Spring 2020 Code Review

CS Capstone 25 *Camera Validation System*

Review by Brian Ozarowicz

Could you clone from Git and build using the README file?

This code is reliant on specific hardware so building it was not part of the review process. However, assuming I had the proper hardware, reading through the documentation in the repo I believe I would be able to get everything running; the instructions are clear and very detailed.

No revision necessary.

Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?

The code has a clean and consistent style across all the files, but there are hardly any comments so it is hard to come in from the outside and follow what everything is doing. Variable names aren't much of a help; I can see generally *what* the different functions are doing, but to understand the *how* comments are definitely needed to walk through some of the flow.

Added extensive comments to the code base explaining the work code blocks/ function perform and why they are needed.

Is it shorter/easier/faster/cleaner/ safer to write functionally equivalent code? Do you see useful abstractions?

Without a better understanding of the implementation details that more comments would provide I don't see anything that stands out as a modification that would have better functionality than the current code.

No revision necessary.

Are there unit tests? Should there be? Are the test covering interesting cases? Are they readable?

There are not defined unit tests, but the program can be checked by showing whether or not it successfully creates and detects the indicated types of video distortion, and it is in the demo video. A test suite to clearly demonstrate specific functionality wouldn't hurt to have though.

I created unit testing suites for both the comparison and corruption unit using the C++ unit testing framework Catch2. Unit tests can be easily run in the preconfigured Raspberry Pi VM image. Details are in the "Unit Testing" section of the README .

Does the code fulfill the requirements?

Yes; the team has successfully created both a corruption unit which introduces abnormalities into a video stream and a comparison unit that detects distortion and freezing in the stream as laid out in the project requirements.

No revision necessary.

Are there other things that stand out that can be improved?

The only reason this can't be run first hand by a reviewer is the dependence on the specific hardware of the Raspberry Pis. I understand that was what you originally designed for because that was what you were developing on, but now that you have the system functioning it would be nice to make it not be dependent on that hardware, allowing input from any system, so more people could run it themselves.

I wouldn't make that a requirement of your project because you have successfully completed your established deliverables, but if it is feasible to make this adjustment in the time remaining you might consider it as a bonus product.

I think it would be outside the scope of the project, and to little benefit for the client, to produce/re-tool the code base so that it is completely agnostic to the hardware. I created a Raspbian (Debian based) VM image with our application installed and all of the dependencies built for testing the code base in an environment which closely mimics the hardware. This VM image can readily be run in VirtualBox or VMWare on any host operating system.

Revised By: Kamron Ebrahimi