Video-based Heart Rate Estimation using Embedded Architectures

Hoda El Boussaki*, Rachid Latif, Amine Saddik
Laboratory of Systems Engineering and Information Technology LISTI
National School of Applied Sciences, Ibn Zohr University, Agadir 80000, Morocco

***Abstract*—Monitoring a driver's heart rate is an important determinant to his health condition. The monitoring system must be accurate and non restrictive to the user's actions. Estimating the driver's change in his usual heart beat pattern can prevent undesirable outcomes. Several methods exist to estimate heart rate without any contact. In this paper, we are focusing on a method that uses remote photoplethysmography (rPPG). rPPG is a technique where heart rate is extracted from a PPG signal. The signal is extracted from the changes in blood flow that corresponds to the color variations recorded through an RGB camera. In this work, a different study that was based on an existing algorithm is presented to determine its processing time. The algorithm we proposed was divided into different global blocks and each block into different functional blocks (FBs). Though evaluating all the blocks' processing time, it was possible to determine the most time consuming functional blocks. The results are implemented on different architectures: Desktop, Odroid XU4 and Jetson Nano to provide a higher performance.**

***Keywords*—Heart rate; driver; photoplethysmography; non-contact; embedded architectures**

I. Introduction

Monitoring Vital signs can be life saving. When it comes to driving, it could save the driver's life as well as anyone who could be affected by a potential accident. People suffering from cardiovascular diseases (CVD), like cardiomyopathy or coronary heart disease (CHD), can become a danger to themselves and any passerby. A rapid heart rate and palpitations can also be caused by a low blood sugar. It can indicate a hypoglycemia for example. Therefore, heart rate is an indicator of several health conditions as it is the first response of the body to a threat. According to the world health organization, an estimated of 17.9 million people died from CVDs in 2019, representing 32% of all global deaths. Heart attacks and strokes were responsible for 85% of these deaths [1]. The death of a driver due to a disease attack is a reasonably common cause of death on the road [2]. As reported by the Center for Disease Control and Prevention (CDC), 1.35 million people are killed every year on the road around the world. Injuries on the road is the eight leading cause of death globally [3]. Therefore, a continuous heart rate monitoring in this context is of great importance as it can save lives.

Measuring heart rate usually requires an ECG that records the electrical heart activity caused by the repolarization and depolarization of the muscle [4], [5]. Different methods exists to estimate heart rate in vehicles. They can be divided into five types depending on what kind of system is used. There is the heart rate monitor integrated into the steering wheel, the seat, the rear-view mirror or the seat-belt or a heart rate monitor using a camera. As an example, J. Priya et al. 2020 used a pulse sensor, GPS and GSM modules are combined to the steering wheel to assess the driver's pulse rate in real time [6]. Using also a steering wheel, Arakawa et al. 2018 developed a system that measures heart rate through a transmitter and a red LED as a receiver [7]. S. Mitani 2018 developed an in-vehicle pulse sensor using the microwave sensor where the sensor is in the driver's seat [8]. Texas instruments developed in 2019 the AWR1642 sensor placed on the rear-view mirror that estimates the heart rate of all the passengers [9]. There are also devices situated in the seat-belt as in the HARKEN concept [10]. Another method is to extract heart rate from the changes of hemoglobin concentration on the surface of the face captured by an RGB camera [11]. Y. lee et al. 2018 used Impulse-Radio Ultra-Wideband (IR-UWB) Radar Technology to monitor vital signs [12]. W. Lv et al. 2021 also used radar technology. They used a frequency-modulated continuous-wave (FMCW) Millimeter Wave Radar in the 120 GHz band [13].

In our work, we propose an algorithm that estimates heart rate through an RGB camera. It is divided into four parts. The first part focuses on the face detection and the forehead extraction. We used the box blurring filter, the edge sobel edge detection technique and morphological operations for face detection and the extraction of the region of interest. The second part extracts the raw signal by calculating the average of each of the channels (red, green and blue). The third part uses only the green channel of the image to estimate the final signal. The result is obtained by normalizing and denoising the signal and also using a detrending filter and a moving average filter. Finally, the fourth part calculates the heart rate based on a frequency analysis. The three latter parts were based on the work of P. rouast et al. 2016 [14]. The summary of our contribution is as follows:

- The proposition of a new algorithm for face detection and forehead extraction.
- The examination of the temporal constraints based on a Hardware/Software Co-Design approach.
- The evaluation of the algorithm on different embedded architectures.

The algorithm based on C/C++ was validated then was accelerated using OpenMP, MPI and CUDA. We chose a hybrid implementation of OpenMP or MPI and CUDA due to the requirements of the algorithm. CUDA uses NVIDIA's Graphics Processing Unit (GPU) to achieve a higher performance time-wise. Using parallel programming gives better results than the naive implementation. We first tested the algorithm on a desktop, but the desktop is not adequate to monitor heart

rate when driving because of its size and power consumption. Therefore, we tried implementing the algorithm on embedded architectures such as Odroid XU4 and Jetson Nano.

This paper is structured as follows: Section I describes the recent works on contactless heart rate monitoring. Section II describes our methodology. Then, Section III highlights the results obtained by implementing the algorithm on different architectures. Finally, a conclusion summarizes this work and gives some future perspectives.

II. Related Work

Various studies were made to estimate heart rate from an RGB camera. It was possible to monitor heart rate by monitoring the variation of the RGB colors of an image induced by the changes of blood flow in the capillaries. The signal extracted from those variations is known as a photoplethysmography (PPG) signal. The human skin is illuminated with a light source and a camera captures the variations of color [15]. The skin's RGB values change with time and are estimated through the reflection of the light [16]. There are two types of reflection: specular and diffuse as shown in Fig. 1. The reflection of a skin pixel is defined in Eq. 1 [16].

$$C_k(t) = I(t).(v_s(t) + v_d(t) + v_n(t)) \quad (1)$$

Where k is the kth pixel, $I(t)$ is the luminance intensity, $v_s(t)$ is the specular reflection that occurs on the surface, $v_d(t)$ is the diffuse reflection on the blood vessels and $v_n(t)$ is the noise.

