Stanford University
CS231B: The Cutting Edge of Computer Vision
Spring 2015

Course Home Syllabus and Lecture Notes Course Projects

[General Policy] [Segmentation Project] [Tracking Project] [Detection Project]

General Policy:

- We provide skeleton code or each project. It's recommended to base your project around this code.
- You may be able to find code online implemented by the authors. While it's ok to look at external code, it's an honor code violation to directly copy code from existing implementations. Your code will be read by the course instructors.
- We allow for both 1-person or 2-person teams for the projects. Each team hands in one copy of the codes and writeup. Grading will be fair regardless of the team size.
- Your project will be evaluated based on many factors, including your understanding of the algorithm, performance of your algorithm, quality of your code, your creativity, write-up, and presentation.
- It is estimated to take you 20-30 hours for each project, depending on your familiarity with the algorithm and how far you want to go for the projects.
- You are not expected to implement all the details of the algorithm or really achieve state-of-the-art performance, but if your algorithm works particularly well, you will get extra credit.

Grading Policy per Project:

- Technical approach and code: 35% (including correctness and completeness of your method, your innovation, etc)
- Experiment evaluation: 35% (including performance and results of your method, thoroughness of your experiments, insights and analysis that you draw from your results)
- Write-up quality: 20% (please read the **write-up sample** carefully)
- Project presentation: 10% (clarity and content of your project presentation)

Late Days:

We allow **seven late days** in total for all the three projects. Once you have used up these late days, the project turned in late will be penalized 20% per late day.

Project 1: Interactive Image Segmentation with GrabCut

Project Description:

- Implement GrabCut. See Rother et al. for a detailed description of the algorithm.
- Refer to Sec.2 of the Rother paper and Boykov & Jolly for details of energy minimization based image segmentation.
- **What you need to implement:** You only need to implement the iterative procedure described in Sec.3.1 and Sec.3.2 of the Rother paper to get a binary segmentation of the image. The user interaction interface in Sec.3.3 is optional. You do not need to implement border matting or foreground estimation in Sec.4.
- **Initialization code:** You can start your implementation from **the code here**. Using this code, you can initialize the segmentation by drawing a bounding box that contains the foreground.
- **Max-flow/min-cut code:** You can download the **max-flow/min-cut energy minimization code from here**. The code is written in C instead of Matlab, though a Matlab wrapper is also available. Thus, you can either implement your code in C or Matlab. **Here** is a brief introduction of mex files. You can implement your code using another library for max-flow if you like. If you do use their code for max-flow/min-cut, please go through their README and code for usage.
- You can read Szeliski et al for more about energy minimization methods in computer vision if you are interested.

Dataset and Experiment:

- **Images:** Download the images that you need to segment from **here**. Each image contains one foreground object.
- **Ground Truth:** Download the segmentation ground truth images from **here**. White and black colors indicate foreground and background respectively.
- Optional **Bounding Boxes:** We provide you with a set of bounding boxes of the authors of "Image segmentation with a Bounding Box Prior" **here**. You can also use your own, if you choose so.
- **Evaluation metric:** There are two metrics that should be evaluated. The first (accuracy), is the number of pixels that are correctly labled divided by the total number of pixels. The second, Jaccard similarity, is the intersection over the union of your predicted foreground region with the ground truth.**You need to compute these values on all 30 images given (report the average across images), and also show segmentations and report these values on at least six individual images: four images that your algorithm perform very well and two images that your algorithm do not perform very well (if there is any)**.
- Analytical study and extensions. You should have at least one or two of these, here are some ideas:
  - The number of iterations of GMM updating and energy minimization.
  - The number of mixture components in your GMM.
  - Whether to use soft GMM labels and do EM, and how that effects other parameters/results.
  - Different ways to represent probabilities other than GMMs.

- 4-neighborhood or 8-neighborhood in your pairwise term.
- The choice of gamma.
- Alternative ways of setting beta.
- A tight initial bounding box or a loose bounding box.
- Better ways of using the bounding box for segmentation.
- Different color spaces or ways of representing pixels.
- Co-segmentation.

- For this project, we expect state-of-the art results (at least on some images). The TA's baseline GrabCut implementation achieves average accuracy=94.5%, Jaccard similarity=82.4, and runs in 17 seconds per image.

References:

- Y.Boykov and M.Jolly. **Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images**. ICCV 2001.
- C.Rother, V.Kolmogorov, and A.Blake. **"GrabCut" - Interactive Foreground Extraction using Iterated Graph Cuts**. SIGGRAPH 2004.
- R.Szeliski, R.Zabih, D.Scharstein, O.Veksler, V.Kolmogorov, A.Agarwala, M.Tappen, and C.Rother. **A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors**. PAMI 2008.
- V.Lempitsky, P.Kohli, C.Rother, T.Sharp. **Image segmentation with a Bounding Box Prior**. ICCV 2009.

