Exercise #3 – Fundamental Threshold Analysis, Segmentation and Tracking

DUE: AS INDICATED on Canvas

Please thoroughly read through Week-5 to Week-6 notes and prior notes to date along with OpenCV examples of Hough shape detection – houghlines.cpp and houghcircles.cpp. Further, read the paper "The world of interactive media systems and applications" and consider the fusion of rendering with computer vision and the concept of rendering a scene that is first fully parsed by computer vision. Do you believe it would ever be possible for computer vision to fully parse and render a scene so that it is indistinguishable from reality? – a Turing test for computer vision. Please see Lab3-Examples for dark room laser spot video.

If you have troubles writing low-level C or C++ code to read in or write out PPM or PGM files, you will find the following tools helpful, however **you can also use OpenCV imread and imwrite**:

1) ***Use vi in hex display mode (****https://vim.fandom.com/wiki/Improved_hex_editing****) or bvi*** – binary vi editor (install with **sudo apt-get install bvi**) and read manual
2) ***eom*** - The Eye of MATE Image Viewer (**sudo apt-get install eom**) and read manual
3) Example V4L2 to capture 1800 frames ready for ffmpeg - simple-capture-1800/

Exercise #3 Goals and Objectives:

1) [10 points] Read the paper "The world of interactive media systems and applications" and summarize key points of the paper. At one point early in the days of ray tracing and computer vision, researchers worried about indistinguishable camera captured video from generated (rendered) video – do you think this was a valid concern?

2) [15 points] Starting with the OpenCV example for Hough Lines, adapt the code so you can compute Hough lines for a continuous camera stream (houghlines.cpp) and refer to class example for OpenCV 3.x as it may help you too (simple-hough-interactive/). Capture an image showing detection of lines for an object you hold up to the camera. What could you use this for?

3) [15 points] Starting with the OpenCV example for Hough Circles, adapt the code so you can compute Hough circles for a continuous camera stream (houghcircles.cpp) and refer to class example for OpenCV 3.x as it may help you too (simple-hough-elliptical-interactive/). Capture and image showing detection of circles for an object you hold up to the camera. What could you use this for?

4) [15 points] Process the video, Dark-Room-Laser-Spot-with-Clutter.mpeg and use color frame differencing for R,G & B to remove the bookshelf background and to preserve the moving

laser spot foreground. How effective is this for removing background clutter? Show an example of the original video display frames with imshow and another with the difference image background removal using imshow for the same frame side by side (looks like - csci612/code/diff-interactive/tick-detection.png). You may find this background removal example for OpenCV 3.x useful and interesting (csci612/code/diff-interactive/).

5) [15 points] Use the Dark Room Laser Spot video (Dark-Room-Laser-Spot.mpeg) and convert the entire video to a grayscale graymap by **using ONE band** (from RGB) **or balanced gray** (see notes for typical balance of each band) - re-encode the video. You can do this using OpenCV and/or a combination of OpenCV and ffmpeg. Note that frames can be saved as a PGM (see http://en.wikipedia.org/wiki/Netpbm_format) and then re-encoded into MPEG4 using ffmpeg. You can use OpenCV if you wish or your own code to write out the PGM frames in a format suitable for ffmpeg encoding. Note that ffmpeg needs filenames to be of the format name%04d.pgm or similar so that frames are encoded in order. Consider more cluttered and lower contrast laser tracker video for extra challenge if you wish (Dark-Room-Laser-Spot-with-Clutter.mpeg, Light-Room-Laser-Spot-with-Clutter.mpeg).

6) [30 points] ***Please implement a bottom-up approach*** with a bright spot detector/tracker you create without using OpenCV and compare to use of an OpenCV approach top-down. Use the Dark Room Laser Spot video you converted to a series of graymap frames (using ONE band from RGB) and determine a threshold function based on your analysis of the characteristics of the edges of this spot in ONE band (using GIMP or MATLAB may help you). Raster each frame to determine the X-bar, Y-bar object COM (Center of Mass) based on X row maximum extents and Y column maximum extents and your threshold detector. Use any edge detector or convolution you want to enhance the image prior to threshold detection of the X and Y edges, or just use the raw data and a well-designed and determined threshold. Mark the COM (Center of Mass) and track it in each frame with crosshairs (at saturation level of 255 - should appear as white lines). Save your annotated video showing tracking as an MPEG-4 (either using OpenCV or by saving frames and then encoding the frames using ffmpeg).