The specular reflection is a mirror-like reflection and does not contain information of the pulse. It can be expressed in Eq. 2 [16].

$$v_s(t) = u_s.(s_0 + s(t)) \quad (2)$$

Where u_s is the unit color vector of the light spectrum and s_0 and $s(t)$ are the stationary and changing parts of the specular reflection.

The diffuse reflection is related to the absorption of the light. It is defined in Eq. 3 [16].

$$v_d(t) = u_d.d_0 + u_p.p(t) \quad (3)$$

Where u_d is the unit color vector of the skin, d_0 is the stationary reflection, u_p is the relative strength of the pulse in the channels and $p(t)$ is the signal.

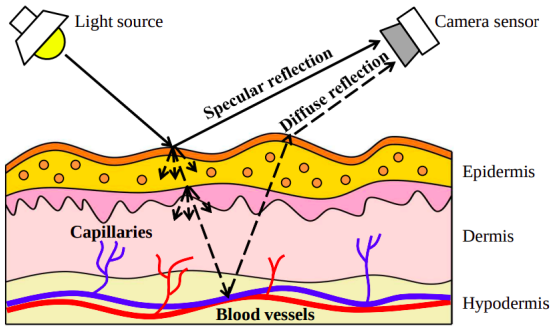

Fig. 1. Reflection of the light on the skin [16].

Different methods were proposed to extract the PPG signal like the green spectrum method where the signal is extracted from the G channel only [17], the Blind source Separation-based (BSS) methods that uses all three channels [18], the CHROM technique that is chrominance-based [19], the Plane-Orthogonal-To-Skin (POS) that defines an orthogonal plane to a normalized skin [16] and the Spatial Subspace Rotation (2SR or SSR) that measures the rotation of the spatial subspace of the pixels [20].

H. Rahman et al. 2016 used an RGB camera to monitor heart rate [21]. They used an Independent Component Analysis (ICA) to extract the PPG signal and the Fast Fourier Transform to convert the signal to the frequency domain. M. A. Hassan et al. 2016 also used an RGB camera using only the green spectrum [22]. V. Jeanne et al. 2013 used an infrared camera instead of a regular RGB camera that requires certain light conditions [23]. Other methods for non-contact heart rate monitoring include radar systems. K. J. Lee et al. 2016 used continuous-wave Doppler Radar to estimation a driver's heart rate. The radar is installed in the seat. The emitted signal gets reflected and contains information about the heart activity. The spectral peak of the reflected signal represents the value of the heart rate. Instead of using a Fast Fourier Transform (FFT), they proposed a method using multiple signal classification (MUSIC) because the contamination of the signal caused by movement of the driver and the vehicle [24]. H. Xu et al. 2021 also used radar technology to estimate heart rate. They used an ultra-wideband (UWB) radar with a mean absolute error (MAE) of 1.32 [25].

This work focuses on the algorithm proposed by P. Rouast et al. 2016 [14]. They used the green spectrum method. The face is detected using the Viola-Jones algorithm as the first method and a deep neural network (dnn) as the second method. Then, it is captured by a camera in order to determine facial landmarks. After that, the forehead, where most of the blood vessels are concentrated, is selected as the region of interest. The average of each pixel color (Red, Green, Blue) of the region is measured over time to extract the PPG signal. Afterwards, the signal is filtered and its peaks are detected to estimate the heart rate. However, the processing time of their work is significant and needs improving. Furthermore, the face detection part is the most time consuming.

III. Methodology

Estimating heart rate is defined in this paper as a four steps methodology: face detection, raw signal extraction, signal filtering and heart rate estimation. Each step represents a block and every block is going to be divided into different functional blocks (FBs).

A. Face Detection

In this work, heart rate is being extracted from the face. The face is the most visible part on the body when a person is captured by a camera. And since the forehead is a surface that has a visible subcutaneous vascular structure, the forehead is the region of interest. Therefore, face detection is an important phase of the algorithm. Different methods exists to detect and track the face. The most used in contactless monitoring are machine learning based methods. They are capable of recognizing facial features through comparing them with an existing database. The main issue with these kind of methods

is that they are time-consuming. Consequently, we propose a different approach for face detection.

1) Original approach: The original algorithm on which this work was based used the Viola-Jones algorithm to detect the face. It's a haar classifier that is trained to detect faces. The OpenCV cascade classifier was used. Once the face is detected, the region of interest (ROI), known as the forehead, is selected. The algorithm can track faces and that means that it works when there is movement but it takes an important amount of time to be executed, approximately 1s. The Haar cascade uses Haar-like features represented by rectangles. Each rectangle is used to detect a region of the face [26]. These features help identifying where pixel intensity suddenly changes. The darker areas have a pixel value of 1 and the lighter areas have a pixel value of 0. When the difference of the sum of the first area's pixels and the sum of the second are's pixels is close to 1, then an edge was detected.

2) First method: The problem encountered in the beginning of this work is that a trained haar classifier cannot be accelerated using parallel computing to reduce its processing time as it is an OpenCV function. The first approach to this problem was to use a sequence of images instead of a video. The processing time decreased but was still significant. The second approach was to use a sequence of images, but instead of detecting the face for each images the region of interest is defined manually. The face is static and so is the ROI. If the ROI's emplacement in the image is known, it can be extracted without going through a trained classifier. The processing time of the ROI extraction using this method was 0,02 ms instead of 1s with a haar classifier.

3) Second method: The second method was proposed because of how limited is the previous one is. Even if the processing time is significantly low, it is not practical to manually set the ROI each time. This method works through three steps: image preprocessing, face identification and forehead extraction.

