images with filter banks, and clusters their box-responses using k-means.
% Then K selected E-SVM detectors are used to detect on test set. AP is
% returned.
%
% - E-SVM detector @esvm_detect
% - filter banks @createFilterBank
% - box-responses @extractFilterResponses
%
% Author: WENBO ZHAO (wzhao1@andrew.cmu.edu)
% Date: Oct 23, 2015
```

```
% Log: (v0.1)-(first draft, written all the functions)-(Oct 23, 2015)
%      (v0.2)-(modified: fixed bug: improved: )
%
close all, clear all
%% set path
addpath(genpath('../utils'));
addpath(genpath('../lib/esvm'));
load('../../data/bus_esvm.mat');
load('../../data/bus_data.mat');
imgDir = '../../data/voc2007';
%% get bounding boxes from all images
params = esvm_get_default_params();
% boundingBoxes = batchDetectImageESVM(modelImageNames, models, params);

%% get bounded images
imageBox = cell(1,length(modelBoxes));
for i = 1:length(modelBoxes)
    fprintf('get bounded image for %s\n', modelImageNames{i});
    image = imread(fullfile(imgDir, modelImageNames{i}));
    boxes = modelBoxes{i};
    imageBox{i} = image(boxes(2):boxes(4), boxes(1):boxes(3), :);
    imshow(imageBox{i})
end
%% get box-filtered-responses
% filter banks
fprintf('Getting filter bank ... \n');
% filterBank = createFilterBank();
% fprintf('Done.\n');
% filter responses
fprintf('Generating filter responses ... \n');
% ======= alpha:sample ====== need tweak ======
alpha = 500; % image size roughly ...
% ============================================
for i = 1:length(imageBox)
    filterResp = extractFilterResponses(imageBox{i}, filterBank);
    randPixels = randperm(size(filterResp,1), alpha); % randomly select ...
        alpha pixels
    filterResp = filterResp(randPixels, :);
    boxResponse(i,:) = filterResp(:);
end
boxResponse = reshape(boxResponse, [length(imageBox)*alpha, ...
    3*size(filterBank,1)]);
fprintf('saving filtered box responses ... \n');
% save('boxResponse', 'boxResponse');
fprintf('Done.\n');

% load boxResponse.mat
%% Cluster and find K examplars
% k-means cluster
fprintf('kmeans clustering ... \n');
% ===== K ===== need tweak ======
K = 65;
% ==============================
[label,centerBox, inClusP2Cdist, P2Cdist] = kmeans(boxResponse, K, ...
    'EmptyAction', 'drop');
% find examplars close to K clusters and average them by
% stating the number of pixels belonging to each cluster in each sampled ...
    image
clusMap = zeros(length(imageBox), K);
for i = 1:K
```

```
68      for j = 1:length(imageBox)
69          temp = label( (j-1)*alpha+1 : j*alpha );
70          clusMap(j, i) = length(find(temp == i));
71      end
72  end
73  imgInClus = cell(K,1); % store image index in each cluster
74  for i = 1:K
75      ind = clusMap(:,i); % label accumulation of each pixel for each ...
            cluster: belongingness to cluster
76      ind_s = sort(clusMap(:,i),'descend');
77      s = [];
78      % ===== top 3 ===== need tweak =====
79      for t=1:3
80      % ==================================
81          tt = find(ind==ind_s(t));
82          s = [s; tt];
83      end
84      imgInClus{i} = s;
85  end
86  %% E-SVM detect with k-detectors and compute AP
87  % + set variables
88  %   - detect
89  numTestImg = length(gtImages); % # of test images
90  %   - AP
91  IOU_ratio = 0.5;
92  draw = true
93  % + Detect and compute AP
94  %   - detect
95  params = esvm_get_default_params();
96  detectBoxes = cell(1, numTestImg);
97  % find K nearest images
98  knImgInd = zeros(1,K);
99  knImg = cell(1,K);
100 newModel = cell(1,K); % and select K models
101 for i = 1:K
102     knImgInd(i) = imgInClus{i}(1);
103     knImg{i} = imread(fullfile(imgDir, modelImageNames{knImgInd(i)}));
104     newModel{i} = models{knImgInd(i)};
105 end
106 for j = 1:numTestImg
107     fprintf('get bounding box for %s\n', gtImages{j});
108     image = imread(fullfile(imgDir, gtImages{j}));
109     detectBoxes{j} = esvm_detect(image,newModel,params);
110 end
111 %   - evaluate AP
112 [¬,¬,ap] = evalAP(gtBoxes, detectBoxes,IOU_ratio,draw)
113
114 fprintf('Save bounding boxes...\n');
115 % save('detectBoxes', 'detectBoxes');
116 fprintf('Done.\n');
117
118 %% Visualize
119 % ---- AP vs. k ----
120 plotAP = 0;
121 if plotAP
122 k = [1 5 15 30 50 60 70];
123 ap = [0.0909 0.0196 0.1990 0.0654 0.3522 0.4182  0.3540];
124 figure
125 plot(k, ap, 'bo');
126 hold on, plot(k, ap, 'k-');
```