Notes – ***For bottom-up***, you may want to use csci612/code/simple-capture/ and avoid using OpenCV altogether, processing the array of pixel data (intensity values) with your threshold and COM determination algorithm of your own design. Or, you may want to use OpenCV to read in the video data into a "Mat", that you then dereference to get color channel intensity (e.g., done here in OpenCV - computer_vision_cv4_tested/simpler-capture-4/brighten.cpp with `image.at<Vec3b>(y,x)[channel]`) or even copy the Mat data for intensity into a new XY array after loading each frame. ***If you use OpenCV in your bottom-up solution, limit your use of OpenCV to the video input, imread, imwrite, imshow for debug, and/or video output and use your algorithm to find the COM.***

Upload all video as encoded MPEG-4 at a reasonable bitrate 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 have done. Include any C/C++ source code you write (or modify) and Makefiles needed to build your code and make sure your code is well commented, documented and follows coding style guidelines. I will look at your report first, so it must be professionally 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 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 Canvas 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 an embedded Linux Jetson system. Please zip or tar.gz your solution with your first and last name embedded in the directory name and/or provide a GitHub public or private repository link. Note that I may ask you to walk through and explain your code. Any code that you present as your own that is "re-used" and not cited with the original source is plagiarism. So, be sure to cite code you did not author and be sure you can explain it in good detail if you do re-use, you must provide a proper citation and prove that you understand the code you are using.***

Grading Rubric

[10 points] Read and summarize the main points of the assigned paper.

Problem	Points Possible	Score	Comments
Three main points of the paper	6		
summary overall	4		
TOTAL	10		

[15 points] Hough lines houghlines.cpp example adaptation for interactive camera streaming.

Problem	Points Possible	Score	Comments
Adaptation of code for camera use and interaction	10		
Example use	5		
TOTAL	15		

[15 points] Hough circles houghcircles.cpp example adaptation for interactive camera streaming.

Problem	Points Possible	Score	Comments
Adaptation of code for camera use and interaction	10		
Example use	5		
TOTAL	15		

[15 points] Clutter removal using differencing from Dark room laser spot with clutter.

Problem	Points Possible	Score	Comments
Adaptation of code for use with OpenCV 4.x	10		
Demonstration of side by side with and without clutter (pixels in motion only)	5		
TOTAL	15		

[15 points] Conversion of Dark room laser spot video to grayscale (ONE band from RGB only)

Problem	Points Possible	Score	Comments
Conversion from RGB to SINGLE BAND grayscale, code to do so, and PGM frames or Mats	10		
Re-encoded grayscale only video using ffmepg or OpenCV	5		
TOTAL	15		

[30 points] Center of laser spot tracking using bottom-up COM detector compared to OpenCV top-down approach.

Problem	Points Possible	Score	Comments
Threshold and enhancement for spot edge finding	5		
Method for laser spot COM determination and marking in code	10		
Final video showing spot tracker	15		
TOTAL	30		

Report file MUST be separate from the ZIP file with code and other supporting materials.

Rubric for Scoring for scale 0…10

Score	Description of reporting and code quality
0	No answer, no work done
1	Attempted and some work provided, incomplete, does not build, no Makefile
2	Attempted and partial work provided, but unclear, Makefile, but builds and runs with errors
3	Attempted and some work provided, but unclear, build warnings, runs with no apparent error, but not correct or does not terminate
4	Attempted and more work provided, but unclear, build warnings, runs with no apparent error, but not correct or does not terminate
5	Attempted and most work provided, but unclear, build warnings, runs with no apparent error, but not correct or does not terminate
6	Complete answer, but does not answer question well and code build and run has warnings and does not provide expected results
7	Complete, mostly correct, average answer to questions, with code that builds and runs with average code quality and overall answer clarity
8	Good, easy to understand and clear answer to questions, with easy-to-read code that builds and runs with no warnings (or errors), completes without error, and provides a credible result
9	Great, easy to understand and insightful answer to questions, with easy-to-read code that builds and runs cleanly, completes without error, and provides an excellent result
10	Most complete and correct - best answer and code given in the current class