Extracting Heart rate from Cell Phone Videos Using Fast Fourier Transform (FFT) and Blind Source Separation (BSS) with Active Face Tracking using MATLAB

Michael Bassis
Electromechanical Engineer
Wentworth Institute of Technology
Boston, Massachusetts
bassism@wit.edu

Rami Hanna
Electromechanical Engineer
Wentworth Institute of Technology
Boston, Massachusetts
hannar1@wit.edu

***Abstract*—In the medical field, there are times when normal contact-based measurements are not ideal due to contamination or other factors that can affect readings. This has led to research into methods for reading vital signs in a contactless method. This paper presents results from the research and implementation of a contactless method for measuring heart rate using video from a cell phone camera. This method also implements a face tracking algorithm to make the system more robust. The success of this project will be measured against reference heart rates taken at the same time as the video sample to compute algorithm accuracy.**

Keywords—heart rate, digital signal processing, video, FFT, fastICA, BSS, face tracking

I. Introduction

The need for contactless measurement of vital signs is nothing new. Vital signs are incredibly important in understanding the condition of the patient and how their body reacts to disease and the treatment they are receiving. Currently, most of these measurements must be taken with sensors which contact the patient. Measurement methods that require contact can be physically intrusive, cause irritation to the patient, spread unnecessary infections, and be cumbersome for the mobility of the patient or the health care workers who treat them. This paper will specifically focus on the heart rate but is also easily applicable to respiratory rate.

Historically, there have been two different methods for reading heart rate without contact. These two methods are radio frequency (RF) imaging and camera-based imaging, which both have their own advantages and disadvantages. While this paper will address RF, the primary focus will be on the camera-based imaging method for detecting heart rate in subjects.

II. Theory of Operation

The driving principle that both RF and camera-based imaging utilize for detecting heart rate is the small movements of the body that can be measured and processed to extract a heart rate frequency [1].

For RF, an unmodulated RADAR signal is reflected off the subject and the phase response is measured to determine the movement of the subject's body. The heart rate would then be determined by finding the periodicity of distance values which would correspond to the small movement due to cardiac activity.

In camera-based imaging, a region of interest (ROI) is defined for the image collected from a camera and the intensity of light signals in the ROI help detect the small movements of the body due to cardiac activity [2]. In a similar fashion to the RF method, the periodicity of these signals is examined, and a heart rate is extracted.

As previously stated, it is very easy for the method provided to be altered for respiratory rate over heart rate. This is because both bodily functions are periodic and result in small movements in the body. To distinguish between them a filter is added to remove frequencies that would fall out of the range of the target vital sign. For example, to read heart rate, a Butterworth bandpass filter can be implemented that only keeps frequencies between 0.8 and 3 Hz. This frequency range corresponds to a heart rate of 40 - 180 BPM which is a reasonable range for human heart rate, but also excludes frequencies in which respiratory function would appear.

III. Implementation of Theory

To test the proposed theory of operation, data was collected in the form of 30 second videos of subjects to measure their heart rate. At the same time their heart rate was collected in conventional manner using devices with heart rate sensors when applicable or measuring manually via the wrist when other methods were not available. The video analysis was performed in MATLAB. First, the face tracking algorithm was implemented to create the ROI for each face using the Computer Vision Toolbox. The ROI was positioned so that it could collect data from the tip of each subject's nose. Then, the red component of the ROI was selected for further analysis. Red was chosen because it can be used to detect color change due to more blood in the area and light intensity which would correspond to head position.

The next step is where the two methods deviate. Two separate methods were implemented and tested to compare their ability to detect a heart rate close to that of the ground truth taken during recording. The two methods used were Fast Fourier Transform (FFT) and Blind Source Separation (BSS) using fast Independent Component Analysis (fastICA) [3]. Both methods are similar in their ability to detect the periodic heart rate signal but differ in exactly how they arrive at that answer.

The FFT was implemented first. The FFT converts the ROI data into the frequency domain and then applies a 2^{nd} order Butterworth filter to remove the extra frequencies which would fall out of a reasonable range of heart rate. The peaks of the filtered signal were then analyzed, and the heart rate can be extracted. Figure 2 shows an example of the process below.