```
127 title('AP vs. K')
128 xlabel('K'), ylabel('AP')
129 end
130
131 % ---- average images of k-bounding boxes ----
132 aveImBox = cell(1, K);
133 reSize = 100; % 100*100
134 for i = 1:K
135     temp = zeros(reSize, reSize,3, 'double');
136     imgInClusTemp = imgInClus{i};
137     for j = 1:length(imgInClusTemp)
138         imTemp = im2double(imageBox{imgInClusTemp(j)}); % im2double!!
139         imTemp = imresize(imTemp, [reSize, reSize]);
140         temp = temp+imTemp;
141     end
142     aveImBox{i} = temp./length(imgInClusTemp);
143 end
144 fprintf('saving average boxes ... \n');
145 % save('aveImBox', 'aveImBox');
146 fprintf('Done.\n');
147 imdisp(aveImBox);
148
149 % addpath ../export_fig
150 % export_fig('average_img_k=50', '-jpg')
```

Question 3.3.2 (10 pts)

In this section different features

(a) HOG

(b) SIFT

(c) dense SIFT (too slow on running, give up... see code below)

are tried. Results are shown below.

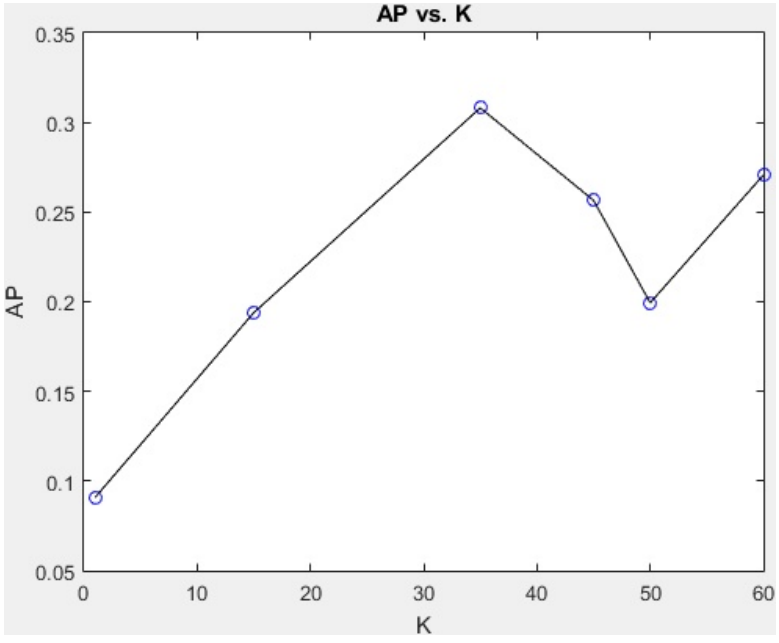


Figure 6: K vs AP (HOG feature)

Figure 7: Average Images (HOG feature, K=60)

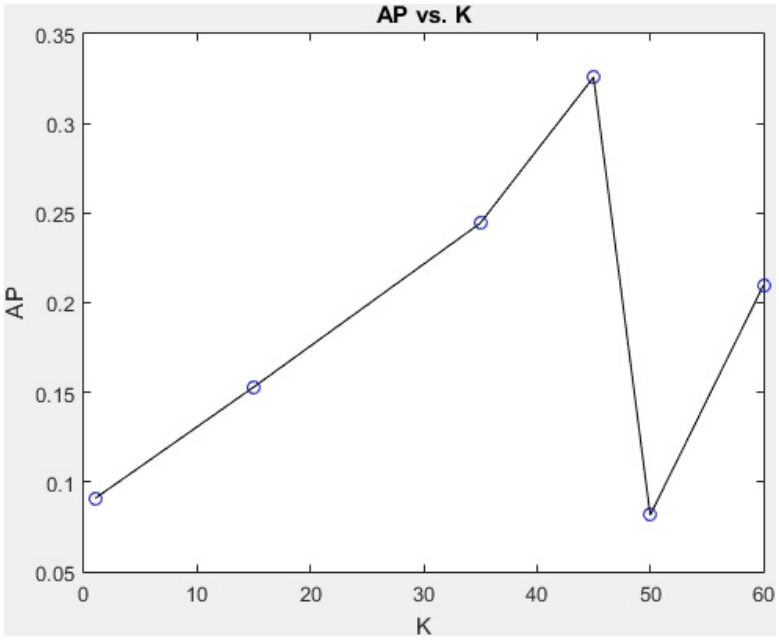


Figure 8: K vs AP (SIFT feature)

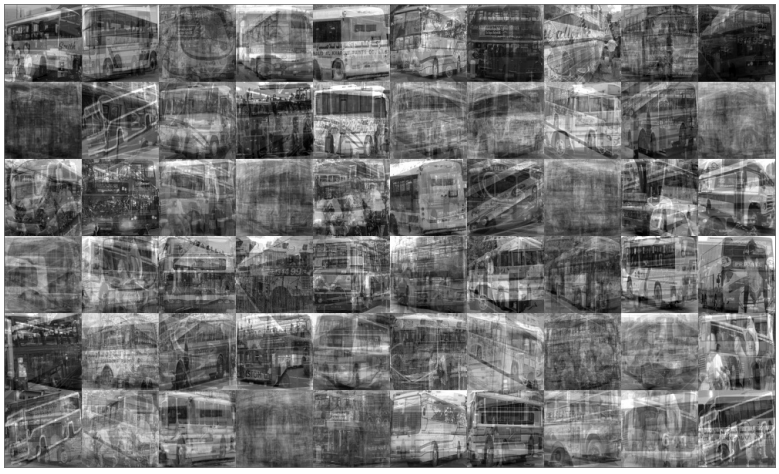


Figure 9: Average Images (SIFT feature, K=60)

