Measuring heart rate using computer vision techniques on Android based mobile phone

MSc Project report

Author:

Weihua Lin

ID: H00141698

Supervisors:

Dr Deepayan Bhowmik

Prof Andrew Wallace

Vision lab

Institute of Sensors, Signals and Systems

School of Engineering and Physical Sciences

Heriot-Watt University

August 2014

Declaration

I, Weihua Lin, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the words of other authors in any form (e.g. ideas, equations, figures, text, tables, programs) are properly acknowledged at any point in their use. A list of the references employed is included.

Signed: .....

Date: .....

Acknowledgements

I would like to thank my supervisor Dr. Deepayan Bhowmik and Prof Andrew Wallace for the patient guidance and continuous support he has provided through the course of my study at Heriot-watt University. His extensive research experience and vision has been the source of inspiration throughout my work.

I am especially indebted to Sthapit Saurav for the in-depth discussions which have made significant contribution to this work. I wish to extend my thanks to my friends and for their valuable comments and support for my experiment.

I would like to express my gratitude to Yu Fu, my husband for his care, continued support and encouragement. He has always been the source of inspiration for me. My special thanks to my parents for their love and understanding.

I acknowledge the support of the Engineering and Physical Research Council through ‘Project Rathlin’, grant references EP/K009931/1 (Programmable embedded platforms for remote and compute intensive image processing applications).

Contents

List of Figures

List of Tables

Glossary

HR	Heart Rate
HRV	Heart Rate Variability
RHR	Resting Heart Rate
PPG	Photoplethysmography
ANC	Adaptive Noise Cancellation
ICA	Independent Component Analysis
ROI	Region of Interest
RGB	Red, Green and Blue
CIE LUV	A color space adopted by the International Commission on Illumination (CIE)
OpenCV	Open Computer Vision
JADE	The Joint Approximate Diagonalization of Eigenmatrices
GPU	Graphics Processing Unit
CCD	Charge-Coupled Device
PC	Personal Computer
CPU	Central Processing Unit
BSS	Blind Source Separation
bpm	Beat Per Second
RAM	Random-Access Memory
DFD	Data Flow Diagram
SDK	Software Development Kit
UI	User Interface
API	Application Programming Interface
JNI	Java Native Interface
RMSE	Root Mean Squared Error
FPS	Frames Per Second
SD	Standard Deviation

Abstract

Non-contact and non-invasive measurements of heart rate (HR) can provide more convenience and comfortable assessment experience. In this project, a mobile phone with camera is used as a remote monitor for HR, based on its powerful processing and video recordings capabilities. This approach bases on PhotoPlethysmoGraphy (PPG), a technology can sense the cardiovascular pulse wave by observing light variations on the skin. The camera of mobile phone is used to recording the human face, and using face tracking technologies for face detection in order to collect optic information of face skin. Independent component analysis (ICA) technology is applied to decompose the optic information into independent components, from which the HR frequency can be extracted

The accuracy of measurement of HR against a HR sensor is evaluated by using correlation and Bland-Altman analysis. The mean ± standard deviation (SD) of the differences between pulse measurements from our application and from other sensor is -0.78 ± 2.83 when the participants were sitting still and -1.4 ± 3.62 in the presence of motion artifacts. The correlation coefficient (r = 0.9313, 0.911 respectively) demonstrated a high correlation between the two measurements. Aforementioned mathematic analysises show a good accuracy of our application.

Key Word: hear rate (HR), PhotoPlethysmoGraphy (PPG), mobile phone, independent component analysis (ICA)

Chapter 1

Introduction

1.1 Context

As one of the simplest cardiovascular parameters, resting heart rate (RHR), has attracted lots of attention due to that high heart rate has been considered as a cardiovascular risk factor [1, 2]. RHR has been proved as a prediction factor of heart disease such as coronary disease and myocardial infarction, as well as diabetes mellitus or hypertension [1]. Moreover, elevated RHR has been demonstrated that it boost the risk of death in different subgroups, for instants, hypertensive, coronary disease patients and even the general population [1, 3]. Various methods for monitoring hear rate (HR) have been proposed, including electrocardiogram, contact sensors and non-contact measurements [4]-[8]. However, in some cases the range of application the contact sensors will be limited. For example, in the case of real-time monitor, the contact sensors can disturb the daily life of the patient. Moreover, adhesive gelling or chest traps were required in some contact sensors that can lead to discomfort and skin irritation.

