4-28-2024

Measurement of Photoplethysmogram in Visible Region of the EM Spectrum

Wesley Nathanael Dyar
University of Alabama in Huntsville

Measurement of Photoplethysmogram in Visible Region of the EM Spectrum

by

Wesley Nathanael Dyar

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

April 28, 2024

Honors Capstone Project Director: Dr. Vinh Nguyen Du Le

Wesley Dyar 04/26/2024

Student Date

Vinh Nguyen Du Le 04/26/2024

Project Director Date

Department Chair Date

Sean Lane Digitally signed by Sean M Lane
Date: 2024.04.30 16:31:35 -05'00'

Honors College Dean Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College

Frank Franz Hall

+1 (256) 824-6450 (voice)

+1 (256) 824-7339 (fax)

honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted with the final manuscript.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Wesley Dyar
Student Name (printed)

Wesley Dyar
Student Signature

04/26/2024
Date

Table of Contents

Abstract

Photoplethysmography (PPG) has been widely used to track heart rate. In commercial wearables such as Apple watch, PPG is often measured at green and infrared wavelengths of the electromagnetic (EM) spectrum. Recent empirical studies have reported that PPG strongly depends on skin absorption which also depends on wavelengths. In this project, an experimental setup will be developed alongside code to test the measurement of PPG with different wavelengths. This setup is designed to be used on both tissue-simulating phantoms and the wrist using a customized fiber optics probe. Ultimately, the setup can be used to compare experimental data with previous empirical results and will help to identify the optimal wavelengths in quantifying heart rate.

Introduction

A photoplethysmogram (PPG) is a way of showing how the volume of something in the human body changes over time. Photoplethysmography has become increasingly popular in recent years as a cheap and non-invasive way of measuring how the blood volume in the microvascular bed changes over time. By making these measurements, we can track short term changes that can be used to measure heart rate as well as long term changes that can track respiration and a variety of other vital functions [1]. This technology has been added to wearables so that people can now have near constant access to important health data in a way that was previously not possible. With this kind of data available, people are not only able to better keep track of their overall fitness but they are also allowing for early detections of cardiovascular disease, heart rate abnormalities, and a variety of other complications [2].

Because the data acquired through photoplethysmography has so much potential to improve medical care for so many people, it is important that we find ways to make sure that this technology can work for anyone who wants to use it. Currently there are studies that show that higher body mass index (BMI), increased skin melanin concentrations, and certain activities can contribute error to PPG measurements; although, this paper will focus on ways to study the first two [3]. While melanin is known to absorb light, especially in the UV region of light, it is debated whether higher skin melanin concentrations really contribute a discernible amount of error. Simulations have shown that up to a 15% decrease in PPG signal can be expected in skin melanin concentrations of 42% volume fraction relative to the same measurement taken with 3%. Obesity (BMI over 30) is known to significantly affect the thickness of many skin layers, decrease the water loss through the epidermis, and reduced blood flow in certain areas like the forearm where PPG measurements are often made. These effects cause higher BMI's to play a significant role in PPG signals with simulations showing as much as a 60% decrease with a BMI of 45 relative to a BMI of 20. The combination of the melanin and BMI effects do not seem to be as significant with a maximum of a 61.2% decrease in PPG signal from someone with 3% melanin concentration and a BMI of 20 to a 42% melanin concentration and a BMI of 45 [2].

PPG is typically measured using a green light source, an infrared emitter, or both. These light sources are used because they allow for the absorption to be measured at different skin depths and because they work well with many people's skin characteristics [4]. By developing a setup that will allow multiple wavelengths to be tested, we can hopefully find a solution that will provide accurate PPG measurements for more people.

Methodology

Setup Design

Before the code could be written to process any data, the experimental setup had to be developed. Originally the setup involved a single laser which allowed me to learn how to collimate and focus a beam so that it could be passed through a fiber optic cable. All that had to be done for this simplified setup was align the beam through two plano-convex lenses and into the light source end of a bifurcated fiber reflection probe. We then used the remaining two ends to connect to the spectrometer and to measure the skin reflectance. This setup increased in complexity as we began using two lasers simultaneously. The two-laser setup is shown in figure 1. In order to introduce a second laser so that we could use two wavelengths at once, we added a dichroic mirror to align the two beams along with two neutral density filters to prevent saturation.

Most of the problems that arose during this phase of the project simply came from a lack of experience putting together optical components. Collimating the beams, aligning the two lasers, and focusing the light onto the fiber cable all required some time to figure out. Once I managed to figure those out, we also ran into some issues with the overall stability of the setup. It was incredibly sensitive to touch which was an issue at first given that we were sharing a small lab with other people. These issues were quickly resolved and I learned how to make the setup less sensitive by more securely fastening many of the optics.