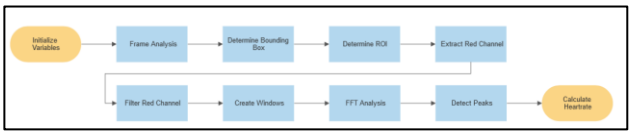

Figure 1: FFT Flowchart

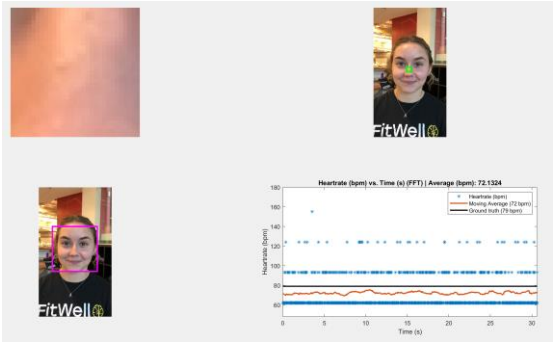

Figure 2: FFT Method, Subject BPM 79

Subplot one is a still frame image of the ROI. Subplot two shows the ROI superimposed onto the original video frame. Subplot three shows the bounding box created by the face tracking algorithm. Subplot four shows the graph over time of the heart rate in BPM. The graph shows individual data points in addition to a moving average and the ground truth reference heart rate.

The BSS method via fastICA was implemented second. This method uses the fastICA function to analyze the image input from the grayscale ROI. fastICA is a technique for separating a multivariate signal into independent, non-Gaussian signals. It works by finding the directions in the signal that have high kurtosis (i.e., high peaks and sharp edges), which are likely to correspond to the underlying independent signals. In the context of this application, those high kurtosis components would be the due to the cardiac activity of the subject. The output of the fastICA can then be used to determine the effective heart rate of the subject being recorded. Figure 4 shows an example of the fastICA process on the same subject below.

Figure 3: fastICA Flowchart

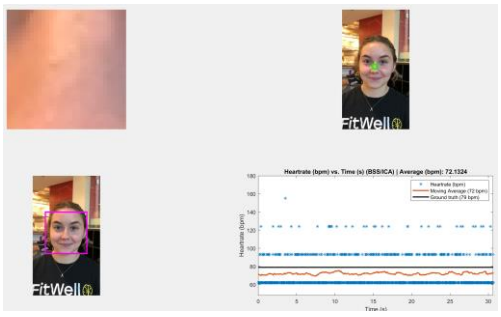

Figure 4: fastICA Method, Subject 79 BPM
Subplot one is a still frame image of the ROI. Subplot two shows the ROI superimposed onto the original video frame. Subplot three shows the bounding box created by the face tracking algorithm. Subplot four shows the graph over time of the heart rate in BPM. The graph shows individual data points in addition to a moving average and the ground truth reference heart rate.

IV. Results Between Methods

To compare the efficacy of both algorithms against the ground truth and against each other, 16 trials were run comparing detected heart rate and time required for computation. This data was tabulated into table 1 below.

Table 1: FFT vs fastICA Performance

Trial #	Ground Truth	Heart Rate BPM	% Error	FFT Execution Time	fastICA Execution Time
1	72	74.60	3.62	284.6	312.5
2	108	57.71	46.6	291.9	313.8
3	108	127.2	17.8	387.9	349.1
4	79	72.13	8.69	368.9	366.8
5	108	84.59	21.7	360.5	365.8
6	76	76.64	0.85	372.1	364.9
7	108	124.9	15.7	351.7	373.1
8	88	60.39	31.4	279.5	314.7
9	85	92.31	8.60	1290.3	1311.1
10	79	46.39	41.3	698.2	750.4
11	104	77.54	25.5	330.6	343.6
12	63	104.8	66.3	1143.9	1223.9
13	88	70.57	19.8	209.5	305.4
14	68	89.81	32.1	267.9	377.7
15	85	88.23	3.73	423.9	452.6
16	130	100.0	26.1	411.6	416.7

