Assignment #2
Due: Week 8 September 16, 18:00
Marks: 20 marks

OBJECTIVE

Design a C or C++ (ANSI standard) program that can extract object edges and detect the presence of straight lines.

BACKGROUND

Automated visual quality control and visual monitoring of industrial assembly lines are the application areas where computer vision has proven its efficiency. Vision systems can operate continuously and provide measurement of critical parameters such as position and orientation of objects, their shape, colour, etc. An obvious advantage of such systems is the possibility of controlling image acquisition process optimising illumination, position of cameras and background colour. As a result, images used for automated visual inspection have good quality in terms of contrast and SNR.

Your program shall include the following stages:
1. Rescale the image down by the factor of 2, to reduce computational complexity of the subsequent stages
2. Extract boundaries of objects using 3x3 gradient operators
3. Save the generated b/w image `gradient.bmp`
4. Detect the presence of straight lines and measure their orientation with the precision 3^o
5. Extract boundaries of objects using binary morphology
6. Save the generated b/w image `morphology.bmp`
7. Detect the presence of straight lines and measure their orientation with the precision 3^o
8. Compare performance and efficiency of both methods and reflect your observations, findings and conclusions in the report `assignment2report.txt`

DESIGN SPECIFICATION

You need to use OpenCV library to implement the solution.
It can be tested using `asm2.bmp` test image. Since information about colour is not required, it contains only 8bit/pix luminance component. The image size is 640x480.

Your program shall produce:
1. Two 8bit/pix black/white (b/w) images `gradient.bmp` and `morphology.bmp` (size `320x240`), which represent object boundaries obtained by two boundary extraction methods. The level of background should be set to `0` (black), while the pixels corresponding to boundaries are set to `255` (white).
Comments: You may try to apply a LPF before the gradient-based method to improve overall accuracy of edge detection. If needed, you can also apply binary morphological

methods (erosion, dilation, etc) to smooth boundaries obtained as a thresholded output of the gradient-based method.

Considering the second solution, before applying the morphological boundary detection method, you need to binarize the input image first. The test image provided has a very high contrast and therefore selection of a threshold is not very difficult. For `asm2.bmp` the threshold can be set around 90-100.

2. For each boundary detection method you need to printout parameters of all detected lines in the following format:

```
Line detection in a b/w image produced by gradient operators:
 Line 1 is detected.  Orientation = … degrees
 Line 2 is detected.  Orientation = … degrees
```

or

```
 No straight lines detected
```

Having implemented and tested the program, you should investigate some practical aspects of boundary extraction and straight-line detection. For example:
- You can compare accuracy and computational complexity of two boundary detection methods
- You can evaluate efficiency of edge detection with LPF pre-processing and without it
- You can measure computational complexity and execution time of Hough Transform
- You can analyze sensitivity of your system to noise by running the program with the second test image `asm2n.bmp`

Your observations, findings and conclusions must be summarized in the report `assignment2report.txt`

SUBMISSION

In this assignment you have to submit:
- `assignment2.c` (or `assignment2.cpp`) source file with appropriate comments
- `assignment2report.txt` (`doc`, or `pdf`) report

The report must provided a 1-2 page description of the implemented solution, its analysis regarding sensitivity to noise and computational complexity, your observations and conclusions, which show your understanding of the theoretical aspects of edge detection, line detection and their practical implementation.

All submitted files must include an assignment header with your name, your student number and your email address. Anonymous submissions without these details will not be assessed and will not be marked.

Do not add a separate header `*.h` file to your source code as it makes evaluation and marking of the assignment more complex.

Add the two files to a ".zip" archive for submission. On the Moodle site for the subject you will find a section with the title "Assignment Submission". Under "Assignment 2" upload your ".zip" file. You are allowed to submit up to 3 times before the deadline.

NOTES:

1. Submit your assignment before the due date. Follow the submission requirements explained in the section SUBMISSION. You are allowed to submit up to 3 times before the deadline. Only the latest submission will be tested and marked.

2. SUBMISSION BY EMAIL IS NOT ACCEPTABLE

3. ASSIGNMENT FILES WITHOUT PROPERLY FILLED HEADERS WILL NOT BE MARKED

4. Enquiries about the marks can only be made within a maximum of 1 week after the assignment results are published. After 1 week the marks cannot be changed.