**Abstract:**

Measurement of human body pulse can provide accurate assessment of health conditions. Our basic aim is to measure the pulse rate of a human body with the help of smart phone camera. The pulse rate can be measured with the help of medical instruments but we have tried to use the easily available devices for the same purpose. Previous methods used for measurements were tedious to use and harmful to human body. In addition to this, they are not portable and costly. In this approach, we recorded the video of human figure which was in contact with camera lens. After detailed analysis of this video, we have successfully generated frame wise frequency response of the captured signal which will help us to calculate the pulse rate of the human body. For the development of the project, we have used the MATLAB software which gives step wise implementations.

**Introduction:**

The methods used before for the determination of pulse rate were very costly as various medical instruments were used for the purpose. Also the radiations from these devices can cause hazardous effects on human body. Hence it is advantageous to use the easily available devices which will be used to capture the video and thereafter we can use the concepts in DSP for further implementations.



We are looking for the graph shown above as our final goal which mainly describes the frequency of the signal which can be utilized in determination of the pulse rate. Concepts in DSP are very useful in this process. Fast Fourier Transform is used to obtain the frequency response of the captured signal. The advantage of FFT over DFT is, the calculations required in FFT are much less compared to DFT.

Filters make control over the band which has to be considered in computations.

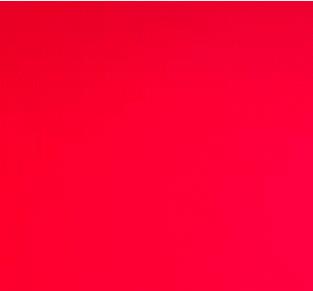
We have used i6 + phone in our project for the demonstration. But any digital camera or smart phone having a camera can be used to work on this project. Thus user does not have to depend on any costly medical equipment for pulse detection. Any layman who has the basic knowledge of handling a mobile device can use this method for generating the pulses. Hence in our opinion, this method which is suggested by MIT people will have huge scope in coming years.

**Computer Simulation:**

For simulation purpose, we have used matlab.

The steps followed for the process are as follows:

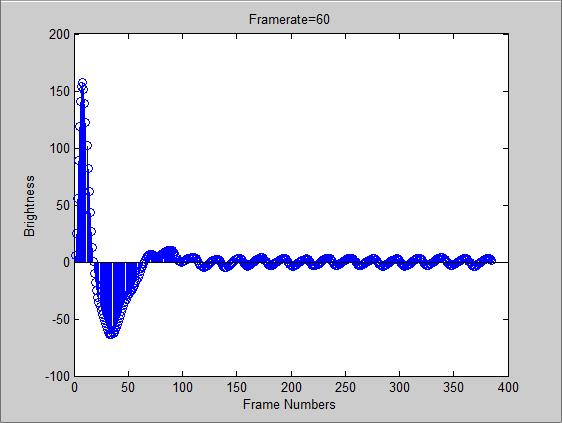
* Video Acquisition:



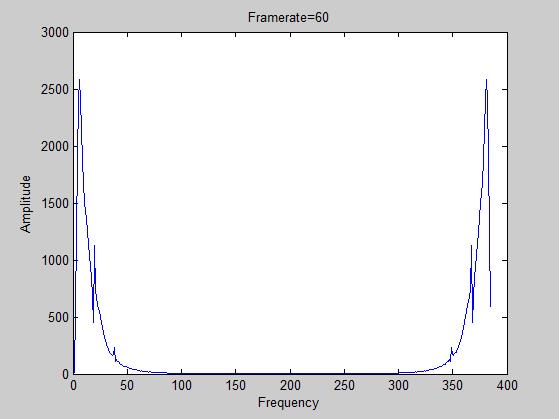
We generated the video, by keeping the finger on the camera and making LED on due to which it gives red coloured output.

* Frame wise brightness calculation:

Generated video is divided in frames and brightness of each frame is calculated by using the formula given in MIT paper.



* Frequency Determination using FFT:

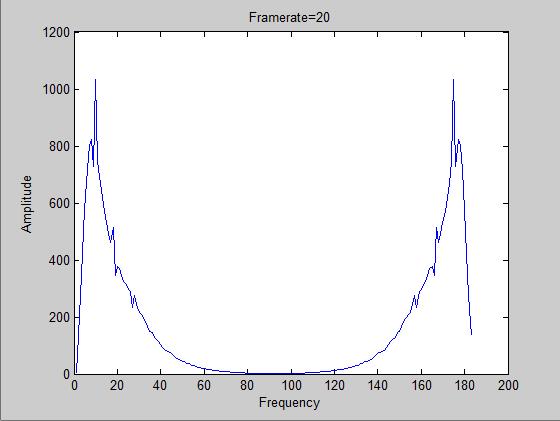
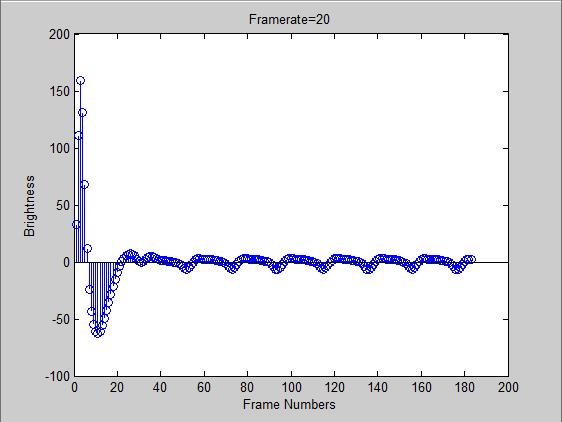


