Intro to Image Understanding (CSC420)
Projects

Report Submission Deadline : April 17, 11.59pm, 2023

Max points for project: 40

Projects can be done individually or in pairs. If a project is done in a pair, each student will still need to defend their part. From the report it should be clear what each student contributed to the project. By April 17 you will need to hand in the project report including code. Make the code organized and documented, possibly including scripts that run your pipeline. In the oral defense you'll need to run some of your code and be able to defend it, as well as answer questions about class material. The oral defense will be scheduled shortly after. The grade will evaluate a project report and an oral presentation, at the total of 40% of the final grade.

Whenever you use ideas, code or data from a paper, forum, webpage, etc, you need to cite it in your report. Whenever you use code available from some paper or method, you need to include a short description of the method showing your understanding of the technique. If there were multiple options for techniques or code, please also explain why you chose a particular one.

The grade will take into account the following factors:

- Your ideas to tackle a problem: how appropriate the techniques you chose are for the problem. Coming up with novel ideas is obviously a big plus.
- Your implementation: the accuracy of the solution, speed, and partly also how organized the code is (scripts that run the full pipeline or specific subtasks, documentation). The grade will take into account also the performance of your solution with respect to other students' results.
- Whether you implemented a technique yourself or used code available online
- How you present your results in the report: ideally you would show your results for each task by including a few pictures in your report. Even more ideally, you would show good cases where your method works and also bad cases where it doesn't. Providing some intuitive explanation of the failure cases is a plus.
- Thoroughness of your report: How well you describe the problem and the techniques you chose, how well you analyzed your results and whether your citations to the techniques you used are appropriate.

Do **not** put the video clips we provide online or share with anyone.

Project 1

A maximum of two students can work together for this project.
This project is about analysis of video clips. Here are the tasks to solve:

(a) Please download three short movie trailers from YouTube with recognizable actors or singers. You can make each trailer shorter, but ensure that each clip is at least 20 seconds long.

(b) Please crawl 20 or more actor or singer photos from the web, for example *imdb.com*. Include also photos of actors/singers that appear in the videos you downloaded above.

(c) Detect shots in the videos. A shot is a set of consecutive frames with a smooth camera motion.

(d) (Manually) Annotate shot boundaries in the video. How would you evaluate how well you are detecting the shots? Why is this a good metric? Compute the performance according to your proposed evaluation metric.

(e) Detect faces in the video.

(f) Perform face tracking by correctly associating a face detection in the previous frame to a face detection in the current frame.

(g) Perform automatic matching of each tracked face to faces in your database from (b), to identify the identity of each face in the video.

(h) Visualize your results: produce a video in which you show bounding boxes around the detected faces. Each face bounding box should have text indicating the name of the actor/singer your algorithm identified in the video.

Project 2

A maximum of two students can work together for this project.
Autonomous driving is one of the major research venues these days. A lot of effort is devoted to it by both the academics as well as industry. In this project you'll familiarize yourself with some of the most important problems that arise in the field of autonomous driving.

The input to your algorithm is a stereo image pair and the camera parameters. You will also have available a set of training images where the cars have been annotated with 2D bounding boxes as well as viewpoint. Furthermore, you'll have a few images where the road has been annotated. Here are the tasks to solve:

1. Compute disparity between the two stereo images. We do not mind if you use existing code as long as you include a description of the algorithm you used, showing you understand what it is doing.

2. Compute depth of each pixel. Compute 3D location of each pixel.

3. Train a road classifier on a set of annotated images, and compute road pixels in your image. Which features would you use? Try to use both 2D and 3D features.

4. Fit a plane in 3D to the road pixels by using the depth of the pixels. Make sure your algorithm is robust to outliers.

5. Plot each pixel in 3D (we call this a 3D point cloud). On the same plot, show also the estimated ground plane.

6. Detect cars in the image. You can use the pre-trained models available here: `http://kitti.is.tue.mpg.de/kitti/models_lsvm.zip`, and detection code available here: `http://www.cs.berkeley.edu/~rbg/latent/`. Alternatively, you can use other detectors available online such as SSD, Fast(er)-RCNN, etc.

7. Train a classifier that predicts viewpoint for each car. The viewpoint labels are in 30° increments, thus train a 12-class classifier. Which features would you use?

8. Show a test image with the detected car bounding boxes and show the estimated viewpoints by plotting an arrow in the appropriate direction.

9. Given the ground plane, estimated depth, and the location of the car's bounding box, how would you compute a 3D bounding box around each detected car? Add the 3D bounding boxes to your plot from 5.

Project 3

This is a project for a single student (no pair work allowed).

In this project the task is to do pixel-level labeling with a set of semantic classes. In particular, you will be given a fashion dataset from the following paper:

Kota Yamaguchi, M Hadi Kiapour, Luis E Ortiz, Tamara L Berg, “Parsing Clothing in Fashion Photographs”, CVPR 2012.
`http://vision.is.tohoku.ac.jp/~kyamagu/research/clothing_parsing/`

The tasks for this project are:

1. (Automatically) label each pixel as either “person” or “background”.
2. (Automatically) label each pixel as either: “background”, “skin”, “hair”, “t-shirt”, “shoes”, “pants”, “dress”.

You’ll be given a train, validation and test split, as well as an evaluation function. In your report please include examples of your results as well as report performance on both tasks.

Note that the paper above provides code. While you can build on top of the paper’s ideas we expect you to implement some of your own ideas as well, using newer techniques/classifiers.

