Real-time, Non-contact Heart Rate Monitor

Advisor: Dr. Lynne Slivovsky

Project Members: Dan Blike

dblike@calpoly.edu

I. Introduction

A. Project Overview

This project aims to create a real-time, non-contact heart rate monitoring program that utilizes a simple camera to capture facial information. Heart rate is one of the most commonly measured physiological makers, and provides the user with important feedback on their current state of health. In the past, it has been necessary to have direct contact to the skin in order to measure one's heart beat. With the advent of powerful computer vision processing libraries, such as OpenCV, and advanced signal filtering techniques, it is now possible to extract accurate heart rate measurements through facial recognition and independent component analysis (ICA). The success of this project would provide users access to accurate heart rate monitoring information--if they are unable to obtain their pulse by themselves--and would give healthcare officials the ability to measure heart rate without needing direct access to a patient's skin surface.

B. Stakeholders

In broad terms, this project serves the healthcare community, but it also assists in giving direct access to accurate heart rate information to anyone with a camera; in an age where "64% of Americans now own a smartphone of some kind"[1], this technology has a relevant impact on the consumer masses; this is even more evident with the growth and popularity of fitness related technology. The ability to obtain physiological measures without the need for such expensive hardware would pave the way for a more ubiquitous approach to healthy living. Further, heart rate can be correlated with other risk factors; one application of this project is in cars or other heavy machinery where heart rate can be used as one marker for the wakefulness of the operator.

C. Goals and Objectives

The primary goal of this project is to deliver a functional proof-of-concept by the end of the second quarter of Senior Project. The minimum value criteria includes a number of constraints.

1. The application can isolate a human face from a video feed
2. The application can extract an estimated heart rate
3. The application can run in real-time (heart rate updates occur at the frame rate of the camera)

At minimum, this project aims to verify the applicability of the ICA algorithm by the end of the first quarter. Beyond that, this project aims to produce some level of functionality by the end of the first quarter. As a stretch goal, OpenCV also has a machine learning component included in the library which can be used to continuously improve the facial recognition algorithm.

D. Outcomes and Deliverables

The main deliverable for this project will be a proof-of-concept application that can perform non-contact, real-time heart rate monitoring. The intended outcome of this project is to display the power of computer vision and image analysis techniques as they relate to the healthcare field. Through the application, we aim to make obtaining an accurate heart rate easy and accessible for a large majority of people. In addition, allowing for real-time heart rate monitoring, as opposed to analysis of archived footage, will hopefully encourage more consumer based health applications that leverage the power of general purpose hardware readily available on most persons.

II. Background

Current, state-of-the art technology used for measuring heart rate are electrocardiograms (ECG). ECGs involved placing a number of electrodes on the body and measuring the change in potential at different angles from pad to pad. This is the most effective way to analyze and record a person's heart beat, as each moment of the cardiac cycle can be viewed in isolation. Typically, ECG machines are only used by healthcare professionals to diagnose cardiovascular disease; machines like these can cost over $3000.

Alternative, more consumer friendly, methods include chest straps and infrared optic techniques[2]. In both cases, specialized hardware is used to measure pulses in the bloodstream. For the chest straps, small microprocessors are paired with electrically conductive fabric to measure subtle electric impulses in the skin during a heartbeat. Similarly, infrared devices emit light through the surface of the skin so that it may be absorbed by the bloodstream; the degree to which this infrared light is absorbed determines when a pulse is counted, or not.

In "Non-contact, automated cardiac pulse measurements using video imaging and blind source separation", a research paper by Poh et. al at the Massachusetts Institute of Technology, they examined photoplethysmography (PPG) using ambient light from standard video recording devices[3]. Their paper forms the basis of this project, as they were able to accurately determine a user's heart rate based on analysis of archived footage. While the researchers at MIT go into depth on their use of signal filtration and detrending methods, an additional goal of this project is to show that the techniques they produced in MATLAB can be repeated and sped up using Python.

Lastly, the scikitlearn website offers a wealth of tutorials on how to incorporate numpy and scipy operations into a project. The implementation of Independent Component Analysis that this project will use is called FastICA, and is found in the scikitlearn package[4].

