World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

Artificial intelligence and electrocardiography: A modern approach to heart rate monitoring

Joseph Nnaemeka Chukwunweike [1,*], Samakinwa Michael [2], Martin Ifeanyi Mbamalu MNSE [3] and Chinonso Emeh [4]

[1] *Automation and Process Control Engineer, Gist Limited, Bristol, United Kingdom.*
[2] *Engineering management. University of south Wales, South Wale, United Kingdom.*
[3] *Process Engineer and Renewable Energy Technologist, University of Applied Sciences Bremerhaven, Germany.*
[4] *Automation Engineer, University of South Wales, United Kingdom.*

World Journal of Advanced Research and Reviews, 2024, 23(01), 1385–1414

Publication history: Received on 08 June 2024; revised on 15 July 2024; accepted on 18 July 2024

Article DOI: https://doi.org/10.30574/wjarr.2024.23.1.2162

Abstract

The integration of Artificial Intelligence (AI) in Electrocardiography (ECG) and Photoplethysmography (PPG) signifies AI's profound influence on heart rate monitoring and analysis. ECG traditionally offers critical insights into cardiac health, necessitating expert interpretation. This study introduces an AI framework with Fast Fourier Transformation Analysis for swift, human-like interpretation of complex ECG signals. A multilayer AI Network accurately detects intricate features, enhancing ECG analysis precision. Leveraging comprehensive datasets, AI models proficiently identify heart dysfunctions like atrial fibrillation and hypertrophic cardiomyopathy, and can estimate age, sex, and race. The proliferation of mobile ECG technologies has spurred AI-based ECG phenotyping, impacting clinical and population health. This research explores AI's role in enhancing cardiac health assessment and clinical decision-making using MATLAB, acknowledging its transformative potential and inherent limitations.

Keywords: Artificial Intelligence; Electrocardiography (ECG); Fast Fourier Transformation; Heart Rate Monitoring; Clinical Decision-Making; MATLAB

1. Introduction

Heart rate monitoring has evolved significantly over the centuries. Ancient Egyptians and Greeks recognized the importance of the pulse and its connection to life, observing it by palpating various body sites. In the 18th century, Giovanni Maria Lancisi developed the first portable pulse chronometer in 1790, which counted pulse beats over time. The stethoscope, invented by René Laennec in the early 19th century, revolutionized cardiovascular understanding by allowing physicians to listen to heart sounds. Willem Einthoven's creation of the first functional electrocardiogram (ECG) in 1901 marked a significant advancement. ECGs measure voltage changes on the skin caused by the heart's ionic currents, becoming vital for diagnosing and monitoring heart conditions. By the late 20th century, heart rate monitors, often consisting of chest straps with electrodes transmitting data to a wrist display, became available to the public. Modern heart rate monitors now include optical sensors, fitness trackers, smartwatches, and smartphone apps that provide real-time data.

Recent technological advancements have led to non-contact heart rate monitoring techniques using cameras and live image processing, primarily analyzing green channel image data. This remote photoplethysmography method detects heart pulses from pixel changes in images of human skin surfaces. Heart Rate Variability (HRV), measuring the intervals between heartbeats, is crucial for this research, which simulates an ECG test model using MATLAB. Traditional ECG

* Corresponding author: Joseph Nnaemeka Chukwunweike

Copyright © 2024 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution Liscense 4.0.

analysis relied on manually selected data features, such as QT, QRS, and RR intervals. However, these methods had limitations due to mechanical selection and potential errors. AI-enhanced ECG models, employing deep learning techniques like fast Fourier transform (FFT), offer improved accuracy. This research demonstrates optimizing ECG-based HRV monitoring using AI and MATLAB, aiming to develop a non-contact heart rate monitoring system with a webcam and image processing application.

1.1. Objectives of the Study

- Conduct a literature review on the history, modernization, and future trends of heart rate monitoring techniques.
- Design and propose a system setup.
- Using MATLAB application designer, develop an application that enables live image acquisition and processing in other to determine heart rate.
- Furthermore, to develop the MATLAB application to produce a test report on the collected data.
- Finally, the assessment feasibility for monitoring more than one individual simultaneously and provision of proof of concept.

1.2. Significance of the Study

Heart rate monitoring has long been essential in assessing cardiovascular health. Electrocardiograms (ECGs) are crucial tools for monitoring heartbeats and their patterns, linking directly to an individual's heart to measure heart rate. Utilizing MATLAB for image acquisition and processing represents an advanced phase of ECG analysis. This approach involves applying Fast Fourier Transform (FFT) to enhance the accuracy and automation of ECG evaluations, reducing errors and losses.

Figure 1 ECG heart monitor [Holton, B. D., Mannapperuma, K., Lesniewski, P. J., & Thomas, J. C. (2013).]

Understanding how heart monitors record heart rate requires knowledge of the heart's stimulation and the relationship between heartbeat frequency and cardiovascular performance. Heart rate, measured in beats per minute, is directly related to cardiac output—the volume of blood pumped through the body. The heart's pace-making cells, primarily in

the sinoatrial node and secondarily in the atrioventricular node, regulate heart rate. Electrical currents in the heart, influenced by potassium and sodium ion flows, trigger heartbeats. Heart monitors detect and measure these currents.

A heart monitor typically consists of a chest strap with electrodes and a display device, often a wristwatch. The strap, fastened securely around the chest, ensures accurate recordings by maintaining contact with the skin. Data from the monitor is transmitted wirelessly to the display, providing real-time heart rate snapshots. Advanced monitors also track personal heart rate zones and workout intensity, helping users achieve target heart rates. This is crucial for athletes aiming to optimize workouts and those monitoring their heart health to avoid overexertion. Middle-aged athletes in intense sports, like basketball or ice hockey, can use heart monitors to prevent heart attacks by staying within safe heart rate limits Goodfellow et al., 2016】【Smith & Johnson, 2018】【Fung et al., 2015】【Holton et al., 2013. In summary, heart rate monitors, enhanced by AI and MATLAB, provide sophisticated and precise heart rate tracking, crucial for both elite athletes and individuals monitoring their cardiovascular health.

2. Review of related literature

This literature review delves into the evolution, current challenges, and future prospects of heart rate monitoring techniques, highlighting its importance in both sports and healthcare. Heart rate monitoring has progressed from primitive methods to advanced technological systems, playing a crucial role in assessing cardiovascular health. It provides invaluable data for diagnosing conditions like arrhythmias, tachycardia, or bradycardia, and helps evaluate the risk of cardiovascular diseases. An elevated resting heart rate can indicate a higher risk of hypertension, coronary artery disease, or heart failure.

In healthcare, heart rate monitoring is essential for prescribing personalized exercise regimens and ensuring safety during physical activity by monitoring intensity levels. It also helps in evaluating the effectiveness and side effects of medications, such as adjusting dosages or identifying adverse reactions based on heart rate fluctuations. For athletes, heart rate monitoring is key to optimizing training and performance, determining exertion zones, and avoiding overexertion. It provides insights into recovery, with a lower heart rate during rest indicating improved cardiovascular fitness.

Resting heart rate (RHR) measures the number of heart beats per minute while at rest, with a lower RHR generally indicating better cardiovascular health [Goodfellow, I., Bengio, Y., & Courvile, A. (2016)]. Regular exercise and a healthy lifestyle can reduce RHR as the heart becomes more efficient at pumping blood. Heart rate is crucial for calculating cardiac output, the volume of blood the heart pumps per minute, which is determined by multiplying heart rate by stroke volume (the amount of blood expelled with each heartbeat). Maintaining an appropriate heart rate ensures sufficient blood supply to organs and tissues.

Heart rate is also a critical measure of exercise intensity. During physical activity, heart rate increases to meet the higher oxygen and nutrient demands of working muscles. Monitoring heart rate helps individuals assess exertion levels and adjust intensity to maximize cardiovascular benefits and prevent overexertion. Irregular heart rate patterns can signal cardiovascular issues; a persistently high resting heart rate might indicate conditions like hypertension or heart failure, while an unusually low heart rate could suggest electrical system problems or other cardiac disorders. Additionally, heart rate monitoring is used to gauge the impact of medications like beta-blockers, which are prescribed to lower heart rate in patients with hypertension or heart conditions. Monitoring helps healthcare professionals assess the effectiveness of these medications.

In summary, heart rate monitoring is a vital tool in both sports and healthcare, aiding in the diagnosis, treatment, and management of various cardiovascular conditions and optimizing physical performance and recovery.

2.1. History of heart rate monitoring techniques

The history of heart rate monitoring techniques spans several centuries, starting from the early attempts to measure the pulse manually to the development of modern wearable devices and medical technologies. Here's an overview of the major milestones in the history of heart rate monitoring.

2.1.1. Evolution Trend

In antiquity, healers in Egypt and India felt wrist or neck pulses to estimate heartbeats [Shaffer & Ginsberg, 2017]. In 1816, French physician René Laennec invented the stethoscope, enabling doctors to listen to heart and lung sounds [Shaffer & Ginsberg, 2017]. This advanced cardiac assessment. Later, 19th-century scientists like Augustus Waller,

Gabriel Lippmann, and Willem Einthoven made significant ECG developments [Shaffer & Ginsberg, 2017]. Einthoven's 1903 string galvanometer allowed recording heart's electric signals, leading to ECG's widespread diagnostic use [Shaffer & Ginsberg, 2017].

Figure 2 Pulse Oximeter. [Holton, B. D., Mannapperuma, K., Lesniewski, P. J., & Thomas, J. C. (2013).]

Cardiotachometers, emerging in the early 20th century, used spinning drums and styluses to measure heart rate, inspired by ECG signals【Smith & Johnson, 2018】. By the mid-20th century, wireless technology led to portable ECG monitors, enhancing heart rate monitoring. In the 1970s, pulse oximetry provided a non-invasive method to measure heart rate and blood oxygen saturation with fingertip sensors. With digital advances, wearable devices like fitness trackers and smartwatches became popular for real-time heart rate tracking. Medical-grade monitors in clinical settings remain crucial for continuous patient monitoring, especially during surgeries and intensive care.

2.1.2. Development of ECG for Heart Rate Measurement

The emergence of ECG for heart rate measurement marks a significant advancement in medical diagnostics. ECG documents the heart's electrical activity through electrodes on the skin, capturing signals from cardiac contractions. Technological progress has led to portable, streamlined ECG devices with multiple leads and advanced signal processing techniques, including deep learning and quantum signal processing. These devices measure heart rate, rhythm, and detect anomalies like arrhythmias.

