Coursework on Mid-Level Vision: Image Segmentation

1. Introduction

This coursework provides the opportunity to design and implement a computer vision algorithm that can segment images. If you require any clarification to these instructions please use the forum on KEATS.

2. The Task

Your code needs to segment colour images into meaningful parts. To assess the quality of the segmentations your code produces its output will be compared to ground-truth segmentations generated by human observers.

At the end of these instructions are examples of 12 colour images, and next to each are four binary images showing the boundaries between different regions of the image as identified by four different human observers. Notice that different human observers do not always agree on how to segment the same image. Hence, it is impossible to produce an algorithm that performs this task perfectly. Furthermore, it is extremely difficult to find an algorithm that will work across different images (in other words, an algorithm, using specific parameter values, might work well for one image, but the same algorithm with the same parameters might produce very poor results on other images). You should, therefore, attempt to implement a method that works reasonably well across a number of images, rather than try to produce perfect results on one or two images. You might consider using any of the methods for image segmentation covered in week 5 and/or methods of edge-detection covered in earlier weeks. You are also free to use other methods not covered in this module.

A zip file containing the training images and the human segmentations shown at the end of these instructions (plus some additional human segmentations) is available from the module's KEATS webpage. You can use these images in any way you wish, but they are provided primarily to allow you to develop your code and evaluate its performance.

3. System Requirements

Your code must include a m-file called "segment_image.m". The main function in this file should have the following definition:

```
function [seg]=segment_image(I)
```

This function takes one input, `I`, which is a MATLAB variable storing an RGB colour image; i.e. `I` will be a 3-dimensional matrix that has been produced using a command such as: `I=im2double(imread('im1.jpg'));`.

This input variable contains the image that is to be segmented. It will be generated by the code that will be used to evaluate the performance of your code. The input variable, therefore, should not be over-written, replaced, or ignored by your code. Your code must run entirely automatically and not need any user intervention or input (e.g. for setting parameter values).

The single output variable, `seg`, should be a 2-dimensional matrix that has the same number of rows and colums as `I`. This variable will define how the image is segmented. It must be in either of the following formats:

1. a image containing only zeros and ones, where values of 1 indicate the locations of boundaries between different regions of the image.
2. an image where each pixel contains a single integer value that is greater than or equal to one, where the value of each pixel defines the region label associated with the corresponding location in the image.

The `segment_image` function may call other functions, but it may not call compiled code (such as .exe files). Unless the other functions are official MATLAB functions, these additional functions should be included in the same m-file, or be include in the zip file you submit. Your code can also load data, as long as this data is included in the zip file you submit. Your code should run on a PC (running Linux) and not need access to special hardware, such as a GPU or RAM in excess of 16GB.

4. Submission

You should submit a single m-file, or a single zip file containing your code and any additional files that it requires to execute. If submitting a zip file you must ensure that when this file is unzipped that the segment_image.m file appears in the directory in which the zip file was unzipped, and not in a sub-directory.

Together with your code you should also submit a one paragraph description of the method you have used. This can be in the form of a separate pdf document or a block of comments at the top of your segment_image.m file. Within this description you must clearly acknowledge any code that you have copied or modified from online sources, books, or articles, and include full details of the source of this code by providing a URL or a citation. Re-using open-source software is good practice and demonstrates your research skills, however, not acknowledging other people's work is both intellectual property theft and plagiarism. Any submission that is identified to include other people's code which has not been clearly referenced will be reported for misconduct. You may **not** share code with fellow students, even if this is acknowledged. You can use any built-in MATLAB function or any official MATLAB toolbox without acknowledgement.

You need to submit through KEATS by the deadline specified on the module's KEATS webpage.

5. Terms and Conditions

Distribution or sharing of your submitted code, or any code derived from it, is forbidden. By submitting your coursework you agree to abide by these terms and conditions.

6. Assessment

The performance of your code will be assessed by testing the accuracy with which it segments 12 testing images. The testing images are distinct from the training images shown below, but like the training images have been segmented by multiple human observers, and the accuracy of your code will be assessed by how well the segmentations produced by your code correspond to those produced by the human observers.

More specificallly, accuracy will be measured by counting the number of true matches between your algorithm's segmentation and the (variable number of) human segmentations. If a boundary produced by your algorithm matches one marked by any of the human observers it counts as a true match. Furthermore, there is a tolerance in the match that is equal to 3 pixels. The higher the number of true matches the higher the score. However, the score will be reduced by false positives (boundaries in your result that do not match any human result) and false negatives (boundaries marked by all human observers that are missing from your result). The exact method of evaluation can be seen in the function `evaluate.m` that is available on the module's KEATS page. To help you visualize the results produced by your segmentation algorithm you can use the function `compare_segmentations.m`. To use this code you will also need the functions `show_results.m` and `convert_seg_to_boundaries.m`. All these functions can be downloaded from KEATS. Do not include any of these provided functions, nor the training images, in your submission: your code will be evaluated using my versions of these functions and unseen, test, images.

Marks will be awarded primarily for how accurately your code solves the segmentation problem. In addition, credit may be given for how sensible an approach you have chosen to implement and how efficiently and correctly that approach has been implemented. Marks will be deducted if your code needs to be fixed in any way to make it execute.

7. Training Images