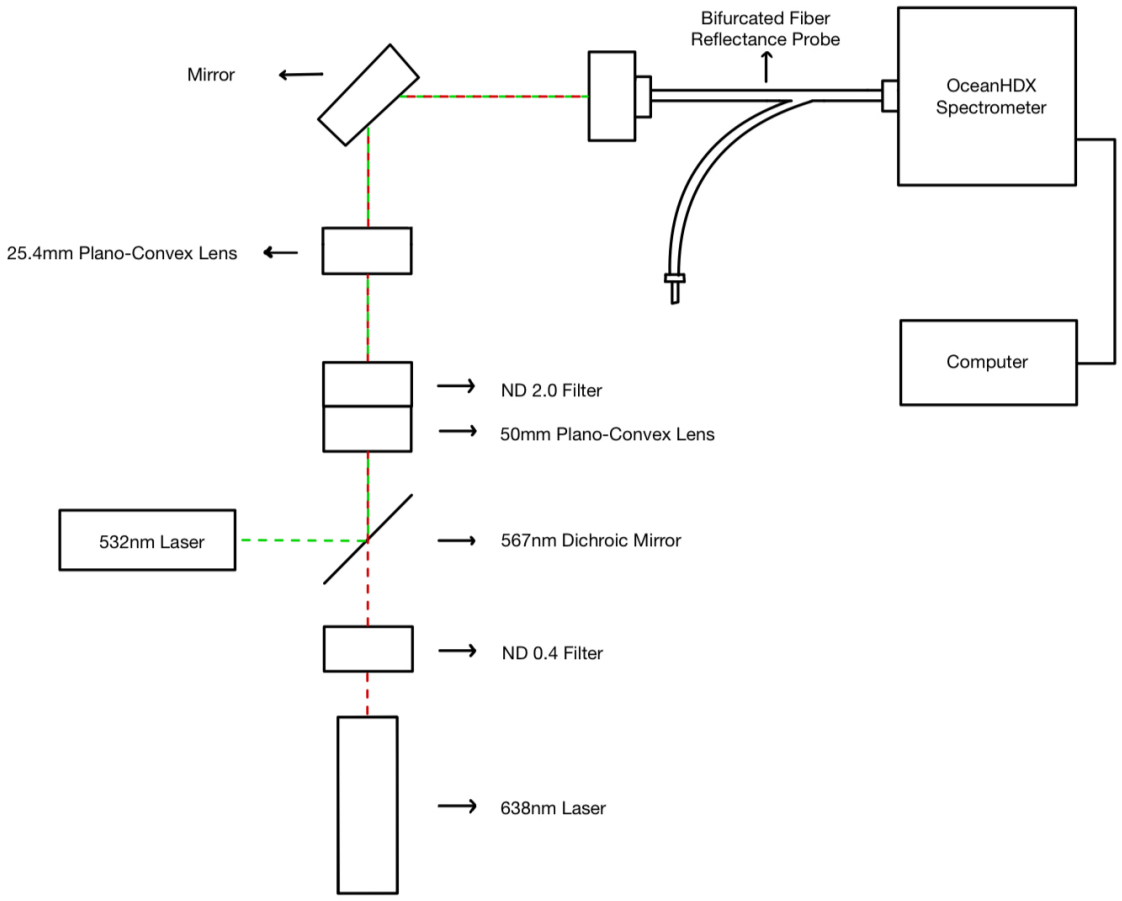

Figure 1: Schematic of two-laser setup.

The main goal of this project is to study how different wavelengths can be used to improve PPG measurements. A good way to do this is to use a broadband light source. The current setup would need to be changed to accommodate this, but the general concept would remain the same. The dichroic mirror would no longer be necessary with a single light source and some of the filters and lenses might need to be replaced to better focus a broadband light source. Aside from changing the light source, developing a better way to rest one's arm as the data is being collected would also be a significant improvement. By its nature, measuring reflectance is very sensitive to movement relative to the probe. Because of this sensitivity, any movement from the arm while collecting data can greatly influence the results. Another solution

to this problem would be to use a tissue-simulating phantom. Phantoms have been proposed as a way to test how changes in the setup and code affect the measurements without having to account for changes in a human's vitals between tests.

Data Collection

The primary focus of the setup is to measure the intensity of the skin reflectance, but several types of data needed to be collected in order to make the final PPG calculation. A dark correction for the spectrometer, a baseline for the laser intensity, and a measure of how the laser intensities change over time all need to be collected in addition to the skin reflectance.

The code to interact with the spectrometer was written in MATLAB. I was provided a program that measured the reflectance intensity but it needed to either be significantly changed or completely rewritten in order to work exactly how we needed it. I used that code to familiarize myself with using MATLAB to connect with the spectrometer and, to a more general degree, to figure out how to use MATLAB as I did not have much prior experience. After becoming familiar with the code, it became apparent that the biggest issue was its run speed. The provided code had many features like real time plotting that, while nice to have, slowed down the code. The original code was able to collect about seven data points per second which is an issue as it does not provide nearly enough detail to begin to see a PPG measurement. For example, if someone's heart rate were 150 bpm, then at seven points per second we would only get about three data points per expected PPG pulse. The program saved data points by looping over a call to the spectrometer to retrieve the spectrum and then adding that spectrum to the end of an array. The problem was that the array size was increased for each iteration which greatly reduced

speed. This, in combination with other issues, made rewriting the code the obvious course of action moving forward.

After writing a new program, we had something that did what we needed and nothing more. The basic function of the code is to get the spectrum from the spectrometer, interpolate the data to whole number wavelengths, and then save the intensities at desired wavelengths alongside the corresponding time. This yields a plot of the skin reflectance intensity over time. The final code allows for the skin reflectance to be measured and plotted at two user inputted wavelengths. Currently those wavelengths correspond to a 533nm laser and a 638nm laser, but they would be simple to change if a transition to a broadband light source is desired. The code also allows for a desired measurement duration to be inputted which it uses to predetermine any array sizes. This change, amongst others, allows the code to record about fifty-eight data points per second.

The methods learned while rewriting the skin reflectance code were applied to collecting the other data sets needed. For the dark correction data, a spectrum is recorded with all lights off so that any error stemming from the spectrometer itself can be identified. For the laser reflectance baseline, the spectrum is recorded using a Spectralon diffuse reflectance standard. This measurement and all other measurements were done in the dark as to avoid any error associated with ambient light. The laser change over time was collected in the same way as the skin intensity data, but using a Spectralon diffuse reflectance standard rather than an arm. The data collected was then divided by the value of the first point to see how it changed over time. The lasers were left running for at least fifteen minutes before any data was collected to allow them to warm up. An example of each of these data sets is shown below.

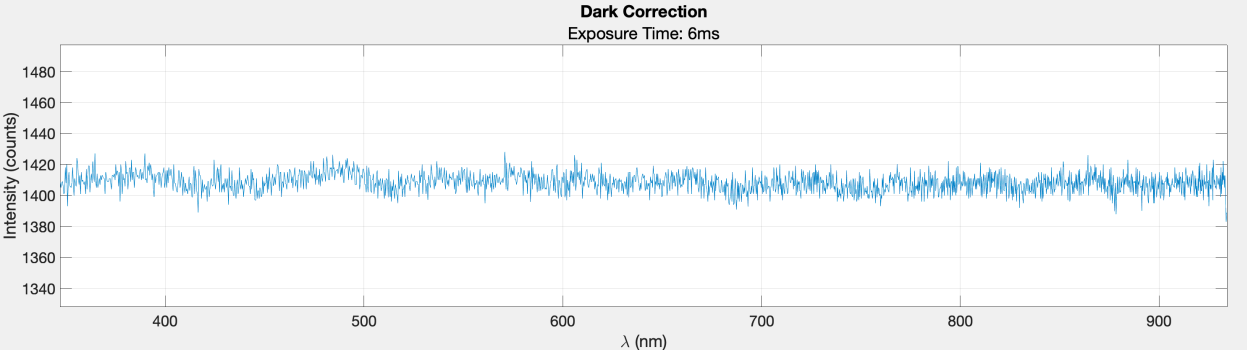

Figure 2: Dark correction data.

