Reproduction of Heart Rate Estimation from Video Using CNNs

Subject Code:	SC3901
Professor:	Dr. Jusak Jusak
Std Name:	Mi Songhua
Std Id:	14399341
Major:	Bachelor of Data Science
Contact:	songhua.mi@my.jcu.edu.au

Content

1. Introduction

Heart rate is a vital health indicator traditionally measured using contact-based devices. However, there is growing demand for contactless methods in applications like patient monitoring, newborn care, and driver health. Remote photoplethysmography (RPPG), which uses camera technology to detect subtle skin color changes, offers a promising contactless solution for accurate heart rate measurement, even with natural head movements.[1]

In this project, we replicate the RPPG technique for heart rate measurement using resources from a GitHub repository. We evaluate various blind source separation methods and face detection algorithms to assess their impact on accuracy. The process includes a thorough understanding of the original project and detailed code analysis to ensure reliable and accurate replication results.

2. Related Work

Research in heart rate estimation from facial videos has advanced significantly over the past decade, focusing on enhancing algorithm stability and robustness under real-life conditions. Blind Source Separation, including techniques like Independent Component Analysis (ICA) and Principal Component Analysis (PCA), has proven crucial for improving accuracy. While both ICA and PCA yield similar results, neither is distinctly preferred.

The choice of Region of Interest (ROI) can also impact accuracy. Traditionally, face detection relied on Viola and Jones' method, which forms the basis of the Haar cascade classifier in OpenCV. Recently, Convolutional Neural Networks (CNNs) have been employed for face and skin segmentation, offering improved resistance to background noise by providing pixel-level masks and reducing inaccuracies.

In this project, replication was constrained by the available source code and dataset. Consequently, the focus is on comparing the effectiveness of CNN-based algorithms with traditional vision haarcascade algorithms in heart rate estimation. This comparison aims to evaluate the improvements in accuracy and robustness provided by CNNs over conventional methods.

3. Methodology

Dataset:

Since the original GitHub project did not provide a dataset, we obtained video data from a new source https://sites.google.com/view/ybenezeth/ubfcrppg .[2]

The methodology involves two primary steps. First, we apply a face detection model to the videos to identify the region of interest (ROI) for heart rate calculation. Next, we decompose the detected ROI into RGB channels and apply Source Separation techniques (PCA/ICA) to extract source signals. These signals are then transformed into the frequency domain for filtering and peak detection, allowing us to estimate the heart rate.

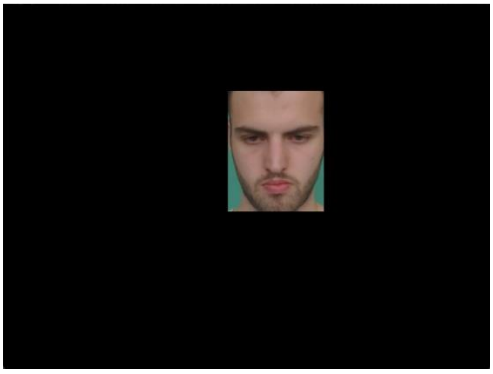
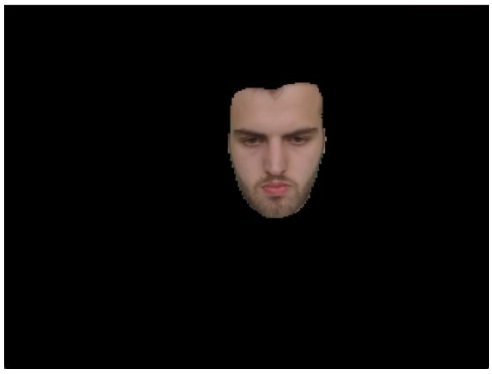