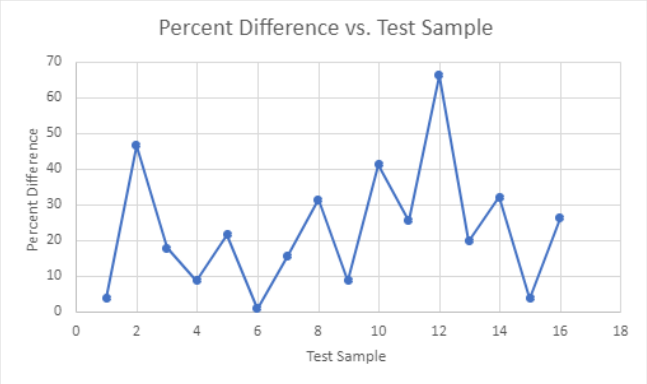

Figure 5: Accuracy of Samples

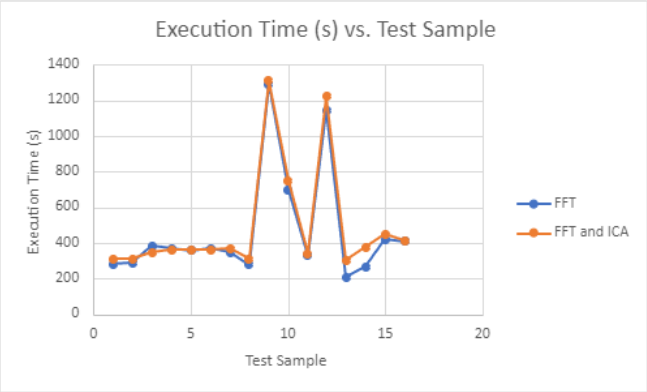

Figure 6: Execution Time vs. Test Samples

The data from the table indicates that fastICA and FFT based algorithm have nearly identical performance, with the only difference being small discrepancies in the execution time for each sample. The FFT has an average execution time of 467.07 seconds while the fastICA has an execution time of 496.39 seconds. So, on average the FFT method is 29.32 seconds faster.

The FFT method was implemented first, and offered a 76.1% accuracy over the trials, with a minimum accuracy of 34.7% and a maximum accuracy of 99.16% for the average heart rate during the video. Since this performance was lackluster in accuracy, the fastICA method was implemented to see if it would provide more accurate heart rate readings. Contrary to our initial hypothesis, the fastICA method performed identically to the FFT method with the only difference in the two methods being a slightly longer average execution time for the fastICA method. This result makes logical sense. The FFT is an optimized DFT, and this task is the perfect application for it. The fastICA is another powerful function, however it is competing against the optimized tool for the job so it makes sense that it would lag.

Based on the new insight of the similar performance, a new hypothesis was formed: since both techniques use different methods for deriving the heart rate signal from the subject, it was hypothesized that the error may not be from the analysis techniques themselves, but from other sources like lighting abnormalities or the face tracking algorithm redefining the ROI. Further experimentation with more constant light sources or a static subject could be explored to test this new hypothesis.

V. Conclusion

In conclusion, this paper has determined that both FFT and fastICA based heart rate detection show great potential for future contactless measurement of periodic vital signs especially when paired with a face tracking algorithm. In the future, the results of this study will benefit from better face tracking algorithms and better lighting. Face tracking will benefit the project because one of the accuracy limiting factors for this project was the ROI placement. Different sized faces and framing of the camera picture can affect what part of the face the ROI ends up collecting data from. A face tracking algorithm that can detect more specific details like forehead space not covered with hair or shadows, or an algorithm that can avoid placing the ROI on a region with glasses would improve the readings collected during the video. The lighting is also key in performance since the algorithm detects differences in the amount of light interacting with the area of the ROI. Inconsistent lighting due to shadows or flickering lights can interfere with readings, so a constant light source would be beneficial for the next iteration of the research. Another potential route for the next phase of research would be spatial and temporal filtering. Spatial and temporal filtering can be used to target and amplify the movements and color shifts that are being examined for heart rate detection.

Acknowledgments

This work was completed with the gracious assistance of students providing video samples for algorithm testing and the guiding challenge provided by Professor Federica Aveta.

References

[1] H Abuella and S Ekin, “Wireless Vital Sisns Monitoring System using Visible Light Sensing (VLS).” June 2018, https://www.slideshare.net/mybalaraja/visible-light-138335083

[2] Lee, Cho, Lee, and Whang, “Vision-Based Measurement of Heart Rate from Ballistocardiographic Head Movements Using Unsupervised Clustering,” Sensors, vol. 19, no. 15, p. 3263, Jul. 2019, doi: https://doi.org/10.3390/s19153263.