Fig. 2 represents the ROI extraction algorithm. The algorithm that represents the first block is divided into different functional blocks. In the first functional block, the RGB image is converted to grayscale and a box blurring filter is applied in order to apply to sobel filter. In the second functional block, the resulting image is converted to a binary image to be able to detect the contours. Once the contours are detected, in the third functional block, inside the contours is filled in white and morphological operations are applied. After that, in the last functional block the new contours are found to determine the top extreme point that represents the top of the head.

a) Preprocessing: In this step, the colored image is turned into a gray scale image because the edge detection technique works with gray scale images only. Then, the image is blurred. Fig. 3 represents the original image and Fig. 4 represents the grayscale image.

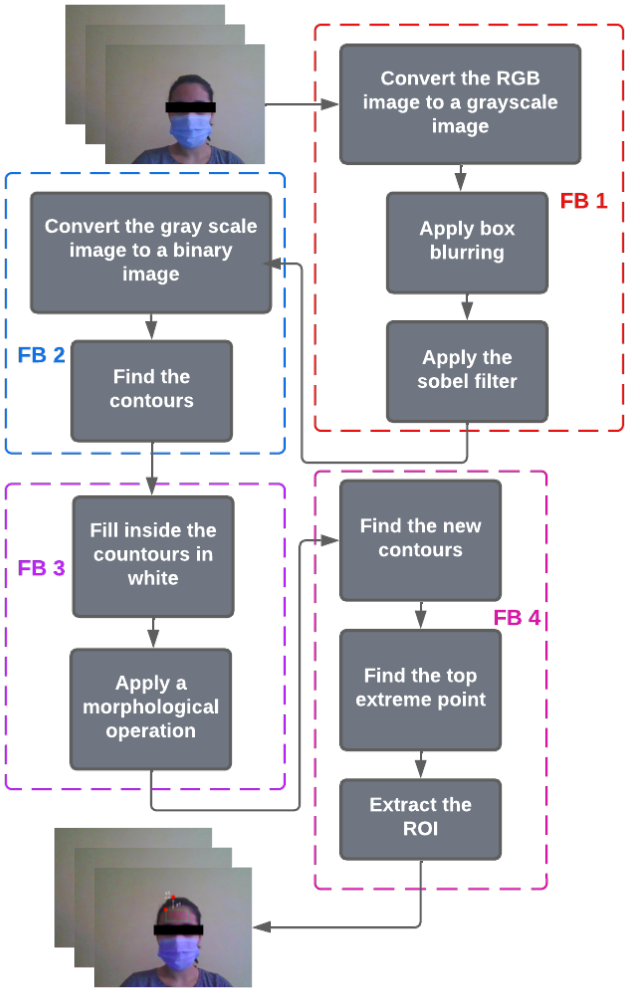

Fig. 2. ROI extraction (Block 1).

Fig. 3. Original image.

Fig. 4. Gray scale image.

Box blurring is a low-pass filter where an image's pixel has a value close to the average value of the pixels surrounding it. It allows the suppression of as much noise as possible. A 3x3 matrix 4 is applied on the image to blur it. The convolution technique is shown in Eq. 5. The center of matrix K corresponds in the image to the pixel's value that's going to change. The value is calculated by adding the product of each

neighboring value with the corresponding kernel's value.

$$K = \frac{1}{3x3}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \tag{4}$$

$$B = A \oplus K \tag{5}$$

Where A is the input image and B is the blurred image. The blurred image is obtained using Eq. 6 [27].

$$B(i,j) = A(i,j) \oplus K(i,j) = \sum_{m=0}^{2}\sum_{n=0}^{2} A(2,2)F(i-m, j-n) \tag{6}$$

Where $0 \leq$ i,m ≤ 2 and $\leq$ j,n ≤ 2

b) Face identification: In order to detect the face, the sobel filter is used to detect the edges of the face by calculating the gradient of the image. According to Himani et al. 2020, the sobel filter is more precise and time-efficient than the canny filter [28]. The filter highlights the edges. It uses two 3x3 matrix Sx and Sy also known as convolution kernels or masks.

$$Sx = \begin{bmatrix}-1 & 0 & +1\\ -2 & 0 & +2\\ -1 & 0 & +1\end{bmatrix} \tag{7}$$

$$Sy = \begin{bmatrix}+1 & +2 & +1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix} \tag{8}$$

Sx is the horizontal mask used for the changes in the horizontal direction and Sy is the vertical mask used for the changes in the vertical direction. Sy is a rotation of the second kernel Sx by 90°. The kernels are applied separately on the image to produce separate calculations of the gradient component in each orientation [29].

$$Gx = Sx \oplus A \tag{9}$$

$$Gy = Sy \oplus A \tag{10}$$

Where A is the input image.

The separate gradients are combined to produce one image using Eq. 11.

$$G = \sqrt{Gx^2 + Gy^2} \tag{11}$$

An approximation of the combined gradients is given by Eq. 12.

$$G = Gx + Gy \tag{12}$$

The edges obtained are shown in Fig. 5.

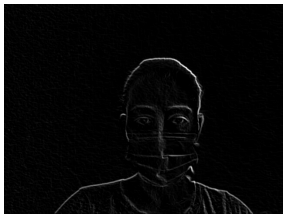

Fig. 5. Image filtered using sobel edge filter.

After using the sobel edge filter, a thresholding is applied on the filtered image. It converts the image from a gray scale one to a binary image. It's an OpenCV technique where a pixel's value becomes 0 if the initial value is smaller than the threshold, otherwise it becomes 255 which is the maximum value a pixel can have. It turns the edges completely white while the rest is black. This technique thickens the edges and make them more visible. Fig. 6 represents the image after using the thresholding technique.

Fig. 6. Image after using the thresholding technique.

Algorithm 1 describes how the sobel edge filter is performed.