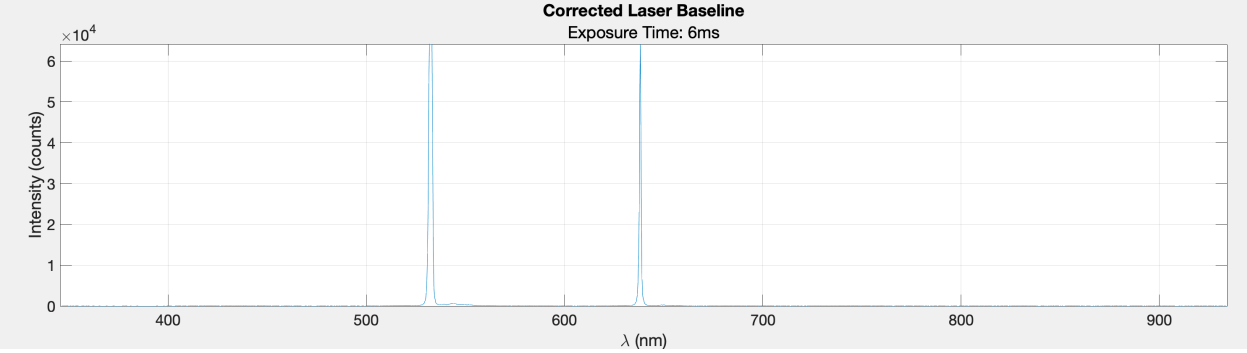

Figure 3: Laser baseline data with the dark correction subtracted.

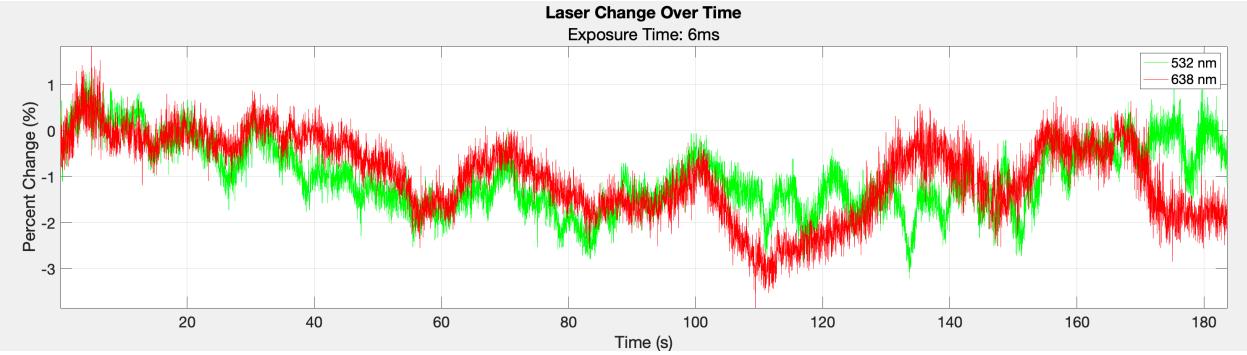

Figure 4: Percent change for the laser over time.

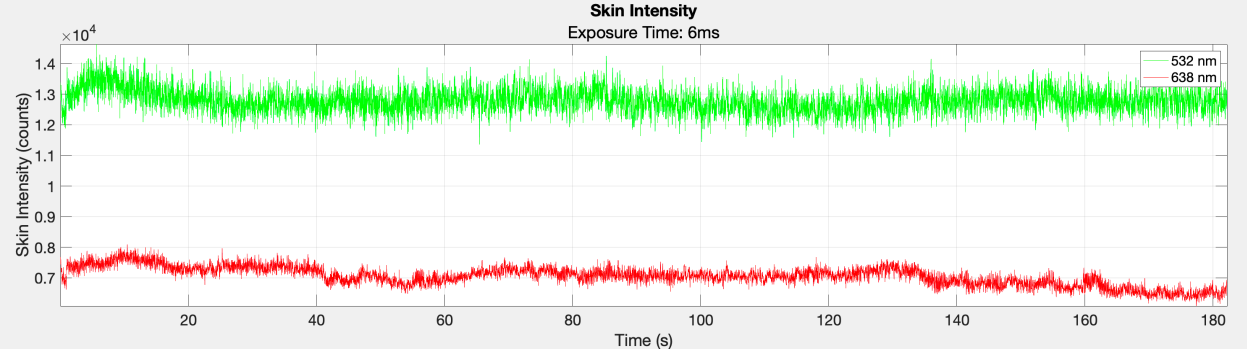

Figure 5: Skin intensity data.

Data Processing

With all of the different data sets collected, they can now be combined into a PPG measurement. This is given by:

$$PPG = -\log\left(\frac{(skin\ intensity\ -\ dark\ correction)}{(laser\ baseline\ -\ dark\ correction)}\right)$$

This calculation yields:

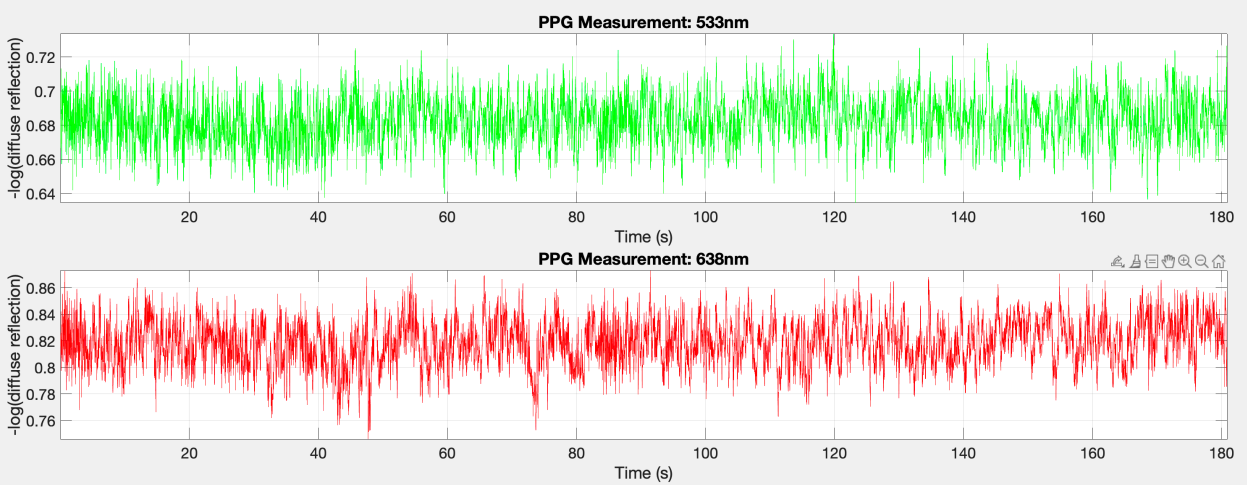

Figure 6: PPG calculation.

Clearly this data is very noisy and therefore it would be difficult to make anything of it. To solve this issue, a moving average was implemented. Per usual with moving averages, one must balance smoothing the data with losing potentially vital information. I decided on two

consecutive moving averages with an averaging window of ten data points for each. With the data smoothed, it becomes easier to see what is happening. In order to check our PPG calculations, we want to be able to use it to measure heart rate and then compare the result to the expected value. To do that the peaks must be counted. The "findpeaks" function was used with parameters defining the minimum peak prominence and the minimum distance between two peaks. The ideal values for prominence, peak separation, and moving average parameters must be experimentally determined by applying them to more data and checking the heart rate found to the known heart rate. With these changes we get a plot shown in figure 7.