```
% This script takes the output bounding boxes from E-SVM detector, extract
% features from VLFEAT toolbox, and cluster their box-feature-responses ...
    using k-means.
% Then K selected E-SVM detectors are used to detect on test set. AP is
% returned.
%
% - E-SVM detector @esvm_detect
% - feature banks @vlfeat toolbox (www.vlfeat.org/
%                                  https://github.com/vlfeat/)
% - box-responses @extractFilterResponses
%
% Author: WENBO ZHAO (wzhao1@andrew.cmu.edu)
% Date: Oct 25, 2015
% Log: (v0.1)-(first draft, written all the functions)-(Oct 25, 2015)
% (v0.2)-(modified: fixed bug: improved: )
%
close all, clear all
%% set path
addpath(genpath('../utils'));
addpath(genpath('../lib/esvm'));
addpath(genpath('./external/vlfeat-0.9.20/'));
% compile vlfeat toolbox
% ******* NEED COMPILE MEX FIRST ****************************************
% ******* MIGHT NEED MEX COMPILER ** SETTINGS DIFFER FOR DIFFERENT DEV ENVS
% (PCX86, PCX64, MACI32, MACI64, etc)
% ** THEN ** RUN SETUP BELOW ******************************************
run ./external/vlfeat-0.9.20/toolbox/vl_setup.m

load('../../data/bus_esvm.mat');
load('../../data/bus_data.mat');
imgDir = '../../data/voc2007';
%% get bounding boxes from all images
params = esvm_get_default_params();
% boundingBoxes = batchDetectImageESVM(modelImageNames, models, params);

