Assignment Weeks 6-10 (pair programming work, summative, 50%)

The Dartboard Challenge

Overview: This assignment is the summative piece of unit coursework. It runs over approximately 4 weeks. The coursework is worth 50% of the unit mark for COMS30121. It is to be completed in your programming teams, which are registered on SAFE. You must report any change to the pair structure to the course director BEFORE starting your assignment.

Submission: Each student is required to upload their full piece of work (incl all source files, and your PDF team report) as a single ZIP file to SAFE before the ***deadline at 17:59 on Monday 3rd December 2018.*** Make sure you submit it early (not last minute!) to avoid upload problems. ***Each team member*** has to upload an identical copy of the zip file containing the team's work.

Assessment: You will be marked on your code, report and a brief presentation. Both team members should understand all of the code written and must be present for the final presentation. During the presentation we will run and discuss the submitted program in sessions to be scheduled in weeks 10, 11 and 12. All the team members will have to attend their presentation slot to get a mark. At your presentation, we will ask you questions about your work, give you feedback and you will be able to showcase your work's merit. Do not attempt to plagiarise, copy code between teams etc. We will ask you intricate questions about your work and code in the presentation. Apart from the Hough transform, you are allowed to use any of OpenCV's functionality – but in any case all the team must understand the basic workings of the used functions.

Time Management: You should be able to work your way through the task in not more than approximately 18h per person including lab slots. For instance, subtask one should take you no more than a 2h lab session, subtask two should take you less than 3h, and task three should take not much more than approx. 10h given your existing, formative work on the subject which you are allowed to use and develop further in this assignment. If you find your team requires significantly more time than suggested or struggles then please flag this early and make sure you seek help in labs and forum.

Task Overview: Detecting Dart Boards

Introduction. Being able to detect and locate instances of an object class in images is important for many computer vision applications as well as an ongoing area of research. This assignment requires you 1) to experiment with the Viola-Jones object detection framework as discussed in the lectures and provided by OpenCV, and 2) 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 a popular sport and past time – darts.

Basic Task Structure: The task runs over approximately 5 weeks. It consists of four subtasks which incrementally build upon each other:

- ***First Subtask*** ***(up to max 10 marks)***: This introductory part will ask you to run and understand the usage of the Viola-Jones detector as provided by OpenCV. In particular, you will experiment with the pre-built off-the-shelf face detector and apply it so some example images. This part lets you gain experience on how and how well the framework can operate in practice.

- ***Second Subtask*** ***(up to max 20 marks):*** In this second task you will build your own object detector by training the Viola-Jones framework on a customized object class, that is `dartboards'. 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.

- ***Third Subtask*** ***(up to max 35 marks):*** In this subtask you will combine your detector with your own implementation of the Hough transform for component configurations of the dartboard shape in order to improve detection performance. (Do not use OpenCV's implementation 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.

- ***Fourth Subask*** ***(up to max 10 marks):*** In order to be able to reach marks above the first class boundary, we ask you to research, understand and use in OpenCV one other appropriate vision approach to further improve the detection efficacy and/or efficiency.

The ***final 25 marks for this coursework*** are reserved for excellence throughout your work, how well you inform your development on evidence and evaluation, and how well you comprehend the techniques used. Here we look for an outstanding understanding, concise presentation, excellent write-up and clean implementation. For significant marks in this category it should become clear during the presentation that a student has reached a level of understanding that allows for contextualisation and reflection well beyond the taught content. Full marks in this category are reserved for work that is novel and publishable in principle.

(Note that students within a team may receive different grades dependent on their understanding of the work & contribution to it.)

Subtask 1: The Viola-Jones Object Detector

You are given some sample code in the file **face.cpp** and 16 real-world images called **dart0.jpg** to **dart15.jpg**:

First, understand, compile and build the provided **face.cpp** source program to generate an executable. (You may need to adjust include paths if you use your own machine).

On lab machines you would compile and build the provided file like this:

g++ face.cpp /usr/lib64/libopencv_core.so.2.4
/usr/lib64/libopencv_highgui.so.2.4
/usr/lib64/libopencv_imgproc.so.2.4
/usr/lib64/libopencv_objdetect.so.2.4

Make sure the built program reads the provided XML file **frontalface.xml,** which contains a strong classifier trained using AdaBoost for detecting frontal human faces. The program also expects the name of an example image file to which it applies the classifier. For instance, given the image **dart4.jpg** and the file **frontalface.xml** in the directory where you built your program in, you can run your compiled and built program as follows:

./a.out dart4.jpg

The program outputs the number of faces found to the console and the resulting detections are finally visualized in the produced output image called **detected.jpg**.

The First Page of your Report (strict limit):

a) Annotate the test images to generate ground truth, then test the face detector's performance (with the given parameters as provided by **face.cpp**) on five given example images: **dart4.jpg, dart5.jpg, dart13.jpg, dart14.jpg** and **dart15.jpg**. Produce the five result images with bounding boxes drawn around detected face candidates and include them in your report.

b) Calculate and note in your report the TPR (true positive rate) for the images **dart5.jpg** and **dart15.jpg**, that is the fraction of successfully detected faces out of all valid faces in an image. Discuss in your team and briefly note in your report 1) some practical difficulties in assessing the TPR accurately, 2) why it is always possible to achieve a TPR of 100% on any detection task, and 3) implement a small piece of code to calculate the F1-score of your face detection system accurately and meaningfully from ground truth and a test run on any given test image set.