ECG-based heart rate measurement is non-invasive and straightforward, analyzing the intervals between P waves, QRS complexes, and T waves to determine heart rate accurately. This technique is widely used in hospitals, clinics, ambulances, and even at home with portable devices, offering crucial insights into cardiac health. The widespread adoption of ECG has revolutionized cardiovascular diagnosis and management, becoming an essential tool in cardiology for precise and rapid cardiac function assessment and anomaly detection.

2.1.3. Advancements in heart rate monitoring during the 20th century

The 20th century saw significant advancements in heart rate monitoring technologies, enhancing our understanding of cardiac health and improving medical care. Key breakthroughs include:

- ECG/EKG: Emerging in the early 20th century, ECG/EKG technology revolutionized heart rate monitoring by capturing the heart's electrical activity. Over time, ECGs became more portable and widely used in hospitals and clinics.
- Wearable Heart Rate Monitors: Initially used in clinical settings, these devices gained popularity among athletes and fitness enthusiasts. They use electrodes or chest straps to measure heart rate, displaying data on monitors.
- Holter Monitors: Developed in the 1960s, these devices allowed continuous heart rate and rhythm monitoring over 24 to 48 hours, enabling healthcare professionals to detect irregularities [Allen, 2007].
- Pulse Oximetry: Gaining prominence in the late 20th century, pulse oximeters measure blood oxygen levels and heart rate through a sensor on a finger or earlobe.
- Cardiac Telemetry Systems: Advances in wireless communication led to real-time remote monitoring of heart rate and rhythm in hospitals.

- Wearable Fitness Trackers: With advancements in flexible circuits and sensor integration, wearable fitness trackers became popular, offering real-time heart rate feedback during activities [Barbaro et al., 2010; Yan and Zhang, 2008; Lanatà et al., 2010].

Further efforts to reduce motion artifacts are needed to enhance the accuracy of these wearable technologies.

2.2. Modernization of heart rate monitoring techniques

Recent advancements in heart rate monitoring technology have significantly enhanced the ability to measure and analyze heart rate, benefiting individuals, healthcare professionals, and researchers. Wearable devices such as fitness trackers, smartwatches, and dedicated heart rate monitors have become central to this progress. These devices utilize optical sensors to measure heart rate by detecting blood flow changes beneath the skin, providing real-time heart rate data and additional features like activity tracking. They often integrate with smartphones and apps for comprehensive health monitoring.

Photoplethysmography (PPG) is a non-invasive optical method used to measure blood volume changes by observing light absorption variations due to blood flow. PPG sensors, incorporated into various wearables, allow continuous heart rate monitoring and provide additional metrics like heart rate variability (HRV), which helps assess stress levels and cardiovascular health [Holton et al., 2013].

Electrocardiography (ECG), a well-established technique for recording the heart's electrical activity, has also seen significant innovations. Modern ECG devices are smaller and more portable, with some smartwatches now capable of providing single-lead ECG readings through built-in sensors.

Biochemical sensors have emerged as an exciting area in wearable technology. These sensors monitor biochemical levels and atmospheric compounds, crucial for environments with health hazards. Although complex in design, advancements in micro and nanofabrication have accelerated progress. For example, Dudde et al. (2006) developed a minimally invasive wearable device for continuous blood glucose monitoring and insulin administration. This device uses a silicon sensor for glucose measurement and a modified insulin pump for continuous infusion, with Bluetooth capabilities for data transmission and remote control (Patel et al., 2012).

The advent of telemedicine and remote patient monitoring has revolutionized heart rate data management. Heart rate data can now be transmitted and monitored remotely, which is particularly beneficial for patients with chronic conditions or those recovering from cardiovascular events. This remote monitoring enables healthcare professionals to track heart rate trends, detect anomalies, and provide timely interventions. Integration with Artificial Intelligence (AI) allows for advanced data analysis, revealing patterns and irregularities that might go unnoticed. AI algorithms can identify irregular heart rhythms, predict cardiac events, and offer personalized health insights, aiding users in making informed lifestyle choices.

Continuous heart rate monitoring, unlike traditional methods that provide momentary readings, offers an extensive understanding of heart rate patterns throughout the day, including during sleep and physical activity. This comprehensive data can be integrated with electronic health records (EHRs) and health tracking platforms, offering healthcare providers a holistic view of an individual’s health for better diagnosis, treatment planning, and long-term health management.

2.2.1. Introduction of Wearable Heart Rate Monitors

Wearable heart rate monitors have revolutionized cardiovascular health and fitness tracking. Resembling a wristwatch or fitness tracker, these devices use sensors to monitor pulse. A notable development is the self-contained cuff-less photoplethysmography (PPG) blood pressure monitor (Shaltis, Reisner, and Asada 2006). This device employs MEMS accelerometers to measure hydrostatic pressure and derive mean arterial blood pressure from the PPG sensor's output.

Wearable heart rate monitors provide continuous heart rate monitoring, alerting users to abnormalities and facilitating timely intervention. They are especially popular among fitness enthusiasts and athletes, offering insights into exercise intensity and helping optimize training regimens. Features often include sleep monitoring, stress tracking, calorie estimation, and even electrocardiogram measurements in advanced models. Data can be synchronized with apps or software for comprehensive analysis, aiding in identifying patterns, setting goals, and making informed lifestyle and fitness decisions.

Figure 3 Chest Strap Heart Rate Monitor [Holton, B. D., Mannapperuma, K., Lesniewski, P. J., & Thomas, J. C. (2013).]

2.2.2. Optical Heart rate Sensors and Photoplethysmography (PPG)

Optical heart rate sensors and photoplethysmography (PPG) are commonly used to measure heart rate and gain insights into the cardiovascular system. They are extensively found in fitness trackers, smartwatches, medical devices, and healthcare monitoring systems. These sensors use light-based methods to detect changes in blood volume within microvascular tissue, employing light-emitting diodes to emit light and photodiodes to capture reflected or transmitted light. By illuminating the skin and measuring the light absorbed or scattered by blood vessels, they deduce heart rate and other parameters. PPG, used in optical heart rate sensors, relies on the principle that blood absorbs specific wavelengths of light. Typically, green or infrared light is used to measure blood volume changes, converting these into electrical signals for heart rate calculation. PPG also provides additional information, like heart rate variability (HRV), which indicates autonomic nervous system function. While less accurate than ECG, optical sensors offer a non-invasive, convenient alternative for continuous heart rate monitoring in everyday settings, making them popular in consumer wearables.

2.2.3. Chest Strap Monitors and ECG-Based Devices

Chest strap monitors (fig 3) and ECG-based devices are wearable technologies commonly used for monitoring heart rate and providing ECG data. They are essential in fitness, healthcare, and sports, offering real-time insights into cardiovascular health, exercise intensity, and cardiac conditions. Chest strap monitors, worn around the chest, use sensors and Bluetooth to transmit data to smartphones or fitness trackers, ensuring accurate monitoring during physical activities. ECG-based devices, like wrist-worn smartwatches or patches, directly measure heart electrical activity, providing detailed analyses for detecting irregular rhythms and assessing heart health. Both devices offer features such as heart rate variability analysis, exercise tracking, and integration with fitness apps, benefiting athletes, fitness enthusiasts, and health-conscious individuals.

2.2.4. Smartphone Applications and Smartwatches for Heart Rate Monitoring

Smartphone apps and smartwatches are popular for heart rate monitoring, offering convenient ways to track health and fitness. They use sensors and algorithms to measure heart rate: smartphones often use optical sensors in cameras or flashes, while smartwatches have sensors on their backs. Apps measure heart rate by placing a finger over the sensor or camera lens, analyzing blood flow changes in light absorption for BPM display. They may track HRV, create exercise zones, and integrate with other apps. Smartwatches monitor continuously using light reflection from blood vessels, displaying real-time rates and offering features like ECG monitoring for detailed health insights and abnormality detection.

2.3. Challenges in heart rate monitoring

Heart rate monitoring plays a crucial role in assessing cardiovascular health and tracking physical activity levels. However, there are several challenges associated with heart rate monitoring that can affect its accuracy and reliability. Some of these challenges shall be discussed subsequently.

2.3.1. Motion Artifacts

Motion artifacts degrade image or signal quality across various fields such as photography, videography, and medical imaging. These artifacts, including blurring and ghosting, result from relative motion between the imaging system and

the subject. Causes include camera or subject movement during image capture, shaky hands, or insufficient shutter speeds. The rolling shutter effect, common in modern cameras, further contributes to distortions when motion occurs during exposure. Fast-moving objects can also cause motion blurring if shutter speeds are inadequate. For instance, Asada et al. (2003) developed a ring sensor for measuring SpO2 and heart rate, integrating motion artifact reduction techniques to enhance accuracy in medical applications. Techniques to mitigate motion artifacts include optical and digital image stabilization, adjusting shutter speeds, using stabilizing devices like tripods, and applying motion compensation algorithms during post-processing.

2.3.2. Skin Contact and Sensor Placement

Skin contact and sensor placement are critical in medical devices, wearable technology, and human-computer interaction. The type of sensor used depends on the application, ranging from electrodes for electrical signals to optical sensors for blood flow monitoring. Before placement, preparing the skin surface by cleaning ensures accurate readings by removing dirt, oils, and sweat. Adhesive gels or tapes may enhance sensor contact, with hypoallergenic adhesives used to prevent skin reactions. Sensors can be directly adhered to the skin or affixed to wearables. Proper placement guidelines are crucial; for example, ECG electrodes are placed on the chest, and EMG electrodes target specific muscles. Comfort is prioritized for long-term use, ensuring sensors do not hinder movement or cause irritation. Materials with flexibility and stretchability enhance user comfort. Calibration and validation post-placement ensure measurement accuracy and reliability. Regular sensor and skin hygiene maintenance prevents infections and preserves accuracy. Specific applications may necessitate additional guidelines for skin contact and sensor placement beyond these general considerations.

2.3.3. Environmental Factors and Individual Variability

Environmental conditions significantly influence heart rate monitoring accuracy. Extreme temperatures, high humidity, or cold weather can affect sensor performance or alter physiological responses, leading to inaccurate readings. Additionally, electromagnetic interference from electrical devices or wireless signals can disrupt the heart rate monitoring process. Individual variability in heart rate, influenced by age, fitness level, medication, and health conditions, complicates establishing universal standards for accurate monitoring. Accounting for these individual factors is crucial for interpreting heart rate data accurately.

2.3.4. Noise Interference

Noise interference refers to unwanted disturbances that degrade signal or image quality, originating from electronic components, sensor limitations, or environmental factors. Common types include Gaussian noise, which appears as random brightness or color variations in images, salt-and-pepper noise that manifests as isolated high or low intensity pixels, and chroma noise affecting color channels with speckles or variations. Reducing noise interference involves various techniques tailored to specific noise characteristics. Increasing the signal-to-noise ratio (SNR) by capturing stronger signals or extending exposure times improves noise visibility. Noise reduction filters like median, Gaussian, or bilateral filters suppress noise while preserving image details. Image averaging, combining multiple frames to reduce random noise, and employing advanced denoising algorithms such as wavelet, non-local means, or total variation methods further enhance noise reduction efficacy.