Non-contact and remote measurements of the physiological parameters can be realized by Doppler radars technology. Non-contact method of detect heartbeats using Doppler radars has proposed in [4] and this technology can also be used to detect respiration signals [5]. Likewise, thermal infrared imaging technique was used to detect stress by collecting physiological data on human faces [6]. More recently, Digital cameras and webcams are used to measurements the HR [7] and heart rate variability (HRV) to detect mental stress [8]. This method bases on PhotoPlethysmoGraphy (PPG), a technology can be employed to sense the cardiovascular pulse wave by observing light variations on the skin.

In recent years, smart phones are becoming more powerful and more popular, which also have been explored as devices in healthcare applications. As smart phones have innate advantage such as inexpensive, portable, multipurpose and ubiquitous, it can be widely used for applications in home-based personal care.

Currently, the method non-contact and remote measurement HR using time-lapse image are almost implemented on computers, which with the limitation of poor mobility and can be only carried out at laboratory environment. Moreover, most approaches are not real-time processing, experiment videos need to be recorded firstly, and then the videos are processed by computer to get the HR. In some references [9, 10], smart phones have been employed to detect HR by collecting finger colour changes [9, 10]. In these cases, a white light should emit diode and the next camera in the mobile phone is using for reflection-mode bio-optical sensor. The PPG information then can collect by putting finger on the diode and the camera. But the method is contact and lack of expansibility. Moreover, it is difficult to combine with other application or to extend with new capabilities.

In order to fill the reach gap, in this project, we develop an application on the smart phone, which can be used for real-time non-contact, motion-tolerant and automated HR detecting.

1.2 Description of the Project

This project is to develop a dedicated application on the smart phone platform, which can be used for non-contact, motion-tolerant and automated HR detecting. This application bases on PPG, face tracking and Blind source separation technology. The camera of mobile is used to recording the human face, and using face tracking for facial detection in order to collect optic information of face skin. Blind source separation technology is employed to decompose the optic information into independent components. Then the HR frequency can be extracted from the independent components.

Over all, the main contributions and novelties and the advantage of this project are highlighted as following:

1. The develop platform is smart phone. Smart phones are inexpensive, portable, multi-purpose supporting and becoming nearly ubiquitous, which can be widely used for applications in home-based personal care.
2. Our system is a real-time application. It can be conveniently carried out for real-time monitor without disturbing the daily life of the patient.
3. The face detecting is used in our project, which make our application with the capability of capturing the face automatically.
4. The application has a good scalability. It can be easily combined with new features, such as facial expression recognition to detect human emotion and stress, or extend to measurement multi people's HR at the same time.

1.3 Dissertation Overview

The remainder of this dissertation is organized as following, which is also the reflection of develop procedure of our application.

- In chapter 2, we will discuss the project background with the foundations of cardiac pulse measurements using by PPG technology. We will introduce how to use the PPG technology. Then a widespread literature review, which researches the similar projects, will be carried out and their method will be analyzed in the following part.

- In chapter 3 we will discuss a statistical technique, the independent component analysis (ICA), which attempts to reveal the independent source signals from a set of observed mixtures. The how the ICA works and the JADE based implementations of ICA will be analysed.

- In chapter 4, the design of the application will be discussed. We will analyse the requirements of the application, including functional requirements and

non-functional requirements. Then the data flowchart, a graphical description of data through a system, of our application is displayed. Using the data flow diagram, the architecture of the software is designed. The structural of the program and behavioural organisation is explained.