Figure 1 and 2. ROI of Haarcascade and CNN

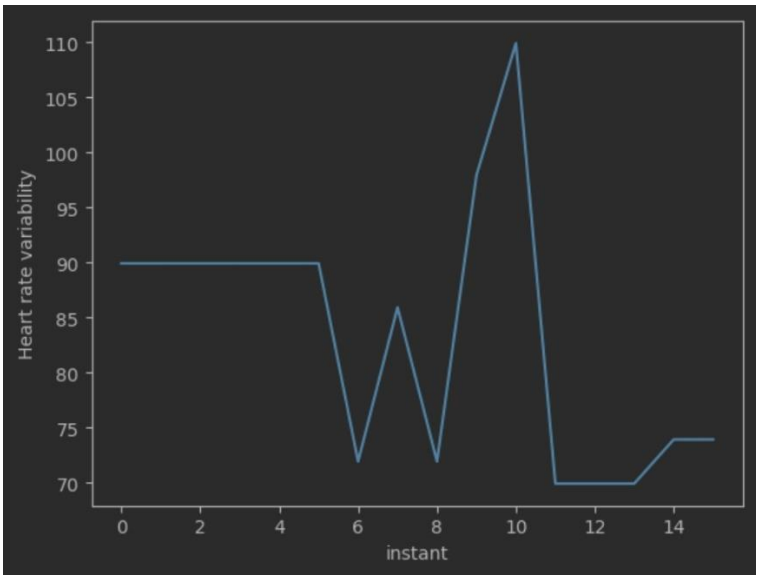

Figure 3. Example of Heart Rate Trend

4. Code Decomposition

4.1 **Function**:

a. random_rotation, random_noise, transleft, transright, horizontal_flip:

These functions can be used for data augmentation by applying random transformations to images, thereby increasing the diversity of the data and improving the model's generalization ability.

b. frame_ps_video:

Retrieves and prints the frames per second (FPS) of a given video.

c. roi_util

Extracts a Region of Interest (ROI) from an image based on the provided face bounding box and adjusts the bounding box to include a specified fraction of the face area. It also optionally removes the eye region from the mask. The function effectively focuses on the face region while allowing for optional exclusion of the eye area, enhancing the accuracy of subsequent analysis by isolating the relevant facial features.

d. distance, roi_extract:

These functions facilitate face detection and tracking across video frames, ensuring that the ROI extraction process remains consistent even as faces move or change between frames.

e. graph_sig:

Visualizes the signal data over time, plotting each component of the signal in a different color. This function is useful for visualizing and analyzing the behavior of multiple signal components over a defined time window.

f. plot_hr, power_gr:

Both functions are used for visualizing different aspects of signal data: one for time-domain heart rate variability and the other for frequency-domain power spectrum analysis.

g. extract_heartrate:

Extracts heart rate from a signal window by normalizing data, decomposing signals with PCA, computing the power spectrum, and identifying the heart rate based on frequency analysis.

4.2 Main Feature

4.2.1 Haarcascade

a. Initialization

The code initializes the video capture, face detection model, and data containers for storing color signals and heart rate values. It sets up variables for frame processing.

b. Video Processing Loop

Each frame is read from the video, and faces are detected and processed to extract the ROI. This loop continues until the video ends.

c. ROI Processing

The ROI is reshaped to extract average color values, which are appended to the colorSig list for heart rate calculation.

d. Heart Rate Calculation

Heart rates are computed from the collected color data by normalizing, decomposing with PCA, and analyzing the power spectrum to determine the heart rate.

e. Masked ROI Handling and Display

Masked regions in the ROI are handled, and the processed image is displayed in real-time using OpenCV.

f. Post-Processing

After video processing, the average heart rate and number of instances are printed, and heart rate data is saved to a CSV file. The video capture and display windows are closed.

4.2.2 CNN

a. Model and Device Setup

The video capture object and face detection model are initialized. Containers for storing average color values and heart rates are set up, and the previous face box is initialized.

b. Video Capture and Initialization

The video capture object and face detection model are initialized. Containers for storing average color values and heart rates are set up, and the previous face box is initialized.

c. Frame Processing

Each frame from the video is read, transformed, and processed by a pre-trained CNN model to perform face segmentation. The frame is resized and normalized before being fed into the model. The model's predictions are used to create a mask for the face region.

d. ROI and Color Processing

The mask is applied to the frame to isolate the face region. Non-face areas are set to transparent. The average color values of the ROI are calculated and stored in colorSig.

e. Heart Rate Calculation

Heart rate is calculated every second based on the collected color data. A window of data is extracted, normalized, and processed to determine the heart rate, which is appended to the heartRates list.

f. Display and Post-Processing

The processed frame with the face mask applied is displayed. After processing all frames, the theoretical mean heart rate is printed, and the heart rate data is saved to a CSV file. The video capture and display windows are closed.