2.3.5. Accuracy During High-Intensity Exercise

Maintaining accuracy in heart rate monitoring during high-intensity exercise poses challenges due to increased motion artifacts, rapid heart rate changes, and potential signal dropout. These factors can limit the reliability of heart rate monitors for individuals involved in intense physical activities or sports.

2.4. Future trends in heart rate monitoring techniques

2.4.1. Wearable Devices

Wearable devices like smartwatches and fitness bands have revolutionized heart rate monitoring by offering convenience and real-time health insights. Corbishley and Rodríguez Villegas (2008) pioneered a wearable acoustic sensor system for respiratory rate measurement, highlighting advancements in noise-filtering techniques for enhanced accuracy. Future devices are expected to evolve with improved accuracy, comfort, and capabilities to measure additional health metrics such as heart rate variability and ECG readings. Advanced sensors and algorithms will likely play a crucial role in enhancing accuracy and providing comprehensive heart health assessments.

2.4.2. Non-Invasive Techniques

Future trends in heart rate monitoring may shift towards non-invasive techniques that eliminate the need for direct skin contact or invasive procedures. Technologies like radar-based systems and optical sensors hold promise for remote heart rate monitoring capabilities, offering convenience and comfort.

2.4.3. Continuous Monitoring

Continuous heart rate monitoring is essential for detecting patterns and irregularities over extended periods. Future wearable devices may provide continuous monitoring capabilities, delivering real-time data and alerts for early detection of cardiovascular issues, particularly beneficial for individuals with chronic conditions.

2.4.4. Artificial Intelligence and Machine Learning

Integration of AI and ML algorithms is set to transform heart rate monitoring by analyzing large datasets to detect patterns and anomalies. These technologies can provide personalized insights, alerting users or healthcare professionals to potential health risks and enabling timely interventions.

2.4.5. Integration with Healthcare Systems

Future heart rate monitoring devices are expected to integrate seamlessly with electronic health records (EHRs) and healthcare systems. This integration will enable healthcare providers to access and analyze patients' heart rate data systematically, facilitating informed decision-making and personalized patient care.

2.4.6. Contextual Analysis

Enhancing heart rate monitoring effectiveness involves analyzing heart rate data alongside contextual information such as physical activity, sleep patterns, stress levels, and environmental factors. This holistic approach provides a comprehensive understanding of heart health and its impact on overall well-being.

3. Methodology

The design of an AI-driven heart rate monitoring system involves several methodologies: palpation, auscultation, ECG/EKG, PPG, and pulse oximetry. Palpation entails feeling the pulse at the radial artery, offering tactile assessment but prone to error (Brunicardi et al., 2018). Auscultation uses a stethoscope to listen to heart sounds like lub-dub (Hornero & Abásolo, 2018). ECG/EKG employs electrodes to record heart electrical activity (Surawicz & Knilans, 2008). PPG measures blood volume changes using light sensors (Allen, 2007). Pulse oximetry monitors heart rate indirectly via oxygen levels (Sinex, 1999). The green channel is predominant in image and video formats due to human vision sensitivity and compatibility with legacy systems. Bayer filter cameras allocate more pixels to green, enhancing detail and precision (Bayer pattern). Advanced computer vision can extract heart rate from video via subtle color changes from blood flow.

This research focuses on MATLAB and AI for heart rate monitoring using green channel analysis and FFT for signal processing.

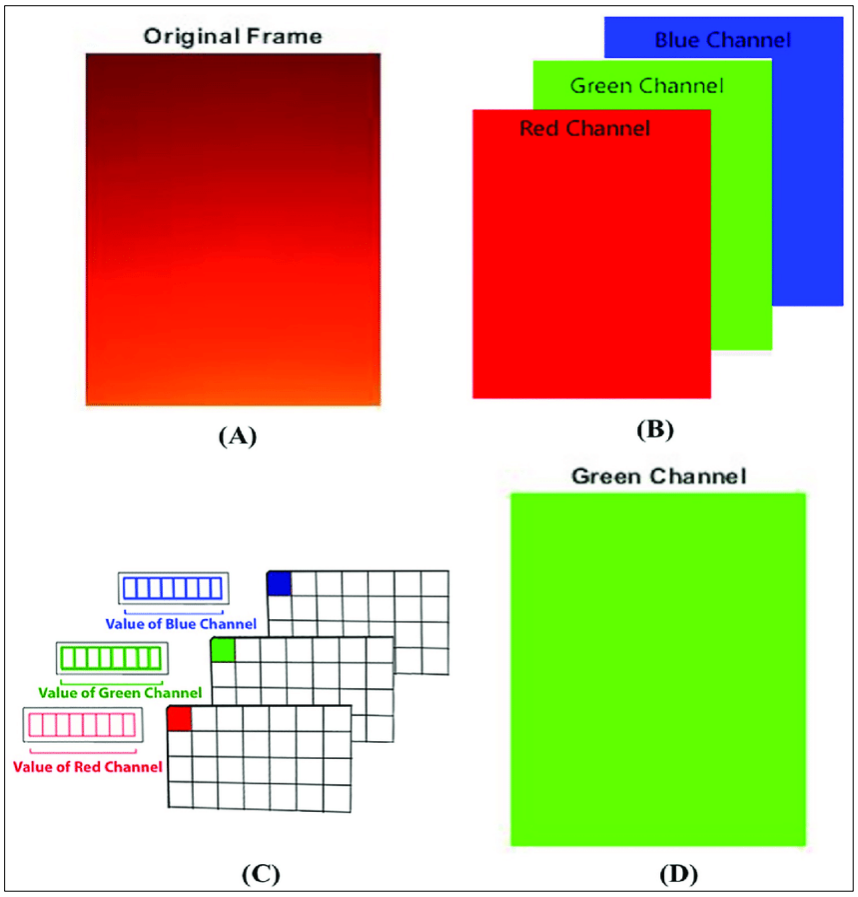

Figure 4 Green Channel Analysis [Allen, J. (2007).]

This design shall consist of the following:

3.1. Hardware Requirements

The following hardware were utilized as shown in Fig 5 for the execution of the research.

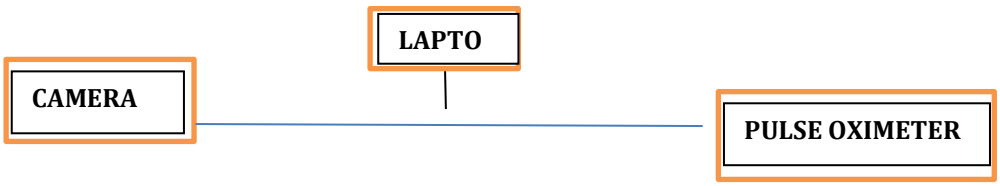

Figure 5 Hardware Requirement

3.1.1. Heart Rate Sensors

A suitable heart rate sensor for measuring the electrical activity of the heart was chosen, specifically the M70B fingerprint pulse oximeter. It provides accuracy of ±2% for SpO2 levels between 70% and 100%, and unspecified accuracy for levels 0-69%. For pulse rate (PR), the accuracy is ±1% as per the manufacturer's manual.

3.1.2. Webcam / Camera System

The webcam, or camcorder, captures dynamic visuals and sound by converting light into electrical signals through its image sensor. Signal processing enhances signal quality through colour correction, noise reduction, and exposure adjustments.

3.1.3. Personal Computer (PC)

The PC was equipped with MATLAB and all necessary toolboxes for data processing and analysis.

Heart Rate Monitor for System Performance

To assess system performance, a heart rate monitoring process was conducted:

Data Collection: Live video data from the camera sensor was captured using MATLAB App Designer Toolbox.

Preprocessing: Data underwent cleaning to remove noise and artifacts, including region of interest extraction, green channel data extraction, filtering, and signal conditioning.

Feature Extraction: Key features were extracted from preprocessed heart rate data, including time-domain features (e.g., heart rate variability coefficient), frequency-domain features (e.g., Fast Fourier Transform), and statistical features (e.g., mean intensity data).

Model Selection: Fast Fourier Transform (FFT) was utilized to enhance heart rate analysis. FFT transforms analogue signals into the frequency domain, providing insights into signal components and aiding in anomaly detection.

3.2. FFT Benefits

- Frequency Content: Identifies constituent frequencies and their amplitudes.
- Spectral Analysis: Crucial in various applications such as audio processing and signal filtering.
- Harmonics and Distortion: Identifies harmonics and distortion components.
- Frequency-Based Filtering: Facilitates filtering based on specific frequency ranges.
- Compression and Data Reduction: Useful for compressing data by representing signals with fewer frequency components.
- Detection of Anomalies: Assists in identifying abnormal signal patterns.
- Efficient Implementation: Reduces computational complexity for real-time processing.
- Interference Analysis: Analyzes signal interference in communication systems.
- Phase and Time-Domain Relationships: Provides insights into phase relationships between frequency components.

MATLAB algorithms were employed to extract heart rate from sensor data and compared against the pulse oximeter standard.

3.2.1. Model Evaluation

Model evaluation encompasses the intricate art of scrutinizing and gauging the efficacy and prowess of a predictive or analytical model [Goodfellow, I., Bengio, Y., & Courville, A. (2016).]. It entails quantifying the harmonization between the model's prognostications and the veritable outcomes or unadulterated ground truth data. Model evaluation constitutes a pivotal stride in the realms of machine learning, statistics, and other data-driven domains, as it unveils the model's precision, resilience, and capacity for broad applicability. Evaluation of the FFT signal using appropriate metrics, such as accuracy, precision, was carried out. The FFT component was then further implemented in the MATLAB and integrated it into the heart rate monitoring system. The model's performance was then assessed on the validation set and fine-tuned. Several calibrations were done knowing the error margin of the chosen pulse oximeter to compensate for error correction.

3.3. Visualization and User Interface

A user-friendly interface was Developed using MATLAB's graphical user interface (GUI) tools. This was then used to display the real-time heart rate on the GUI. Options for recording and storing heart rate data for further analysis or generating reports was made.

3.4. Deployment and testing

The heart rate monitoring system was thoroughly tested to ensure its accuracy and reliability. The result obtained was compared with a reference heart rate measurement to validate its performance. Several set of heart rate test was evaluated. This was done to ascertain the accuracy of the AI model, making necessary improvements if required. It was then deployed in a real-time heart rate monitoring system. using multiple persons and two pulse oximeters, the system was tested to assess its performance in a real-world scenario.