- We will explore implement our project in Chapter 5. Equipment and the development environment will be discussed. Then we will give an overview of the different module development. This will includes the implementation of the arithmetic, the multiprocessing development and the use of java native interfaces to call dynamic library.

- In the following chapter 6, we will describe the test program. The application is tested and the measurement result will compare with the result obtained from monitor using Bland-Altman method. The robustness of our system for HR measurements in the presence of motion artifacts were also evaluated

- In the last chapter, we summarize the process of project. Besides a final conclusion of this project, an overview of possible future work will also be given in this chapter.

Chapter 2

Literature Review

The project background with the foundations of cardiac pulse measurements using by PPG technology will be discussed in this chapter. A widespread literature review, which research the similar projects, will be carried on and their method will be analysed in the following part.

2.1 Photoplethysmography (PPG)

The PPG, a low-cost and portable technique that can measure blood volume changes through collecting the variations in reflected or transmitted light, is used for non-invasively physiological measurements [11]. The worthy information regarding to the cardiovascular system can be provided by this optic technique, for example the blood pressure, blood oxygen saturation, HR, HRV and cardiac output [11]. Recently, The PPG technique attracts lots of research interest again due to the developments of digital cameras and software data analysis tools. Some researchers have used digital cameras to detect HR by PPG technique using normal surrounding light as the lighting source [7, 8, and 12]. The PPG was also used on smart phone to monitor physiological varying [9, 10].

However, it is known that PPG is quite sensitive to motion artefact, and overcoming the motion artefact is treated as the most challenging problems for PPG technologies [7, 11]. Furthermore, it is ineffective using linear filtering with cut off frequencies due to the frequency band of noise is the same with the physiological signal of interest. Adaptive noise cancellation (ANC), which employs accelerometers as a noise reference, is proposed to reduce the affection of motion artefact [13]. Similarly, ICA, which is used for discovering the independent source signals, has been employed to decrease motion artefacts in PPG measurements [7]. Some other

researchers chosen the CIE LUV colour to reduce fluctuations due to light variation or head movements [8].

2.2 Measurement Hear Rate by Video

In most references [7, 8, 12], the main equipment is camera for recording the videos and personal computer (PC) for image processing. The participants need to seat in front of the camera at a distance of around 0.5 meter for about a minute.

The general steps can be shown by the following figure.

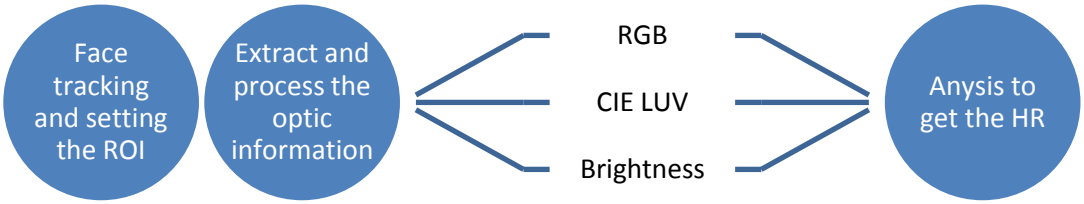

Figure 1 Flow chart of HR measurement

The general principle and procedure of the measurement can be described as following.

Firstly, face tracker is used for face detection in the video and setting the measurement region of interest (ROI) for each frame. In most literatures [7, 8], cascade of boosted classifier with Open Computer Vision (OpenCV) is widely used for face detecting. Then the algorithm isolates skin pixels (the ROI) which contain the PPG signal from the original frame.

Secondly, the optic information in the ROI is extracted. Various optic parameters were used, such as red green and blue (RGB) colour model [7], the International Commission on Illumination (CIE LUV) [8] colour space and brightness [12].

Finally, the optic information is processed to get the HR. Different technologies are used to get cardiac pulse, including ICA for decreasing motion artefacts [7], spline interpolation for correcting and regulating the measured illumination [12], Continuous

Wavelet Transform for removing trends and high frequency noise [8], and fast Fourier transform (FFT) to get the power spectrum [7].

