Princeton University

COS429 Computer Vision

Problem Set 5: Finding Nemo

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.

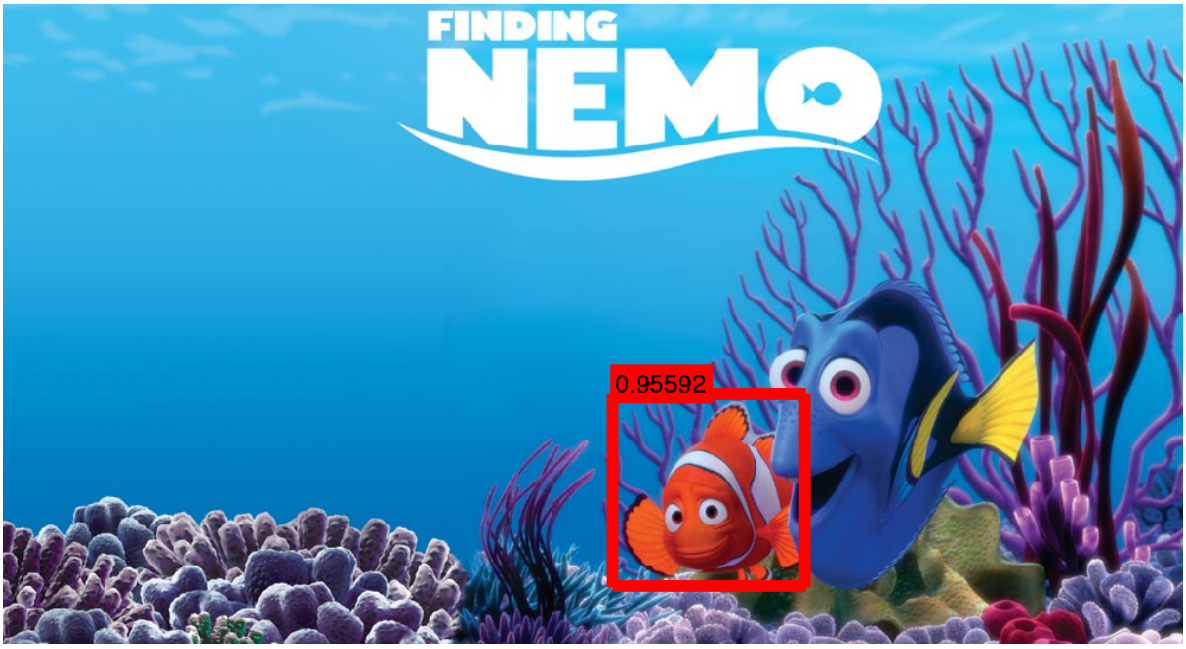


Figure 1: Examplar-SVM for detecting Nemo.

Problem 1 *Describe Steps*

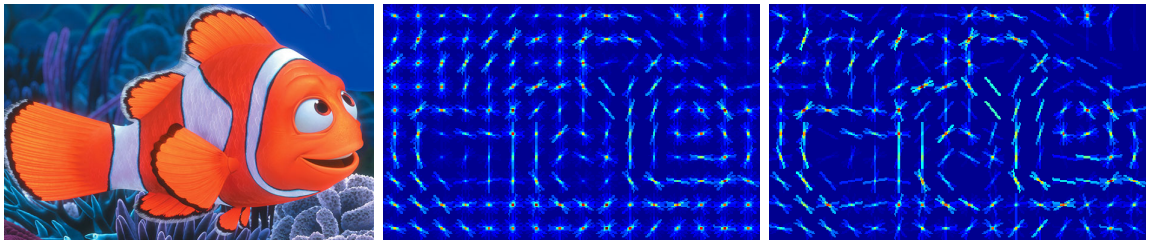
Describe the main steps for both training and testing an Exemplar-SVM object detector.

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?



(a) Image of a positive examplar (b) HOG feature of the examplar (c) Trained SVM weights

Figure 2: Visualization of a positive training examplar.

Problem 4 *Visualizing Examplars*

Download the code and data we provided and open it in MATLAB. The code comes with three pre-trained exemplar detectors. For each detector, the pre-trained model contains the image area that we used for training the exemplar (e.g. Figure 2(a)), the HOG feature vector computed from the image area (e.g. Figure 2(b)), and the trained linear SVM weights corresponding to each HOG feature dimension (e.g. Figure 2(c)).

You task is to visualize these three components for each of the three pre-trained model to generate a figure look like Figure 2. Write a script to load each pre-trained model, visualize the HOG feature by calling the function "showHOG" that we provided, visualize the SVM weights by calling the "showHOG" function as well, save the three images for the three visualization, and include all of them into the report PDF file as the answer to this question. Then, look at the visualization and try to understand the correspondences among the three visualization. Describe how you see the pattens in the HOG and SVM weight visualization.