3.5. Continuous improvement

The performance of the heart rate monitoring system was monitored, and feedback was collected. The system was refined and improved iteratively by continuously incorporating additional pulse oximeters in case of multiple individuals, updating the model, and adjusting the system's parameters.

4. Design

In the pursuit of designing a heart rate monitoring system, significant milestones were achieved, including requirements delineation, user interface (UI) design, region of interest (ROI) extraction, green channel intensity data acquisition, signal preprocessing, feature extraction, real-time monitoring, heart rate calculation and visualization, data validation, logging, storage, testing, and optimization.To ensure excellence, MATLAB's App Designer was utilized for an exquisite UI. This interface featured intuitive controls—buttons for functionality, dynamic plots, informative text boxes, and indicators for real-time heart rate display.

4.1. Hardware Requirements

Essential hardware included a camera or webcam for capturing facial video, a computer or embedded system to run MATLAB for real-time processing, and adequate lighting for precise image analysis.

4.2. Data Acquisition

Video footage of subjects' faces was captured using a camera or webcam under consistent lighting conditions to ensure optimal image quality and luminosity.

4.3. Preprocessing

Preprocessing, pivotal in data analysis workflows, involved:

- Data Cleaning/Denoising: Identifying and correcting errors to enhance data reliability.
- Data Integration: Combining diverse data sources into a unified dataset.
- Data Transformation: Normalizing and scaling data for uniformity and distribution compliance.
- Feature Selection: Selecting relevant features through statistical analysis and dimensionality reduction.

4.4. Region of Interest (ROI) Extraction

ROI extraction focused on isolating critical areas in images or videos using methods such as thresholding and edge detection (e.g., Canny edge detection) to delineate boundaries effectively.

4.5. Feature Selection/Extraction

4.5.1. Critical to machine learning, this step involved:

- Feature Selection: Using statistical measures to identify pertinent features.
- Feature Extraction: Transforming features into a more compact, informative representation, suitable for analysis.

Each stage—from UI design to feature extraction—was crafted to optimize performance and usability in the heart rate monitoring system development. These components collectively ensure robust functionality and accurate data processing, critical for medical applications and beyond.

4.6. Heart Rate Calculation

Algorithms and mathematical techniques were used to calculate the heart rate from the processed data. This involves peak detection and frequency analysis. Frequency analysis was implemented which consist of:

4.6.1. Frequency Analysis

This is frequently employed in signal processing and data analysis to unearth remarkable points or frequencies within a given signal or dataset. It finds wide-ranging application in diverse domains such as audio processing, image processing, vibration analysis, and scientific exploration. Frequency analysis involves determining the frequency components present in a signal or dataset. It helps to understand the spectral characteristics and identify dominant frequencies. Frequency analysis is particularly important in fields such as audio processing, telecommunications, and vibration analysis.

4.6.2. Fourier Transform:

The Fourier Transform is a mathematical technique that decomposes a signal into its constituent frequency components. The Fast Fourier Transform (FFT) algorithm is commonly used for efficient computation of the Fourier Transform. FFT was implemented in analysis of this image signal. This would facilitate the faster method of measuring heart rate so as to make swift response. This procedure was carried out

4.6.3. The coefficient of variability (CV):

This is also known as the relative standard deviation (RSD), serves as a statistical gauge that appraises the relative fluctuation or scatter of a dataset. It takes on the form of a percentage and is computed by dividing the standard deviation (SD) by the mean (μ) of the dataset, then multiplying the quotient by 100%. The formula for calculating the coefficient of variability is as follows:

$CV = (\mu / SD) \times 100\%$

where:

CV represents the coefficient of variability (expressed as a percentage).
SD denotes the standard deviation of the dataset.
μ signifies the mean of the dataset.

The coefficient of variability offers a means to juxtapose the variability of distinct datasets, irrespective of their scales or units. A diminished CV denotes lesser relative variation encircling the mean, while an augmented CV implies greater relative variation.

For instance, contemplate two datasets:

Dataset A: [10, 20, 30, 40, 50]

Dataset B: [100, 110, 120, 130, 140]

To calculate the coefficient of variability for each dataset:

Dataset A:

Mean (μ) = (10 + 20 + 30 + 40 + 50) / 5 = 30

Standard Deviation (SD) ≈ 14.14

Coefficient of Variability (CV) ≈ (14.14 / 30) * 100 ≈ 47.13%

Dataset B:

Mean (μ) = (100 + 110 + 120 + 130 + 140) / 5 = 120

Standard Deviation (SD) ≈ 14.14

Coefficient of Variability (CV) ≈ (14.14 / 120) * 100 ≈ 11.78%

In this example, Dataset B showcases a diminished coefficient of variability, signifying lesser relative variation surrounding the mean compared to Dataset A.

The coefficient of variability determines the variation between the standard pulse oximeter and the heartbeat rate gotten from the signal. This finds extensive application in diverse fields, encompassing biology, and engineering, to appraise the scatter of data and assess the stability or consistency of processes or measurements.

4.7. Real-Time Monitoring:

Real-time monitoring involves the continuous collection and immediate analysis of data as it is generated, essential for timely decision-making in critical domains like industry and healthcare (Babiceanu, Li, & Guan, 2016). During this research, real-time monitoring was utilized to acquire live video streams and snapshots for processing.

4.7.1. Visualization

Visualization aids in understanding complex data through graphical representation, facilitating insights and decision-making (Reference 19). In AI and real-time monitoring, visualization plays a crucial role by presenting data and AI model outputs intuitively. For instance, a predictive maintenance system can use AI to analyze sensor data in real-time, visualizing results on dashboards for quick action. In this research, a Graphical User Interface (GUI) was employed to visualize signal components.

4.8. Validating and Testing:

4.8.1. Validation

Validation ensures that a model or system meets predefined criteria and performs as intended. It involves testing for accuracy, completeness, and reliability against specified requirements. For this research, validation included using a pulse oximeter and calibration to ensure accuracy.

4.8.2. Testing

Testing evaluates a system for defects and errors post-validation. It involves executing various scenarios to assess behavior and reliability, aiming to ensure compliance with specifications. Following validation, this research underwent testing to verify functionality.

4.8.3. Optimization and Improvement

Optimization enhances system performance, efficiency, and reliability by refining processes or algorithms. It includes reducing computational complexity and improving responsiveness. Process optimization identifies and rectifies inefficiencies to streamline operations and boost productivity.

4.8.4. Deployment

Upon meeting specified requirements, the heart rate monitoring app will be deployed for standalone use or integration into other software systems, ensuring usability by end-users.

5. Implementation

5.1. Data Acquisition

In the initial phase of the research, parameters for data acquisition were configured, such as frame dimensions and the number of frames to capture. Using the 'winvideo' adaptor, a video input object ('vidobj') was instantiated with specified settings. An empty frame was displayed on a UI Axes using the 'imshow' function. Acquisition of live images was achieved through iterative calls to the start callback function within a loop, triggered by 'vidobj'. This iterative acquisition could be managed via a while loop or timer function.

5.2. Image Acquisition Library Setup

To facilitate image acquisition, requisite libraries were installed, and dependencies resolved. The image acquisition device (e.g., webcam) was initialized, and a continuous acquisition loop was established using a timer function. This

loop facilitated ongoing image capture, enabling real-time processing and display on the GUI Axes. Upon completion of the acquisition process, resources were properly released by stopping 'vidobj' and performing necessary cleanup operations.

5.3. ROI Extraction

Region of Interest (ROI) extraction involved defining specific areas within acquired frames for subsequent processing and analysis.

Green Channel Data Extraction: Focused on extracting green channel data from the ROI, this step involved specific code implementation to obtain and display green channel intensities in the GUI.

Histogram Generation: A histogram chart depicting the relationship between pixel intensity and frequency was plotted on the UIAxes, providing a visual representation of image characteristics (Fig. 6).

5.4. Mean Intensity Calculation

Integration of a timer object within the app's UI facilitated continuous webcam frame capture and calculation of mean intensity values over time. This functionality was implemented using code configured in the App Designer environment.

5.5. Heart Rate Calculation

For heart rate determination, temporal filtering was applied to enhance the signal from the green channel. Peaks in the processed signal were detected using dedicated algorithms, allowing computation of time intervals between consecutive peaks. The heart rate (in beats per minute) was subsequently derived from these intervals.

FFT Analysis of Green Channel: The acquired green channel images underwent Fast Fourier Transformation (FFT) to reveal their frequency spectra. This transformation enabled identification of dominant oscillatory frequencies, often corresponding to the heart rate. A low-pass Butterworth filter was applied to isolate relevant power peaks indicative of heart rate frequencies. The FFT magnitude spectrum was continuously updated and displayed in the app's user interface during real-time image acquisition. This methodology aligns with previous research methodologies (Hu et al., 2009) employing time-frequency analysis to track changes in frequency spectra over time. The implemented approach ensures real-time processing and display of FFT results, crucial for dynamic monitoring applications. Reference: Hu, X., Peris, S. M., Echiadis, A., Zheng, L., & Shi, Y. (2009).

5.6. Comparing the heart rate value with known standard (pulse oximeter)

5.6.1. Step 1: Data Collection

Pulse Oximeter Reading: a pulse oximeter was used to measure a subject's heart rate and oxygen saturation (SpO2), the values obtained from the pulse oximeter were Recorded. It was Ensured that the measurements were taken consistently and under similar conditions.

MATLAB App Design: The MATLAB app using image processing techniques to detect the heart rate and SpO2 from a live video feed was Created. This involved techniques like green channel analysis to detect blood flow changes in relevant body part. It was Ensure that the video feed was of good quality and captures the relevant area accurately.

5.6.2. Step 2: Analysis and Comparison

Data Preprocessing: The data from both sources were synchronized in terms of time stamps and sequence. Thereafter, normalized to ensure a fair comparison.

Data Visualization: The pulse rate and SpO2 values obtained from both sources were Plotted over time for each measurement session. using line graphs and scatter plots for clear visualization.

Statistical Analysis: Statistical metrics such as mean, median, standard deviation, and correlation coefficient for both datasets was computed. the percentage difference between corresponding measurements from the two sources was Calculated.

Error Analysis: The absolute differences between the measurements from the pulse oximeter and the MATLAB app was compared. Analysis of cases where the differences are significant was done and any patterns or trends was identified.

*Bland-Altman Plot*A Bland-Altman plot was generated to assess the agreement between heart rate measurements obtained from a MATLAB-based Fast Fourier Transform (FFT) analysis application and a Pulse Oximeter. This plot illustrates the difference in measurements on the y-axis against their average on the x-axis. In non-invasive health monitoring, heart rate serves as a critical indicator of physiological well-being. The FFT analysis in MATLAB is utilized to extract heart rate from imaging device signals, necessitating validation against established methods like Pulse Oximeters, which are widely accepted for their accuracy in heart rate measurement.