III. Engineering Specifications

This project will produce a proof-of-concept application that will estimate the heart rate of the user in real-time. The following is our engineering specification:

1. Will be a Python application
2. Runs on Python(x,y) 2.7, Scikitlearn 0.16, and OpenCV 2.4
3. Be able to use a computer’s attached camera to capture video footage
4. Will process video frames with a target resolution of 640 by 480
5. Will estimate heart rate with at least 90% accuracy
6. Will have a processing time of less than 1 video frame (real-time)

The first requirement was chosen due to the Python’s substantial open source modules. The most important modules were OpenCV, Scikitlearn, Numpy, and Scipy. OpenCV was necessary for implementing the facial recognition algorithm; Scikitlearn contained the ICA algorithm and requires both Numpy and Scipy in order to work properly; Numpy and Scipy are both statistical modules that helped in performing the signal analysis and filtration. The second requirement specifying the minimum versions for the different modules was extracted from the

Python(x,y) package. This package contained Python, Scikitlearn, and OpenCV as an optional add-on. The specified versions are the ones contained within Python(x,y) 2.7.

Specification three is necessary for obtaining the required video footage for analysis. A minimum target resolution is included in the fourth requirement in order to normalize a specific resolution to accurately estimate heart rate. This requirement is not a true minimum because we cannot predict the camera hardware and we will not attempt to restrict users based on their hardware. Nevertheless, we choose to support a minimum target resolution below which we cannot guarantee optimal results to meet customer requirements. Specification five was chosen to represent the minimum level of accuracy required by a heart rate monitor to render its results useable. Ideally, any well-functioning heart rate monitor should be able to perform at an even higher accuracy level.

Specification six was determined based on the requirement to have the application operate in real-time. Rather than place a fixed limit on the running time, the only true requirement of a real-time system--in this use case--is that the results are accurate to the lowest quantum of measurement. For a system that relies on a video camera to capture its data, a quantum is a frame: 33.3ms for a 30fps camera, and 66.6ms for a 15fps camera.

Key: *L - Low, M - Medium, H - High, I - Inspection, T - Test*

Spec #	**Description**	**Target**	**Tolerance**	**Risk**	**Compliance**
2	Library Versions	Python(x,y) - 2.7 Scikitlearn - 0.16 OpenCV - 2.4	min	L	I

4	Target Resolution	640 by 480	min	L	I
5	Min Accuracy	90%	±10%	H	T
6	Max Algorithm Time	1 frame	±20%	M	T

Table 1: Formal Engineering Requirement Table

A. Personas

Our first persona is Dr. Gerald Moore. He is a 35 year old neonatal practitioner at Mission Viejo Hospital. Frequently, Dr. Moore wishes to examine the heart rate and other physiological markers of the newborns without disturbing their sleep. While there is preexisting hardware for completing this task, the hardware will only work for one baby at a time. Further, the cost of this additional hardware is multiplied by the number of babies present in the ward at a given time.

If this application was developed and launched in an overhead camera system, the facial recognition system could be tuned to that of a baby's face and the heart rates of all the babies could be tracked in real-time. Not only would this system save a significant amount of hardware, but there would also be reduced human intervention required on behalf of nurses and doctors examining local heart rate monitors. Rather, they would instead only have to examine one monitor, and that monitor could even be programmed to alert medical staff when one of the baby's heart rates left the normal expected range.

Our second persona is Pubert Thomas. He is a 26 year old software developer working for Tesla. Pubert doesn't get outside enough and is worried that his physical health is degrading.

In order to improve his physical fitness, he begins to workout on a regular basis. Rather than buy a trendy smart watch that can provide a myriad of physical biometrics, Pubert instead decides to be cheap. He uses his phone to track his mileage and takes his own pulse. Having just joined the fitness scene, however, Pubert struggles to reliably take his pulse and it fluctuates wildly, even within minutes of readings.

Using our application, Pubert is able to download a version that works with his phone's camera. Now when he's done running, he can accurately take down his heart rate measurements to see his cardiovascular fitness improve over time. Exploring the possible uses of the technology, Pubert pitches an idea to his boss, Elon Musk. Soon, this application is present in all Teslas, constantly monitoring the heart rates of its drivers. When their heart rate drops out of the expected zone, the driver is alerted and encouraged to pull over. The new software is heralded as yet another breakthrough in smart car technology.