Algorithm 1 Box blurring and sobel edge filter (FB1)

Input: Image
Output: Image after using the sobel filter
Function `cvtcolor`**:**
  GrayImage ← 0.299.R + 0.587.G + 0.114.B
Function `blur`**:**
  Create a 3x3 kernel
  Apply the kernel to the image
Create the horizontal mask Sx
Function `filter2D`**:**
  Compute correlation between Sx and the GrayImage
  Get Gx
Create the vertical mask Sy
Function `filter2D`**:**
  Compute correlation between Sy and the GrayImage
  Get Gy
G ← Gx + Gy

c) Forehead extraction: The first step to detect the forehead is to find the coordinates of the contours. They are found using the OpenCV function: findcontours. The function detects the sudden changes in the image's color. Once the change is detected, the coordinate are retrieved. The function implements an algorithm introduced in 1985 by S. Suzuki et al. [30].

When the contours are retrieved, it becomes possible to color the face in white with the OpenCV function fillpoly. Now, we want to delete the forms left on the background but also leave only the upper part of the face. For this purpose, a mathematical operation, known as an opening, is applied on the filled image to make the image more clear. The image contains small white filled forms that need to be removed, hence darkened. An opening is the process of applying erosion followed by dilatation on an image [31]. These two operations are achieved by using a 5x5 structuring element x on an image A as shown in the following formula.

$$B = (A \ominus x) \oplus x \tag{13}$$

Where the first operation represents the erosion and the second represents the dilatation.
The dilatation and erosion give a new binary image. In the dilatation, the pixel's value of image A is set to 1 when any of the neighboring pixels is equal to 1. Whereas in the erosion, the pixel's value of A is set to 0 when any of the neighboring pixels is equal to 0.

Now that only some parts of the image are left and the forehead is very visible, it is possible to detect the forehead by finding the top extreme point on the image which is the top of the head.

In order to find the coordinates of the top extreme point, we apply the OpenCV function findcontours. The use of this function for a second time gives us the new contour's values because the image have been modified. Then, a loop is used to compare between all the new coordinates of the contours to find the top point. Algorithm 2 describes these steps.

Algorithm 2 Finding the top extreme point (FB4).

Input: Image after using the opening operation
Output: The Top extreme point
Function `findcontours`**:**
  Retrieve the contours from the image
Create a point Top
 for *i = 0 to Contours.size()* **do**
  Create a point P
   Create a vector NewContours
   NewContours ← Contours[i]
   for *j = 0 to NewContours.size()* **do**
    Create a point CurrentP
     CurrentP ← NewContours[j]
     if *y coordinate of CurrentP ¡ y coordinate of P* **then**
      P ← CurrentP
     end
   end
  Top ← P
end

The top extreme point is represented in red in Fig. 7. The last step is to subtract a value x1 to the x coordinate of the top point and add a value y1 to the y coordinate. It allows us to get an ROI that starts from these new coordinates a little below the top of the head where the forehead is situated. Fig. 8 represents this step.

Fig. 7. The Top extreme point on the original image

Fig. 8. Roi extraction.

B. Raw Signal Extraction

The raw signal is extracted from the image by using a function that calculates the average of each channel's pixels. An image contains 3 channels: red, green and blue. The average of each channel is added to the signal. And, for every frame, new values are added to it. At this stage, the signal represents the changes of the pixel's values from one frame to another. Fig. 9 represents the raw signal extraction algorithm.

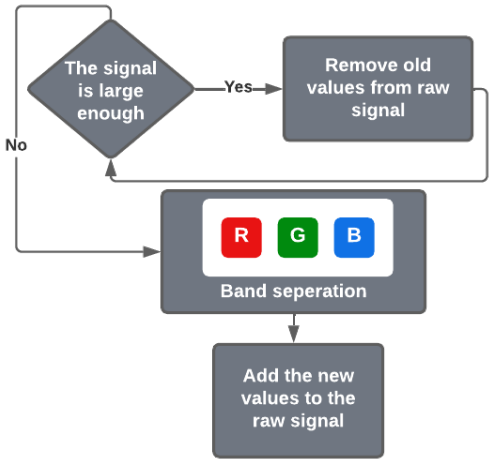

Fig. 9. Raw signal extraction (Block 2).

C. Signal Filtering

Different methods exist to obtain the final signal. After extracting the average of each channel, only the green channel is left. W. Verkruysse et al. 2008 explained that the green channel contains the most information about a PPG signal. The main reason for that is the better absorption of green light than red and blue by hemoglobin [32]. Fig. 10 represents the signal filtering algorithm which represents the second block of the algorithm.

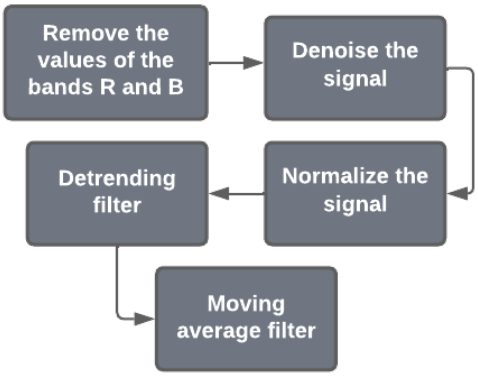

Fig. 10. Signal filtering (Block 3).

This block is executed only if the signal is large enough. There is enough data after exactly 35 frames. The block, represented in Fig. 11, contains four steps. First, a filter is used to remove unwanted spikes from the signal. Then, normalize the signal and apply a high pass and a low pass filter to cut off low and high frequencies corresponding to 0.7 and 3 Hz [33]. These frequencies are generally caused by noise and sudden light change. P. rouast et al. 2016 used detrending filter as an equivalent to the high pass filter and a moving average filter as an equivalent to the low pass filter.

D. Heart Rate Estimation

The HR is measured using a frequency analysis. This block remained the same as the original algorithm and is executed only if the signal is large enough. In this case, the discreet Fourier transform (DFT) is used. The heart rate is calculated with Eq. 14.

$$BPM = (MaxFr * fps * 60)/Size \tag{14}$$

Where:

MaxFr is the maximum frequency

fps is the number of frames per second

Size is the size of the signal

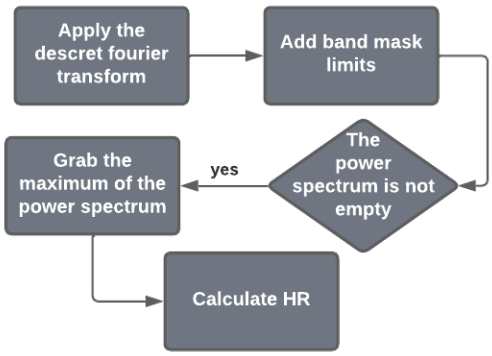

Fig. 11. Heart rate estimation (Block 4).