Problem 5 *Testing Procedure*

Use the pre-trained models, and implement the testing procedure for the exemplar SVM detector. We have basically provided all the functions you need to call. You task is to put them all together to slide one exemplar detector over the image to find Nemo. You will write all your code to complete the function "bbdet = testfeaturePym(imageRgb,Model)". Look at the comments in the code with major steps listed.

Problem 6 *Draw the Detection Result*

Use the function "draw_2d" that we provide to draw your detection results, and include the figures in your report.

Problem 7 *Training Procedure*

We have provided almost all the code for training an Exemplar-SVM detector in the file "TrainingMain_hog.m", except the hard negative mining step. You task contains:

1. Complete the hard negative mining step by reusing the code have have written for the testing procedure.

2. Explain what is hard negative mining, why we need hard negative mining, and why you can reuse the code.

3. Run the training procedure to train the Exemplar SVM for the other three positive training data (We provided pre-trained results for three exemplars. You need to train the remaining three exemplars).

4. Visualize the three new exemplars that you train by including the three images as in Problem 4.

5. Re-run the testing code with all the seven exemplars and regenerate the testing results as in Problem 6.

Extra Credit [Optional] *Detecting Other Objects [Highly Recommended]*

Watch the video and read the Kickstarter page at `http://vision.ai` to get some ideas about what other objects we can detect besides Nemo. Find an interesting object to detect (e.g. detecting Stan Marsh in South Park. Be creative!!!). Download images from the Internet, and write a Matlab labeling tool to label the bounding boxes for training a detector. Run it on a list of testing image and show that it works (or it doesn't and explain why).

Extra Credit [Optional] *Evaluation*

Compute the precision-recall curve to evaluate the detection performance. Compute the average precision as in the standard PASCAL VOC criteria.

Extra Credit [Optional] *Calibration*

Implement the calibration between exemplars as described in [4].

Extra Credit [Optional] *Label Transfer*

Transfer some label (e.g. segmentation or 3D) from the exemplar as described in [4] or [1].

Extra Credit [Optional] *Cross-domain Image Matching*

Use the exemplar-based approach for cross-domain image matching, as describe in [5].

Extra Credit [Optional] *iHOG*

Inverse visualization of HOG features, as describe in [7].

Extra Credit [Optional] *LDA*

Instead of SVM, use LDA to train the model as described in [3] and [2].

Extra Credit [Optional] *Sliding Shapes*

Re-implement the 3D Sliding Shapes algorithm in [6].

Extra Credit [Optional] *Context-pruning*

Use context to remove some false positive detection and boost some false negative detection with low confidence. See `http://web.mit.edu/torralba/www/carsAndFacesInContext.html`.

What to submit: You need to submit two files: one PDF file for the report that contains your name, Princeton NetID, all the pictures taken and text to answer the questions; one ZIP file (not RAR or any other format) that contains all source code for your system, and a "runMe.m" file that takes no parameter as input and run directly in Matlab to generate the results reported in your PDF file. Both the PDF and ZIP file should be named using your Princeton NetID underscore cos429ps5. As an example using my account, they should be named "xj_cos429ps5.pdf" and "xj_cos429ps5.zip". To **verify your result** and **detect plagiarism** to make sure there is no cheating, we will use an automatic program to run your code and compare your code with other students' (including both this year and all previous years) and public available implementations (e.g. from Google, Bing, Github). Therefore, please follow the file format to make our grading job easier. Failure to follow these rules will result in losing your grade.

References

[1] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models[1]. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[2] M. Gharbi, T. Malisiewicz, S. Paris, and F. Durand. A gaussian approximation of feature space for fast image similarity[2]. 2012.

[3] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification[3]. In *European Conference on Computer Vision (ECCV)*, 2012.

[4] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond[4]. In *International Conference on Computer Vision (ICCV)*, 2011.

[5] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros. Data-driven visual similarity for cross-domain image matching[5]. *ACM Transaction of Graphics (TOG) (Proceedings of ACM SIGGRAPH ASIA)*, 30(6), 2011.

[1] `http://www.di.ens.fr/willow/research/seeing3dchairs/index.html`
[2] `http://people.csail.mit.edu/tomasz/papers/gharbi_techreport_2012.pdf`
[3] `http://www.cs.berkeley.edu/~bharath2/pubs/pdfs/bharatheccv2012.pdf`
[4] `http://www.cs.cmu.edu/~tmalisie/projects/iccv11/`
[5] `http://graphics.cs.cmu.edu/projects/crossdomainmatching/`

[6] S. Song and J. Xiao. Sliding Shapes for 3D object detection in RGB-D images[6]. In *European Conference on Computer Vision (ECCV)*, 2014.

[7] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. HOGgles: Visualizing Object Detection Features[7]. *International Conference on Computer Vision (ICCV)*, 2013.

[6] http://slidingshapes.cs.princeton.edu

[7] http://web.mit.edu/vondrick/ihog/