B. Use Case

The following passage represents the standard use case for this application. The application is intended to be used by anyone who wishes to obtain theirs or someone else's heart rate via video camera. In practice, this group includes healthcare practitioners and general population technology owners. All of these users have similar needs with respect to this application. As professionals, they need accuracy and reliability. As ordinary users, they would prefer a fast execution time, good responsiveness, and a small application footprint. The following use case illustrates who we believe our clients to be:

Use Case: Identifying one’s heart rate via video camera

Actor: End-User

Goal: Accurately determine the user’s heart rate

Description: User starts application. The application verifies that a camera is present and available. If not, the user is notified via an error message. To give the camera a chance to obtain focus, there is a pause of approximately 100 frames before the User Interface appears. Once this startup period has elapsed, the application then creates an appropriately sized frame to contain the video feed. The user can zoom in and out of the frame, change the focus, and examine individual pixel values. Once analysis has begun, a green rectangle will be drawn around the user’s face. A smaller, red rectangle will be drawn within the green rectangle; this is the region that will be analyzed. If there is no rectangles being drawn to the frame, then the application is not identifying a face, and the frame will not be stored for analysis. Once the minimum frames necessary to obtain an accurate heart rate is captured, an estimated heart rate will then be displayed to the user on the screen.

This scenario is a use case because the end-user is interacting with the system to take a picture of a petri dish and have the application analyze it. Thus, it is an interaction of the user and the system to achieve a goal.