IV. Hardware and Software Results

The algorithm was first validated using the C/C++ language. Then, in order to achieve better results in term of time consumption, we separated the algorithm into four blocks, each with a specific function. This step was essential to estimate the processing time of the blocks and to determine the blocks that consume the most. The first block is for face detection and forehead extraction, the second is for the raw signal extraction, the third is for signal filtering and finally the fourth block calculates the heart rate. In our case, the first block was the most consuming, hence its separation into four functional blocks. The second temporal evaluation on the algorithm revealed that the first functional block (FB1) takes most of the block's processing time.

A. System Specification

This work was implemented on an Intel i7-1165G7 desktop that has a NVIDIA GeForce MX330 GPU based on a Pascal architecture and that supports CUDA. It was also implemented on two different embedded architectures: Odroid XU4 and NVIDIA Jetson Nano. The Odroid XU4 has an exynos 5422 processor, an ARM A15 CPU with 2 Ghz and an ARM A7 with 1.4 Ghz. Finally, the NVIDIA Jetson Nano has an ARM A57 CPU with 1.43 Ghz and a Maxwell based GPU. Table I represents the systems' specifications.

TABLE I. Specification of the Systems Used

Type	Desktop	Odroid XU4	Jetson Nano
Processor	11th Gen Intel Core™	Exynos 5422	Tegra SoC
CPU	Intel i7	ARM Cortex A15/A7	ARM A57
GPU	NVIDIA GeForce MX330	Advanced Mali	Nvidia Maxwell
Suport language	C/MPI/OpenMP/ Cuda/OpenCL	C/MPI/OpenMP/ OpenCL	C/MPI/OpenMP/ Cuda
Frequency	2.8GHz	2GHz/1.4GHz	1.43Ghz
Weight	1,78kg	60g	136G
Energy	90W	5W	10W

B. Sequential Implementation of the Algorithm

The algorithm was implemented on each of the different architectures based on the C/C++ language. It contains four blocks and the processing time of each block was calculated. Table II summarizes the time evaluation.

TABLE II. Processing Time of Each Block

Blocks	Desktop	Odroid XU4	Jetson Nano
B1	50,41 ms	503,21 ms	16,56 ms
B2	0.51 ms	0.52 ms	0.033 ms
B3	0.86 ms	2.22 ms	0.1 ms
B4	0.52 ms	0.5 ms	0.07 ms
Total	52.3 ms	506.45 ms	17.39 ms

The time evaluation in Table II shows that the Jetson Nano consumes the less when compared to the other architectures with a global processing time of 16.76 ms. The desktop consumes 52.3 ms and the Odroid XU4 consumes 506.45 ms. Fig. 12 to 15 represent the processing time of block 1 to block 4 respectively on three different architectures. The Jetson Nano has a lower processing time for all blocks. Block 1 consumes the most for all architectures. For that reason, the functional blocks of Block 1 are going to be evaluated. Block 1 is about 50.41 ms for the desktop, 503.21 ms for the Odroid XU4 and 16.56 ms for the Jetson Nano. The next step is to evaluate

the processing time of the functional blocks of B1 and use OpenMP, MPI and CUDA in order to accelerate the global processing time of the algorithm.

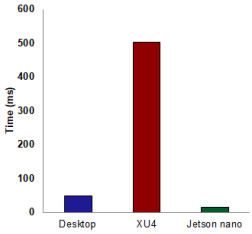

Fig. 12. Processing Time of Block 1.

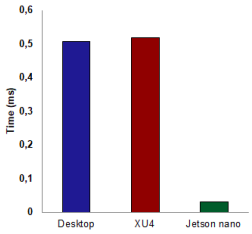

Fig. 13. Processing Time of Block 2.

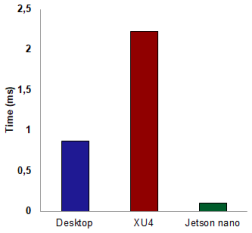

Fig. 14. Processing Time of Block 3.

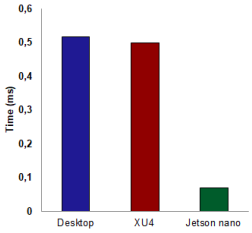

Fig. 15. Processing Time of Block 4.

C. OpenMP and MPI based Implementation

In this, we are going to use both OpenMP and MPI to accelerate the algorithm and determine which one gives better results. However, the main issue we encountered is the difficulty of accelerating the first block using directives. The reason for that is the fact that Block 1 contains mainly OpenCV functions, we couldn't reduce its processing time using OpenMP and MPI. OpenMP and MPI directives wouldn't be effective. On the contrary the processing time increased. Therefore, the implementation of the algorithm using OpenMP and MPI was done only on Blocks 2, 3 and 4. Table III represents the processing time of blocks 2, 3 and 4 with OpenMP and MPI.

The temporal evaluation in Fig. 16 represents the processing time of Block 2, Block 3 and Block 4 with C/C++, OpenMP and MPI implemented on the desktop. It shows better results with MPI as the time is significantly lower than with OpenMP and even more when compared with C/C++. MPI is 6.2 times faster than OpenMP for Block 2, 2.2 times faster for Block 3 and Block 4. Fig. 17 and 18 represent the comparison between the processing time of same blocks using C/C++, OpenMP and MPI but implemented on Odroid XU4 and Jetson Nano.