[3] M.-Z. Poh, D. J. McDuff, and R. W. Picard, “Non-contact, automated cardiac pulse measurements using video imaging and blind source separation,” Optics Express, vol. 18, no. 10, p. 10762, May 2010, doi: https://doi.org/10.1364/oe.18.010762.

[4] H.-Y. Wu, M. Rubinstein, E. Shih, J. Guttag, F. Durand, and W. Freeman, “Eulerian video magnification for revealing subtle changes in the world,” ACM Transactions on Graphics, vol. 31, no. 4, pp. 1–8, Jul. 2012, doi: https://doi.org/10.1145/2185520.2335416.

[5] U. Upadhyay, “Heartrate Detection using Camera,” Intel Software Innovators, Jun. 03, 2019. https://medium.com/intel-software-innovators/heartrate-detection-using-camera-d34b3289e272 (accessed Apr. 17, 2023).

[6] W. Verkruysse, L. O Svaasand, and J. S. Nelson, “Optica Publishing Group,” opg.optica.org, Dec. 12, 2008. https://opg.optica.org/oe/fulltext.cfm?uri=oe-16-26-21434&id=175396 (accessed Apr. 17, 2023).

[7] A. Ahmed, A. El, and R. Elnakib, “Heart rate measurement using webcam Supervised by.” Accessed: Apr. 17, 2023. [Online]. Available: https://engfac.mans.edu.eg/images/files/engpdf/projects/comm/pro2/heart-rate-book.pdf

Appendix

Table 2: Work Allocation Table

Task	Michael Bassis Contribution	Rami Hanna Contribution
Research	50%	50%
Coding	30%	70%
Report Writeup	90%	10%
Class Presentation	10%	90%

Table 3: MATLAB add-ons Utilized.