Subtask 2: Building & Testing your own Detector

The second subtask requires you to build an object detector that recognises dartboards. 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.

In order to give you a head start you are provided with readily compiled versions of two OpenCV tools you can use to build your object detector: **opencv_createsamples** and **opencv_traincascade**. (These binaries are part of all standard OpenCV installations.) You are also given an image **dart.bmp** containing a dartboard that can serve as a prototype for generating a positive training image set.

In addition, a large set of negative images (containing no dart boards) is provided in a directory called **`negatives'** and a text file **negatives.dat** listing all filenames in the directory.

dart.bmp

First, create your positive training data set of 500 dartboard samples from the single prototype image provided. To do this, you can run the tool **opencv_createsamples** via the following command if you use lab machines and execute it in a folder that contains the **negatives.dat** file, the **dart.bmp** image and the **negatives** folder:

./opencv_createsamples -img dart.bmp -vec dart.vec
-neg negatives.dat -w 20 -h 20 -num 500
-maxidev 80 -maxxangle 0.8 -maxyangle 0.8 -maxzangle 0.2

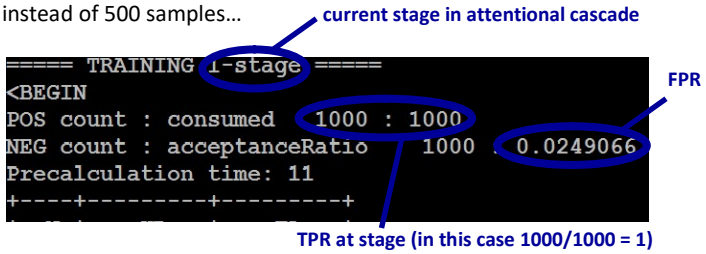
This will create 500 tiny 20×20 images of dart boards (later used as positive training samples) and create the file **dart.vec**, which contains all these 500 small sample images. Each of the sample images is created by randomly changing viewing angle and contrast (up to the maximum values specified) to reflect the possible variability of viewing parameters in real images better.

Now use the created positive image set to train a dartboard detector via boosting. To do this, create a directory called **dartcascade** in your working directory. Then run the **opencv_traincascade** tool with the following parameters in your working directory:

./opencv_traincascade -data dartcascade -vec dart.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 as done in Subtask 1 of the assignment.

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…



The boosting procedure considers the 1000 positive images and employs sampled patches from the 1000 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.

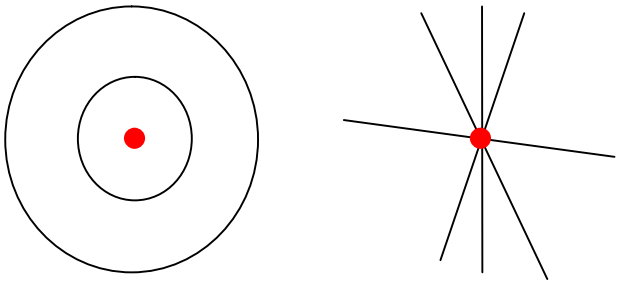
The Second Page of your Report (strict limit):

a) 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). Collate this information into a scatter plot 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) Test the dartboard detector's performance on all given example images. Produce the result images with bounding boxes drawn around detected dartboard candidates and include 4 of them in your report. In tabular form, calculate the overall F1 score per image and the average across all 16 images. Briefly discuss the performance achieved and compare it to the level of performance achieved in a).

Subtask 3: Integration with Shape Detectors

For the third subtask, use one Hough Transform on the query images in order to detect important shape configurations of dartboards. Feel free to use and/or adapt your implementation of the Hough Transform from former formative tasks. You must implement your own Hough transform, otherwise you can also use any OpenCV functionality. Utilize the information (e.g. on lines, circles, ellipses) to aid dartboard detection. For instance your system could consider the center of concentric circles or ellipses, and points where multiple line segments intersect:



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.

The Third Page of your Report (strict limit):

a) Show in your report for two of the given example dart 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, 3) the result images showing final detections using bounding boxes.

b) Evaluate your detector on all of the example images. Provide the F1-score of overall detection performance for the set of example test images in a table. Document your overall detection results (for instance by using precision/recall measures) and briefly note in bullet points the key merits and shortcomings of your implementation.

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

Subtask 4: Improving your Detector

The final subtask allows you to develop the dartboard detector further into a direction you choose. We ask you to identify and utilize 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.

The Forth Page of your Report (strict limit):

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

b) Visualize important aspects of your technique in two of the given example dart images selected to best exhibit the merit of your approach.

c) Evaluate your final detector on all of the example images, show the improvements in F1-score. Document your overall detection results and briefly note in bullet points the key merits and shortcomings of your final implementation.

Program Structure for Submission

We ask you to submit a zip file to SAFE before the deadline. The file should contain all your team's source code and your PDF report, if needed include a readme.txt to explain how to compile/built. Your final detector program will be presented in your scheduled presentation and should take one parameter, that is the input image filename, and produce at least an image **detected.jpg**, which highlights detected dartboards by bounding boxes. You can use your own machines or lab machines to develop. Make sure you regularly save/backup your work, meet regularly with your team partner and discuss your work together throughout the duration of the project.