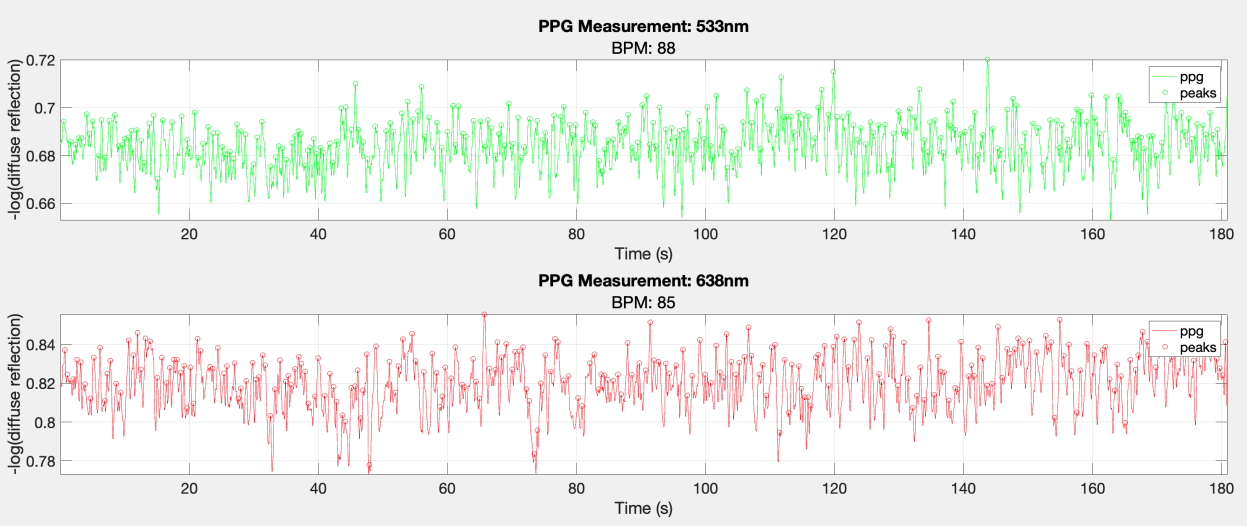

Figure 7: Smoothed PPG calculation with bpm found using findpeaks.

This is clearly easier to see than before the smoothing, but an option was added in the program to choose a time range from the data shown so that that range will be replotted with a recalculated bpm. This is helpful because the setup is sensitive to movement so it is easier to focus on shorter sections of data that appear less affected by movement. An example of a zoomed in PPG measurement is shown in figure 8.

Results

As mentioned previously, the only way to optimize the parameters for smoothing the PPG and calculating heart rate is to collect data and to compare it to an accepted heart rate measurement. The current parameters are two 10-point moving averages, a 0.002 minimum peak prominence, and a 17 point minimum peak separation. A few data sets will be presented here along with heart rates measured either by hand or with an Apple Watch SE. Figure 8 shows a PPG where the heart rate measured by hand was 92 bpm. Figure 9 shows a PPG measurement where the heart rate measured by hand was 125 bpm. I believe that the percent change for the green laser recorded before this PPG measurement is incorrect as it shows as zero percent change over the course of three minutes while the red laser remains within four percent.

Figure 8: Zoomed in PPG with 92 bpm heart rate measured by hand.

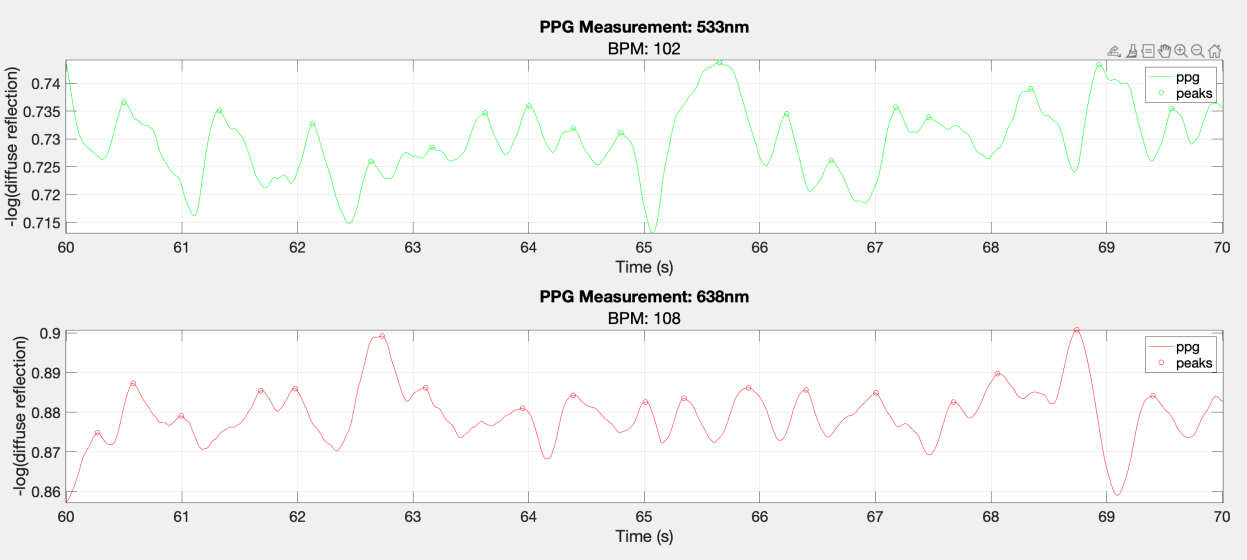

Figure 9: Zoomed in PPG with 125 bpm heart rate measured by hand.

The heart rate data found in figure eight closely matches the expected value. While the data in figure nine does not correlate as closely, the trend in the heart rates is correct between the figures. Figure ten shows the percent change for the lasers before the data shown in figures eleven and twelve were taken. These show PPG with heart rates measured by an Apple Watch SE to be 100 and 120 respectively.

