Real-time Tele-Medicine in Ophthalmology

Architecture Assignment

Team 7

CS499
October 17, 2019

Author	Contribution
Sam McCauley	460
Shelby Stocker	1,013
Cole Terrell	1327
Bryce Kushner	250

Customer

Paras Vora
Dr. Eric Higgins

Developer Notebook: https://github.com/coleterrell97/SeniorDesign

High Level Design

Our main architecture we are using in this project is a server client model for the remote viewing half of this project. The server will be created using Python Flask. To capture the camera feed we will be using OpenCV. The camera feed will be passed to the server where it will be hosted for clients such as other doctors or students to access. The local half is all hard wired from the cameras to a computer and then to a screen for the VR headset. Below is a diagram which represents this. We will create a simple user interface which has control settings such as exposure, contrast, etc. that will be able to be controlled locally. The remote viewer will not have access to these settings unless the local physician grants them access.

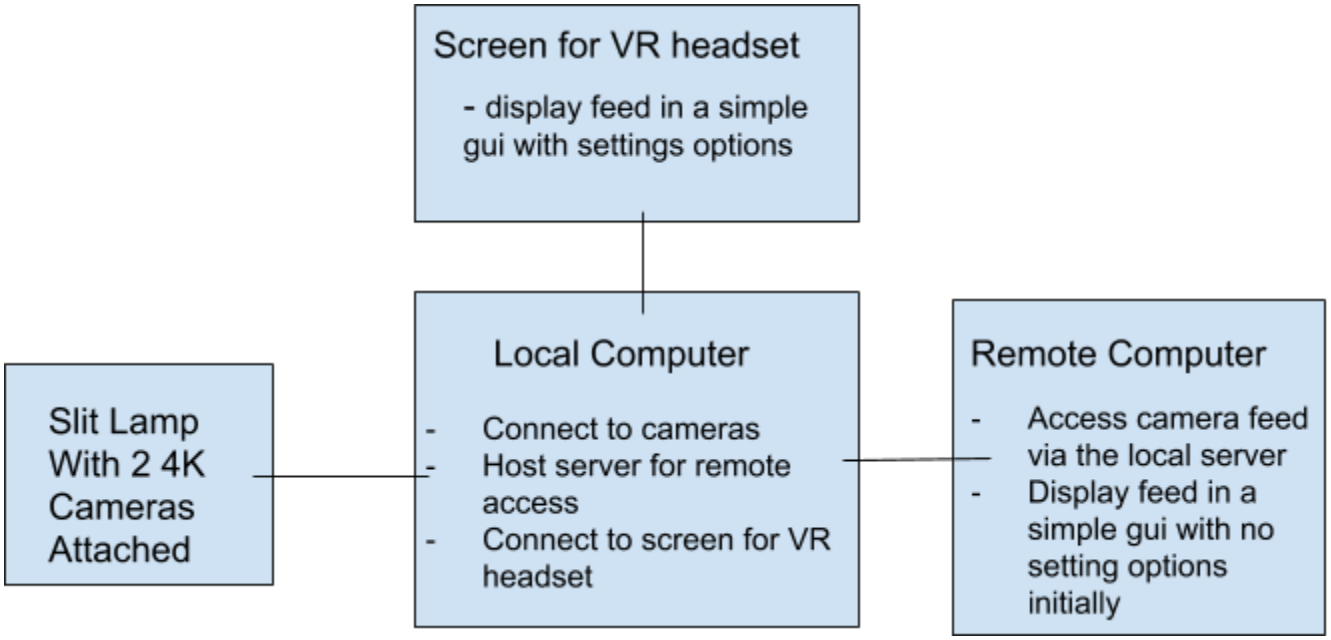
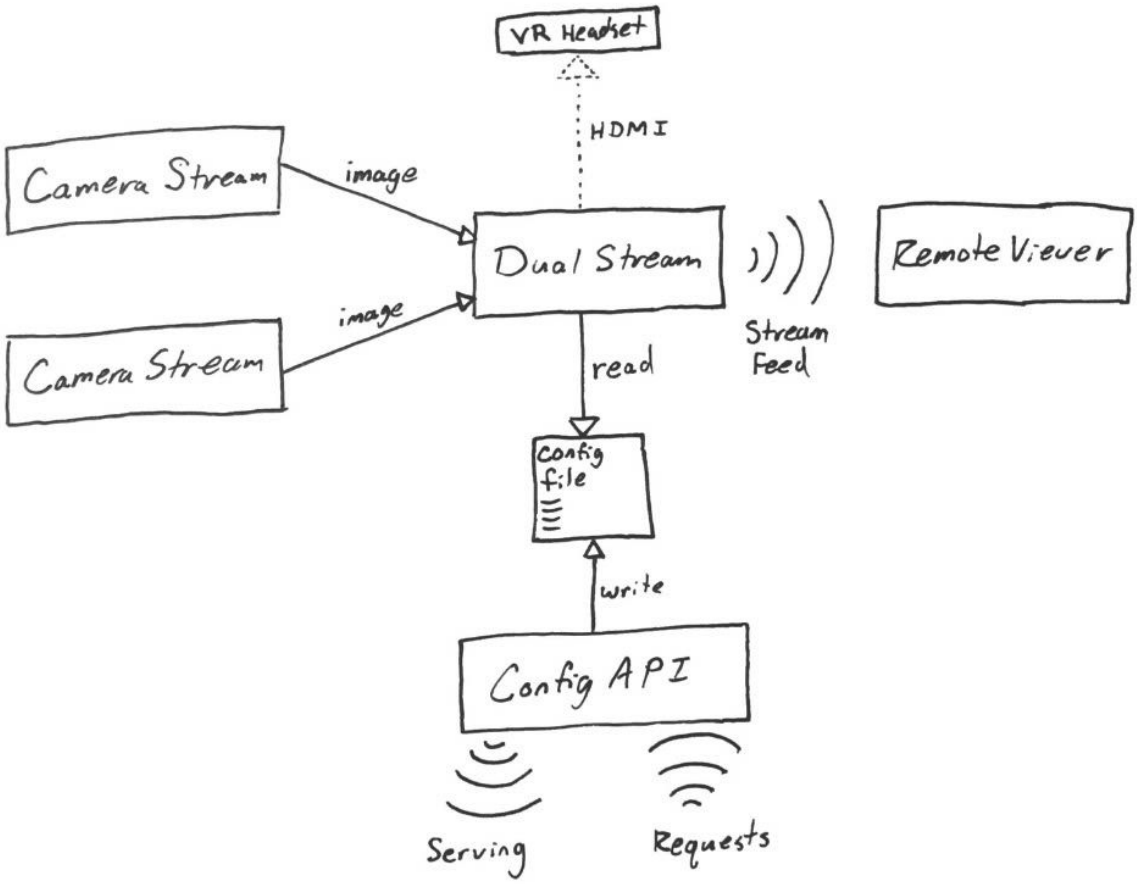