IV. Final Design Concept

A. Algorithm Flow

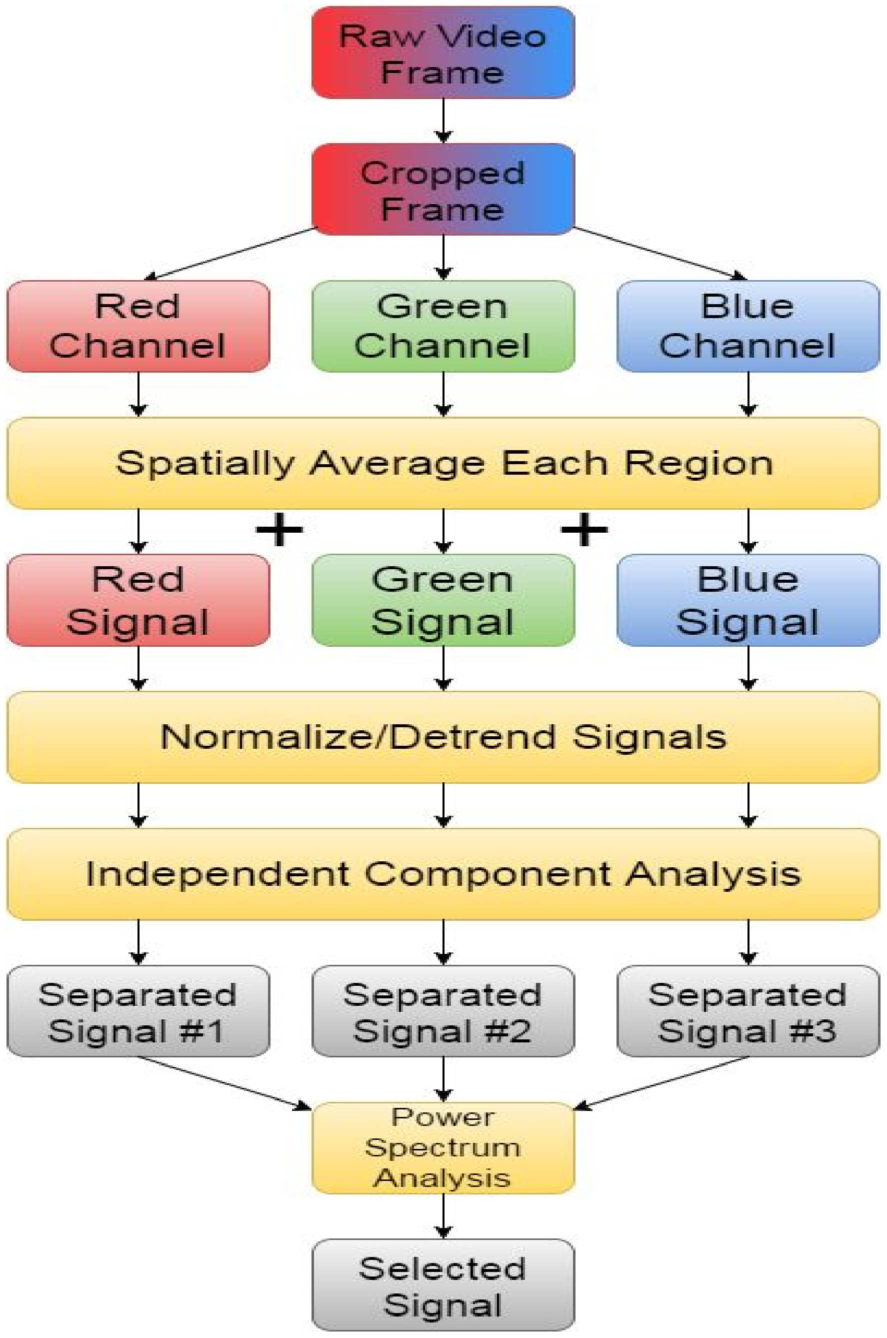


Figure 1: *Algorithm diagram for obtaining separated source signal*

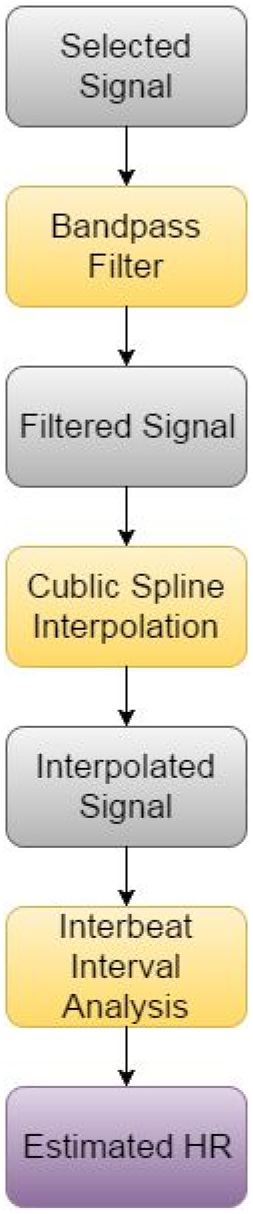


Figure 2: *Algorithm diagram for obtaining estimated heart rate from separated source signal*

<u>To obtain the separated source signal from the raw signal traces</u>:

1. We begin with an uncropped frame from the video camera
2. This frame is then cropped to the size of the detected face
3. The frame is separated by channel, and each channel is spatially averaged to obtain a single channel value

4. This value is added as a time step in each channel’s ongoing signal
5. The complete signal for each channel is detrended and normalized
6. Independent component analysis can now be performed on the three signals
7. The separated signal with the highest power spectral density is the candidate signal for heart rate analysis

<u>To obtain an estimated heart rate from the separated source signal</u>:

1. The signal is bandpass filtered from 0.7-4Hz
2. The resulting signal is then interpolated with a cubic spline
3. Resulting peaks form the Interbeat Interval, and this forms the basis of the heart rate calculation
4. From the period of the Interbeat Interval, an estimated heart rate is produced

B. User Interface

The application’s user interface (UI) was designed with simplicity in mind. The goal of the design was to provide feedback to the user that the facial recognition system was functioning properly. For a quick summary of the UI, see Figure 3. As noted in the use case, the user can zoom in and out of the frame, change the area of focus, and examine individual pixel values. The outer green rectangle is the face as determined by the OpenCV facial recognition algorithm. As can be seen in Figure 3, however, this region often incorporates non-facial features as well, such as hair or clothing. Therefore, an inner red rectangle making up 60% of the size of the original rectangle is used as the region of interest, because it’s more likely to be composed of solely skin regions than the outer green rectangle. Lastly, the overall size of the frame is fixed to the size of the capturing camera, so in this instance the frame size is 640 by 480.