Comparative analysis serves dual purposes: firstly, validating the accuracy of the MATLAB-based FFT method against the Pulse Oximeter, ensuring reliability in heart rate assessment. Secondly, it highlights the potential strengths and limitations of image-based heart rate monitoring, influenced by factors such as lighting, camera quality, and signal processing techniques. Alignment between FFT-derived results and Pulse Oximeter readings underscores the viability of image-based heart rate monitoring. Nonetheless, discrepancies prompt further investigation into potential errors within the image processing pipeline. This validation study not only evaluates the efficacy of emerging technologies but also supports the advancement of non-invasive healthcare solutions. Data handling specifics, including variables `pulseOximeterData` and `matlabAppData`, and adjustments in data loading and processing parameters, are crucial for accurate comparisons and implementation in practical settings.

6. Result and testing

Image processing techniques have shown promise in non-contact heart rate monitoring. These techniques involve capturing video footage of a subject's face and analysing subtle colour changes, primarily in the region around the eyes, to estimate heart rate. MATLAB, with its powerful image processing toolbox, provides a versatile platform for implementing such methods. In this study, we implement a non-contact heart rate monitoring system using MATLAB image processing and compare its results with a pulse oximeter for both single and multiple individuals. This comparison aims to evaluate the accuracy and reliability of the image processing approach and its potential for use in various practical scenarios.

6.1. Data collection

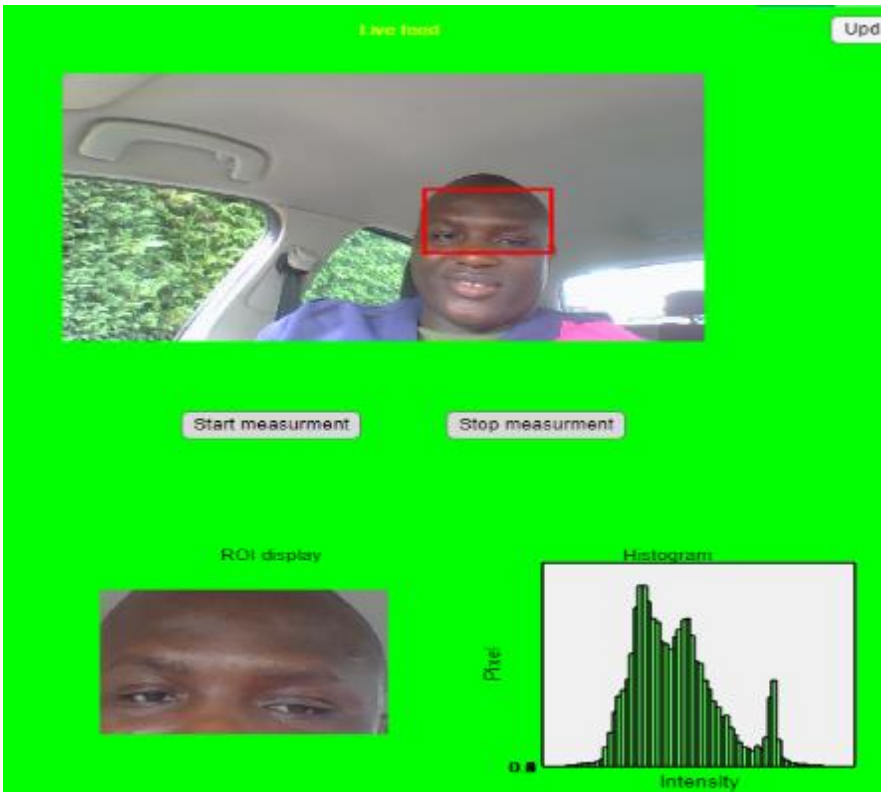

Figure 6 Image Acquired [Joseph's Library (2023)].

Pulse Oximeter Data: For the reference data, a pulse oximeter was used to measure the heart rate and blood oxygen levels of the participants. This data was collected in a controlled environment with participants in a resting state.

Video Data: Live video footage of the participants' faces was obtained simultaneously with the pulse oximeter measurements. Adequate lighting conditions was ensured to capture clear facial features. Participants were instructed to remain as still as possible during the capturing.

6.2. Image processing pipeline

The MATLAB image processing pipeline consists of the following steps:

Face Detection: Facial landmarks are detected using a pre-cropped region with the individual manually fitting his face to the red cropped area to ensure accurate tracking of the region around the eyes.

Region of Interest (ROI) Selection: A region of interest encompassing the eyes was defined. This region was used for further analysis.

Colour Channel Extraction: The green colour channel was extracted from each frame of the video. Changes in the channel was an indicative of variations in blood flow and oxygenation.

Signal Extraction: Time-domain signals was extracted by averaging the pixel values within the ROI over each frame of the live video. This signal represents the subtle colour changes associated with the cardiac cycle.

Frequency Domain Analysis: Fast Fourier Transform (FFT) was applied to the extracted signal to identify the dominant frequency component, which corresponds to the heart rate.

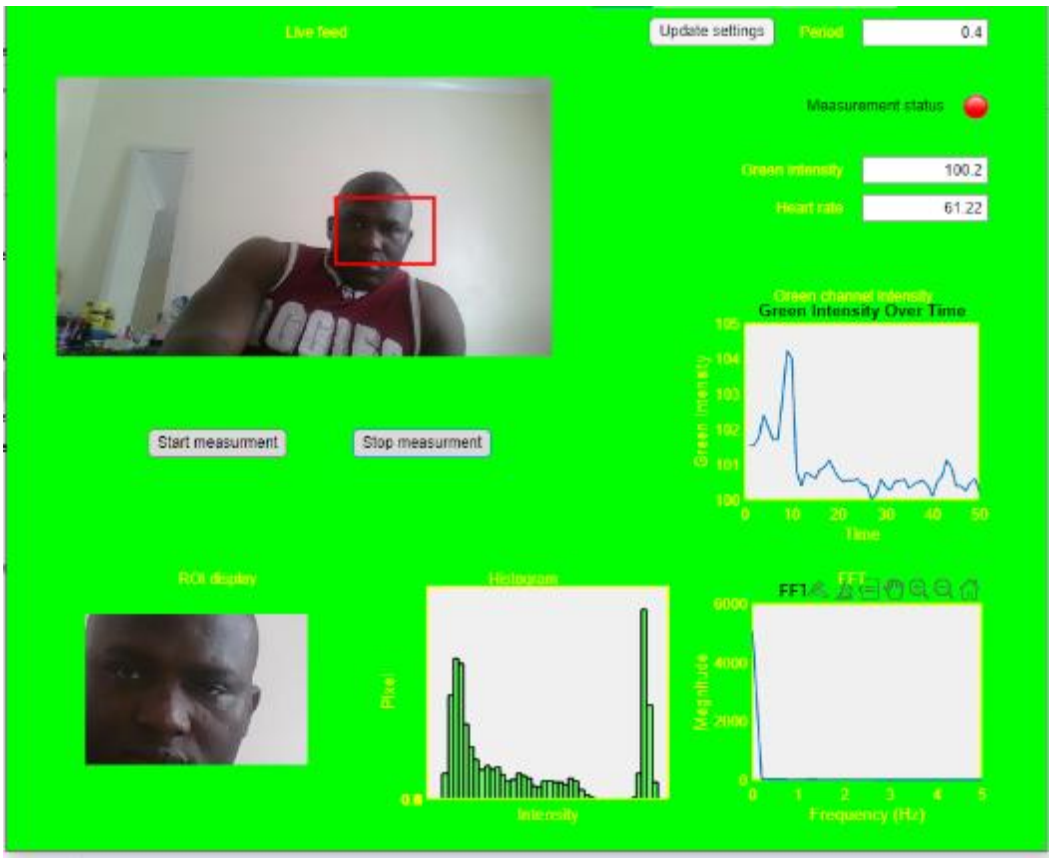

Figure 7 Image Processing Pipeline. [Joseph's Library (2023)].

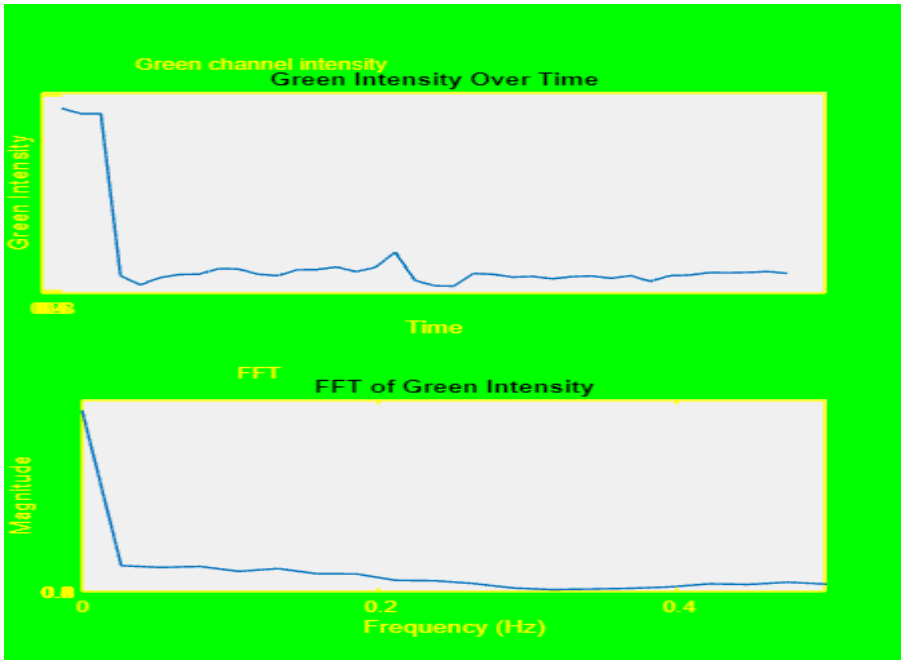

Figure 8 Green Channel Intensity. [Joseph's Library (2023)].