TABLE III. PROCESSING TIME OF EACH BLOCK WITH OPENMP AND MPI

Blocks	Desktop	Odroid XU4	Jetson Nano
OpenMP			
B2	0,42 ms	0.52 ms	0.03 ms
B3	0.28 ms	1.55 ms	0.092 ms
B4	0.27 ms	0.31 ms	0,067 ms
Total	1.24 ms	2.38 ms	0.19 ms
MPI			
B2	0.069 ms	0.59 ms	0.03 ms
B3	0.12 ms	2.037 ms	0.093 ms
B4	0.12 ms	0.34 ms	0.067 ms
Total	0.31 ms	2.97 ms	0.19 ms

For the Odroid XU4, the results show a nearly same processing time for Block 2. OpenMP was found to be 1.4 times faster for Block 3 and 1.6 times faster for Block 4. MPI was found to be 1.1 times faster for Block 3 and 1.4 times faster for Block 4. This concludes that OpenMP shows a better result than MPI on XU4 with an improvement in global processing time of all the blocks of x1.4 for OpenMP and x1.1 for MPI. In regards to the Jeston Nano, the results show the same results for OpenMP and MPI. However, MPI shows a better result on the desktop with an improvement in global processing time of x6 when OpenMP shows an improvement of x2.

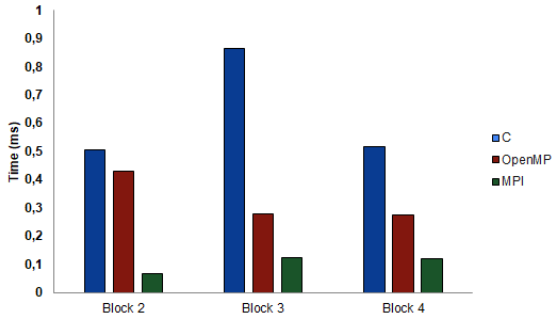

Fig. 16. Improved processing time on desktop.

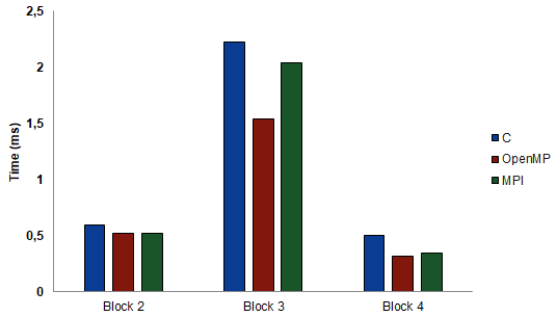

Fig. 17. Improved processing time on Odroid XU4.

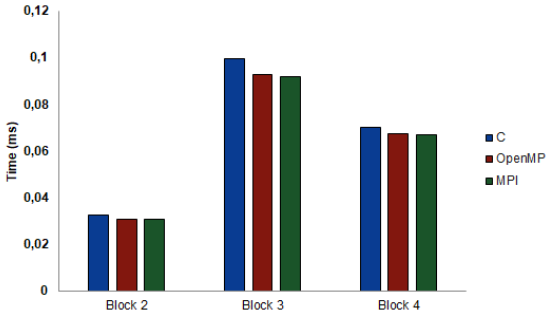

Fig. 18. Improved processing time on Jetson Nano.

D. Cuda based Implementation

In this part, we focused only on the first block. Block 1 was divided into five functional blocks (FBs) as previously shown in Fig. 2. Since we couldn't improve its processing time using OpenMP or MPI, we opted for CUDA to exploit the advantages of a GPU. Table IV represents the processing time of the four different functional blocks of Block 1 on the desktop and the Jetson Nano.

TABLE IV. PROCESSING TIME OF B1'S FUNCTIONNAL BLOCKS

Blocks	Desktop	Jetson Nano
FB1	23,21 ms	7,56 ms
FB2	9.8 ms	2,47 ms
FB3	7,41 ms	5,23 ms
FB4	3,07 ms	1,18 ms

For both the desktop and the Jetson Nano, the first functional block is the most consuming. FB1 consumes a time of 23,21 ms for the desktop and 7,56 ms for the Jetson Nano. Consequently, FB1 will be accelerated using CUDA. Fig. 19 summarizes the processing times of the different functional blocks.

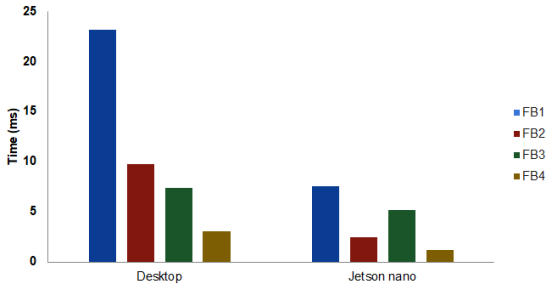

Fig. 19. Processing time of the different functional blocks of B1 in desktop and Jetson Nano.

FB1 contains three sub-blocks (SBs). The first one converts an image from RGB to grayscale, it takes an average of 5.12 ms for the desktop and 1.25 ms for the Jetson Nano. The second applies a blurring filter with an average of 3.6 ms for the desktop and 1 ms for the Jetson Nano.The last sub-block filters the image using a sobel filter and takes 14.49 ms for the desktop and 5.3 ms for the Jeston Nano. We then opted to accelerate the first and the last sub-blocks as shown in Fig. 20 for both architectures.