%% get bounded images
imageBox = cell(1,length(modelBoxes));
for i = 1:length(modelBoxes)
    fprintf('get bounded image for %s\n', modelImageNames{i});
    image = imread(fullfile(imgDir, modelImageNames{i}));
    boxes = modelBoxes{i};
    imageBox{i} = single((rgb2gray(image(boxes(2):boxes(4), ...
        boxes(1):boxes(3), :))));
%     imshow(imageBox{i})
end
%% get box-feature-responses using different features
method = 'HOG'
% method = 'SIFT'
% method = 'dSIFT'
switch lower(method)
    case 'hog'
        fprintf('Extracting %s features ... \n', method);
        % ================================
        cellSize = 8 ;
        alpha = 100; % randomness
        % ================================
        hogFeat = [];
        numPixelPerImg = [];
        for i = 1:length(imageBox)
            hogFeatTemp = vl_hog(imageBox{i}, cellSize, 'verbose') ;
```

```
59              %  ---- plot ----
60              plot = 1;
61              if plot
62              imhog = vl_hog('render', hogFeatTemp, 'verbose') ;
63              imagesc(imhog) ; colormap gray ;
64              end
65              % ---------------
66              hogFeatTemp = ...
                    reshape(hogFeatTemp,[size(hogFeatTemp,1)*size(hogFeatTemp,2), ...
                    31]);
67              randSel = randperm(size(hogFeatTemp,1), min(alpha, ...
                    size(hogFeatTemp,1)));
68              numPixelPerImg = [numPixelPerImg; i*ones(numel(randSel),1)];
69              hogFeat = [hogFeat; hogFeatTemp(randSel, :)];
70          end
71          fprintf('saving box-feature-responses ... \n');
72  %         save('hogFeat', 'hogFeat');
73          fprintf('Done.\n');
74          feat = hogFeat;
75      case 'sift'
76          % ==========================
77          alpha = 100;
78          % ==========================
79          siftFeat = [];
80          numPixelPerImg = [];
81           for i = 1:length(imageBox)
82               [f,d] = vl_sift(imageBox{i}) ;
83               % ---------- plot ---------
84               plot = 0;
85               if plot
86               perm = randperm(size(f,2)) ;
87               sel = perm(1:5) ;
88               h1 = vl_plotframe(f(:,sel)) ;
89               h2 = vl_plotframe(f(:,sel)) ;
90               h3 = vl_plotsiftdescriptor(d(:,sel),f(:,sel)) ;
91               end
92               % -------------------------
93               d = d';
94              randSel = randperm(size(d,1), min(alpha, size(d,1)));
95              numPixelPerImg = [numPixelPerImg; i*ones(numel(randSel),1)];
96              siftFeat = [siftFeat; d(randSel, :)];
97           end
98          fprintf('saving box-feature-responses ... \n');
99  %         save('siftFeat', 'siftFeat');
100         fprintf('Done.\n');
101         feat = double(siftFeat);
102
103     case 'dsift'
104         % ============================
105         binSize = 8 ;
106         magnif = 3 ;
107         alpha = 100;
108         % ============================
109         dsiftFeat = [];
110         numPixelPerImg = [];
111         for i = 1:length(imageBox)
112             Is = vl_imsmooth(imageBox{i}, sqrt((binSize/magnif)^2 - .25)) ;
113             [f, d] = vl_dsift(Is, 'size', binSize) ;
114             f(3,:) = binSize/magnif ;
115             f(4,:) = 0 ;
```

```
116             [f_, d_] = vl_sift(imageBox{i}, 'frames', f) ;
117              % ---------- plot ---------
118              plot = 0;
119              if plot
120              perm = randperm(size(f_,2)) ;
121              sel = perm(1:5) ;
122              h1 = vl_plotframe(f_(:,sel)) ;
123              h2 = vl_plotframe(f_(:,sel)) ;
124              h3 = vl_plotsiftdescriptor(d_(:,sel),f_(:,sel)) ;
125              end
126              % -------------------------
127             d_ = d_';
128             randSel = randperm(size(d_,1), min(alpha, size(d_,1)));
129             numPixelPerImg = [numPixelPerImg; i*ones(numel(randSel),1)];
130             dsiftFeat = [dsiftFeat; d_(randSel, :)];
131         end
132         fprintf('saving box-feature-responses ... \n');
133 %         save('dsiftFeat', 'dsiftFeat');
134         fprintf('Done.\n');
135         feat = double(dsiftFeat);
136     otherwise
137         disp('no defined method!\n');
138 end
139
140 % load boxResponse.mat
141 %% Cluster and find K examplars
142 % k-means cluster
143 fprintf('kmeans clustering ... \n');
144 % ===== K ===== need tweak ======
145 K = 60;
146 % ================================
147 [label,centerBox, inClusP2Cdist, P2Cdist] = kmeans(feat, K, ...
        'EmptyAction', 'drop');
148 % find examplars close to K clusters and average them by
149 % stating the number of pixels belonging to each cluster in each sampled ...
        image
150 clusMap = zeros(length(imageBox), K);
151 for i = 1:K
152     for j = 1:length(imageBox)
153         temp = label( numPixelPerImg == j );
154         clusMap(j, i) = length(find(temp == i));
155     end
156 end
157 imgInClus = cell(K,1); % store image index in each cluster
158 for i = 1:K
159     ind = clusMap(:,i); % label accumulation of each pixel for each ...
            cluster: belongingness to cluster
160     ind_s = sort(clusMap(:,i),'descend');
161     s = [];
162     % ===== top 3 ===== need tweak =====
163     for t=1:3
164     % ==================================
165         tt = find(ind==ind_s(t));
166         s = [s; tt];
167     end
168     imgInClus{i} = s;
169 end
170 %% E-SVM detect with k-detectors and compute AP
171 % + set variables
172 %  - detect
```