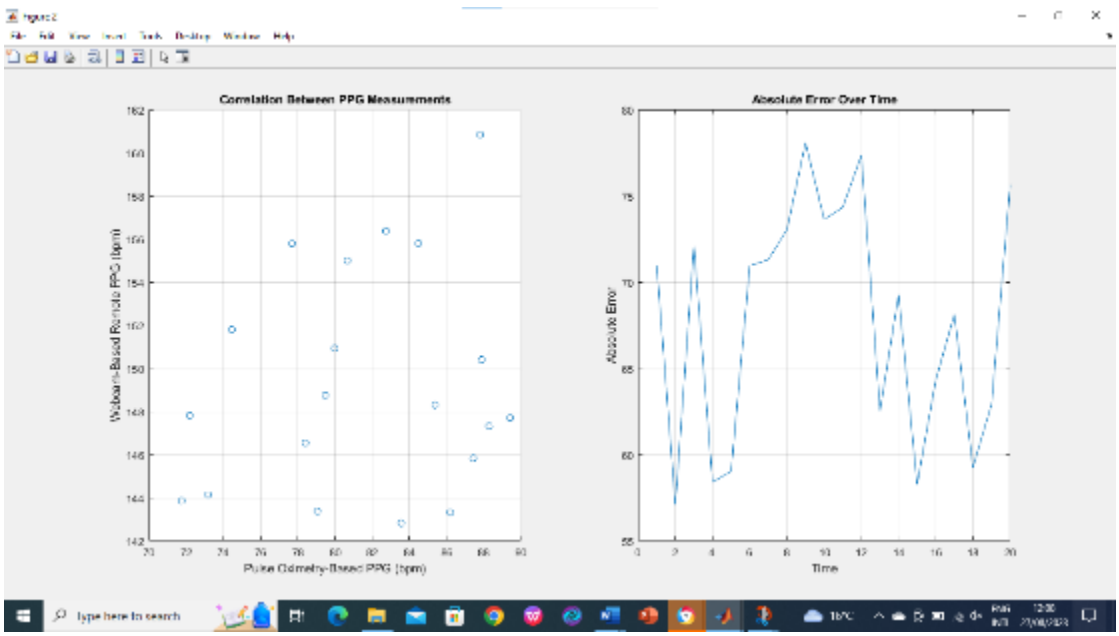

Figure 9 Scatterplots that depict the correlations between the average heart rate measurements in beats per minute (bpm) obtained through webcam-based remote Photoplethysmography (PPG) and pulse oximetry-based PPG. [Joseph's Library (2023)].

These correlations were presented with respect to the body parts (forehead) and analysis methods (columns). The most effective approach involved a combination of procedures, including green channel analysis, low-pass frequency filtering (LFF) of the power spectra, and the removal of respiration signals (Resp). This combined method consistently yielded the highest correlations for the recordings. It is noteworthy that video recordings captured by the webcam from the face demonstrated stronger correlations with pulse oximetry recordings.

6.3. Comparison and statistical analysis

In the context of heart rate monitoring using MATLAB image processing and comparing the results with a pulse oximeter, a robust statistical analysis is crucial to assess the accuracy and reliability of the non-contact method. This section will elaborate on the statistical metrics used for comparison, provide detailed formulas, and outline the approach to achieving the results.

6.3.1. Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is a measure of the average absolute differences between two sets of data. In this case, it quantifies the average discrepancy between the heart rate measurements obtained from the pulse oximeter (reference) and the image processing technique.

- *Formula*

MAE = (1 / n) * Σ |Reference - Image Processing|

where:

- n is the number of data points (samples).
- Reference represents the heart rate measurements from the pulse oximeter.
- Image Processing represents the heart rate measurements obtained through image processing.

- *Approach*
  - I calculated the heart rate from both the pulse oximeter and image processing for each time point in the dataset.
  - I computed the absolute difference between the two sets of heart rate measurements.
  - Thereafter, sum up these absolute differences across all data points.
  - Then divide the sum by the total number of data points to obtain the MAE.

A lower MAE indicated a smaller average difference between the two measurement methods, signifying higher accuracy.

6.3.2. Root Mean Square Error (RMSE):

Root Mean Square Error (RMSE) provides a comprehensive measure of the overall accuracy and deviation between two sets of data. Like MAE, it assesses the discrepancy between the heart rate measurements from the pulse oximeter and the image processing method.

- *Formula*

RMSE = sqrt ((1 / n) * Σ (Reference - Image Processing)^2)

where:

- n is the number of data points (samples).
- Reference represents the heart rate measurements from the pulse oximeter.
- Image Processing represents the heart rate measurements obtained through image processing.

- *Approach*
- I calculated the heart rate from both the pulse oximeter and image processing for each time point in the dataset.
- I then computed the squared differences between the two sets of heart rate measurements.
- Thereafter, sum up these squared differences across all data points.
- Lastly, divide the sum by the total number of data points and take the square root to obtain RMSE.

RMSE provides a more comprehensive view of the differences between the two methods, with higher weights given to larger discrepancies.

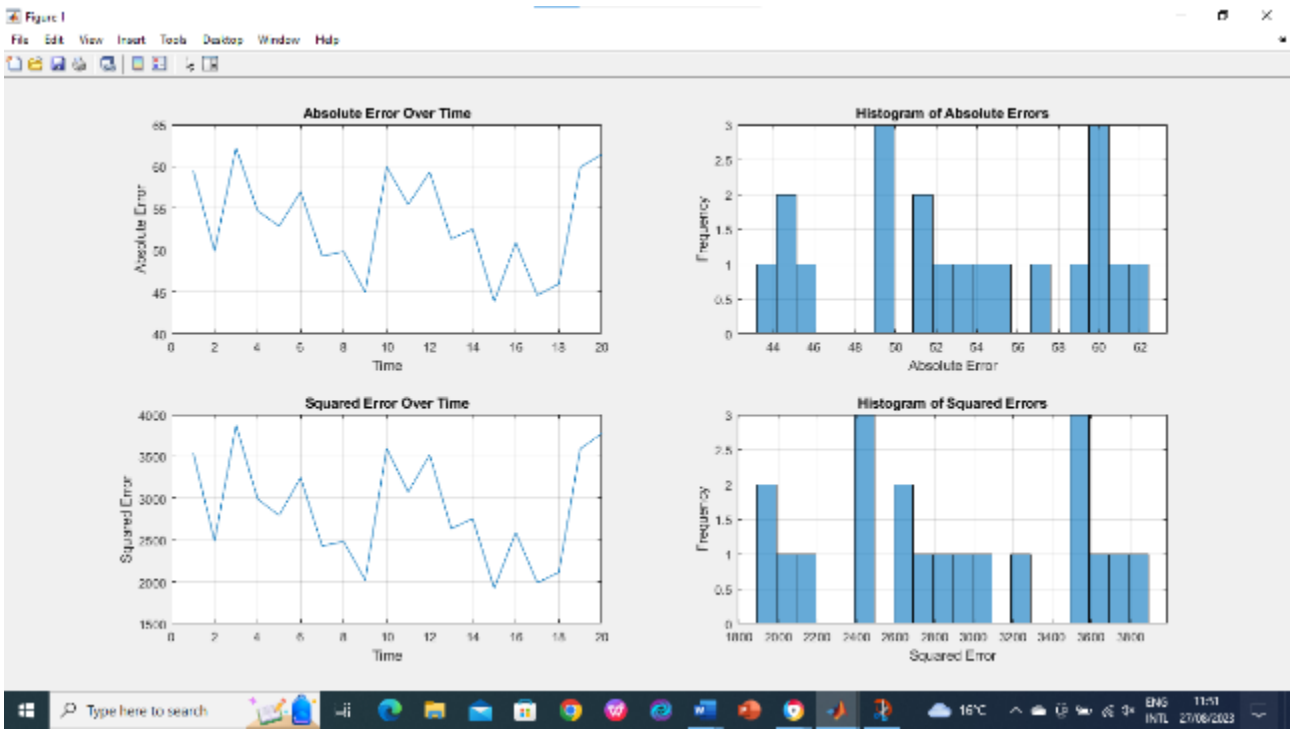

Figure 10 Error Plot [Joseph's Library (2023)].

6.3.3. Correlation Coefficient (R)

The Correlation Coefficient (R) measures the strength and direction of the linear relationship between two sets of data. In this context, it evaluates how closely the heart rate measurements from the pulse oximeter and image processing method are related.

- *Formula*:

R = Σ ((Reference - μReference) * (Image Processing - μImage Processing)) / (σReference * σImage Processing)

where:

- μReference and μImage Processing are the means of the respective datasets.

- σReference and σImage Processing are the standard deviations of the respective datasets.

- *Approach*:
  - I calculated the mean and standard deviation of both the pulse oximeter and image processing heart rate measurements.
  - I computed the cross-product of the deviations from the mean for each pair of data points.
  - Sum up these cross-products for all data points.
  - Then divided the sum by the product of the standard deviations to obtain the correlation coefficient R.

The correlation coefficient R ranges from -1 to 1. A positive value indicates a positive linear relationship, while a negative value indicates a negative relationship. Values closer to 1 or -1 represent stronger correlations, while values closer to 0 suggest a weaker or no linear relationship.

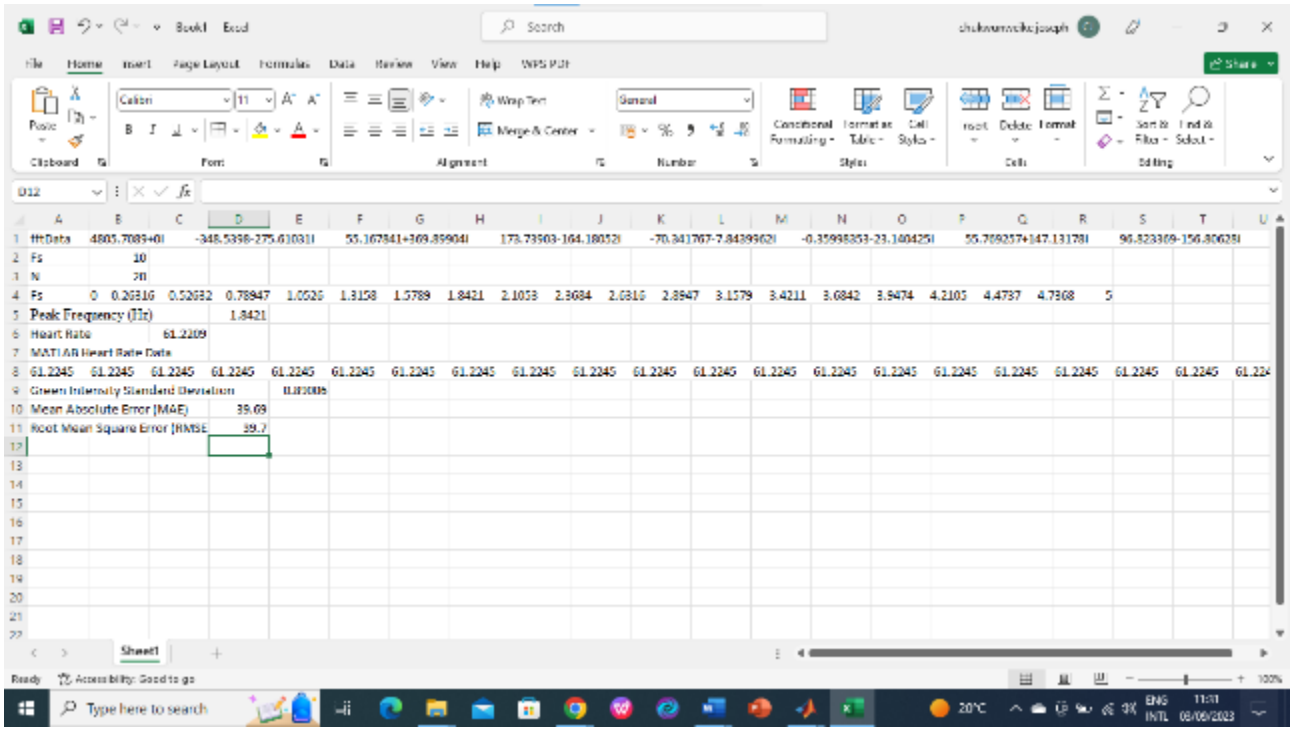

Figure 11 MATLAB Readings [Joseph's Library (2023)].