2.3 Concrete Example Review

In this section, we will review some concrete cases regarding PPG detection in detail. The feature of these examples has shown in the Table 1. In [12], a charge-coupled device (CCD) camera is needed, and there is no great advantage. While the case in [8] has some great advantage like motion-tolerant and automatic, but the accuracy of the approval cannot be guaranteed. However, the example in [7] not only has the motion-tolerant and automatic advantage, but also the guaranty the accuracy.

Author	Equipment	Method & Technology	Advantage
Takano C, Ohta Y [12]	CCD camera PC	Brightness, Spline interpolation, Low pass filter,	Non-contact
Poh M Z, McDuff D J, Picard R W [7]	Webcam Laptop	Face detection ICA FFT RGB	Non-contact Motion-tolerant Automatic High accuracy
Bousefsaf F, Maaoui C, Pruski A [8]	Webcam PC	Face detection Skin detection CIE LUV Wavelet transform	Non-contact Motion-tolerant Real-time Automatic

Table 1 Analyses of HR detecting Methods

2.3.1 Example Regarding Brightness

The approach in [12], the changes of brightness were employed. Firstly the ROI needed be set as a rectangular area in cheek image. Then assemble the average

brightness information of ROI at intervals of about 200 MS. After that spline interpolation is employed to interpolate and regulate the brightness data into those with interval of 100 MS. The following processing includes first-order derivative, a low pass filter of 2 Hz, and Auto-Regressive spectral analysis. Then the peaks of heartbeat and breathing can be observed by the wavelet transform and the AR modelling.

However, in this case the person who was detecting HR need stationary because of the ROI is fixing. Moreover, the conditions of lighting must be strictly controlled due to the auto iris function on the camera, leading to limited environments.

2.3.2 Example Regarding ICA

In [7], the RGB colour model is chosen and ICA is used to decrease motion artefacts in this example and the general steps are shown in figure 2.

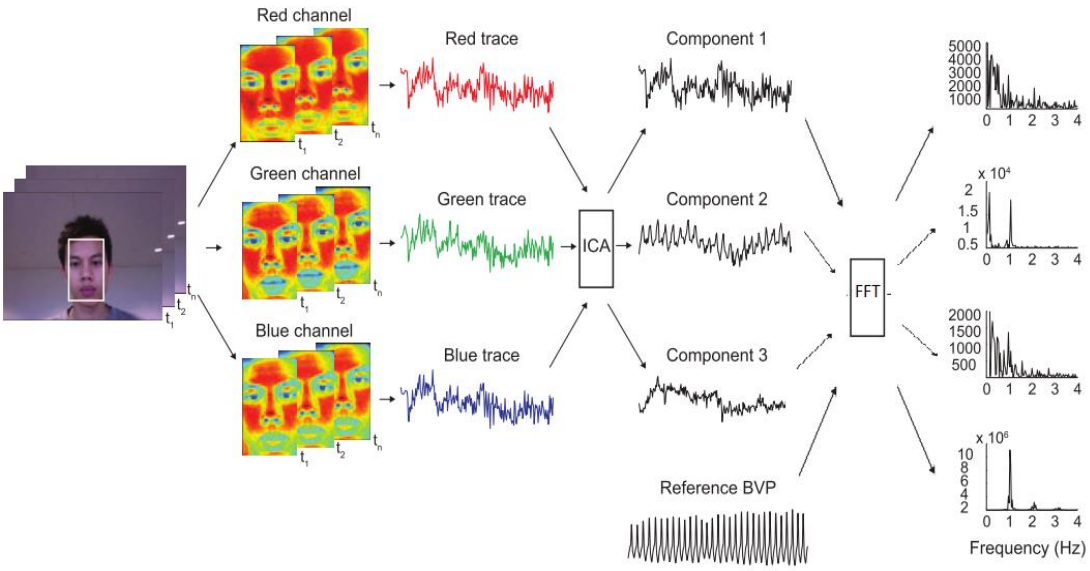

Figure 2 Steps of recovering the blood volume pulse by RGB [7]

Firstly face tracker is employed for face detection in the video and setting the measurement region of interest (ROI) in each frame. The coordinates of face location can be obtained by OpenCV, and the face detection based on a cascade of boosted

classifier. Then the ROI is set with the full height and centre 60% width of face box to wipe off the face segmentation mistakes in order to prove well performance of system.