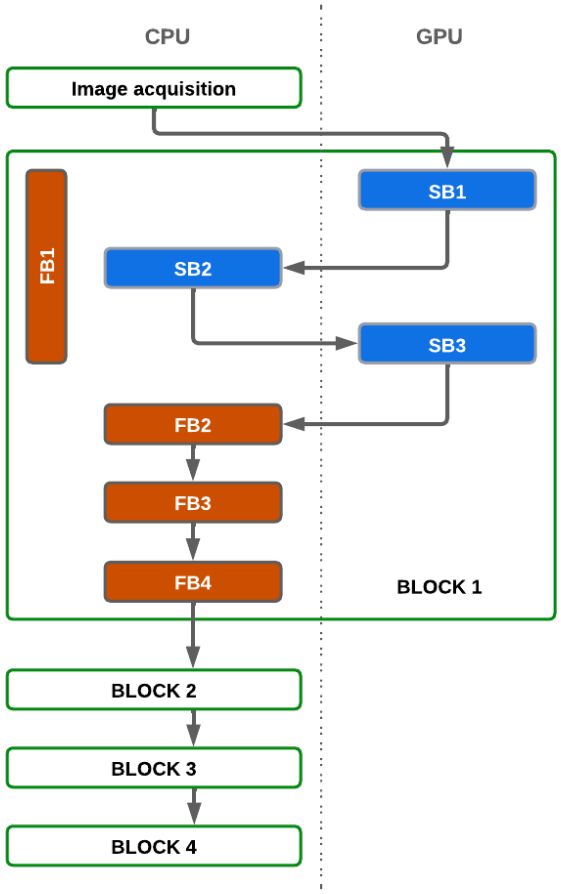

Fig. 20. CPU-GPU implementation based on CUDA.

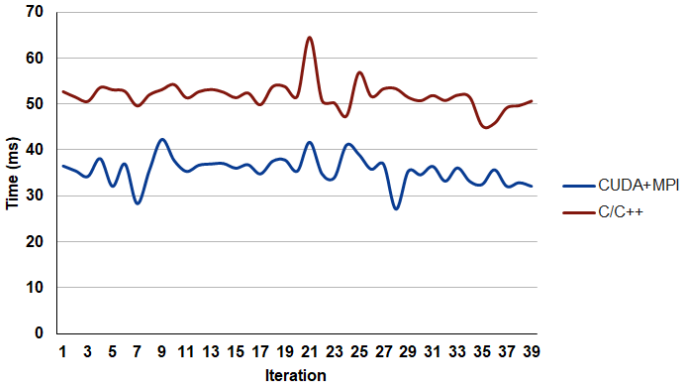

Fig. 21. Improved global processing time on desktop.

Fig. 21 shows the improved global processing of the algorithm on the desktop and Fig. 22 shows the improved global processing on the Jetson Nano. We obtained an average improved time 35.54 ms for the desktop, hence an overall improvement of x 1.5. For the Jetson Nano, we achieved an improved time of 12.7 ms which is 1.32 times faster. We

used a hybrid implementation of CUDA for the first block and OpenMP/MPI for the other three blocks.

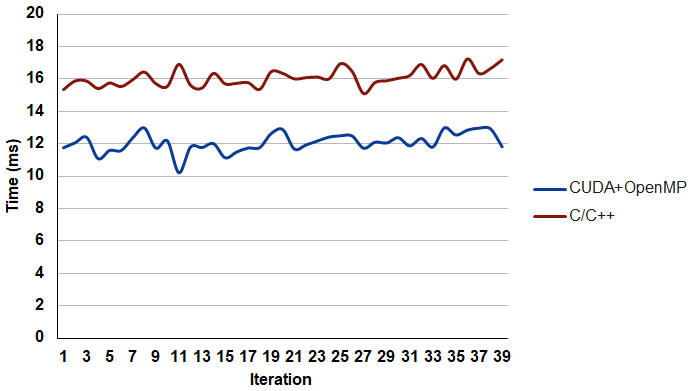

Fig. 22. Improved global processing time on Jetson Nano.

V. Conclusion

In this paper, a non contact heart rate monitoring algorithm is proposed to measure the driver's heart rate. The algorithm was studied to be implemented on different architectures such as Odroid XU4 and Jetson Nano. The time evaluation of the C/C++ implementation showed better results on the Jetson Nano than the other architectures. We were able to exploit the advantages that presents a Nvidia GPU in CPU-GPU architectures by using CUDA. A hardware/software co-design approach was implemented and showed that the Jetson Nano remains the best choice. The sequential implementation consumes a lot of time. Hence, it is not real-time. For this reason, an acceleration of the algorithm was proposed. The acceleration is based on OpenMP, MPI and CUDA on the different architectures used. Future works consist of improving the face detection algorithm when there is movement and further accelerating the algorithm based on CUDA.

Acknowledgment

We owe a debt of gratitude to the National Center for Scientific and Technical Research of Morocco (CNRST) for their financial support (grant number: 27UIZ2022) and for the financial support of the project Cov/2020/109.

References

[1] World health organisation Homepage [Online] Retrieved 2022-06-15 from https://www.who.int.

[2] T. Tervo, E. Räty, P. Sulander , J. M. Holopainen, T. Jaakkola, and K. Parkkari, "Sudden death at the wheel due to a disease attack," Sudden death at the wheel due to a disease attack, 14(2), pp.138-144, 2013.

[3] Road Traffic Injuries and Deaths—A Global Problem. (2020, December 14). Centers for Disease Control and Prevention. Retrieved October 12, 2022, from https://www.cdc.gov/injury/features/global-road-safety/index.html

[4] S. Mejhoudi, R. Latif, W. Jenkal, A. Saddik, and A. Elouardi, "Hardware Architecture for Adaptive Dual Threshold Filter and Discrete Wavelet Transform based ECG Signal Denoising," International Journal of Advanced Computer Science and Applications, 12(11), 2021.

[5] S. Mejhoudi, R. Latif, A. Saddik, W. Jenkal, and A. Elouardi, "Speeding up an Adaptive Filter based ECG Signal Pre-processing on Embedded Architectures," International Journal of Advanced Computer Science and Applications, 12(5), 2021.

[6] J. Priya, T. S. Reshmi, and M. Gunasekaran, "Smart Steering Wheel for Real-Time Heart Rate Monitoring of Drivers," International Journal of Innovative Technology and Exploring Engineering, 9(4), pp.3040–3043, 2020.

[7] T. Arakawa, N. Sakakibara, and S. Kondo, "Development of Non-Invasive Steering-Type Blood Pressure Sensor for Driver State Detection," Int. J. Innov. Comput. Inf. Control, 14, pp.1301–1310, 2018.