Computer Vision Toolbox
Signal Processing Toolbox
PCA and ICA Data Package

heartrate_fft.m

```
clear
close all
clc

tic
% Ramiface_1,2 were recorded at
79 bpm
% Create CascadeObjectDetector
object
faceDetector =
vision.CascadeObjectDetector();

filename = 'rami_130.MOV';

% Read video file
video =
VideoReader(['C:\Users\hannar1\
OneDrive - Wentworth Institute
of Technology\Dsp\',
filename]);
% video =
VideoReader('IMG_4172.mov');
frameRate = video.FrameRate;
max_frames = video.Duration *
frameRate;
max_frames = cast(max_frames,
'uint32');

matches = regexp(filename,
'_\d+\.', 'match');
if ~isempty(matches)
    ground_truth =
str2double(matches{1}(2:end-
1));
else
    % handle case where no
match is found
end
ground_truth = ground_truth *
ones(1, max_frames);

time = linspace(0,
video.Duration, max_frames);

heartrates = zeros(1,
floor(max_frames));
time_current = zeros(1,
ceil(video.Duration));

% Figure for visualization
figure

for i = 1 : floor(max_frames)
    % Read current frame
    frame = readFrame(video);

    % Detect face in the frame
    try
        bboxes =
step(faceDetector, frame);
    catch ME
        warning(['Error
detecting face in frame ',
num2str(i), ': ', ME.message]);
        continue;
    end

    % Check if any face is
detected
    if isempty(bboxes)
        continue;
    end

    if size(bboxes, 1) > 1 % if
multiple faces are detected
        disp('mult faces')
        bbox_areas = bboxes(:,
3) .* bboxes(:, 4); % compute
areas of all bounding boxes
        [~, idx] =
max(bbox_areas); % get index of
largest face
        bbox = bboxes(idx, :);
% select the largest face
    else % if only one face is
detected
        bbox = bboxes;
    end

    % compute the horizontal
center of face
    x_center = bbox(1) +
bbox(3) / 2;
```

```
    % Define new bounding box
for forehead
    forehead_box = [x_center-
20, bbox(2) + bbox(4)*.45, 30,
30];

    % Extract ROI from forehead
bounding box
    forehead_roi =
imcrop(frame, forehead_box);

    % Convert ROI to red
channel
    if ndims(forehead_roi) == 3
%         roiGray =
rgb2gray(forehead_roi);
%         roiGray =
im2double(roiGray);
        roiRed =
forehead_roi(:,:,1);
        roiRed =
im2double(roiRed);
        roiRed = mean(roiRed);
    else
        disp('ROI is not a 3D
matrix')
        % continue;
    end


    % Apply a bandpass filter
to the pulse signal
    % Define the passband
frequency range (e.g., 0.8-3 Hz
= 40 - 180 bpm)
    fpass = [0.8, 3];
    fs = frameRate;
    % Design a Butterworth
bandpass filter
    [b, a] = butter(2, fpass /
(fs / 2), 'bandpass');
    % Apply the filter to the
pulse signal
    roiFiltered = filtfilt(b,
a, roiRed);

    % Apply windowing to obtain
short stationary segments
    winLen = round(fs * 5 /
60); % Length of window = 5
heartbeats
    overlap = round(winLen /
2); % 50% overlap
    [nSamples, nWindows] =
size(buffer(roiFiltered(1:lengt
h(roiFiltered)), winLen,
overlap, 'nodelay'));

    % Apply FFT to each window
    fftWindows =
fft(buffer(roiFiltered(1:length
(roiFiltered)), winLen,
overlap, 'nodelay'));
%     fftWindows =
fft(roiFiltered);

    % Compute the magnitude
spectrum of each window
    magSpec =
abs(fftWindows(1:round(winLen /
2 + 1), :));
%     magSpec =
abs(fftWindows);

    % Identify the peak
frequency in each window
    [~, loc] = max(magSpec);

     % Compute the
corresponding frequencies
    f = linspace(0, fs / 2,
winLen / 2 + 1);

    % Convert the peak
locations to frequencies
    peakFreqs = f(loc);

    % Compute the mean peak
frequency across all windows
    meanPeakFreq =
mean(peakFreqs);

    % Convert the mean peak
frequency to heart rate in
beats per minute (BPM)
    heartRate = meanPeakFreq *
60;
```

```
    % Store the heart rate and
the current time in their
respective arrays
    heartrates(i) = heartRate;
    time_current(i) = time(i);


    % display ROI for
visualization
    imgWithROI =
insertShape(frame, 'rectangle',
forehead_box, 'LineWidth', 10,
'Color', 'green');
    imgWithBBOX =
insertShape(frame, 'rectangle',
bbox, 'LineWidth', 10, 'Color',
'magenta');


    clf; % clear previous data
    subplot(2,2,1)
    imshow(forehead_roi);
    subplot(2,2,2)
    imshow(imgWithROI)
    subplot(2,2,3)
    imshow(imgWithBBOX)
    subplot(2,2,4)
    % Append the new data to
the plot
    % Set the window size for
the moving average
    windowSize =
max_frames*.10;

    % Calculate the moving
average of the heart rates
    movingAvg =
movmean(heartrates(1:i),
windowSize);

    try

        if(heartRate < 40 &
heartrates(1)>0 & heartrates(i-
1)>0)
            disp('Issue with
ROI. Heartrate of Frame:',
num2str(i), 'is heartrates(i-
1)')
            heartrates(i) =
heartrates(i-1); % there is
currently a bug where the ROI
isn't distinguished properly
resulting in a non-3D matrix,
so the heartrate at the
iteration is set to the
movingAvg(i-1)
            continue;
        else
            heartrates(i) =
heartRate;
        end
    catch
        continue
    end

    disp(['Frame: ',
num2str(i), '  Heart rate: ',
num2str(heartRate), ' bpm']);


    time_current(i) =
video.CurrentTime;
    % Plot the heart rates and
the moving average
    plot(time_current(1:i),
heartrates(1:i), '*');
    hold on;
    plot(time_current(1:i),
movingAvg, 'LineWidth', 2);
    hold on;

plot(time_current(1:i),ground_t
ruth(1:i), 'LineWidth', 2,
'Color', 'black');
    titlestring = ['Heartrate
(bpm) vs. Time (s) (FFT) |
Average (bpm): ',
num2str(mean(heartrates(1:i)))]
;
    title(titlestring);
    xlabel('Time (s)');
    xlim([0 time_current(i)])
    ylabel('Heartrate (bpm)');
```

```
    legend('Heartrate (bpm)',
['Moving Average (',
num2str(round(movingAvg(i))), '
bpm)'], ['Ground truth (',
num2str(ground_truth(1)), '
bpm)']);
    ylim([48 180])
    drawnow;
%     pause(.1);
end

% Compute the mean heart rate
across all frames
meanHeartRate =
mean(heartrates);
perDiff = (abs(meanHeartRate -
ground_truth)/ground_truth)*100
;

timeElapsed = toc;
disp(['Time for execution: ',
num2str(timeElapsed)]);
disp(['Average heart rate: ',
num2str(meanHeartRate)]);
disp(['Percent Difference: ',
num2str(perDiff)]);
```