Secondly, they separated the ROI into the RGB channels and get the RGB measurement value for each frame by averaging over all pixels in the ROI to form the raw RGB traces.

The following processing, including normalizing, ICA and FFT, is performed by a 30 s moving window with 1 s increment. Normalizing the raw RGB traces aims to obtain the zero-mean and unit variance normalized traces. Then the joint approximate diagonalization of eigenmatrices (JADE), one method of ICA, is used to decompose the normalized RGB into three independent source signals. Finally, the power spectrum of the second component source signal is obtained by the FFT. Then a peak match the HR frequency can be seen, which is closely agree with the reference that finger blood volume pulse signal, in the operational range [0.75, 4] Hz.

In this case, the participants can move their body or head slowly due to the robustness of the proposed methodology. The ICA technology also can increased the level of correlation, the result showed high accordance with finger sensor result the average of the different is -0.05 ± 2.29 bpm when the participants were motionless and 0.64 ± 4.59 bpm while motion artifacts.

2.3.3 Example Regarding CIE LUV

In [8], the CIE LUV colour space is used to reduce fluctuations. They point that the PPG information will be more sensitive when using the U component the indicator of red to green colour, that is result of Oxy and feaeration haemoglobin have a better absorption coefficient in the wavelength of green colours. The processing the system is illustrated in figure 3.