[8] OMRON, Development of the in-vehicle pulse sensor [Online] Retrieved 2022-06-15 from https://www.omron.com/global/en/assets/file/technology/omrontechnics/vol50/OMT\verb|_|Vol50\verb|_|006.pdf

[9] TEXAS INSTRUMENTS, Using TI mmWave Sensors for Heart-Rate Monitoring [Online] Retrieved 2022-06-15 from https://e2e.ti.com/blogs\verb|_|/b/behind\verb|_|the\verb|_|wheel/posts/ti-mmwave-technology-for-car-interior-sensing.

[10] HARKEN [Online] Retrieved 2022-06-15 from https://harken.ibv.org/index.php/about

[11] G. Okada, T. Yonezawa, K. Kurita, and N. Tsumura, "Monitoring Emotion by Remote Measurement of Physiological Signals Using an RGB Camera," ITE Trans. MTA, 6, pp.131–137, 2018.

[12] Y. Lee, JY. Park, and YW. Choi, "A Novel Non-contact Heart Rate Monitor Using Impulse-Radio Ultra-Wideband (IR-UWB) Radar Technology," Sci Rep 8, 13053, 2018.

[13] W. Lv, W. He, X. Lin, and J. Miao, "Non-Contact Monitoring of Human Vital Signs Using FMCW Millimeter Wave Radar in the 120 GHz Band," Sensors (Basel, Switzerland), 21(8), 2732, 2021.

[14] P. V. Rouast, M. T. P. Adam, R. Chiong, D. Cornforth, and E. Lux, "Remote heart rate measurement using low-cost RGB face video: a technical literature review," Frontiers of Computer Science, 12(5), pp.858-872, 2018.

[15] A. Bella, R. Latif, A. Saddik, and F. Z. Guerrouj, "Monitoring of Physiological Signs and Their Impact on The Covid-19 Pandemic: Review," E3S Web of Conferences, 229, 01030, 2021.

[16] W. Wang, A. C. den Brinker, S. Stuijk, and G. de Haan, "Algorithmic Principles of Remote PPG," IEEE Transactions on Biomedical Engineering, 64(7), pp.1479–1491, 2017.

[17] T. Ysehak Abay, K. Shafqat, and P. A. Kyriacou, "Perfusion Changes at the Forehead Measured by Photoplethysmography during a Head-Down Tilt Protocol," Biosensors, 9(2), 71, 2019.

[18] R. H. Goudarzi, S. Somayyeh Mousavi, and M. Charmi, "Using imaging Photoplethysmography (iPPG) Signal for Blood Pressure Estimation," 2020 International Conference on Machine Vision and Image Processing (MVIP), 2020.

[19] G. De Haan, and V. Jeanne, "Robust Pulse Rate From Chrominance-Based rPPG," IEEE Transactions on Biomedical Engineering, 60(10), pp.2878-2886, 2013.

[20] W. Wang, S. Stuijk, G. de Haan, "A Novel Algorithm for Remote Photoplethysmography: Spatial Subspace Rotation," IEEE Transactions on Biomedical Engineering, 63(9), pp.1974–1984, 2016.

[21] H. Rahman, M. U.Ahmed, S. Begum, and P. Funk, "Real Time Heart Rate Monitoring from Facial RGB Color Video using Webcam," The 29th Annual Workshop of the Swedish Artificial Intelligence Society SAIS 2016, 129.

[22] M. K. Hassan, A. B. Malik, D. Fofi, N. M.Saad, and F. Meriaudeau, "Novel health monitoring method using an RGB camera," Biomedical Optics Express, 8(11), 4838, 2017.

[23] V. Jeanne, M. Asselman, B. den Brinker and M. Bulut, "Camera-based heart rate monitoring in highly dynamic light conditions," 2013 International Conference on Connected Vehicles and Expo (ICCVE), Las Vegas, NV, USA, pp. 798-799, 2013.

[24] K. Y. Lee, C. Park, and B. Lee, "Tracking driver's heart rate by continuous-wave Doppler radar," International Conference of the IEEE Engineering in Medicine and Biology Society, 2016.

[25] H. Xu, M.P. Ebrahim, K. Hasan, F. Heydari, P. Howley, and M.R. Yuce, "Accurate Heart Rate and Respiration Rate Detection Based on a Higher-Order Harmonics Peak Selection Method Using Radar Non-Contact Sensors," Sensors 2022, 22, 83.

[26] A. B. Shetty, Bhoomika, Deeksha, J. Rebeiro, and Ramyashree, "Facial recognition using Haar cascade and LBP classifiers," Global Transitions Proceedings, 2(2), pp.330–335, 2021. https://doi.org/10.1016/j.gltp.2021.08.044

[27] A. Saddik, R. Latif, A. Elouardi, M. A. Alghamdi, and M. Elhoseny, "Improving Sustainable Vegetation Indices Processing on Low-Cost Architectures," Sustainability, 2022.

[28] H. Rana, and H. Sirohia, "Comparative Study Between Canny and Sobel Edge Detection Techniques," 2022.

[29] X. J. Jiang, and P. J. Scott, "Characterization of free-form structured surfaces," Advanced Metrology, pp.281–317, 2020.

[30] S.Suzuki, and K. be, "Topological structural analysis of digitized binary images by border following," Computer Vision, Graphics, and Image Processing, 30(1), pp.32–46, 1985.

[31] A. M. S. Khairul, and B. J. Asral, "A study of image processing using morphological opening and closing processes," International Journal of Control Theory and Applications, 9, pp.15-21, 2016.

[32] W. Verkruysse, L. O. Svaasand, J. S. Nelson, "Remote plethysmographic imaging using ambient light," Optics Express, 16(26), 21434, 2008.

[33] A. Saddik, R. Latif, and A. Bella, "ECG signal monitoring based on Covid-19 patients: Overview," Journal of Intelligent Systems and Internet of Things, Vol. 2 , No. 2 , pp.45-54, 2021.