

## Assignment 3: Automatic Color-based Face Detection (12%)

### TDI2131 Digital Image Processing



Automatic face detection in images is a very popular application that makes good use of image processing techniques (and also some pattern classification methods) and it is useful for many real-life imaging automations such as face identification for biometric authentication, surveillance of human activity, image retrieval and image classification. In color images, face detection may have an additional advantage – the color feature.

How do we determine which regions of an image could possibly belong to a human face?

Is there a certain range of colors that characterizes human skin colour?

Can shape be used to differentiate between face areas and limb areas?

### TASK

This assignment should be done in a group of 1-2 people. You are required to design and implement an automatic face detection algorithm, which is able to **efficiently** and **accurately** detect human faces in a color image.

Look under “Algorithm Design Guide” for some hints on how to design your algorithm. Similar to previous assignments, there are more than one ways of achieving the task. Since this is a group assignment, you should try to discuss and work together to come up with some good methods.

### FILES FOR DOWNLOAD

All students will be processing the same set of images.

Please access the Assignment 3 download page: <http://pesona.mmu.edu.my/~johnsee/teaching/tdi2131/as3>

#### I. Training Images and Ground Truth Data

Training images are sample images that you can use to design your algorithm, and they are particularly useful for fine-tuning and improving on the algorithm. With a variety of training images available, you will find them useful for designing an algorithm that is *invariant* (or able to accommodate) towards various skin colours, face pose and background illumination (lighting).

Ground truth data gives you an indication of where the “correct” face regions are. Your algorithm should eventually try to locate these “correct” regions automatically, as close to the ground truth data as possible.

There are altogether 4 training images. Here's the list of images and their respective ground truth data:

- [group1.jpg, group1\\_gtd.png](#)
- [group2.jpg, group2\\_gtd.png](#)
- [group3.jpg, group3\\_gtd.png](#)
- [group4.jpg, group4\\_gtd.png](#)

All images are regular JPEG images.

In the ground truth images, face regions are labeled with ones, while background regions are labeled with zeros.

## II. Evaluation Program

Your face detection algorithm will be evaluated using this MATLAB function: [evaluate.m](#) (adapted from Chuo-Ling Chang, Stanford University, 2003)

## III. Test Images

Test images (unlike training images) are additional images, which will be used to evaluate your algorithm for its effectiveness in processing new images, after you have done the earlier training. They **ARE NOT ALLOWED** to be used as training images. Test images will be released a week before the deadline.

## INSTRUCTIONS

There are 3 steps in this face detection assignment:

- I. Design and implement your algorithm (as a function) in MATLAB based on the given training images and ground truth data.
- II. Test your algorithm with the training images using the evaluation program.
- III. Evaluate performance of your algorithm with training and test images (which will be released one week before deadline).

### I. Implementation of your algorithm

#### *Format*

Your main function has to be in this format:

```
function outFaces = faceDetection(inImage)
```

**Input:** An image matrix formed by

```
inImage = double(imread(imageFilename)),
```

 where `imageFilename` is the name of one of the training images, or the final test images

**Output:** A N-by-2 matrix named `outFaces` containing the centroid coordinates for each detected face region, where

`N` is the number of faces you've detected

```
outFaces(:, 1)
```

 contains the detected vertical coordinates (row index)

```
outFaces(:, 2)
```

 contains the detected horizontal coordinates (column index)

You can call other sub-functions under this main function. This main function `faceDetection` serves as the interface with the evaluation program, so no alteration should be done to the function name, inputs and outputs. In the evaluation program (you can open to check), the main function is called by

```
outFaces = faceDetection(inImage)
```

### *Time-limit*

Your algorithm should not exceed 7 minutes (420 seconds) as clocked by the evaluation program, or it will be rendered inefficient. The evaluation program checks if you are within the time-limit only after your function finishes running.

Optional: You are allowed to downsample your image to reduce the amount of computation, but downsampling must be part of your own function. However, you still have to return the coordinates of the face regions based on the original image size.

## II. Testing with Evaluation Program

### *Format*

```
[finalScore, detectScore, numHit, numRepeat, numFalsePositive, distance,
runTime, bonus] = evaluate(inImageFilename, refImageFilename)
```

### **Input:**

**inImageFilename:** filename of the training image (or test image)

**refImageFilename:** filename of the corresponding ground truth data

### **Output:**

**finalScore:** detectScore + bonus

**detectScore:** numHit - numRepeat - numFalsePositive

**numHit:** number of faces successfully detected

**numRepeat:** number of faces repeatedly detected

**numFalsePositive:** number of cases where a non-face is reported (or a face falsely reported)

**distance:** root mean squared (RMS) distance between detected centroid point of face region and centroid of the ground truth data

**runTime:** total run time of your face detection function

**bonus:** total bonuses accumulated from accuracy bonus and time bonus

### *Usage example*

```
[finalScore, detectScore, numHit, numRepeat, numFalsePositive, distance,
runTime, bonus] = evaluate('group1.jpg', 'group1_gtd.png')
```