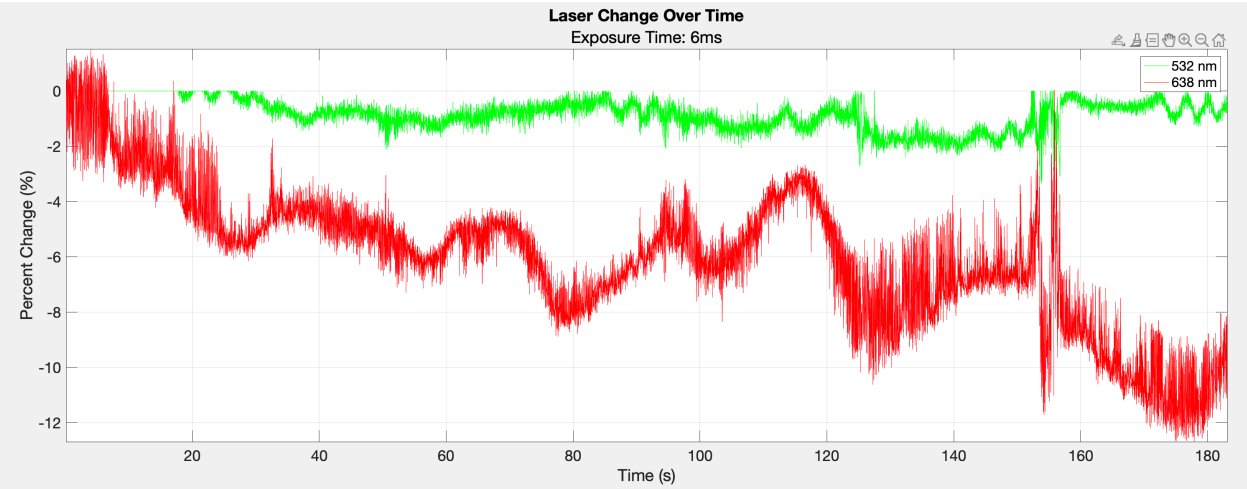

Figure 10: Percent change for data shown in figures ten and eleven.

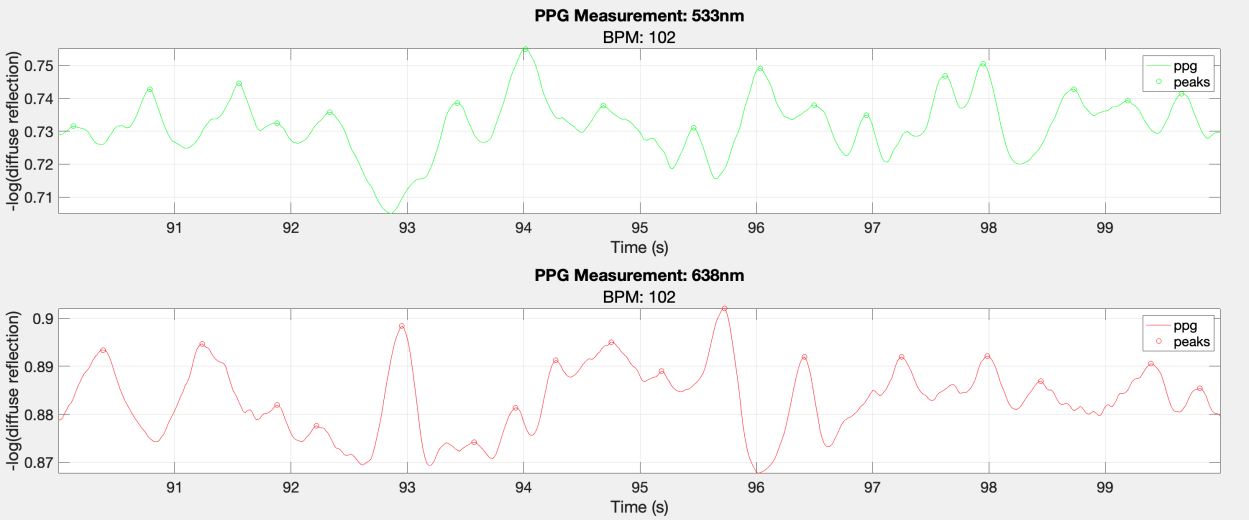

Figure 11: Zoomed in PPG with 100 bpm measured by Apple Watch SE.

Figure 12: Zoomed in PPG with 122 bpm measured by Apple Watch SE.

These plots yield promising heart rate calculations with the data in figure twelve being very close to the expected value.

The error in the data must be considered. With so few data sets the parameters are hard to optimize. With more data we could determine if there is a set of parameters that consistently provides an accurate heart rate measurement, and therefore proof that the PPG measurements are

being made correctly. There are a few different factors that point to error in the data. First, with some of the laser change data exceeding ten percent, the setup itself is a source of error. The sensitivity to movement provides an unknown amount of error, but it must be considered. Both of these forms of error could be mitigated by zooming in to shorter time scales when looking at the data rather than considering the longer data sets on their own. The other factor that is most concerning when looking at the data is that we don't seem to see clear PPG signals. Additionally, the peaks don't seem to have any correlation between wavelengths which we would expect to see if they are a result of a heartbeat.

Conclusion

A setup has been developed with the goal of making PPG measurements through the use of a bifurcated fiber reflectance probe. A code was developed to prioritize measurement speed so that as much detail could be shown as possible in the PPG measurements. The PPG was calculated, smoothed, and used to make a heart rate measurement.

The data found certainly shows promise as a single set of parameters was able to accurately show trends in heart rate data and for the most part give a measurement close to the expected value. There are concerns generated by a small number of data sets and simply by the appearance of the PPG waveforms. In order to continue to improve this moving forward, there are a series of steps that can be taken. First, I would take a large number of data sets from different people and compare it to an accepted method for measuring heart rate. This data could be used to optimize the parameters. Next, I would transition to a broadband light source so that different wavelengths can be tested with parameters that have already been determined. If the broadband light source were to be used before finding the best set of parameters, then it would

complicate understanding whether the wavelength being tested was performing poorly or if the parameters were simply doing a poor job of presenting the data collected.

This setup should set the groundwork for future work to be accomplished. Measuring PPG has allowed far more people access to preventative healthcare simply by giving them a way to consistently check many of their vitals. The switch to a broadband light source would allow for wavelengths to be tested against a wide range of BMIs and melanin concentrations so that we can hopefully make this technology and its benefits a reality for even more people.

