COMS30068 - Image Processing and Computer Vision with Coursework (2022-23)

Shape Detection and 3D from Stereo

Assignment Weeks 9-11

COMS30030

School of Computer Science, Electrical and Electronic Engineering, and Engineering Maths

Overview: This assignment is the summative piece of unit coursework for COMS30068 for those who elected to do the CW version of COMS30030. The assignment comes in TWO major parts. The CW focusses on classical computer vision algorithms for object and shape detection, and stereo analysis. It runs over 3 weeks and must be completed individually.

Assessment: You will be marked on both code and report for both parts. Please do not attempt to copy code or report text, etc. You are allowed to use any of OpenCV's functionalities – except where stated (i.e. you must use your own Hough Transform in Part 1) - but, in any case, you must understand the algorithmic workings of the used functions. If you do use other code/text that is not your own, please declare this in your report with appropriate citation. **Note, plagiarism is taken very seriously based on university plagiarism policies and processes and plagiarism detection software will be used. Please avoid it**.

Submission: Each student is required to upload their full piece of work (incl all source code files, and your two separate reports in PDF). One single ZIP file containing two folders must be submitted. The folders should be named as follows: "youruserid-Part1" and "youruserid-Part2". Each folder should contain the source code and the report relevant to the corresponding part of this coursework. Your reports should be named as youruserid-PartX.pdf, replacing X with 1 or 2 correspondingly.

Your single ZIP file should be submitted on Blackboard under "Assessment, submission and feedback" **on UNIT COMS30068** before the ***deadline at 13:00 on Thursday 8th December 2022.*** Make sure you submit it an hour early or so (not last minute!) to avoid upload problems. **Even if your upload is a single second late you will immediately have 10% deducted (per day) from your mark by the system (a universal BB imposition - out of our control).**

Managing your assignment: Limited support will be provided on Teams. TAs will occasionally monitor the COMS30030 Teams channel in case there are queries to answer. Please do not ask "Does this look correct enough to get the marks?" Note, the TAs cannot provide academic solutions to the coursework, as per CS Department's instructions.

PART I - Shape Detection ⛔
(50 marks)

Task Overview

Introduction. Detecting (locating & classifying) instances of an object class in images is an important application in computer vision as well as an ongoing area of research. This assignment requires you 1) to experiment with the classical Viola-Jones object detection framework as discussed in the lectures and provided by OpenCV, and 2) to combine it with other detection approaches to improve its efficacy. Your approach is to be tested and evaluated on a small image set that depicts aspects of an important traffic sign – No Entry.

Basic Task Structure for Part 1: The task consists of 3 subtasks which incrementally build upon each other:

- ***First Subtask*** ***(up to max 15 marks)***: In this task you will build your own object detector by training the Viola-Jones framework on a customised object class, that is `no entry signs'. You will then test the detector's performance and interpret your results on an image set. This part lets you gain experience on how to train and evaluate the Viola Jones framework in practice.

- ***Second Subtask*** ***(up to max 15 marks):*** In this subtask you will combine your detector with **your own implementation** of the Hough transform to detect component configurations of the shape of no entry signs in order to improve detection performance. (**Do not use OpenCV's implementations of the Hough transform!**) It is up to you to decide if you opt to implement the Hough transform for lines, circles, ellipses, your own parameterized shape or the generalised Hough transform or combinations of them for this task. This task will test your engineering skills of practically applying taught concepts and integrating them within an application context.

- ***Third Subtask*** ***(up to max 10 marks):*** In order to be able to reach marks close to and above the first class boundary, we ask you to research, understand and use in OpenCV at least one other appropriate vision approach to further improve the detection efficiency of the system.

The ***final 10 marks for this part of the coursework*** are reserved for excellence throughout your work, how well you inform your development on evidence and evaluation, and how well you demonstrate your comprehension of the techniques used. Here, we look for outstanding understanding, concise presentation, excellent write-up and clean implementation.

Submission

Report - For significant marks in this category, it should become clear in **a (maximum 3-page) report** that the student has reached a level of understanding that allows for contextualisation and reflection well beyond the taught content. More details of what should be in the report are given in the Subtask descriptions below.
Code - Must be clearly commented, explaining the key components and the functionality. Note: the code will be tested and so please make sure that any required inputs and outputs are clearly explained at the top of the code.