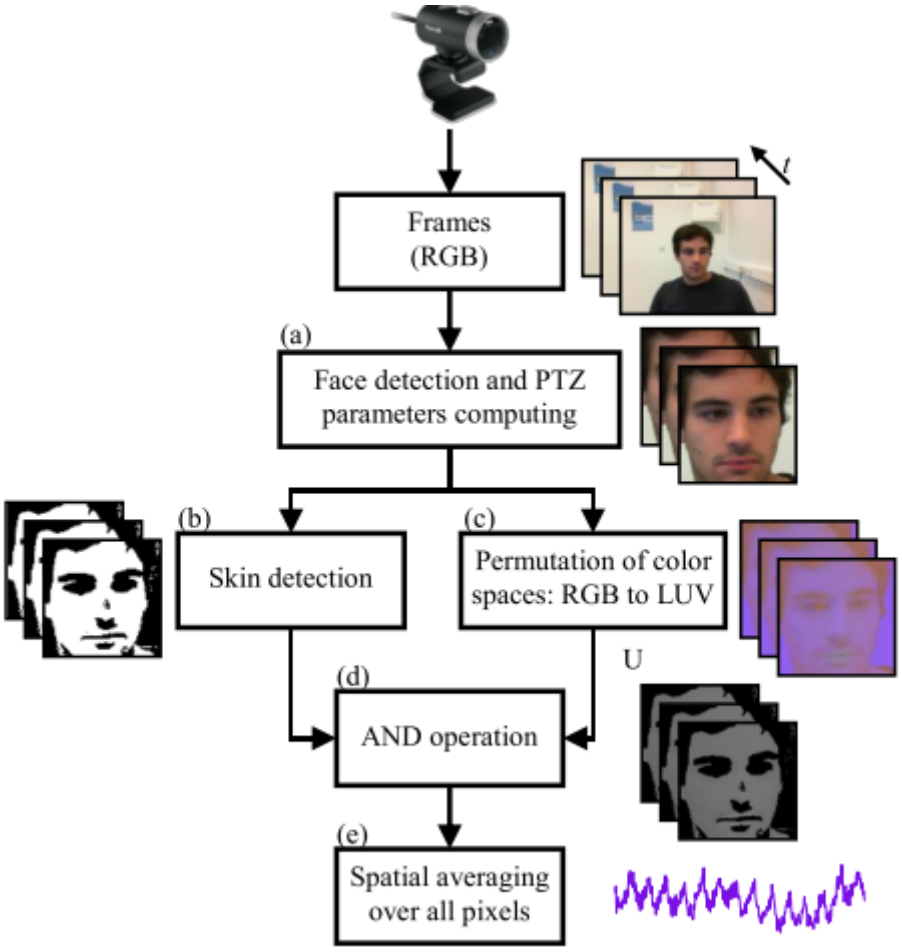

Figure 3 The processing of detection HR by CIE LUV [8]

Firstly, a cascade of boosted classifier is employed on each frame to detect the face. Then set the pan and tilt parameters for tracking the face with the passing of the time. Secondly, skin detection is applied on the face to extract the skin pixels, and set these pixels as ROI. At the same time RGB information is changed to LUV information. After that the U component value of every frame is obtained by spatial averaging the U component of all pixels in the ROI. Then the raw signal source is formed by a set of frames. The raw signal source is operated by a Wavelet Transform filtering to deplete high frequency noise and trends. Then instantaneous HR trace can be computed by detecting peaks.

In this example, CIE LUV colour space is used to reduce fluctuations. This is based on the feature of PPG information that is more easy reaction on the U

component. However, the paper did not accuracy test to compare with any other HR sensor, so we do not know the performance of the system.

2.4 Smart phone

These methods in the above examples all run in PC or laptop, and some algorithm computational complexity greatly, like ICA, FFT and face detection. So the device of our project should have camera and powerful computation.

However, Mobile processing capabilities and memory specifications are speedily advancing making more processor-intensive applications possible. For example, Sony Xperia Z1 compact included a Qualcomm Anapdragon 800 2.2GHz quad-core data processing unit (CPU), 2GB random-access memory (RAM) and an award-winning G Lens. The Samsung Galaxy S4 has two processors including a quad-cord 1.9 GHz Krait 300 CPU and an Adreno 320 GPU, 13MP camera and 32 GB internal storage. This renders modern smartphones can process many tasks that were done on laptops or desktops such as playing full high-definition videos, editing pictures and play 3D games. However, compared with the PC platform, the mobile phone platform has many limitations such as computational power and input modalities. Moreover, the GPU has become into a greatly powerful co-processor. With the emergence of programmable GPU, application developers used the GPU to moderate the burden form CPU. So if necessary the GPU can be applied to accelerate in order to realize the real-time application in our project.

Moreover, smartphone ownership has been reported sharp growth and Android has become the most popular development. According to Go-Gulf.com (2012) [24], there are over 1.08 billion people use smartphone in the world. ABI Research (2013) [25] claimed that by the end of 2013 the total number of the active smartphones will reach 1.4 billion and 57% of this base will use Android. This means more and more people use their phones as primary types of communication. Additionally,

Besides, the smartphone have been investigated as a device in healthcare applications [10]. This is due to cell phone are getting more advanced with higher processing power and greater resolution on their cameras. Recently, smart phone was also to monitor physiological varying by using the PPG technology [9]. Mobile phone can accurately detect physiological parameter, consider its ability to assemble and process the difference colour signals. Similar, smart phone is employed to analysis HR and HRV by recording finger tips [10]. However, this application is contact and lack of expansibility. The approach only simply recording finger tips to collect PPG information then detect physiological parameter, but it is difficult to combine with other application or to extend with new capabilities.

Smart phones are becoming more powerful and more popular, leads to they are not only phones but also processor. Moreover, smart phones are inexpensive, portable, multi-purpose supporting and becoming nearly ubiquitous, which can be widely used for applications in home-based personal care.