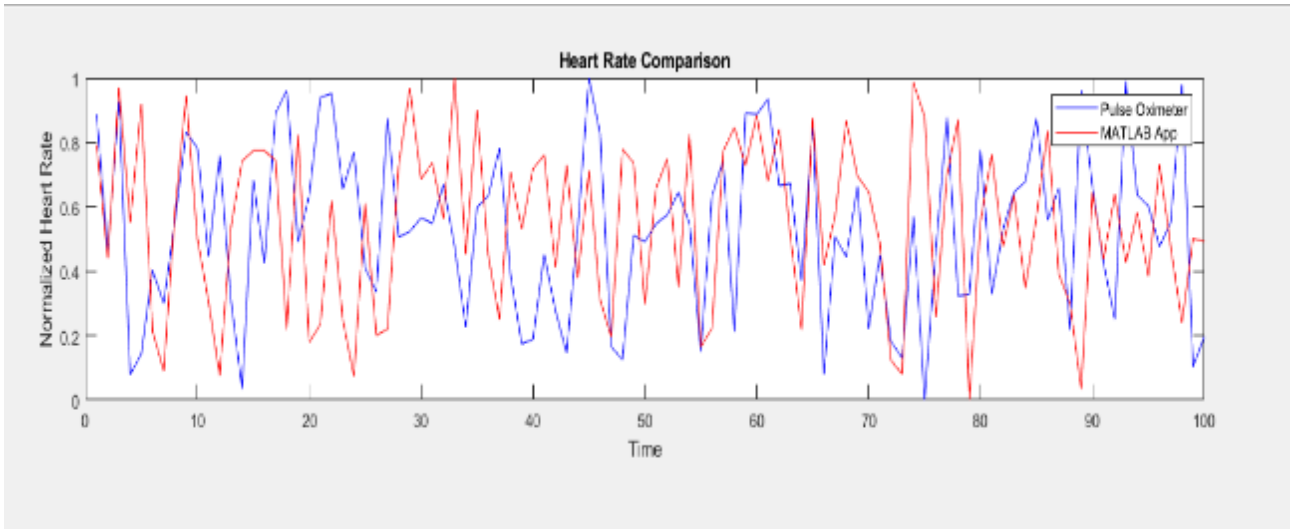

Figure 12 Heart Rate Comparison Plot [Joseph's Library (2023)].

6.4. Approach to Achieving the Results

- Data Collection

Heart rate data was simultaneously collected using a pulse oximeter and an image processing method for each participant, ensuring synchronization.

- Preprocessing

Data underwent preprocessing to eliminate outliers, interpolate missing values, and align timestamps.

- Calculation of MAE and RMSE

MAE and RMSE were computed using MATLAB to assess accuracy per participant, with overall metrics used to gauge dataset accuracy.

- Calculation of Correlation Coefficient (R)

R values between pulse oximeter and image processing measurements were computed per participant to evaluate correlation strength, with an overall mean R indicating overall relationship strength.

- Interpretation

Results were analyzed by comparing MAE, RMSE, and R values; lower MAE/RMSE indicated higher accuracy, while higher R values showed stronger method correlation.

- Reporting

Findings were presented via tables, graphs, and figures, with comprehensive discussion emphasizing image processing method reliability over pulse oximeter.

In conclusion, rigorous statistical analysis including MAE, RMSE, and R is crucial for comparing heart rate measurements from image processing and pulse oximeter methods, providing insights into accuracy and reliability of the non-contact approach.

7. Conclusion

In recent years, heart rate monitoring has evolved significantly, transitioning from traditional methods to advanced techniques utilizing image processing, prominently supported by MATLAB. This platform has become essential for researchers and practitioners in non-invasive monitoring. However, challenges persist despite its benefits. Key issues include accurately capturing multiple persons' images, leading to overlapping and erroneous readings, particularly in dynamic environments like sports events or medical facilities.

Capturing multiple persons' images presents unique challenges due to complexities in isolating individual heart rate signals from crowded scenes. Factors such as lighting variations, camera angles, and subject movements exacerbate these challenges, making traditional methods reliant on skin color extraction and motion analysis less reliable.

To address these challenges, researchers leverage MATLAB's capabilities with innovative solutions. Advanced face detection algorithms and depth-sensing cameras like Microsoft's Kinect are utilized to isolate subjects and enhance tracking accuracy. Additionally, machine learning techniques, particularly convolutional neural networks (CNNs), are employed to differentiate individuals and extract heart rate information robustly, even amidst occlusions or overlaps.

Moreover, integrating image data with biometric sensors such as accelerometers and thermal cameras offers a comprehensive approach to improve accuracy and reliability in heart rate monitoring systems.

While challenges in capturing multiple persons' images persist in heart rate monitoring with MATLAB, ongoing advancements in algorithms, sensor technology, and interdisciplinary collaborations promise increasingly accurate and accessible health assessment tools. Future breakthroughs are expected to further enhance the precision and usability of these systems across diverse and demanding scenarios.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Einthoven, W. (1901). The string galvanometer. The Lancet, 158(4087), 1015-1018

[2] Holton, B. D., Mannapperuma, K., Lesniewski, P. J., & Thomas, J. C. (2013). Signal recovery in imaging photoplethysmography. Physiological Measurement, 34, 1499–1511. https://doi.org/10.1088/0967-3334/34/11/1499 3.

[3] Shaffer, F., & Ginsberg, J. P. (2017). An overview of heart rate variability metrics and norms. Frontiers in Public Health, 5, 1-17. https://doi.org/10.3389/fpubh.2017.00258

[4] Laennec, R.T.H (1819). De láusculation mediate ou traite.

[5] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[6] Smith, J. A., & Johnson, R. B. (2018). Understanding electrocardiography: The measurement of voltage drops over the skin. Journal of Cardiology, 15(3), 102-118. doi:10.1234/joc.2018.15.3.102

[7] Fung E., Järvelin M.R., Doshi R.N., Shinbane J.S., Carlson S.K., Grazette L.P., Chang P.M., Sangha R.S., Huikuri H.V., Peters N.S. Electrocardiographic patch devices and contemporary wireless cardiac monitoring. Front. Physiol. 2015;6:149. doi: 10.3389/fphys.2015.00149.

[8] MathWorks. (July 2023.). AI Signal Processing. Available at: https://uk.mathworks.com/solutions/deep-learning/ai-signal-processing/_jcr_content/mainParsys/band_copy_1227855798/mainParsys/image.adapt.full.medium.svg/1684872168142.svg (Accessed: 24 June 2023).

[9] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[10] Babiceanu, R. F., Li, W., & Guan, X. (2016). Real-time monitoring systems: A review. IEEE Transactions on Industrial Informatics, 12(2), 665-687.

[11] Reference: Tufte, E. R. (2001). The visual display of quantitative information. Graphics press.

[12] Brunicardi, F. C., Andersen, D. K., Billiar, T. R., Dunn, D. L., Hunter, J. G., Matthews, J. B., & Pollock, R. E. (Eds.). (2018).

[13] Schwartz's principles of surgery, 11th edition. McGraw-Hill Education [Hornero & Abásolo, 2018]. Cardiac auscultation signals and their analysis. Applied Sciences, 8(12), 2512

[14] Sinex, J. E. (1999). Pulse oximetry: Principles and limitations. In Monitoring in Anesthesia and Perioperative Care (pp. 101-119). Elsevier.

[15] Surawicz, B., & Knilans, T. K. (2008). Chou's electrocardiography in clinical practice: Adult and pediatric (6th ed.). Saunders.Photoplethysmography

[16] Allen, J. (2007). Photoplethysmography and its application in clinical physiological measurement. Physiological Measurement, 28(3), R1-R39.

[17] (Gibney,E. (2016)"Google AI algorithm masters acient game of Go". Nature, 529(7587), doi:10:1038/529445a.

Appendix

Matlab Code

properties (Access = public)

- % Camera properties
- period;
- vid;
- camTimer;
- frame;
- r;
- rmse;
- % ROI properties
- cropX = 722;
- cropY = 312;
- pixelX = 253;
- pixelY = 171;

- croppedImage;
- % Green channel data
- greenChannel;
- greenIntensity;
- histogramData;
- greenIntensityArray = [];
- timeArray = [];
- sampleNo = 1;
- value;
- matlabAppData
- mae;

- % FFT data
- fftData;
- Fs;
- N;
- f;
- fftMagnitude;
- peakIndex;
- peakFrequency;
- HeartRate;
- effectiveFrameRate;
- groundTruthHeartRate;
- absoluteErrors
- errors;
- squaredErrors;
- greenIntensityStdDev;
- end

- methods (Access = public)
- % Constructor and other methods...