Strictly nothing after the 3rd page *for this part of the assignment* will be used for marking, so please do not let your subtask descriptions spill onto page 4!

Subtask 1: The No-Entry Sign Detector

This subtask requires you to build an object detector that recognises no entry signs. Please download the material from the unit github website: https://github.com/UoB-CS-IPCV/CW-I-Shape-Detection. The initial steps of this subtask introduce you to OpenCV's boosting tool, which you can use to construct an object detector that utilises Haar-like features. We provide 16 real-world images, called **NoEntry0.bmp** to **NoEntry15.bmp**.

In order to give you a smooth start, if you don't use lab machines, please create virtual environment with Python 3.6, activate your environment, and then install OpenCV packages.

```
conda create -n ipcv36 python=3.6
conda activate ipcv36
conda install -c menpo opencv
```

You are given **no_entry.jpg** containing a no entry sign that can serve as a prototype for generating a whole set of positive training images.

In addition, a large set of negative images (do not contain any no entry signs) is provided in a directory called **negatives** and a text file **negatives.dat** listing all filenames in the directory.

no_entry.jpg

First, create your positive training data set of 500 samples of no entry signs from the single prototype image provided. To do this, you can run the tool **opencv_createsamples** via the following single command if you use lab machines and execute it in a folder that contains the **negatives.dat** file, the **no_entry.jpg** image and the **negatives** folder:

```
opencv_createsamples -img no_entry.jpg -vec no_entry.vec  -w 20 -h 20 -num 500 -maxidev 80 -maxxangle 0.8 -maxyangle 0.8 -maxzangle 0.2
```

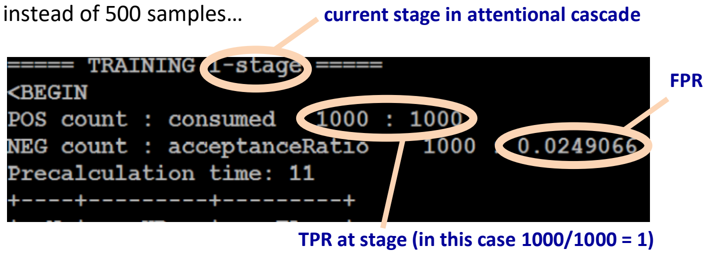
This will create 500 tiny 20×20 images of no entry signs (later used as positive training samples) and store in the file **no_entry.vec**, which contains all these 500 small sample images. Each of the sample images is created by randomly changing viewing angle and contrast (up to the maximum values specified) to reflect the possible variability of viewing parameters in real images better.

Now use the created positive image set to train a no entry sign detector via AdaBoost. To do this, create a directory called **NoEntrycascade** in your working directory. Then run the **opencv_traincascade** tool with the following parameters as a single command in your working directory:

```
opencv_traincascade -data NoEntrycascade -vec no_entry.vec -bg negatives.dat -numPos 500 -numNeg 500 -numStages 3 -maxDepth 1 -w 20 -h 20 -minHitRate 0.999  -maxFalseAlarmRate 0.05 -mode ALL
```

This will start the boosting procedure and construct a strong classifier stored in the file **cascade.xml,** which you can load in an OpenCV program for later detection as done in Lab4: Face Detection (face.py).

During boosting the tool will provide updates about the machine learning in progress. Here is an example output when using 1000 instead of 500 samples…



```
===== TRAINING 1-stage =====
<BEGIN
POS count : consumed   1000 : 1000
NEG count : acceptanceRatio    1000 : 0.0249066
Precalculation time: 11
+----+---------+---------+
```

current stage in attentional cascade
FPR
TPR at stage (in this case 1000/1000 = 1)

The boosting procedure considers all the positive images and employs sampled patches from the negative images to learn. The detector window will be 20×20. To speed up the detection process, the strong classifier is built in 3 parts (numStages) to form an attentional cascade as discussed in the Viola-Jones paper. The training procedure may take up to 15min for reasons discussed in the lectures – stop the training and restart if it exceeds this time.