## III. Performance of your algorithm

The performance of your algorithm on both training and test images will be judged by the outputs of the evaluation program. The performance criterion is as follows:

- **Performance score** – based on the detection score + total bonuses you have accumulated from time and accuracy
- **Run-time** – make sure your run-time is within the time limit. Otherwise, your program will have to be terminated without getting any results. All computation time < 60 seconds\* may be entitled to time bonus.
- **Distance** – The smaller the RMS distance value, the closer and more accurate your centroid coordinates

are to the centroid coordinates of the ground truth data. Accuracy is the least important criteria in this evaluation, but it does contribute bonus as well. Distance amounts  $< 12$  pixels\* may be entitled to distance bonus.

[\* Qualifying time and distance values will be tuned again later according to overall performance of ALL assignment hand-ups.]

## ALGORITHM DESIGN GUIDE

- Since color images are used, the search for faces can be narrowed down by finding areas on the image that correspond to human skin color. Is there a compact range of skin color in the RGB model, or are there other color models (such as HSI, YCbCr, CIE) that can better represent skin color?
- The given ground truth data of the training images may be useful for finding out more about the representation of skin color.
- If you *only* attempt to locate skin color regions, you may also need to find out how to separate face regions from other skin-colored regions, such as arms, hands, legs, or even other background objects that are very close to skin color!
- There are many methods that you can use to determine whether a skin region belongs to a face. One way is to find out if a region is elliptical enough to qualify as a face (since faces are either round or elliptical whereas hands and legs are not). Another way is using template matching, which requires you to use a “face template” to find matches or its correlation with all the detected skin regions. There are many more ways...
- Don't forget that sometimes enhancement steps (for general improvement of image) or binary morphological operations (for cleaning up your detected regions) may help, if needed.
- How do we process so many regions in one image? Label them first!
- Your final output variable `outFaces` should contain the centroid coordinates for each detected face region. The centroid coordinates of a region can be easily determined by calculating the average position (i.e. average x and y coordinates) among all pixels in a region.

## GROUP REGISTRATION

Please register your group (maximum 2 people) with your instructor as soon as you have one. This is for administrative and logistic purposes. Upon registration, you will be provided with a [group number](#).

## DELIVERABLES

1. You are required to submit a short report, elaborating in detail on the algorithm that you have proposed, and the evaluation results on your training and test images. (If you have more than one techniques at hand, you can evaluate all, but state your choice of the best technique.) Provide a brief analysis of your proposed techniques/methods or discuss any other relevant issues that you faced. There is **NO** format for the report. Feel free to present it the way you think is best. Images to illustrate your algorithm would be a useful addition. Documentation of codes should be done using fixed-width fonts (such as `Courier New`, `Consolas`).
2. Include your `faceDetection` function, and other supplementary Matlab files that accomplishes your task. If you are submitting other functions, please comment clearly on its usage.

Please name your submission files (both report and program ZIP file) with the following naming style, by filling in your group number where indicated by 'XX' (for e.g. 02, 13, 21). All your program files ([faceDetection](#) and other supplementary files) should be ZIPed.

Report	<a href="#">face_groupXX.pdf</a> (due to compatibility issues, please use PDF only. For people who don't know how to make PDF files, you can get a free converter/driver from <a href="#">here</a> or <a href="#">here</a> ).
Program ZIP file	<a href="#">face_groupXX.zip</a>

You are **NOT** allowed to use any graphic/image-editing softwares to enhance your images. After all, you have to hand-up your proposed steps to solve the problem, and your codes will also be run to verify your proposed solution.

### FAQ: How will you be evaluated on this assignment?

If you have doubts over how this assignment will be evaluated, you can be assured that you will not be evaluated *entirely* on the performance of your algorithm (such as detection score, time and distance values), although an algorithm that performs more accurately and efficiently than others should deserve some credit. The performance evaluation is part of encouraging “friendly” competition amongs one another.

You will be evaluated based on

- Novelty/originality of method
- Systematic, clear, and justified use of different methods/processing steps
- Good documentation of your method, convincing delivery of presentation

### SUBMISSION

Do not submit any printed hardcopy. Save the trees!

All files to be submitted should be zipped up and sent to [tdi2131@gmail.com](mailto:tdi2131@gmail.com) with

Subject: **[AS3] Group <Group Number>**

Please do not make multiple submissions or spam the email. Only submit your final version of files.

Deadlines and important dates for Assignment 3:

- **11.59PM, 28 April 2010 (Wednesday, Week 14)** – according to email timestamps.
- **29 April 2010 (Thursday, Week 14)** – Presentation of work during usual lecture slot (4-6pm). All members of the group **MUST** be present during the presentation. Please prepare some simple slides to aid your presentation.

Plagiarised work (report & program included) and late submissions will be penalized accordingly.

**APPENDIX**  
Training Images