Figure 1.1: High Level Diagram of Project Architecture

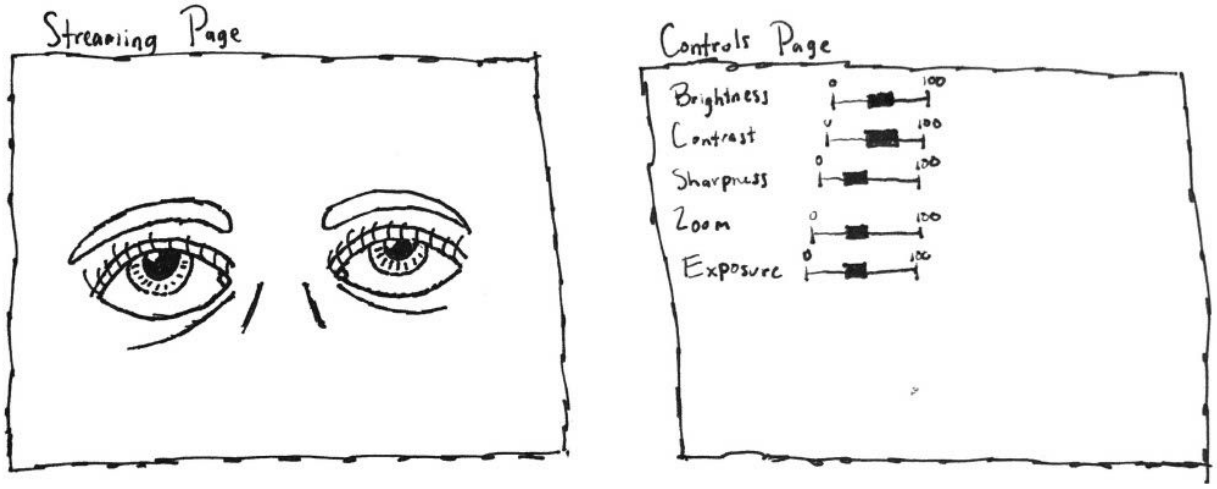
Detailed Design

Class Diagram



As you can see, there are a lot of different classes that we will need to implement for our project. Each class has a dedicated task that will aid in accomplishing our goal. The main classes that we will be using are:

CameraStream(): The CameraStream object is a threaded OpenCV class that takes a camera source and continually reads the frames in a subprocess, thus allowing other applications to simultaneously do work. Other classes will be able to efficiently read the frames that the CameraStream subprocess fetches.
DualStream(): The DualStream class reads two separate instances of the CameraStream classes. This class's main tasks are to concatenate the two images into a VR viewable stream, send the VR frames to a local VR headset over HDMI, and also to send the VR frames out over the internet to a remote location. It will also read a config file that's on the server to control the camera settings like zoom, brightness, saturation and perform the necessary image modifications.
ConfigAPI(): This class is responsible for hosting the UI webpage that will allow the operators to make changes to the camera streams. It will write the users desired configurations to a local .json file.
RemoteViewer(): The RemoteViewer class will catch the streamed frames from the internet and show them at a remote location.



User Interface Design: One of the key components of this project is creating a user interface that is simple to use. This interface will be displayed locally and remotely. The largest portion of the screen will be devoted to displaying the feed from the cameras. There will be a separate web page that will have a list of settings. These settings will include sliders for adjusting the contrast, saturation, brightness, etc. These settings will only be able to be adjusted locally. The primary physician/local viewer must grant the remote viewer access before the remote viewer can adjust anything.

UI Back-end Design: The UI described above will be enabled by a separate web server that will accept post requests from the user that will manipulate data in a configuration file. This configuration file will be read in by the image processing class (reads will occur on interrupts) and this will result in the desired changes to the qualities of the image to appear at the front-end of the UI.

Testing

The quality of the final product that we will deliver is tied strongly to the quality of the user interface that we produce. Because the user interface is of such importance and will be of relatively low complexity, our primary testing method will be manual UI tests. Additionally, we will unit test methods where computation or data manipulation occurs such as the methods controlling our image manipulation UI tools and image cropping tools. Due to the nature of the data that serves as the input to our system (image data from cameras), it is not possible to devise test cases with the intention of breaking our input capture scheme; thus, our testing will serve the purpose of debugging UI options provided to the user and verifying performance. The tests we will conduct along with their descriptions are presented below.

1.) Frame Rate Test: We will need to ensure that our product produces an image both on a local and remote machines with a frame refresh rate of roughly 30Hz. This test will consist of verifying the frame rate; this can be done through methods included in the OpenCV package that we are using to capture and stream the image.

2.) Latency Testing: Latency testing will be done to ensure that our image reaches the viewer with acceptable delay for the application (certain applications can accept much higher latency).