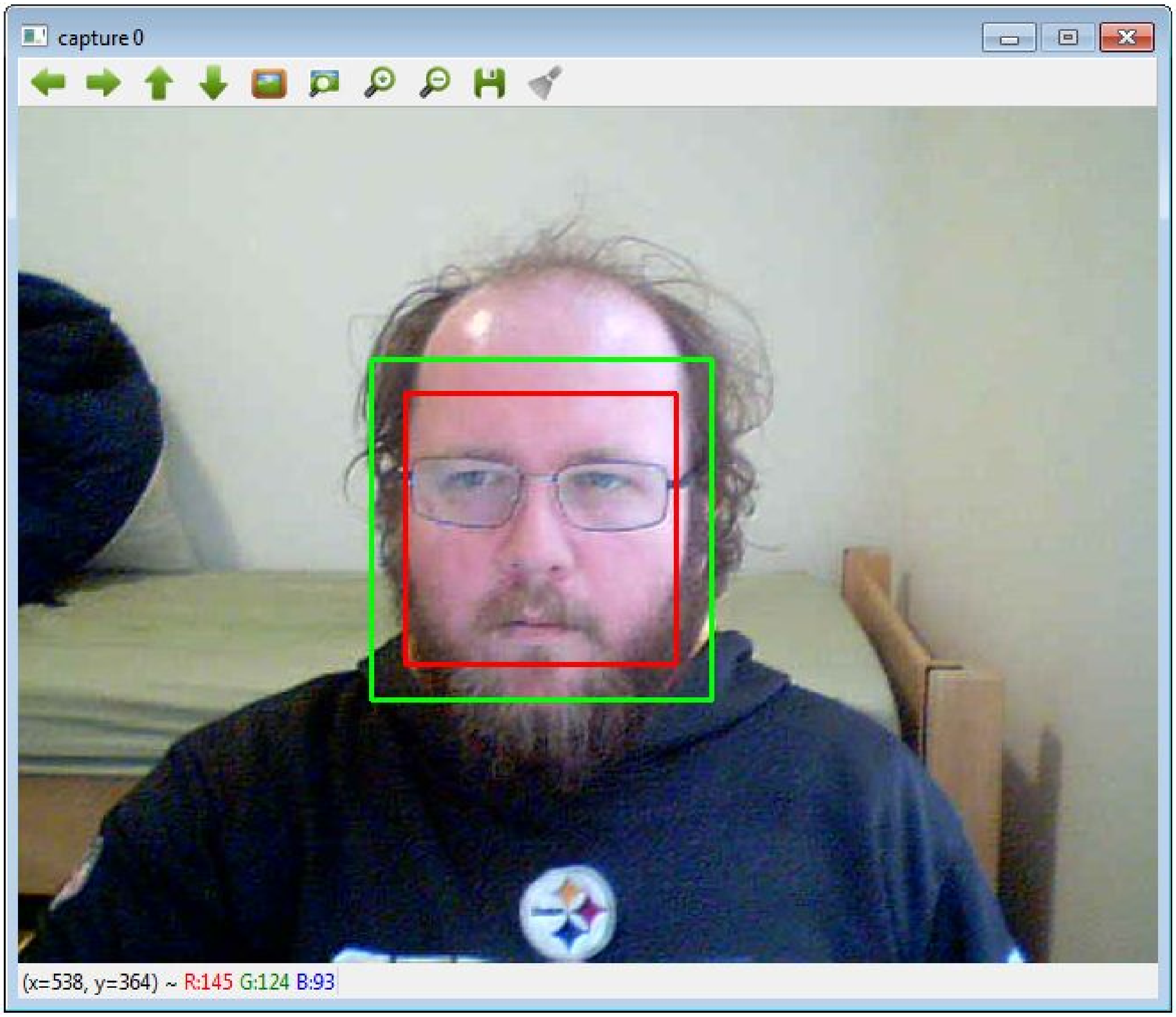


Figure 3: *User interface with red/green rectangles for region of interest*

C. Other Considerations

In order to test the accuracy of the application, it was necessary to compare against a reference heart rate monitor. Ultimately, we decided on a Garmin Premium Heart Rate Monitor.

This particular model is a chest strap; funds permitting, future teams may wish to explore the fingertip based heart rate monitors for their ease of use and higher recording resolution. This device communicates via an ANT protocol receiver. Many wearable fitness applications make use of the ANT protocol, so it's a fairly ubiquitous choice. The ideal platform for testing would be to receive the Garmin data while simultaneously comparing it to the application's estimated heart rate. However, the only readily available software package for interfacing with raw ANT data was Golden Cheetah, an all-purpose program designed to communicate with various fitness monitors. While it allowed for the real-time observation of the Garmin data, there was no way to interface with it in real-time. Therefore, all testing would have to be done on archive data, making the process more laborious than ideal.

V. System Integration & Testing

Failure Mode Effects and Analysis Overview

When designing a system, it is important to consider how the system could fail and what the results of the failure could be. Most of the failures we account for have to do with the camera and inconsistent frame quality. One of the things that can go wrong with the application is that the computer does not have an attached camera. In this case, the application should throw an error and say that there is no available camera to use; at the moment, this application will follow that behavior. Along with camera failure, there is also the possibility that the camera is already in use by another application. In this case, the application will throw an error message saying that the camera is already in use. At this point, the user will need to close the other application in order to proceed. The final issue we considered is when the camera captures a low quality frame. Unfortunately, there is nothing fool-proof that we can do about this particular issue. Something future work groups could do is look into automating this process with OpenCV; so that when a low quality frame is captured, the application can notify the user that the quality of the frame will affect results and can prompt the user to, likely, refrain from moving as much. To see our formal FMEA, see Appendix B.