heartrate_WIP2.m

```
clear
close all
clc

tic
% Ramiface_1,2 were recorded at
79 bpm
% Create CascadeObjectDetector
object
faceDetector =
vision.CascadeObjectDetector();

filename = 'annie_79.mov';

% Read video file
video =
VideoReader(['C:\Users\hannar1\
OneDrive - Wentworth Institute
of Technology\Dsp\',
filename]);
% video =
VideoReader('IMG_4172.mov');
frameRate = video.FrameRate;
max_frames = video.Duration *
frameRate;
max_frames = cast(max_frames,
'uint32');

matches = regexp(filename,
'_\d+\.', 'match');
if ~isempty(matches)
    ground_truth =
str2double(matches{1}(2:end-
1));
else
    % handle case where no
match is found
end

ground_truth =
ground_truth*ones(1,max_frames)
;

time = linspace(0,
video.Duration, max_frames);

heartrates = zeros(1,
floor(max_frames));
time_current =
zeros(1,ceil(video.Duration));
figure % create figure

for i= 1:floor(max_frames)
    frame = readFrame(video);

    % Detect face in the frame
    try
        bboxes =
step(faceDetector, frame);
    catch ME
        warning(['Error
detecting face in frame ',
num2str(i), ': ', ME.message]);
        continue;
    end

    % Check if any face is
detected
    if isempty(bboxes)
        continue;
    end

    if size(bboxes, 1) > 1 % if
multiple faces are detected
        disp('mult faces')
        bbox_areas = bboxes(:,
3) .* bboxes(:, 4); % compute
areas of all bounding boxes
        [~, idx] =
max(bbox_areas); % get index of
largest face
        bbox = bboxes(idx, :);
% select the largest face
    else % if only one face is
detected
        bbox = bboxes;
    end

    % compute the horizontal
center of face
    x_center = bbox(1) +
bbox(3) / 2;

    % define new bounding box
for forehead
```

```
    forehead_box = [x_center-
20, bbox(2) + bbox(4)*.45, 30,
30];

    % extract ROI from forehead
bounding box
    forehead_roi =
imcrop(frame, forehead_box);

    % convert ROI to grayscale
and normalize pixel values to
range [0,1]
    if ndims(forehead_roi) == 3
%         roiGray =
rgb2gray(forehead_roi);
%         roiGray =
im2double(roiGray);
        roiRed =
forehead_roi(:,:,1);
        roiRed =
im2double(roiRed);
        roiGray = mean(roiRed);
    else
        disp('ROI is not a 3D
matrix')
        % continue;
    end

    % apply the BSS algorithm
(e.g., FastICA or SOBI) to
separate the pulse signal
    % from the other sources of
noise and motion artifacts
    [icasig, A, W] =
fastICA(roiGray,min(length(roiG
ray(1)),length(roiGray(2))));
%     [icasig, A, W] =
fastICA(roiGray',2);
%     [icasig] =
fastICA(roiGray,2);



    % select the first
component as the pulse signal
    pulseSignal = icasig(1,:);

    % Apply a bandpass filter
to the pulse signal
    % define the passband
frequency range (e.g., 0.8-3
Hz)
    fpass = [0.8, 3];
    fs = frameRate;
    % design a Butterworth
bandpass filter
    [b,a] =
butter(2,fpass/(fs/2),'bandpass
');
    % apply the filter to the
pulse signal
    pulseSignalFiltered =
filtfilt(b, a, pulseSignal);
     % Apply windowing to
obtain short stationary
segments
    winLen = round(fs * 5 /
60); % Length of window = 5
heartbeats
    overlap = round(winLen /
2); % 50% overlap
    [nSamples, nWindows] =
size(buffer(pulseSignalFiltered
(1:length(pulseSignalFiltered))