Important Dates:

- 4-5 slides for presentation due to jkrause@cs: Sun, Apr 19 (5:00 pm)
- A short presentation of your work: Mon, Apr 20 (in class)
- Deadline of submitting the code and write-up (submit via CourseWork): Tues, Apr 21 (5:00 pm)

Project 2: Tracking

Project Description:

- In this project, you will design the learning and detection components of an online single-target tracker such as TLD.
- We have provided you a modified version of the original Kalal et al. tracker to get you started: **starter_code**

We do not expect you to use any other files from the original OpenTLD implementation

- **What you need to implement:** You are expected to fill the sections marked as "TODO" in the following files:
  - run_TLD_on_video.m
  - tld/tldDetection.m
  - tld/tldGeneratePositiveData.m
  - tld/tldGenerateNegativeData.m
  - tld/tldInit.m
  - tld/tldLearning.m
  - tld/tldNN.m
  - tld/tldPatch2Pattern.m
  - tld/tldProcessFrame.m
  - tld/tldTrainNN.m

Carefully read through each of the TODO sections before implementing them.

- Please note that this is an open ended assignment, and we expect you to play around with the model parameters in run_TLD_on_video.m, as well as the features, detection and learning methods to understand the tracking problem.

Dataset and Experiments:

- **Evaluation:** We have also provided an evaluation script tldEvaluateTracking.m in tld/ folder. This code accepts a ground-truth text file and the detection text file generated by tldExample.m to provide the final tracking performance. We will use two evaluation metrics: average overlap, and mean Average Precision. In addition, for this project we also expect you to report the time taken for tracking the object in each video.
- **Dataset:** Download the tracking dataset from **tiny_tracking_data**.
  - The dataset contains a validation set and a test set. You are expected to fine-tune your model parameters on the validation set and retain the same values for the test set.
  - The validation and test splits are provided in val_test_splits.txt. The validation set contains 4 videos and test sets contain 5 videos.
  - The video sequences also contain a ground-truth text file. Every line of the file provides the true bounding box co-ordinates for the box in the corresponding frame. The "img" folders in the video sequences also contain an "init.txt" file which provides the intializing bounding box.
- **Getting started:**
  - The starter code uses opencv to implement the KLT tracker. If you plan to run the code on corn.stanford.edu, we have provided a compile_corn.m file. Run this file to generate the mex files. If you have a local installation of OpenCV, then change the dependencies in the compile_all_platforms.m file accordingly.
  - To get started, test your code on a sample video as shown in main.m A reasonable implementation should get you a mAP of 1.0 and average overlap ≥ 0.8 on the first 20 frames of the provided "Car4" sequence. You should also be able to get an average overlap ≥ 0.68 and mAP ≥ 0.78 on the full "Car4" sequence.
- **Analytical study and extensions:** You should have at least one or two of these, here are some ideas:

- Feature represantations for detection/learning. Try different features such as HOG or binary features like BRIEF, FREAK, BRISK etc.
- Keypoint/grid/pyramid sampling strategies for feature extraction.
- Classification methods such as linear/kernel SVM, Random forests, boosting, decision trees.
- Strategies for generating positive and negatives such as warping, noise addition, shifting of confident positive and negative patches.
- An online learning method for speed-up.
- Strategies for fusing the dtection bounding boxes from your learned detector, and the boxes obtained from the KLT tracker.

References:

- Kalal et al. **Tracking-Learning-Detection**. PAMI 2010.
- Ojala et al. **Local Binary Patterns (LBP)**. PAMI 2002.
- Calonder et al. **BRIEF** . ECCV 2010.

Important Dates:

- Presentation slides due to alahi@stanford.edu: Sun, May 10 (5:00 pm)
- A short presentation of your work: Mon, May 11 (in class)
- Deadline of submitting the code and write-up: Tues, May 12 (5:00 pm)

Project 3: Detection with R-CNN

Project Description:

- In this project, you will be implementing the core algorithm of an R-CNN.
- Starter code is available **here**.
- **What you need to implement:** As detailed in 'readme.txt' within the starter code, you are expected to implement the following:
  - extract_region_feats.m
  - train_rcnn.m
  - train_bbox_reg.m
  - test_rcnn.m

If you'd like to use python instead, that's fine, but be prepared to re-implement the given starter code. In particular, hooking up python with caffe may take some work (but is not impossible: caffe has a python wrapper).

Dataset and Experiments:

- **Evaluation:** Detection evaluation is based on mean average precision (mAP), and you should report your average precision for each class as well as show precision recall (PR) curves. Code has been provided to help with these calculations.
- **Dataset:** Download the images **here**.
  - The dataset contains a training and a test set. In practice, one would also have a validation set to tune parameters. Since R-CNNs take a while to run, we're skipping that for the class.
  - Details of the dataset format are available in readme.txt
- **Analysis and extensions:** You should have at least a few of these (depending on the depth of the extension), here are suggestions:
  - Feature representation. Compare CNN features with more traditional vision features, e.g. HOG, LLC, fisher vectors.
  - Which layer of the CNN to use.
  - Parameters used during training (regularization, overlap thresholds, etc.)
  - Bounding box offset parameterizations for bounding box regression.
  - Training your own, larger CNN.
  - Fine-tuning the CNN on the given images.
  - Other methods for detection, e.g. DPM.
  - Simultaneous detection and segmentation.

References:

- Girshick et al. **Rich feature hierarchies for accurate object detection and semantic segmentation**. CVPR 2014.

Important Dates:

- Presentation slides due to jkrause@cs: Tues, June 2 (5:00 pm)
- A short presentation of your work: Wed., June 3 (in class)
- Deadline for submitting code and write-up: Thurs, June 4 (5:00 pm)