References

[1] Allen, J. (2007). Photoplethysmography and its application in clinical physiological measurement. *Physiological Measurement*, *28*(3). https://doi.org/10.1088/0967-3334/28/3/r01

[2] Ajmal, Tananant Boonya-Ananta, Andres J. Rodriguez, V. N. Du Le, and Jessica C. Ramella-Roman, "Monte Carlo analysis of optical heart rate sensors in commercial wearables: the effect of skin tone and obesity on the photoplethysmography (PPG) signal," Biomed. Opt. Express 12, 7445-7457 (2021)1616

[3] Bent, B., Goldstein, B.A., Kibbe, W.A. *et al.* Investigating sources of inaccuracy in wearable optical heart rate sensors. *npj Digit. Med.* **3**, 18 (2020). https://doi.org/10.1038/s41746-020-0226-6

Appendix

Dark Correction Code

```
% Determine the dark correction to be applied to both the laser baseline
% and the real time spectrum.
cd('C:\Users\tblua\Desktop\WESLEY')
close all;
clear
clc

% Define the spectrometer
spectrometer = icdevice('OceanOptics_OmniDriver.mdd');
connect(spectrometer);

% Spectrometer Parameters
exposure_time = 6;  % (ms) 6ms -> 10s
int_time = exposure_time * 1000;
spect_index = 0;
channel = 0;
enable_flag = false;

% Check that a spectrometer is properly showing up:
num_of_spectrometers = invoke(spectrometer, 'getNumberOfSpectrometersFound');
if num_of_spectrometers > 1
    disconnect(spectrometer);
    delete(spectrometer);
    error('Too many spectrometers. Cannot support more than 1.')
elseif num_of_spectrometers < 1
    disconnect(spectrometer);
    delete(spectrometer);
    error('No spectrometers detected')
end

% Apply spectrometer parameters
invoke(spectrometer, 'setIntegrationTime', spect_index, channel, int_time);
invoke(spectrometer, 'setCorrectForDetectorNonlinearity', spect_index,
channel, enable_flag);
invoke(spectrometer, 'setCorrectForElectricalDark', spect_index, channel,
enable_flag);

% Getting the Spectrum
spectrometer_wavelengths = invoke(spectrometer, 'getWavelengths',
spect_index, channel);
dark_corr = invoke(spectrometer, 'getSpectrum', spect_index);


% Plotting the Spectrum

plot(spectrometer_wavelengths, dark_corr);
title('Dark Correction');
subtitle_text = ['Exposure Time: ' int2str(exposure_time) 'ms'];
subtitle(subtitle_text);
```

```
ylabel('Intensity (counts)');
xlabel('\lambda (nm)');
grid on
axis tight

% Save the full spectrum for normalization
save("data/dark_correction.mat", "dark_corr", "spectrometer_wavelengths")


% Disconnect devices and delete object
disconnect(spectrometer);
delete(spectrometer);
```

Laser Baseline Code

```
% calculate the spectralon intensity minus the dark correction.
cd('C:\Users\tblua\Desktop\WESLEY')
close all;
clear
clc

% Define the spectrometer
spectrometer = icdevice('OceanOptics_OmniDriver.mdd');
connect(spectrometer);


% Spectrometer Parameters
exposure_time = 6;  % (ms) 6ms -> 10s
int_time = exposure_time * 1000;
spect_index = 0;
channel = 0;
enable_flag = false;

% Check that a spectrometer is properly showing up:
num_of_spectrometers = invoke(spectrometer, 'getNumberOfSpectrometersFound');
if num_of_spectrometers > 1
    disconnect(spectrometer);
    delete(spectrometer);
    error('Too many spectrometers. Cannot support more than 1.')
elseif num_of_spectrometers < 1
    disconnect(spectrometer);
    delete(spectrometer);
    error('No spectrometers detected')
end

% Apply spectrometer parameters
invoke(spectrometer, 'setIntegrationTime', spect_index, channel, int_time);
invoke(spectrometer, 'setCorrectForDetectorNonlinearity', spect_index,
channel, enable_flag);
invoke(spectrometer, 'setCorrectForElectricalDark', spect_index, channel,
enable_flag);

% Getting the Spectrum
```

```
spectrometer_wavelengths = invoke(spectrometer, 'getWavelengths',
spect_index, channel);
spectralon_intensity = invoke(spectrometer, 'getSpectrum', spect_index);


% Subtracting dark correction.
load("data/dark_correction.mat", "dark_corr");
spectralon_intensity = spectralon_intensity - dark_corr;
spectralon_intensity(spectralon_intensity < 0) = 0;


% Plotting the Spectrum
plot(spectrometer_wavelengths, spectralon_intensity);
title('Spectralon Intensity');
subtitle_text = ['Exposure Time: ' int2str(exposure_time) 'ms'];
subtitle(subtitle_text);
ylabel('Intensity (counts)');
xlabel('\lambda (nm)');
grid on
axis tight

% Save the full spectrum for normalization
save("data/spectralon_intensity.mat", "spectralon_intensity",
"spectrometer_wavelengths")


% Disconnect devices and delete object
disconnect(spectrometer);
delete(spectrometer);
fprintf('baseline saved as Desktop\\WESLEY\\data\\spectralon_intensity.mat
');
```

Skin Intensity Code

```
% measure the reflection from the skin

cd('C:\Users\tblua\Desktop\WESLEY')
close all;
clear
clc


% Define parameters for spectrum acquisition

% User Inputs
lower_focus_wavelength = 533; % nanometers
higher_focus_wavelength = 638; % nanometers
exposure_time = 6;  % (ms) 6ms -> 10s
seconds_recorded = 180;
points_saved = 58 * seconds_recorded;

nm=(lower_focus_wavelength-20):(higher_focus_wavelength+20);
[min_nm, max_nm] = bounds(nm);
nm_533=find(nm==lower_focus_wavelength);
```

```
nm_638=find(nm==higher_focus_wavelength);
integrateTime = exposure_time * 1000;
spectroIndex = 0;
channelIndex = 0;
enableElectricalDarkCorrection = false;
recording = 0;


% Make sure OmniDriver is on path
omniDriverPathStr = ...
    fullfile('C:\Program Files\Ocean Optics\OmniDriver\OOI_HOME');

if ~contains(path, omniDriverPathStr)
    addpath(omniDriverPathStr)
end

% Create spectrometer object and connect to device
omniDriverFilename = 'OceanOptics_OmniDriver.mdd';

Spectrometer = icdevice(omniDriverFilename);
connect(Spectrometer)

% Determine how many spectrometers are connected
numSpectrometers = invoke(Spectrometer, 'getNumberOfSpectrometersFound');
fprintf('Found %d Ocean Optics spectrometer(s).\n\n', numSpectrometers);

% Throw error if more than one spectrometer connected
if numSpectrometers > 1
    disconnect(Spectrometer);
    delete(Spectrometer);
    error('Too many spectrometers. Cannot support more than 1.')
end

% Display names and serial numbers for each spectrometer
for iSpec = 0:(numSpectrometers - 1)
    specName = invoke(Spectrometer, 'getName', iSpec);
    specSerialNum = invoke(Spectrometer, 'getSerialNumber', iSpec);

    fprintf('Spectrometer Index: %d\n', iSpec);
    fprintf('\t    Name: %s\n', specName);
    fprintf('\tSerial #: %s\n', specSerialNum);
end

fprintf('\n');

% Set spectrum acquisition parameters
invoke(Spectrometer, 'setIntegrationTime', spectroIndex,channelIndex,
integrateTime);
invoke(Spectrometer, 'setCorrectForElectricalDark', spectroIndex, ...
    channelIndex, enableElectricalDarkCorrection);

% Get set of wavelengths
spectrometer_wavelengths = ...
    invoke(Spectrometer, 'getWavelengths', spectroIndex, channelIndex);
```

```
% Check if the spectrometer can yield the desired wavelengths
[min_spectro_nm, max_spectro_nm] = bounds(spectrometer_wavelengths);
if ~isequal(nm, spectrometer_wavelengths)
    if min_nm < min_spectro_nm
        min_nm = min_spectro_nm;
        fprintf('the lower focus wavelength is lower than the spectrometer
can provide');
    end
    if max_nm > max_spectro_nm
        max_nm = max_spectro_nm;
        fprintf('the higher focus wavelength is higher than the spectrometer
can provide')
    end
    nm = min_nm:max_nm;

end

% Define vectors/arrays for time and wavelength data
timeVec_sec = zeros(1,points_saved);
intensity_533 = zeros(1, points_saved);
intensity_638 = zeros(1, points_saved);

% start the timer
tic

% Loop to collect and plot data:
i=0;
% fprintf("Data recording will stop when the space key is hit.\n");

while i < points_saved
    i = i + 1;

    % Acquire spectrum
    spectrumIntensityVec_counts = ...
        invoke(Spectrometer, 'getSpectrum', spectroIndex);

    % Interpolate for desired wavelength range
    intensity=interp1(spectrometer_wavelengths,
spectrumIntensityVec_counts,nm);

    intensity_533(i) = intensity(nm_533);
    intensity_638(i) = intensity(nm_638);
    timeVec_sec(i) = toc;

end

fprintf('Increase the value of the points_saved variable to record for
longer.');
save_file_name = string(datetime('now','Format','yyyy_MM_dd_HH_mm_ss')) +
"_skin_intensity.mat";

save("data/"+save_file_name, 'timeVec_sec', 'intensity_638', 'intensity_533',
"nm", "nm_533", "nm_638");
```

```
fprintf("\n\nData saved to: \n"+ save_file_name);
fprintf("\n");
% Disconnect devices and delete object
disconnect(Spectrometer);
delete(Spectrometer);
fprintf('\nSpectrometer disconnected');
close all;
```