, winLen, overlap, 'nodelay'));

    % Apply FFT to each window
    fftWindows =
fft(buffer(pulseSignalFiltered(
1:length(pulseSignalFiltered)),
winLen, overlap, 'nodelay'));
%     fftWindows =
fft(roiFiltered);

    % Compute the magnitude
spectrum of each window
    magSpec =
abs(fftWindows(1:round(winLen /
2 + 1), :));
%     magSpec =
abs(fftWindows);

    % Identify the peak
frequency in each window
    [~, loc] = max(magSpec);
```

```
    % Compute the
corresponding frequencies
    f = linspace(0, fs / 2,
winLen / 2 + 1);

    % Convert the peak
locations to frequencies
    peakFreqs = f(loc);

    % Compute the mean peak
frequency across all windows
    meanPeakFreq =
mean(peakFreqs);

    % Convert the mean peak
frequency to heart rate in
beats per minute (BPM)
    heartRate = meanPeakFreq *
60;

    % Store the heart rate and
the current time in their
respective arrays
    heartrates(i) = heartRate;
    time_current(i) = time(i);

    % display ROI for
visualization
    imgWithROI =
insertShape(frame, 'rectangle',
forehead_box, 'LineWidth', 10,
'Color', 'green');
    imgWithBBOX =
insertShape(frame, 'rectangle',
bbox, 'LineWidth', 10, 'Color',
'magenta');

    % display detected heart
rate
    disp(['Frame: ',
num2str(i), '  Heart rate: ',
num2str(heartRate), ' bpm']);

    clf; % clear previous data
    subplot(2,2,1)
    imshow(forehead_roi);
    subplot(2,2,2)
    imshow(imgWithROI)
    subplot(2,2,3)
    imshow(imgWithBBOX)
    subplot(2,2,4)
    % Append the new data to
the plot
    % Set the window size for
the moving average
    windowSize =
max_frames*.10;

    % Calculate the moving
average of the heart rates
    movingAvg =
movmean(heartrates(1:i),
windowSize);

    try

        if(heartRate < 40 &
heartrates(1)>0 & heartrates(i-
1)>0)
            disp('Issue with
ROI. Heartrate of Frame:',
num2str(i), 'is heartrates(i-
1)')
            heartrates(i) =
heartrates(i-1); % there is
currently a bug where the ROI
isn't distinguished properly
resulting in a non-3D matrix,
so the heartrate at the
iteration is set to the
movingAvg(i-1)
            continue;
        else
            heartrates(i) =
heartRate;
        end
    catch
        continue
    end

    time_current(i) =
video.CurrentTime;
    % Plot the heart rates and
the moving average
    plot(time_current(1:i),
heartrates(1:i), '*');
    hold on;
```

```
    plot(time_current(1:i),
movingAvg, 'LineWidth', 2);
    hold on;

plot(time_current(1:i),ground_t
ruth(1:i), 'LineWidth', 2,
'Color', 'black');
    titlestring = ['Heartrate
(bpm) vs. Time (s) (BSS/ICA) |
Average (bpm): ',
num2str(mean(heartrates(1:i)))]
;
    title(titlestring);
    xlabel('Time (s)');
    xlim([0 time_current(i)])
    ylabel('Heartrate (bpm)');
    legend('Heartrate (bpm)',
['Moving Average (',
num2str(round(movingAvg(i))), '
bpm)'], ['Ground truth (',
num2str(ground_truth(1)), '
bpm)']);
    ylim([48 180])
    drawnow;
%     pause(.1);

end

timeElapsed = toc;
meanHeartRate =
mean(heartrates);
perDiff = (abs(meanHeartRate -
ground_truth)/ground_truth)*100
;

disp(['Time for execution: ',
num2str(timeElapsed)]);
disp(['Average heart rate: ',
num2str(meanHeartRate)]);
disp(['Percent Difference: ',
num2str(perDiff)]);
```