Design Verification Plan and Report

To see our test plans and test report, see Appendix C.

Overall System Analysis

Any hardware issues that may cause a problem are discussed on the table in Appendix B. At times the application will crash due to the camera being “in use” despite a previous run of the application terminating. At this point in time, the reason for the unfreed camera resource is unknown, but unplugging and plugging the camera back in will restore functionality. Further, the camera used in this project was the Microsoft Lifecam VX-3000, a very old camera dating back to 2006. It was long assumed that camera captured 640 by 480 resolution frames at 15 frames per second. However, the technical specifications never listed the frame rate, and further testing points to the possibility of running at 10 frames per second instead. While this does provide more time for the algorithm to complete, allowing for a larger frame window, this does lead to possibility of incurring too much error between frame captures. While the average human heart rate is unlikely to beat fast enough that even 10 fps would be too little, it’s still possible for 10 fps to be too slow to capture a periodic trend in the frame window.

The facial recognition worked well in this project, as the default cascade classifier for identifying human faces provided by OpenCV worked well enough without further training. Future teams looking to obtain more accurate frame data could look to train individualized classifiers for each user and incorporate other feature classifiers like that for hair, mouths, eyes, and other non-whole skin features.

The core heart rate algorithm was based on the MIT paper that explored the same topic on archive footage. In that paper, the algorithm is loosely explained as a series of steps, with the majority of the code being represented in MATLAB. That being said, it was difficult to make the conversion to Python, and many of the techniques described in the paper did not yield comparable results, despite efforts to replicate the conditions. For example, a window of 300 frames was collected, detrended, normalized, and ran with independent component analysis. However, none of the three extracted signals exhibited any significantly higher power spectral density upon analysis, leading to the conclusion that ICA failed to isolate a dominant signal. This means that either the detrending algorithm was improperly ported to Python (the normalization is trivially implemented) or there was an issue with the camera setup for gathering the frames. In either case, it was impossible to obtain any usable data for making accurate heart predictions.

Therefore, one of the core takeaways from this project was the application of the remainder of the algorithm to observe whether or not it was practical to run it in real-time. To test this, the running time for the various algorithm steps was recorded at different frame windows to determine the maximum window length that could be run in one frame. Determining a maximum window length is important, because the longer the window, the more likely to ascertain the periodicity of the heart rate. This project made use of an i5-2500K Intel processor; results of the test can be seen in Figure 5.