Latency verification can be done using the tools provided by the OpenCV package as well as qualitatively by simply observing the delay between changes in the viewing field of the cameras and changes on the produced image.

3.) Image Manipulation Tools Testing: Our UI will include various tools for manipulating the image in real-time. Sliders for adjusting the contrast, saturation, brightness, etc. of the image must be tested for basic functionality (do they affect the image quality that they should).
    a.) We will unit test the methods that enable these UI tools to ensure that the sliders scale the selected image quality uniformly and without overflow or other arithmetic abnormalities. This will allow us to prevent possible disorientation for a user viewing the image through a VR viewer.

4.) Resolution Testing: The image we produce will likely be streamed to small LCD screens or other mobile devices that will not have the capability to display true 4K images. Thus, we will need to verify the resolution that is produced on the screen both fits the screen and is of the highest resolution possible for that screen. This can, once again, be done using methods provided by the OpenCV package that we are using.

5.) Functional Test: Once our product is in a near final stage, we will also conduct a functional test with the assistance of our customers. At this time, our customer will attempt to view various structures of the human eye through our product in order to properly identify them. If our customer is able to identify the requisite structures, we will deem this test a pass, and this will confirm the overall function of our product.

Test Case Number	Description	Conditions	How To Test	Expected Result
1	Camera feed is being displayed locally	The cameras are attached to the slit lamp and the server is running	Set up system and check the screen to see if the video feed is displayed	The local screen will display the video feed
2	Camera feed is being displayed remotely	The cameras are attached to the slit lamp and the server is running	Set up system and check the screen to see if the video feed is displayed	The remote screen will display the video feed
3	The frame refresh is approximately 30 Hz locally and remotely	The camera feed is being displayed locally and remotely	Use a method in OpenCV package	The result of the method will verify that the frame refresh rate is about 30 Hz
4	The latency in which the video	The camera feed is being displayed	-Use a method in OpenCV package	The delay between the

	feed reaches the viewer is minimized	locally and remotely	- Manually observe the difference between camera view and resulting image	camera view and the resulting image should be minimized to as close to real time as possible
5	Test that the brightness slider works	The camera feed is being displayed locally and remotely and the UI is viewable	Manually change the slider to different values and check if the image is affected	The image's brightness should adjust as the slider is changed
6	Test that the contrast slider works	The camera feed is being displayed locally and remotely and the UI is viewable	Manually change the slider to different values and check if the image is affected	The image's contrast should adjust as the slider is changed
7	Test that the zoom slider works	The camera feed is being displayed locally and remotely and the UI is viewable	Manually change the slider to different values and check if the image is affected	The amount the image is zoomed in should adjust as the slider is changed
8	Test that the sharpness slider works	The camera feed is being displayed locally and remotely and the UI is viewable	Manually change the slider to different values and check if the image is affected	The image's sharpness should adjust as the slider is changed
9	Test that the exposure slider works	The camera feed is being displayed locally and remotely and the UI is viewable	Manually change the slider to different values and check if the image is affected	The image's exposure should adjust as the slider is changed
10	The primary viewer must grant a remote viewer access to view the feed	The camera feed is being displayed locally and remotely	Have the remote viewer try to access the video feed without requesting permission and with requesting permission	The remote viewer should not be able to see the video feed until they are granted access from the primary viewer
11	Verify the resolution	The camera feed is being displayed	Use a method from an openCV	The resolution should appear to

	produced fits the screen and is of the highest resolution possible	locally and remotely	package and verify the image fits the screen and the resolution looks good	be good quality and the image should fit the screen

Metrics

Size Estimate

The size estimate for our project will be given using story points, as the number of functions/lines of code required is not well understood at this early stage of our design process. The scale will range from 1-5, with 1 signifying a task that is simple and will require little effort and 5 signifying a task that is complex and will require a great effort.