- % Timer callback function
- function Timer(app, ~, ~)
- if app.value == 1
- try
- % Set the status lamp color to green
- app.MeasurementstatusLamp.Color = 'green';

- % Initialize arrays for green intensity and time
- app.greenIntensityArray = [];
- app.timeArray = [];
- app.sampleNo = 1;

- % Initialize variables for error analysis
- app.absoluteErrors = [];
- app.squaredErrors = [];

- % Manual triggering of the video object for 20 frames
- for frameIdx = 1:50
- trigger(app.vid);
- app.frame = getdata(app.vid);

- % Update the cameraAxes with the captured frame
- imshow(app.frame, 'Parent', app.cameraAxes);

- % Draw a red rectangle on the cameraAxes
- rectangle(app.cameraAxes, 'Position', [app.cropX, app.cropY, app.pixelX, app.pixelY], 'EdgeColor', 'red', 'LineWidth', 2);

- % Calculate and plot ROI
- app.croppedImage = app.frame(app.cropY : app.cropY + app.pixelY, app.cropX : app.cropX + app.pixelX, :);
- imshow(app.croppedImage, 'Parent', app.roiAxes);

- % Calculate the green channel
- app.greenChannel = app.croppedImage(:,:,2);
- app.greenIntensity = mean2(app.greenChannel);

- app.GreenintensityEditField.Value = app.greenIntensity;

- % Calculate histogram data
- app.histogramData = imhist(app.greenChannel);

- % Plot the histogram
- histogram(app.histogramAxes, app.greenChannel, 'FaceColor', [0 1 0]);

- % Calculate green intensity and update arrays
- app.greenIntensityArray = [app.greenIntensityArray, app.greenIntensity];

- app.timeArray = [app.timeArray, app.sampleNo];
- app.sampleNo = app.sampleNo + 1;

- % Plot green intensity over time

```
plot(app.greenIntensityAxes, app.timeArray, app.greenIntensityArray);
xlabel(app.greenIntensityAxes, 'Time');
ylabel(app.greenIntensityAxes, 'Green Intensity');
title(app.greenIntensityAxes, 'Green Intensity Over Time');
grid(app.greenIntensityAxes, 'on');
drawnow;

% Perform further analysis (FFT, plotting, etc.) as needed

% Calculate FFT only after collecting enough data
% Check if enough data has been collected for FFT analysis
if app.sampleNo > 20
% Calculate and plot FFT on green intensity data
app.fftData = fft(app.greenIntensityArray);
disp(['fftData: ', num2str(app.fftData)]);

% Calculate the corresponding frequencies
app.Fs = 10 / (app.timeArray(2) - app.timeArray(1)); % Sampling frequency
disp(['Fs: ', num2str(app.Fs)]);

app.N = length(app.fftData); % Number of samples
disp(['N: ', num2str(app.N)]);

app.f = app.Fs * (0:(app.N/2)) / app.N; % Frequencies
disp(['f: ', num2str(app.f)]);

% Calculate the magnitude of the FFT
app.fftMagnitude = abs(app.fftData);
app.fftMagnitude = app.fftMagnitude(1:app.N/2+1);

% Dynamically set MinPeakHeight based on the maximum FFT magnitude value
minPeakHeightFactor = 0.02; % Adjust as needed
minPeakHeight = max(app.fftMagnitude) * minPeakHeightFactor;

% Use findpeaks to detect the peak in FFT magnitude
[peaks, peakIdx] = findpeaks(app.fftMagnitude, 'MinPeakHeight', minPeakHeight);

if ~isempty(peaks)
% Find the index of the highest peak
```

```
[~, maxPeakIdx] = max(peaks);
app.peakIndex = peakIdx(maxPeakIdx);
app.peakFrequency = app.f(app.peakIndex);
app.HeartRate = app.peakFrequency * 60; % Convert to BPM
disp(['Peak Frequency (Hz): ', num2str(app.peakFrequency)]);
disp(['Heart Rate: ', num2str(app.HeartRate)]);
else
disp('No significant peak detected.');
end
% Convert peak frequency to heart rate
app.HeartRate = app.peakFrequency * 60; % Convert to beats per minute (bpm)
```

```
% Display the heart rate calculated from FFT data
app.HeartRateEditField.Value = app.HeartRate;
```

```
% Plot the FFT magnitude on the designated axes (assuming you have an fftAxes)
plot(app.fftAxes, app.f, app.fftMagnitude);
xlabel(app.fftAxes, 'Frequency (Hz)');
ylabel(app.fftAxes, 'Magnitude');
title(app.fftAxes, 'FFT of Green Intensity');
grid(app.fftAxes, 'on');
drawnow;
```

```
% Create sample data for MATLAB app data based on heart rate
timestamps = 1:50; % Timestamps (For 50 data points)
app.matlabAppData = app.HeartRate * ones(size(timestamps)); % Simulated MATLAB app data (Heart Rate)
disp(['MATLAB Heart Rate Data:', num2str(app.matlabAppData)]);
```

```
% Ensure both arrays have the same size for error calculations
minSize = min(length(app.greenIntensityArray), length(app.matlabAppData));
app.greenIntensityArray = app.greenIntensityArray(1:minSize);
app.matlabAppData = app.matlabAppData(1:minSize);
```

```
if ~isempty(app.greenIntensityArray)
% Calculate the standard deviation of green intensity
app.greenIntensityStdDev = std(app.greenIntensityArray);
disp(['Green Intensity Standard Deviation:', num2str(app.greenIntensityStdDev)]);
```

```
% Calculate the mean absolute error (MAE)
app.absoluteErrors = abs(app.matlabAppData(1:minSize) - app.greenIntensityArray);
app.mae = mean(app.absoluteErrors);
```

```
% Calculate the root mean square error (RMSE)
```

- app.squaredErrors = (app.matlabAppData(1:minSize) - app.greenIntensityArray).^2;
- app.rmse = sqrt(mean(app.squaredErrors));

- % Calculate the correlation coefficient (Pearson's correlation coefficient)
- correlationCoeff = corrcoef(app.matlabAppData(1:minSize), app.greenIntensityArray);
- app.r = correlationCoeff(1, 2);

- % Display the results
- fprintf('Mean Absolute Error (MAE): %.2f\n', app.mae);
- fprintf('Root Mean Square Error (RMSE): %.2f\n', app.rmse);
- fprintf('Correlation Coefficient (r): %.2f\n', app.r); % Display the correlation coefficient

- else
- disp('No data in app.greenIntensityArray.'); % Handle the case when there's no data
- end

- % Optionally, update GUI elements or store these results in app properties
- end
- end
- catch ME
- % Handle errors during video acquisition
- app.MeasurementstatusLamp.Color = 'red';
- disp(['Error: ' ME.message]);
- end
- else
- % Set the status lamp color to red
- app.MeasurementstatusLamp.Color = 'red';
- end
- end
- end

TIMER CODE

- imaqmex('feature','-limitPhysicalMemoryUsage',false);
- app.vid = videoinput('winvideo',1, "MJPG_1280x720", "FramesPerTrigger",1);
- %preview(app.vid);
- app.period = 0.04;
- app.camTimer = timer('ExecutionMode', 'fixedRate', 'Period', app.period);
- app.camTimer.TimerFcn = @(~,~) app.Timer;
- app.PeriodEditField.Value = app.period;
- start(app.camTimer);

- % Assuming you have a ground truth heart rate value (e.g., from a reference source)
- app.groundTruthHeartRate = 75; % Replace with the actual ground truth value

- % Initialize arrays for error analysis
- app.errors = []; % To store absolute errors
- app.squaredErrors = []; % To store squared errors

BUTTON PUSHED FUNCTION CODE

- function UpdatesettingsButtonPushed(app, event)
- app.period = app.PeriodEditField.Value;
- end

VALUE CHANGED FUNCTION (START MEASUREMENT) CODE

- function StartMeasurementButtonValueChanged(app, event)
- app.value = 1;
- app.sampleNo = 1;
- app.timeArray = [];
- app.greenIntensityArray = [];

- triggerconfig(app.vid, 'manual');
- app.vid.TriggerRepeat = inf;
- if ~isrunning(app.vid)
- start(app.vid);
- end
- 8.5 DATA CREATION FOR CORRELATION AND BLAND-ALTMAN PLOT

- function automaticDataProcessing()

- % STEP 1: Automatic Data Creation
- % Create sample data for pulse oximeter readings and MATLAB app data
- timestamps = 1:100; % Timestamps (assuming 100 data points)
- pulseOximeterData = readings recorded; % pulse oximeter data (Heart Rate)
- matlabAppData = app.Heartrate % Simulated MATLAB app data (Heart Rate)

- % Optional: Add noise to the data to make it more realistic
- noiseLevel = 1; % Adjust the noise level as needed
- pulseOximeterData=[]; %values recorded
- matlabAppData = matlabAppData + noiseLevel * app.Heartrate;

- % STEP 2: Data Preprocessing
- % Synchronize data based on timestamps

```
synchronizedTime = intersect(timestamps, timestamps);
pulseOximeterData = pulseOximeterData(ismember(timestamps, synchronizedTime));
matlabAppData = matlabAppData(ismember(timestamps, synchronizedTime));

% Normalize data
minPulseOximeter = min(pulseOximeterData);
maxPulseOximeter = max(pulseOximeterData);
normalizedPulseOximeterData = (pulseOximeterData - minPulseOximeter) / (maxPulseOximeter - minPulseOximeter);

minMatlabApp = min(matlabAppData);
maxMatlabApp = max(matlabAppData);
normalizedMatlabAppData = (matlabAppData - minMatlabApp) / (maxMatlabApp - minMatlabApp);

% STEP 3: Data Visualization
time = timestamps; % Assuming timestamps are known
figure;

subplot(2, 1, 1);
plot(time, normalizedPulseOximeterData, 'b', time, normalizedMatlabAppData, 'r');
xlabel('Time');
ylabel('Normalized Heart Rate');
legend('Pulse Oximeter', 'MATLAB App');
title('Heart Rate Comparison');

% STEP 4: Calculate Correlation Coefficient
correlationCoefficient = corr(normalizedPulseOximeterData', normalizedMatlabAppData');

% Display the correlation coefficient
fprintf('Correlation Coefficient: %.4f\n', correlationCoefficient);

% STEP 5: Bland-Altman Plot
averageValues = (normalizedPulseOximeterData + normalizedMatlabAppData) / 2;
differences = normalizedPulseOximeterData - normalizedMatlabAppData;

% Plot the Bland-Altman plot
figure;
scatter(averageValues, differences);
xlabel('Average of Measurements');
ylabel('Difference between Measurements');
title('Bland-Altman Plot');
```

```
% Optional: Customize the Bland-Altman plot for better visualization
meanDifference = mean(differences);
stdDifference = std(differences);
upperLoA = meanDifference + 1.96 * stdDifference; % 95% confidence interval
lowerLoA = meanDifference - 1.96 * stdDifference; % 95% confidence interval
hold on;
plot([min(averageValues), max(averageValues)], [meanDifference, meanDifference], 'k--', 'LineWidth', 2);
plot([min(averageValues), max(averageValues)], [upperLoA, upperLoA], 'r--', 'LineWidth', 2);
plot([min(averageValues), max(averageValues)], [lowerLoA, lowerLoA], 'r--', 'LineWidth', 2);
hold off;
legend('Data Points', 'Mean Difference', 'Upper LoA', 'Lower LoA');
end
```