In this assignment, we will study a full pipeline of a sliding window approach for 2D object detection. Specifically, we will implement an Exemplar-SVM algorithm. We will use all the steps described in [4] except the calibration step in Section 3.1 (i.e. for simplicity purpose, we just assume we don’t need calibration between different exemplars which is not true). Your task is to train an object detector for Nemo from a set of images as the training data, and then run the detector to find Nemo in a testing set of images that are different from the training set.

**Problem 1**
Describe Steps
Describe the main steps for both training and testing an Exemplar-SVM object detector.

**Problem 2**
Pyramid
Why do we need to compute image pyramid in the object detection pipeline?

**Problem 3**
Non-Maximum Suppression
Why do we need to do Non-Maximum Suppression (NMS) in the object detection pipeline?
Exemplar-SVM

- Still a rigid template, but train a separate SVM for each positive instance.

For each category it can has exemplar with different size aspect ratio.
Problem 1 Describe Steps

Training :
   To train one exemplar SVM what we do

Testing :
   For each image what we do ?

In this assignment we use 6 images as training and others as testing.
Problem 2 why use image pyramid?

When we compute HoG for each image we compute the HoG for different scale of image.

- What problem we are trying to solve here?
- If there any alternative solution? Why not as good as this one?
Problem 2 Why Non-Maximum Suppression?

Look at the code see where we do non-maximum suppression (call the function nmsMe).

• What does it do?
• Why we do this?
Problem 3 Visualizing Examplars

• Visualize the exemplar image : Model.posRgb
• Visualize the exemplar HoG feature : showHoG(Model.posfeature)
• Visualize the trained svm weight : showHoG(Model.svm.w)
Problem 3 Visualizing Examplars

- Try to understand the correspondences among the three visualization.
- Describe the patterns in the HOG and SVM weight visualization.
Problem 4 Testing Procedure

• Goal: complete the function `bbdet = testfeaturePym(imageRgb, Model)`

• What it does: slide one exemplar detector over the image to find Nemo

• Past your code in the report.
Problem4 Testing Procedure

Basic steps (Listed in the code comments)

1. Compute feature pyramid

2. For each level of the feature pyramid
   - Do convolution get the detection score for each window
   - Convert the score to bounding box in image domain

3. Do non-maximum-suppression
Convolution get the detection score

• Dot product between trained svm weight and feature for each window
  • score = fconv(one level of feature, {weight}, 1, 1);
  • The score stores at the left top corner of the window in feature domain.

• If image at this level is too small, will not do convolution on this and smaller level
Convert the score to bounding box in image domain

• Use:
  – the scale of the this level
  – HOG bin size 8x8 pixels
  – size of this exemplar
• Example in white board
Problem 1
Describe Steps

Describe the main steps for both training and testing an Exemplar-SVM object detector.

Problem 2
Pyramid

Why do we need to compute image pyramid in the object detection pipeline?

Problem 3
Non-Maximum Suppression

Why do we need to do Non-Maximum Suppression (NMS) in the object detection pipeline?

- Paste result in report
- Can you find Nemo? When it fails?
Problem 7 Training Procedure

• After you finish the function testfeaturePym
  Congratulations! You can train your own detectors.
• Run rumMe.m with step.train =1
• To train 6 exemplars and test on the images
• What’s the difference of the results?
[Extra Credit] Detecting Other Objects

• Find an interesting object to detect
• Get images
• Get ground truth box by label it by yourself
[Extra Credit] Evaluation

Precision = \frac{TP}{TP + FP}

Recall = \frac{TP}{TP + FN}

Intersection

Over

Union

= \frac{\text{Intersection}}{\text{Union}}
[Extra Credit ] Evaluation

- X : recall = true positive / total ground truth
- Y : precision = true positive / total detection
Calibration

- Idea: make different exemplar detector's score compatible.

- Good exemplar vs Bad exemplar
Calibration

- **Idea**: make different exemplar detector's score compatible.

1. On Validation set:
   - Sort detection by score,
   - classify true/false detection

2. sigmoid function

- **All detector’s scores are from 0 to 1.**
- **The calibrated score reflects each detector’s precision**
Extra Credit Label Transfer

iHoG

- Webpage: http://web.mit.edu/vondrick/ihog/
- They have online demo and code for download

```matlab
>> feat = features(im, 8);
>> ihog = invertHOG(feat);
>> imagesc(ihog);
```