A.) The most critical function of our product is the transfer of a high-resolution video stream from cameras mounted to an ophthalmologist's slit lamp to a portable viewing device as well as a virtual reality viewing device.

Story Points: 3, while this is the most critical function of our product, we already have some code to work from, and we are not as concerned with potential latency issues for the local stream as we are with the remote video stream. Overall, this task is moderately complex and will require a moderate effort to complete.

B.) The next function that our product shall have is the ability to stream the high-resolution video over a network connection to a remote viewer (with VR capabilities also). This stream must have low-to-moderate latency, as the stream may be used for applications such as guiding a resident through a procedure in real-time or acquiring a second opinion from a remote-located physician.

Story Points: 5, this function is not as critical as the local stream, but our customers have demonstrated the immense potential that such a function could have to revolutionize remote medicine; thus, it is still of great importance to our team. Minimizing the latency over the network connection will likely be a challenge in addition to the fact that our customers require ultra-high resolution video. Overall, this task will likely take the majority of our time and effort to complete, and it will almost certainly be the most challenging part of our project.

C.) The last function of our product will be to record and store the video stream that we capture for later use. This function may present storage challenges, as the 4k resolution video will likely be very large. We may need to learn about video compression or invest in large amounts of external storage.

Story Points: 2, this function will not be as challenging to implement, as saving the video stream should be quite simple to implement on any of the machines that are receiving the stream. The tricky part will be the efficient storage of the video so that it is portable and scalable, but there are simple solutions in existence already.

Lines of Code

We have only written a small amount of code thus far. Initial, we were sent some starter code by Dr. Higgins and Paras Vora. The starter code was a bare bones `CameraStream` class as well as a simple Flask server--both written in Python. The Camera class uses OpenCV to open a camera source and pulls the image. The Camera class is also threaded so that the two separate feeds can run simultaneously and efficiently. This class has six functions and 38 lines of code. We still need to add configuration functions to this class such as exposure changes, resolution changes, and image cropping. The Flask server, `app.py`, pulls the image(s) and sends them to a simple hosted html file. This file has four functions and 47 lines of code. To this, we have added simple prototyping functionality that accepts user source and allows one *or* two camera feeds for easy debugging. This file will handle the remote stream as well as the local stream. We will also make a simple server that accepts configuration changes as POST requests to update the feeds.

Complexity

These two files are not extremely complex compared to the work we still need to do. They are intended to create a flask server and connect to the cameras to obtain the video feed. Beyond that, we still need to send the information to the hardwired local display. We also have much work to do in terms of latency optimization.

Complexity of Overall System

Thus far, the only complexity currently in the system is of the `CameraStream` class as well as the Flask server.

Product Size

Covered in section 1 of this document (size estimate).

Product Effort

So far, we have each spent approximately ten hours on this project. We have met with our customer multiple times now and developed a strategy for maintaining open communication with him.

Defects

We have not had any defects yet.

Review

After reviewing this assignment and taking into account the feedback we got regarding our previous submission, we realized there were quite a few things we needed to rework. First of all, we did not have a section for the user interface design. Secondly, the detailed design section did not have a diagram and did not provide a breakdown of each method. Thirdly, we had a testing section but it did not explicitly describe each test case we plan to perform, it was more of a general overview. Lastly, we did not have a review section. We took all of these issues into account and feel as though we resolved them in this copy of the architecture assignment. We did however take a step back and analyze our current high level design. We acknowledged the fact that achieving a high quality stream, with minimal latency, and streaming it, is going to be feasible, but very difficult. This isn't necessarily a defect in our process we have established now because it is certainly feasible to get the desired results, but it could be difficult in the remaining time. This led us to do some extra research on potentially transitioning our process to run a stream through Twitch. They have spent millions of dollars in developing an effective high resolution streaming service and so we are exploring methods to potentially utilize this. We have not yet officially made the switch, but is just something we are exploring as a backup plan in the event that we think we need to pivot from our original design.