Laser Change Over Time Code

```
% measure the laser change over time

cd('C:\Users\tblua\Desktop\WESLEY')
close all;
clear
clc


% Define parameters for spectrum acquisition

% User Inputs
lower_focus_wavelength = 533; % nanometers
higher_focus_wavelength = 638; % nanometers
exposure_time = 6;  % (ms) 6ms -> 10s
seconds_recorded = 180;
points_saved = 58 * seconds_recorded;

nm=(lower_focus_wavelength-20):(higher_focus_wavelength+20);
[min_nm, max_nm] = bounds(nm);
nm_533=find(nm==lower_focus_wavelength);
nm_638=find(nm==higher_focus_wavelength);
integrateTime = exposure_time * 1000;
spectroIndex = 0;
channelIndex = 0;
enableElectricalDarkCorrection = false;
recording = 0;


% Make sure OmniDriver is on path
omniDriverPathStr = ...
    fullfile('C:\Program Files\Ocean Optics\OmniDriver\OOI_HOME');

if ~contains(path, omniDriverPathStr)
    addpath(omniDriverPathStr)
end

% Create spectrometer object and connect to device
omniDriverFilename = 'OceanOptics_OmniDriver.mdd';

Spectrometer = icdevice(omniDriverFilename);
connect(Spectrometer)

% Determine how many spectrometers are connected
numSpectrometers = invoke(Spectrometer, 'getNumberOfSpectrometersFound');
```

```
fprintf('Found %d Ocean Optics spectrometer(s).\n\n', numSpectrometers);

% Throw error if more than one spectrometer connected
if numSpectrometers > 1
    disconnect(Spectrometer);
    delete(Spectrometer);
    error('Too many spectrometers. Cannot support more than 1.')
end

% Set spectrum acquisition parameters
invoke(Spectrometer, 'setIntegrationTime', spectroIndex,channelIndex,
integrateTime);
invoke(Spectrometer, 'setCorrectForElectricalDark', spectroIndex, ...
    channelIndex, enableElectricalDarkCorrection);

% Get set of wavelengths
spectrometer_wavelengths = ...
    invoke(Spectrometer, 'getWavelengths', spectroIndex, channelIndex);

% Check if the spectrometer can yield the desired wavelengths
[min_spectro_nm, max_spectro_nm] = bounds(spectrometer_wavelengths);
if ~isequal(nm, spectrometer_wavelengths)
    if min_nm < min_spectro_nm
        min_nm = min_spectro_nm;
        fprintf('the lower focus wavelength is lower than the spectrometer
can provide');
    end
    if max_nm > max_spectro_nm
        max_nm = max_spectro_nm;
        fprintf('the higher focus wavelength is higher than the spectrometer
can provide')
    end
    nm = min_nm:max_nm;

end

% Define vectors/arrays for time and wavelength data
change_time = zeros(1,points_saved);
intensity_533 = zeros(1, points_saved);
intensity_638 = zeros(1, points_saved);

% start the timer
tic

% Loop to collect and plot data:
i=0;
% fprintf("Data recording will stop when the space key is hit.\n");

while i < points_saved
    i = i + 1;

    % Acquire spectrum
    spectrumIntensityVec_counts = ...
        invoke(Spectrometer, 'getSpectrum', spectroIndex);
```

```
    % Interpolate for desired wavelength range
    intensity=interp1(spectrometer_wavelengths,
spectrumIntensityVec_counts,nm);

    intensity_533(i) = intensity(nm_533);
    intensity_638(i) = intensity(nm_638);
    change_time(i) = toc;

end

fprintf('Increase the value of the points_saved variable to record for
longer.');

% Calculating Percent Change
first_point_533 = intensity_533(1);
first_point_638 = intensity_638(1);
percent_change_533 = ((intensity_533 - first_point_533)/first_point_533)*100;
percent_change_638 = ((intensity_638 - first_point_638)/first_point_638)*100;

% Plot the percent change over time.
plot(change_time, percent_change_533, 'g')
hold on
plot(change_time, percent_change_638, 'r')
title('Percent Change Over Time')
subtitle_text = ['Exposure Time: ' int2str(exposure_time) 'ms'];
subtitle(subtitle_text)
ylabel('Percent Change (%)')
xlabel('Time (s)')
ylim([-10 10])
grid on
axis tight
legend('532 nm', '638 nm')

%Saving the file
save_file_name = string(datetime('now','Format','yyyy_MM_dd_HH_mm_ss')) +
"_percent_change.mat";

save("data/"+save_file_name, 'percent_change_638', 'percent_change_533',
"change_time");
fprintf("\n\nData saved to: \n"+ save_file_name);
fprintf("\n");
% Disconnect devices and delete object
disconnect(Spectrometer);
delete(Spectrometer);
fprintf('\nSpectrometer disconnected');
close all;
```

