Exercise #4 – Continuous Transformations for Scene Understanding

DUE: AS INDICATED on D2L (before Midnight)

Please thoroughly read through chapter 5 in Computer and Machine Vision by E.R. Davies and notes on Segmentation; please also note the scanned special handout from E.R. Davies Computer and Machine Vision Chapter 9 on methods to skeletonize an image (machine vision process) uploaded on D2L. The goal of this lab is to introduce concepts and theory related to continuous scene segmentation, basic shape recognition (with skeletal thinning), depth estimation, and behavioral analysis, which feed into recognition methods. It includes both top-down OpenCV application development and understanding as well as bottom-up algorithm implementation and understanding.

Exercise #4 Requirements:

1) [10 points] Read the paper Use of the Hough Transformation to Detect Lines and Curves in Pictures, Richard Duda & Peter Hart (also available on Blackboard) and summarize the papers key points and contributions to computer vision.

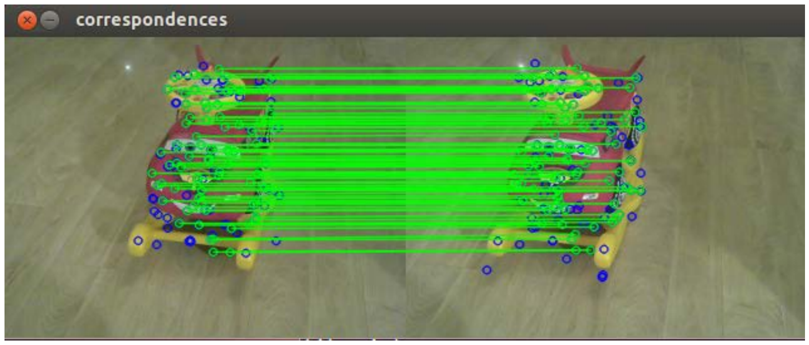
2) [10 points] Build and run the basic Hough linear example provided and paste a transformed image of your choice into your report.

3) [20 points] Using a top-down OpenCV approach, adapt the example code found in capture_transformer to use the skeletal.cpp transform on continuous frames (like captureskel.cpp) from your camera, but use the much simpler approach of simpler_capture rather than V4L2 camera capture. Gesture in front or your camera and see if you can get a reasonable continuous skeletal transform of your arm and hand. Record example frames (up to 3000 for 100 seconds of video – JPEG frames are fine) and encode your results to an MPEG video. Upload the modified code with your report.

4) [30 points] Use the methods presented in E.R. Davies Chapter 9 handout on Blackboard to write your own algorithms from the ground up to transform video frames of basic arm gestures to create a skeletal model of the arm. First, eliminate background, then convert the image to a binary bit map (as we discussed in class), then apply the skeletal thinning algorithms from E.R. Davies so that your frames track only the movements of the arm skeleton over time. Comment on whether your bottom-up algorithm implementation is better than the OpenCV top-down in terms of quality and efficiency. [Use ffmpeg to decode and save a single frame from the MPEG video]

5) [10 points] Read the paper Distinctive Image Features from Scale-Invariant Keypoints, by David Lowe (also available on Blackboard) and summarize the papers key points and contributions to computer vision.

6) [10 points] Build and run the basic OpenCV keypoint comparison detector code provided in sift detector_extractor_matcher.cpp and paste a transformed image of your choice into your report. It should provide a comparis on like this between two images:



Take two snapshots of the same scene (as was done above) from a left/right offset and run the detector comparison code and paste the result into your report.

7) [10 points] Build and run the disparity depth estimator example in example_stereo using two cameras and describe how it works based on your reading and understanding of the code and just in terms of performance and accuracy. Paste in an example left-eye, right-eye, depth map set of three images that correspond which you captured with your cameras. Are keypoints used for passive depth estimation, and if not, could they be?

Upload all video as encoded MPEG-4 at a reasonable bit-rate and quality.

Overall, provide a well-documented professional report of your findings, output, and tests so that it is easy for a colleague (or instructor) to understand what you've done. Include any C/C++ source code you write (or modify) and Makefiles needed to build your code. I will look at your report first, so it must be well written and clearly address each problem providing clear and concise responses to receive credit.

In this class, you'll be expected to consult the Linux and OpenCV manual pages and to do some reading and research on your own, so practice this in this first lab and try to answer as many of your own questions as possible, but do come to office hours and ask for help if you get stuck.

Upload all code and your report completed using MS Word or as a PDF to Blackboard and include all source code (ideally example output should be integrated into the report directly, but if not, clearly label in the report and by filename if test and example output is not pasted directly into the report). ***Your code must include a Makefile so I can build your solution on Ubuntu VB-Linux. Please zip or tar.gz your solution with your first and last name embedded in the directory name.***