Frequency of the signal is determined by calculating its FFT.

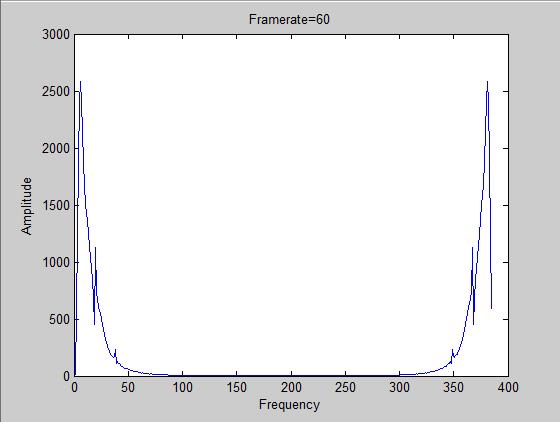
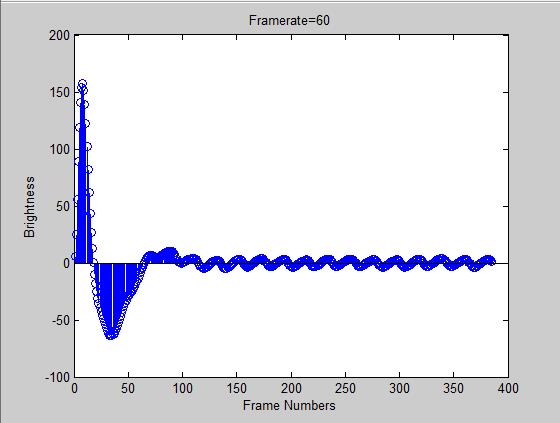
**Performance Evaluations:**

We tried to generate the output under different conditions. We examined the variations in output by changing the frame rate. The variations in brightness values and frequency response is shown below

* When Frame rate is 20fps:

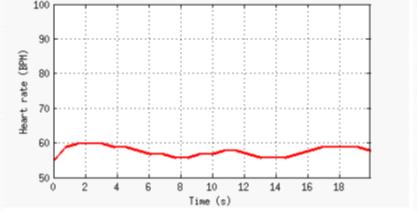


* When Frame rate is 60fps:



The changes in output can be observed from above figures. Bandwidth of the signals increases with increase in frame rate.

The final goal we are looking to achieve is as shown below:



Above result can be obtained once the generated FFT signal goes through smoothing & pulse detection.

**Matlab Code:**

%%%%%%%%%%%%%%%%%%%% Video Display %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

videoFReader = vision.VideoFileReader('C:\Users\User\Desktop\Spring\DSP\Project 1\I6+.MOV');

videoPlayer = vision.VideoPlayer;

while ~isDone(videoFReader)

videoFrame = step(videoFReader);

step(videoPlayer, videoFrame);

pause(0.04);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Brightness Calculation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

brightness=zeros(1,385);

videoPlayer=VideoReader('C:\Users\User\Desktop\Spring\DSP\Project 1\I6+.MOV');

for i = 1:385

frame = read(videoPlayer, i); % frame-by-frame reading

redPlane = frame(:,:,1); % redplane gives intensity information

brightness(i) = sum(sum(redPlane)) / (size(frame, 1) \* size(frame, 2));

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Butterworth Filter %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BPM\_L=40;

BPM\_H=230;

FrameRate=60;

[b, a] = butter(2, [((BPM\_L / 60) / FrameRate \* 2), ((BPM\_H/ 60) / FrameRate \* 2)]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Decision of end parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% fprintf('Values of b are \n');

% disp(b);

% fprintf('Values of a are \n');

% disp(a);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Brightness Filtering %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

filtBrightness = filter(b, a, brightness);

figure(1);

stem(filtBrightness);

xlabel('Frame Numbers');

ylabel('Brightness');

title('Framerate=60');

fftbright=abs(fft2(filtBrightness));

figure(2);

plot(fftbright);

xlabel('Frequency');

ylabel('Amplitude');

title('Framerate=60');