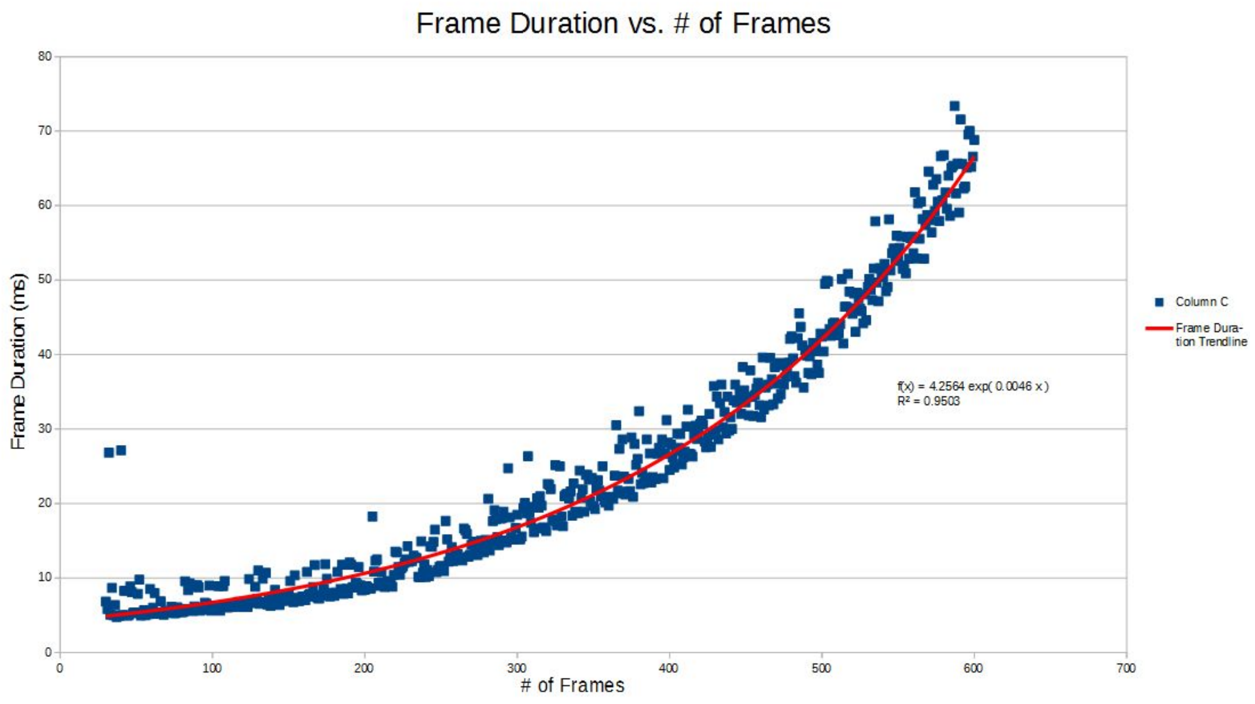


Figure 4: *A graph comparing frame window size to frame algorithm duration*

As evidenced in the graph, the relationship between window size and frame duration is exponential. For this project, a maximum window size can be estimated at over 600 frames, due to the 10 fps limit placed by the camera used. For a 15 fps camera, a more conservative estimate is <600 frames, and <300 frames for a 30 fps camera. The hardware used for this project is still relatively powerful compared to many machines, so these figures may be larger than is practical for many systems. The MIT paper ran 30s traces @ 15fps, yielding a frame window of 450 frames. Based on the data gathered here, it might have been possible for their algorithm to run in real-time, but it would have to have been running on relatively powerful hardware. Future work on this project should attempt to establish a definitive relationship between window size and accuracy, so that future attempts at a real-time application can be cognizant of the minimum necessary window length for acceptable performance.

VI. Appendix

A. References

[1] Aaron Smith, 'U.S. Smartphone Use in 2015', [Online]. Available: http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/. [Accessed: 27 - May - 2016].

[2] M.M.A. Hashem, et. al, 'Design and Development of a Heart Rate Measuring Device using Fingertip', 2015. [Online]. Available: http://arxiv.org/ftp/arxiv/papers/1304/1304.2475.pdf. [Accessed: 27 - May - 2016].

[3] M. Z. Poh, D. J. McDuff and R. W. Picard, "Advancements in Noncontact, Multiparameter Physiological Measurements Using a Webcam," in *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 1, pp. 7-11, Jan. 2011.

[4] Scikit-learn.org, 'Blind source separation using FastICA', 2015. [Online]. Available: http://scikit-learn.org/stable/auto_examples/decomposition/plot_ica_blind_source_separation.html#example-decomposition-plot-ica-blind-source-separation-py. [Accessed: 27 - May - 2016].

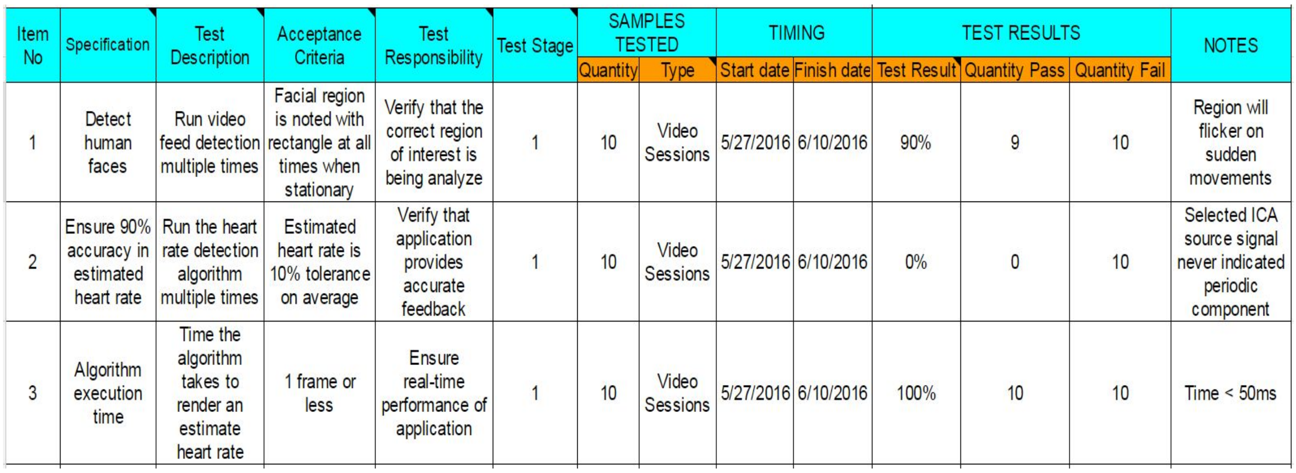
[5] Rouse, Margaret. "Use Case Definition." Search Software Quality. Tech Target. Web. <http://searchsoftwarequality.techtarget.com/definition/use-case>.