```
numTestImg = length(gtImages); % # of test images
%  - AP
IOU_ratio = 0.5;
draw = true
% + Detect and compute AP
%  - detect
params = esvm_get_default_params();
detectBoxes = cell(1, numTestImg);
% find K nearest images
knImgInd = zeros(1,K);
knImg = cell(1,K);
newModel = cell(1,K); % and select K models
for i = 1:K
    knImgInd(i) = imgInClus{i}(1);
    knImg{i} = imread(fullfile(imgDir, modelImageNames{knImgInd(i)}));
    newModel{i} = models{knImgInd(i)};
end
for j = 1:numTestImg
    fprintf('get bounding box for %s\n', gtImages{j});
    image = imread(fullfile(imgDir, gtImages{j}));
    detectBoxes{j} = esvm_detect(image,newModel,params);
end
%  - evaluate AP
[¬,¬,ap] = evalAP(gtBoxes, detectBoxes,IOU_ratio,draw)

fprintf('Save bounding boxes...\n');
% save('detectBoxes', 'detectBoxes');
fprintf('Done.\n');

%% Visualize
% ---- AP vs. k ----
plotAP = 0;
if plotAP
k = [1 15 35 45 50 60]; % hog
ap = [0.0909 0.1939 0.3081 0.2565 0.1994 0.2707];
% k = [1 15 35 45 50 60]; % sift
% ap = [0.0909 0.1530  0.2444 0.3258 0.0815 0.2098];
figure
plot(k, ap, 'bo');
hold on, plot(k, ap, 'k-');
title('AP vs. K')
xlabel('K'), ylabel('AP')
end

% ---- average images of k-bounding boxes ----
aveImBox = cell(1, K);
reSize = 100; % 100*100
for i = 1:K
    temp = zeros(reSize, reSize, 'double');
    imgInClusTemp = imgInClus{i};
    for j = 1:length(imgInClusTemp)
        imTemp = im2double(imageBox{imgInClusTemp(j)}); % im2double!!
        imTemp = imresize(imTemp, [reSize, reSize]);
        temp = temp+imTemp;
    end
    aveImBox{i} = temp./length(imgInClusTemp);
end
fprintf('saving average boxes ... \n');
% save('aveImBox', 'aveImBox');
fprintf('Done.\n');
```

```
imdisp(aveImBox);

% addpath ../export_fig
% export_fig('hog_average_img_k=35', '-jpg')
```

4 Extra credit: Segmentation transfer using ESVM (20 pts)

If you have attempted this extra-credit section please include a summary of your efforts here and include all relevant work in the folder `segTransfer`.

Thoughts: The basis concept is replacing the detected bounding boxes of targets (using HOG features and E-SVM detectors) with the corresponding masks, either the masks be meta-data or segmentation superpixels. Specifically, one can first train the E-SVM detector with examplar HOG features, and then detect targets on the test set. For the detected targets, they correspond to a pre-defined meta-data (or segmentation superpixels). What we need to do is aligning this meta-data to the detected bounding boxes (resizing the meta-data to the box size).

References

[1] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, (New York, NY, USA), pp. 233–240, ACM, 2006.