What to say in your Report (roughly 1 page):

a) ***TRAINING PERFORMANCE:*** The training tool produces a strong classifier in stages. Per stage the tool adds further features to the classifier and prints the achieved TPR and FPR (false positive rate) for that point on the training data (see Figure above). Collate this information into a graph that plots TPR vs FPR on the training data for the three different stages. Produce this graph in your report and briefly interpret what it shows.

b) ***TESTING PERFORMANCE:*** Test the no entry sigh detector's performance on all given example images. Produce the result images with bounding boxes drawn around detected no entry sign candidates (in green) and ground truth (in red) and include 3 of them in your report. In tabular form, calculate the overall TPR and F1 score per image and the average of these scores across the 16 images. Briefly discuss the performance achieved and give reasons for the different TPR values compared to the performance achieved in a).

Subtask 2: Integration with Shape Detectors

For the second subtask, use at least one Hough Transform on the query images in order to detect important shape configurations of no entry signs. Feel free to use and/or adapt your implementation of the Hough Transform from former formative tasks. You must implement your own Hough transform(s). You may reuse your code from Lab 3: Coin Counter Challenge**.** Utilize the information (e.g. on lines, circles, ellipses) to aid no entry sign detection~~.~~ Think carefully how to combine this new evidence with the output of your Viola-Jones detector in order to improve on results. Implement and evaluate this improved detector.

What to say in your Report (roughly 1 page):

a) ***HOUGH DETAILS:*** Show in your report for two of the given example no-entry sign images which best exhibit the merit and limitations of your implementation: 1) the thresholded gradient magnitude image used as input to the Hough Transform, 2) a 2D representation of the Hough Space(s), 3) the result images showing final detections using bounding boxes (in green) and the ground truth (in red).

b) ***EVALUATION:*** Evaluate your detector on all of the example images. Provide the TPR and F1-score for each of the test images and their average across the 16 images in a table. Note in extra table columns the difference w.r.t. to the detection performances using only the Viola-Jones detector. Briefly note in bullet points the key merits and shortcomings of your now enhanced implementation.

c) ***DETECTION PIPELINE:*** In a flow diagram, depict how you have combined evidence from the Hough Transform(s) and Viola-Jones detector. In bullet points, explain briefly your rationale behind the way you have combined evidence.

Subtask 3: Improving your Detector

The final subtask allows you to develop the no entry sign detector further into a direction you choose. We ask you to identify and utilise some image aspects/features able to improve detection results further. This will generally include identifying, researching, understanding and using in OpenCV one other appropriate vision approach to further improve the detection efficacy and/or efficiency.

What to say in your Report (roughly 0.5-1 page):

a) ***IDEA:*** In bullet points, explain briefly your rationale behind selecting the approach you have taken.

b) ***VISUALISE:*** Visualise important aspects of your technique in two of the given example images of no entry signs selected to best exhibit the merit of your approach.

c) ***EVALUATE:*** Evaluate your final detector on all of the example images, show the improvements in TPR and F1-score compared to previous approaches. Briefly note in bullet points the key merits and shortcomings of your final implementation.

Notes on Your Submission for Part 1

Include your source code and your maximum 3-page PDF report, and if needed, include a readme.txt to explain how to compile/build. Your final detector program should take one parameter, that is the input image filename, and produce at least an image **detected.jpg**, which highlights detected no entry signs by bounding boxes. You can use your own machines or lab machines to develop your program, but please test that it runs on the lab machines seamlessly when it comes to marking. Make sure you regularly save/backup your work and monitor your workload throughout the duration of the project

Part II - 3D from Stereo

(50 marks)

This part of the assignment assumes that you are familiar with the 3-D simulator used in the 3-D from Stereo Lab Sheets I and II and that you have completed all the tasks in those lab sheets. The assignment involves determining corresponding points in two views of a scene and using the correspondences to reconstruct 3-D points and spheres in the scene. In your report, you must describe the tasks and experiments completed, including an analysis of the results. You are also required to submit your code. Both are required to obtain marks.