[6] "Use Cases." Usability.gov. U.S. Department of Health & Human Services. Web. <http://www.usability.gov/how-to-and-tools/methods/use-cases.html>.

B. Failure Mode Effects and Analysis

Process Function	Potential Failure Mode	Potential Effect(s) of Failure	Sev	Class	Potential Cause(s)/ Mechanism (s) of Failure	Occur	Current Process Controls	Detec	RPN	Recommended Action(s)
Video Camera	Bad Frame	The algorithm won't be able to include the frame	4		Too much movement	8	Discard Frame	2	64	
	Camera Fails	No frames will be captured	7		Camera lens is broken	4		1	28	
	Camera being used by another application	No frames will be captured	6			4		3	72	
	Camera is too low quality	The algorithm won't be able to extract enough information	7			3		5	105	Determine a minimum resolution of effectiveness
Computer Processor	Application crashes	Application may need to be restarted	5		Inadequate resources, I/O error	2		7	70	
Determining Heart Rate	Heart Rate too high	Inaccurate heart rate displayed	6		Incorrect algorithm or bad frame	8	Comparison to reference monitor	2	96	
	Heart Rate too low	Inaccurate heart rate displayed	6		Incorrect algorithm or bad frame	8	Comparison to reference monitor	2	96	

C. Test Plans



Item No	Specification	Test Description	Acceptance Criteria	Test Responsibility	Test Stage	SAMPLES TESTED		TIMING		TEST RESULTS			NOTES
						Quantity	Type	Start date	Finish date	Test Result	Quantity Pass	Quantity Fail	
1	Detect human faces	Run video feed detection multiple times	Facial region is noted with rectangle at all times when stationary	Verify that the correct region of interest is being analyze	1	10	Video Sessions	5/27/2016	6/10/2016	90%	9	10	Region will flicker on sudden movements
2	Ensure 90% accuracy in estimated heart rate	Run the heart rate detection algorithm multiple times	Estimated heart rate is 10% tolerance on average	Verify that application provides accurate feedback	1	10	Video Sessions	5/27/2016	6/10/2016	0%	0	10	Selected ICA source signal never indicated periodic component
3	Algorithm execution time	Time the algorithm takes to render an estimate heart rate	1 frame or less	Ensure real-time performance of application	1	10	Video Sessions	5/27/2016	6/10/2016	100%	10	10	Time < 50ms

Test Procedure 1:
Determine: Whether the application correctly detects a face
Materials: One video camera; a computer; the application
Safety: Wear your fun hat! :D
Procedure:

1. Assure application is properly installed
2. Open application
3. Ensure there is a present and available camera
4. Wait predetermined number of frames for UI to display
5. Verify that UI is the correct size for the frame
6. Identify two rectangles: green and red
    a. Ensure the green rectangle occupies the majority of the user’s facial region
    b. Ensure the red rectangle occupies a smaller region, within the green rectangle, composed almost entirely of skin

Test Procedure 2:
Determine: Whether the application provides an estimated heart rate
Materials: One video camera; a computer; the application
Safety: Wear your fun hat! :D
Procedure:

1. Assure application is properly installed
2. Open application
3. Ensure there is a present and available camera
4. Wait predetermined number of frames for UI to display
5. Verify that UI is the correct size for the frame

6. Identify two rectangles: green and red
    a. Ensure the green rectangle occupies the majority of the user’s facial region
    b. Ensure the red rectangle occupies a smaller region, within the green rectangle, composed almost entirely of skin
7. Wait predetermined number of frames for analysis to begin
8. Identify a number representing estimated heart rate appear on the UI