5. Results

	Actual Heart Rate	Haarcascade Heart Rate Result	CNN Heart Rate Result
Subject 3	100	90	87
Subject 4	110	85	95
Subject 5	98	84	99
Subject 25	91	82	94
Subject 27	89	81	85
Subject 42	94	99	98

Figure 1. Comparison of results

	Haarcascade	CNN
Error	1.694	0.667

Figure 2. Error of two method

6. Discussion

In this study, we compared heart rate estimation results obtained using two different face detection methods: Haar cascade and CNN-based segmentation, alongside evaluating the impact of window size and source signal decomposition techniques.

Comparison of Heart Rate Results:

The heart rate estimates derived from CNN-based segmentation generally show higher accuracy compared to those obtained using the Haar cascade method. Specifically, the CNN method yields a lower mean absolute error (0.667) compared to

Haar cascade (1.694), reflecting its superior performance in extracting accurate face regions for heart rate estimation. For instance, the average heart rate errors for various subjects were notably lower using CNN, with the heart rate estimates being closer to the actual values compared to Haar cascade results.

Impact of Sliding Window Size:
The prediction accuracy is influenced by the size of the sliding window used during analysis. We determined the optimal window size based on FPS analysis of the video, balancing between capturing enough data and maintaining accurate heart rate calculations. The chosen window size directly affects the algorithm's performance, as variations can lead to inconsistent heart rate estimations due to insufficient or excessive data.

Outliers and Algorithm Performance:
Outliers in heart rate measurements may indicate that the algorithm failed to correctly identify pulse signals, leading to the selection of incorrect dominant frequencies. These anomalies often result in heart rate estimates that deviate significantly from actual values, thus affecting the overall accuracy of the algorithm. Accurate heart rate estimation is contingent upon correctly identifying the pulse signal amidst other noise.

7. Conclusion

The CNN-based segmentation method demonstrates a significant advantage over the Haar cascade approach due to its ability to provide pixel-level face contours, which improves the accuracy of the ROI extraction. In contrast, Haar cascade provides a less precise bounding box, which can lead to inaccuracies in capturing the relevant face region for heart rate analysis.

8. Learning Process

The learning process involved four key stages:

Initial Setup: Acquired the GitHub project repository from the advisor and reviewed essential software and dataset requirements. This phase involved understanding the scope and objectives of the project and setting up the necessary tools and resources.

Environment Setup and Understanding: Configured the coding environment by installing dependencies and tools. Thoroughly studied the original research paper and project documentation to grasp the theoretical concepts and implementation details. This step was crucial for building a solid foundation for the project.

Replication Attempt: Conducted initial attempts to replicate the code from the GitHub repository. During this phase, encountered and documented various issues and challenges. Sought feedback and guidance from the advisor to address these problems and refine the replication process.

Finalization: Successfully completed the replication of the project by integrating the solutions to the encountered issues. Analyzed the results and compiled a detailed report, documenting the methodology, findings, and conclusions. This phase ensured that the project was accurately reproduced and the outcomes were effectively communicated.

9. References

Jayantj1j. (2023). *Heart rate estimation from video using CNN*. GitHub. Retrieved August 12, 2024, from https://github.com/jayantj1j/Heart-rate-estimation-from-video-using-CNN

Benezeth, Y. (n.d.). *UBFC-RPPG dataset*. Retrieved August 12, 2024, from https://sites.google.com/view/ybenezeth/ubfcrppg

Yeafi, A. (2021). *UBFC-P: A dataset for heart rate estimation from facial videos*. Kaggle. Retrieved August 12, 2024, from https://www.kaggle.com/datasets/ashfakyeafi/ubfc-2