Project 4

This is a project for a single student (no pair work allowed).

Imagine a phone app where you take a picture of a book and the app tells you which book it is. In this project, the goal is to localize and recognize a book cover in an input image given a database of a large set of book covers. You will need to implement the following paper: Nister, Stewenius, *Scalable Recognition with a Vocabulary Tree*, CVPR 2006, `http://www-inst.eecs.berkeley.edu/~cs294-6/fa06/papers/nister_stewenius_cvpr2006.pdf`
Here are the tasks:

1. Take a few pictures of a few books that you have at your disposal (if you don't own a book, you can go to a library). Put the books on the table or the floor, and take photos from different angles. We will refer to these photos as *test images*.
2. Crawl at least 100 book cover images from the web. Also download images of covers of books from above. For example, Amazon has a wide selection of covers.
3. Implement homography estimation with RANSAC to match one book cover image to a test image.
4. Implement the efficient retrieval approach by Nister and Stewenius.
5. For a test image, retrieve top 10 matches (book cover images from the database you created) returned via the implemented approach. Compute a homography with your implementation from (1) for each retrieved book cover image and the test image.
6. Find the book cover image from (5) with the highest number of inliers. Plot the test image with the localized book cover and as well as the best retrieved book cover.

Project 5

This is a project for a single student (no pair work allowed).

In this project, you will use the latest 3D deep learning technology. Please read the paper `https://arxiv.org/abs/2003.08934`. Code for this work can be found here `https://github.com/bmild/nerf`. The goal of the project is to reconstruct 3D objects from multiview photographs and perform novel view synthesis.

1. Take multiview photos of a few selected objects, but moving the camera around the object. The nerf method typically utilizes 40 to 60 photos per object.
2. Compute camera matrices for all the photos. Please try to write your own implementation. Please also check the nerf github repository for instructions on using existing packages. Please compare your own camera estimates with the ones using existing packages.
3. Implement nerf by yourself.
4. Provide visualizations of novel view synthesis results.

Useful Code and Techniques

Shot Detection. A simple way of detecting shot boundaries is to look at differences in color histograms between two consecutive frames. An even better way is to look at Displaced Frame Distances, described in Section 2.1 of this paper:

Makarand Tapaswi, Martin Baeuml and Rainer Stiefelhagen, *"Knock! Knock! Who is it" Probabilistic Person Identification in TV Series*, CVPR 2012, `https://cvhci.anthropomatik.kit.edu/~mtapaswi/papers/CVPR2012.pdf`

More options are discussed in this paper:
Y. Yusoff, W. Christmas, and J. Kittler, *A Study on Automatic Shot Change Detection. Multimedia Applications and Services*, 1998, `http://www.cs.utoronto.ca/~fidler/slides/CSC420/papers/shots.pdf`.

Face Detection. A few options:

- The first paper that did great on faces was Viola-Jones face detector:
  Paul Viola and Michael Jones, Rapid object detection using a boosted cascade of simple features, CVPR, 2001, `https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf` There is lots of code online.

- Paper:
  P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, *Object Detection with Discriminatively Trained Part Based Models* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, Sep. 2010 `http://cs.brown.edu/~pff/papers/lsvm-pami.pdf`
  Code: `http://www.rossgirshick.info/latent/`
  with a trained model called `dpm_baseline.mat` available here:
  `https://bitbucket.org/rodrigob/doppia/src/tip/data/trained_models/face_detection/?at=preparing_v2`. This detector may work a little better. You can also check a detailed analysis and code for different face detectors here: `http://markusmathias.bitbucket.org/2014_eccv_face_detection/`

- Paper:
  X. Zhu, D. Ramanan. *Face detection, pose estimation and landmark localization in the wild*, CVPR 2012, `http://www.ics.uci.edu/~xzhu/face/`.
  This detector however also gives you the keypoints for the facial landmarks.

Tracking Here are some options (papers), choose the one it suits you best:

- Tracking via dynamic programming: `https://engineering.purdue.edu/~qobi/papers/acs2012b.pdf`

- Tracking via Hungarian method: `http://www.cvlibs.net/publications/Geiger2014PAMI.pdf`, first 3 paragraphs of Section 4.1

- Tracking via Kalman filter: lots of tutorials online

Available code:

- Check http://www.ics.uci.edu/~dramanan/ for paper (and code): 'H. Pirsiavash, D. Ramanan, C. Fowlkes. "Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects", Computer Vision and Pattern Recognition (CVPR), 2011
- http://www.cvlibs.net/software/trackbydet/
- http://research.milanton.de/dctracking/

Stereo For autonomous driving, it's best to check the stereo challenge of the KITTI dataset (http://www.cvlibs.net/datasets/kitti/). It has links to code and gives you running time of each algorithm. Some options:

- http://userpage.fu-berlin.de/spangenb/
- http://ttic.uchicago.edu/~dmcallester/SPS/index.html

Flow For autonomous driving, it's best to check the stereo challenge of the KITTI dataset (http://www.cvlibs.net/datasets/kitti/). It has links to code and gives you running time of each algorithm.

- http://people.csail.mit.edu/celiu/OpticalFlow/
- Matlab has some functions to compute flow: http://www.mathworks.com/help/vision/ref/opticalflow.html
- OpenCV has flow and trackers, and other useful stuff: http://opencv.org/documentation.html

Other For features, super pixels, classifiers, etc, look under Resources of class webpage: http://www.cs.utoronto.ca/~fidler/teaching/2019/CSC420.html