1. Download and run the assignment version of the 3-D simulator used in lab sessions from the unit github website: https://github.com/UoB-CS-IPCV/CW-II-3-D-from-stereo. Refer to the 3-D from Stereo Lab Sheet I for details of how to run the code.
2. When you run the simulator, you should see 6 spheres of different sizes located on a plane. You should also find two images in your current directory corresponding to the images captured by two virtual cameras (VCs). Each time you run the code you will see a different arrangement of spheres of different sizes and images captured from different viewpoints. You should also be able to switch to a second visualisation which shows the plane and centres of the spheres.
3. Either use your own Hough circle detector (if you developed one for Part I) or the OpenCV HoughCircles() to detect circles in each VC image. Check and document the results, noting any dependencies on any parameter settings.
4. Select one of the VCs as the reference view and for each detected circle centre in its image, compute the corresponding epipolar line for the other VC view. Draw the line and check that it is correct.
5. Hence use the epipolar line for each detected circle centre in the reference view to find the corresponding circle in the other VC view.

2.

6. For each circle correspondence, compute the 3-D location of the associated sphere centre using the 3-D reconstruction algorithm described in the lectures.
7. Display the estimated sphere centres alongside the ground-truth centres in the visualisation. Compute the errors in the sphere centre estimates. Note: think carefully about how you should compute the errors.

3.

8. Using the circle detections in the reference and viewing images, estimate the radius of each sphere.
9. Hence display the estimated spheres alongside the ground truth spheres and compute the error in the radius estimates. You will need to think carefully about how you display the estimated and ground truth spheres so as to allow comparison, since comparison may be difficult if you display the estimates as solid spheres. For example, you may want to draw a single circle to represent each sphere to allow comparison.
10. Test and evaluate the performance of your implementation on multiple instances of running the simulator. Ideally it should be successful for every case. If not, note when it does not work and try to find out why.

Submission

Code - that is clearly commented, explaining the key components and the functionality. Note: the code will be tested and so please make sure that any required inputs and outputs are clearly explained at the top of the code.

Report - detailing, discussing and analysing the tasks and experiments completed and the results obtained, including example visualisations and showing quantitative and qualitative results of the estimation and any failure cases. This should be a maximum of 4 pages. There is no set format for the report - you should present and include relevant material as you see fit.

Marking Scheme This part is worth 50% of the CW mark for the unit. Marks will be allocated as follows. For components 1-3, marks will be based on the content of the report w.r.t description, analysis and evaluation and on the code.

1. Sphere correspondence - 15%
2. Centre reconstruction - 15%
3. Sphere reconstruction - 10%
4. Overall standard of presentation, analysis and evaluation - 10%

Administrative Notes from the Department of Computer Science

Deadline
The deadline for submission of all optional unit assignments is 13:00 on Thursday 8th of December (due to the fact that the University discourages Friday deadlines). Students should submit all required materials to the “Assessment, submission and feedback” section of Blackboard - it is essential that this is done on the Blackboard page related to the “With Coursework” variant of the unit.

Time commitment
You are expected to work on both of your optional unit courseworks in the 3-week coursework period as if it were a working week in a regular job - that is 5 days a week for no more than 8 hours a day. The effort spent on the assignment for each unit should be approximately equal, being roughly equivalent to 1.5 working weeks each. It is up to you how you distribute your time and workload between the two units within those constraints.

You are strongly advised NOT to try and work excessive hours during the coursework period: this is more likely to make your health worse than to make your marks better. If you need further pastoral/mental health support, please talk to your personal tutor, a senior tutor, or the university wellbeing service.

Academic Offences
Academic offences (including submission of work that is not your own, falsification of data/evidence or the use of materials without appropriate referencing) are all taken very seriously by the University. Suspected offences will be dealt with in accordance with the University’s policies and procedures. If an academic offence is suspected in your work, you will be asked to attend an interview with senior members of the school, where you will be given the opportunity to defend your work. The plagiarism panel are able to apply a range of penalties, depending the severity of the offence. These include: requirement to resubmit work, capping of grades and the award of no mark for an element of assessment.

Extenuating circumstances
If the completion of your assignment has been significantly disrupted by serious health conditions, personal problems, periods of quarantine, or other similar issues, you may be able to apply for consideration of extenuating circumstances (in accordance with the normal university policy and processes). Students should apply for consideration of extenuating circumstances as soon as possible when the problem occurs, using the following online form:
https://www.bristol.ac.uk/request-extenuating-circumstances-form

You should note however that extensions of any significant length are not possible for optional unit assignments. If your application for extenuating circumstances is successful, you may be required to retake the assessment of the unit at the next available opportunity (e.g. during the summer reassessment period).