PPG Calculation Code

```
% calculate and plot ppg.

cd('C:\Users\tblua\Desktop\WESLEY')
```

```
close all;
clear
clc

% number of  points to be used in the moving average.
average_range = 10;
moving_averages = 2;
peak_prominence = 0.002;

% request the skin_intensity file for which the plot is wanted.
filename = input('Enter the desired skin_intensity file name: ', 's');
file_date = filename(1 : end - 18);

% load the skin intensity, dark corrected spectralon intensity, and the
% dark correction data files.
load("data/spectralon_intensity.mat", "spectralon_intensity",
"spectrometer_wavelengths");
load("data/dark_correction.mat", "dark_corr");
load("data/" + filename, "intensity_533", "intensity_638", "timeVec_sec",
"nm", "nm_533", "nm_638");

% interpolate the spectra so that we can use whole number wavelengths.
spectralon_intensity=interp1(spectrometer_wavelengths,
spectralon_intensity,nm);
dark_corr=interp1(spectrometer_wavelengths, dark_corr,nm);

% calculate diffuse reflection
diffuse_refl_533 = ((intensity_533 - dark_corr(nm_533))/
spectralon_intensity(nm_533));
diffuse_refl_638 = ((intensity_638 - dark_corr(nm_638))/
spectralon_intensity(nm_638));

% calulate ppg measurements
ppg_533_no_smoothing = -log10(diffuse_refl_533);
ppg_638_no_smoothing = -log10(diffuse_refl_638);

% moving averages
i = 0;
ppg_533 = ppg_533_no_smoothing;
ppg_638 = ppg_638_no_smoothing;
while i < moving_averages
    ppg_533 = movmean(ppg_533, average_range);
    ppg_638 = movmean(ppg_638, average_range);
    i = i + 1;
end

% finding the peaks
[peaks_533, peak_index_533] = findpeaks(ppg_533, 'MinPeakProminence',
peak_prominence, 'MinPeakDistance', 20);
[peaks_638, peak_index_638] = findpeaks(ppg_638, 'MinPeakProminence',
peak_prominence, 'MinPeakDistance', 20);

peak_times_533 = timeVec_sec(peak_index_533);
peak_times_638 = timeVec_sec(peak_index_638);
```

```
num_peaks_533 = numel(peaks_533);
num_peaks_638 = numel(peaks_638);

% calculating bpm
total_time = max(timeVec_sec);
bpm_533 = round((num_peaks_533/total_time)*60);
bpm_638 = round((num_peaks_638/total_time)*60);

save_file_name = file_date + "ppg_calculation.mat";

save("data/"+save_file_name, 'ppg_533', 'ppg_638',
'ppg_533_no_smoothing', ...
    'ppg_638_no_smoothing', 'timeVec_sec', "peak_times_638",
"peak_times_533", "peaks_638", "peaks_533");
fprintf("\nData saved to: \n"+ save_file_name);

% Plotting the smoothed ppg and peaks
figure;

subplot(2,1,1);
plot(timeVec_sec, ppg_533, 'g-' ,peak_times_533, peaks_533, 'go');
title('PPG Measurement: 533nm');
subtitle_text = ['BPM: ' int2str(bpm_533)];
subtitle(subtitle_text);
legend('ppg', 'peaks');
ylabel('-log(diffuse reflection)');
xlabel('Time (s)');
grid on
axis tight
ax = gca;
ax.FontSize = 18;

subplot(2,1,2);
plot(timeVec_sec, ppg_638,'r-' , peak_times_638, peaks_638,'ro');
title('PPG Measurement: 638nm');
subtitle_text = ['BPM: ' int2str(bpm_638)];
subtitle(subtitle_text);
legend('ppg', 'peaks');
ylabel('-log(diffuse reflection)');
xlabel('Time (s)');
grid on
axis tight
ax = gca;
ax.FontSize = 18;

% Ask if a shorter time scale is desired
fprintf("\n\nWould you like to replot with a specific time range?\n");
shortening_input = input("Enter y for yes or n for no: " , "s");

%shorten the time frame and replot the data.
if strcmpi(shortening_input, 'y')
%    Find the index in timeVec_sec that is closest to the user input.
```

```
    start_input = input("Enter the time (seconds) you want the data to start?
");
    end_input = input("Enter the time (seconds) you want the data to end? ");
    start_diff = abs(timeVec_sec - start_input);
    end_diff = abs(timeVec_sec - end_input);
    [min_start_diff, start_index] = min(start_diff);
    [min_end_diff, end_index] = min(end_diff);
    trim_ppg_533 = ppg_533(start_index : end_index);
    trim_ppg_638 = ppg_638(start_index : end_index);
    trim_time = timeVec_sec(start_index : end_index);

    % finding the peaks
    [peaks_533, peak_index_533] = findpeaks(trim_ppg_533,
'MinPeakProminence', peak_prominence);
    [peaks_638, peak_index_638] = findpeaks(trim_ppg_638,
'MinPeakProminence', peak_prominence);

    peak_times_533 = trim_time(peak_index_533);
    peak_times_638 = trim_time(peak_index_638);

    num_peaks_533 = numel(peaks_533);
    num_peaks_638 = numel(peaks_638);

    % calculating bpm
    total_time = (max(trim_time) - min(trim_time));
    bpm_533 = round((num_peaks_533/total_time)*60);
    bpm_638 = round((num_peaks_638/total_time)*60);

    % Plotting the smoothed ppg and peaks
    figure;

    subplot(2,1,1);
    plot(trim_time, trim_ppg_533, 'g-' ,peak_times_533, peaks_533, 'go');
    title('PPG Measurement: 533nm');
    subtitle_text = ['BPM: ' int2str(bpm_533)];
    subtitle(subtitle_text);
    legend('ppg', 'peaks');
    ylabel('-log(diffuse reflection)');
    xlabel('Time (s)');
    grid on
    axis tight
    ax = gca;
    ax.FontSize = 18;

    subplot(2,1,2);
    plot(trim_time, trim_ppg_638,'r-' , peak_times_638, peaks_638,'ro');
    title('PPG Measurement: 638nm');
    subtitle_text = ['BPM: ' int2str(bpm_638)];
    subtitle(subtitle_text);
    legend('ppg', 'peaks');
    ylabel('-log(diffuse reflection)');
    xlabel('Time (s)');
    grid on
    axis tight
```

```
    ax = gca;
    ax.FontSize = 18;


elseif strcmpi(shortening_input, 'n')
    disp("okie dokie :]");

else
    disp('You entered something other than y or n\n');
end
```