2.5 Chapter Summary

In this chapter, the foundation of PPG technology was discussed and various concrete methods were analysed. Some complicated algorithm, such as a cascade of boosted classifier applied to detect face, ICA used to decrease motion artefacts in PPG measurements and FFT employed to obtain the power spectrum, will be analysed in the following chapter.

Chapter 3

Algorithmic Learning

In last chapter, we have analysed various concrete methods of measurement HR. The approach in [7] was adopted due to the advantages of high accuracy, non-contact and motion-tolerant. In this chapter, we will study the complicated algorithm in this method, includes the cascade of boosted classifier, ICA and FFT.

3.1 A Cascade of Boosted Classifier

Booting, a general method, which can be used to improve the performance of learning algorithm by reduce the error of “weak” algorithm [27]. It is through the repeated operation of a given weak learning algorithm on varieties of distributions over the training data, and then combining classifier produced by the weak learner for a combined classifier.

A cascade of classifiers is a degraded decision tree [26]. It uses a group simple classifier can be constructed which reject many of the negative sub-windows while detecting almost all positive instances (figure 4). In each stage, a classifier will be trained using boosting to build a weighted voted. Each stage acts as a filter, rejecting a grand number of easy cases, and passing the hard cases to the next stage. The stages become progressively more expensive, but are used progressively less often. Globally the computation cost decreases dramatically.

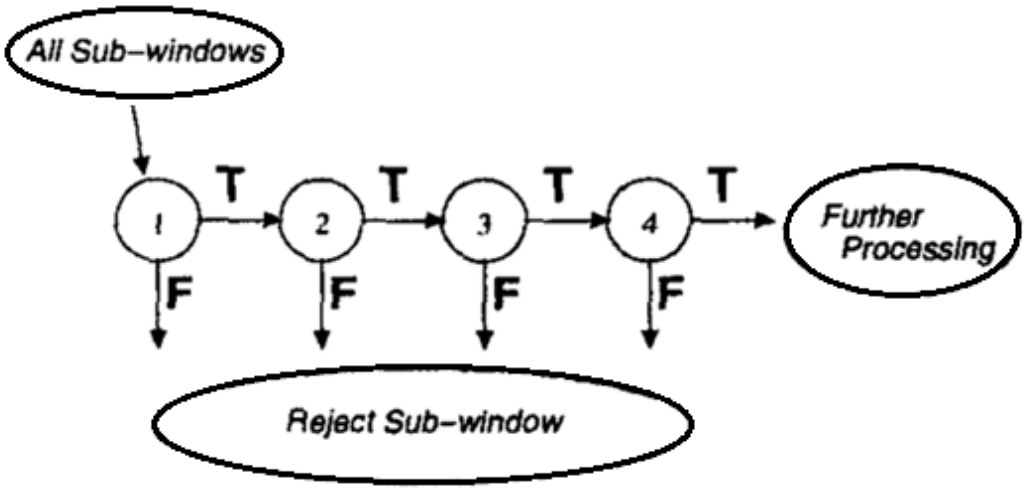

Figure 4 Cascade of classifiers with N stages [20]

In our project, the classifier is used to frontal faces detecting. In each stage, it will decide if the region of interest contains a face, only events labelled as yes are passed to the next stage else the area is rejected. In order to identify the positive matches of different sizes, the dimensions of the area of interest is changed sequentially.

3.2 Independent Component Analysis (ICA)

ICA, a statistical technique that attempts to uncover the independent source signals from a set of observed mixtures, is used for settling the blind source separation (BSS) [14, 15]. BSS aims to recover unobserved signals from a set of observations that assumed to be linear mixtures of the some underlying sources. The “blind” property of the BBS relates to the fact that nothing is known about the source signals and no information is available about the way they were mixed. The “cocktail party problem” can interpret the BBS very well [18]. In this model, a group of microphones ranged in a cocktail party room, in which people are talking simultaneously, records sound that is mixture of people’s voice in the room (figure 5). The BBS is used to separate out each voice into its own signal from the recording obtained from all microphones, to remove pollution from any other person's voice, and completely match the original sound.

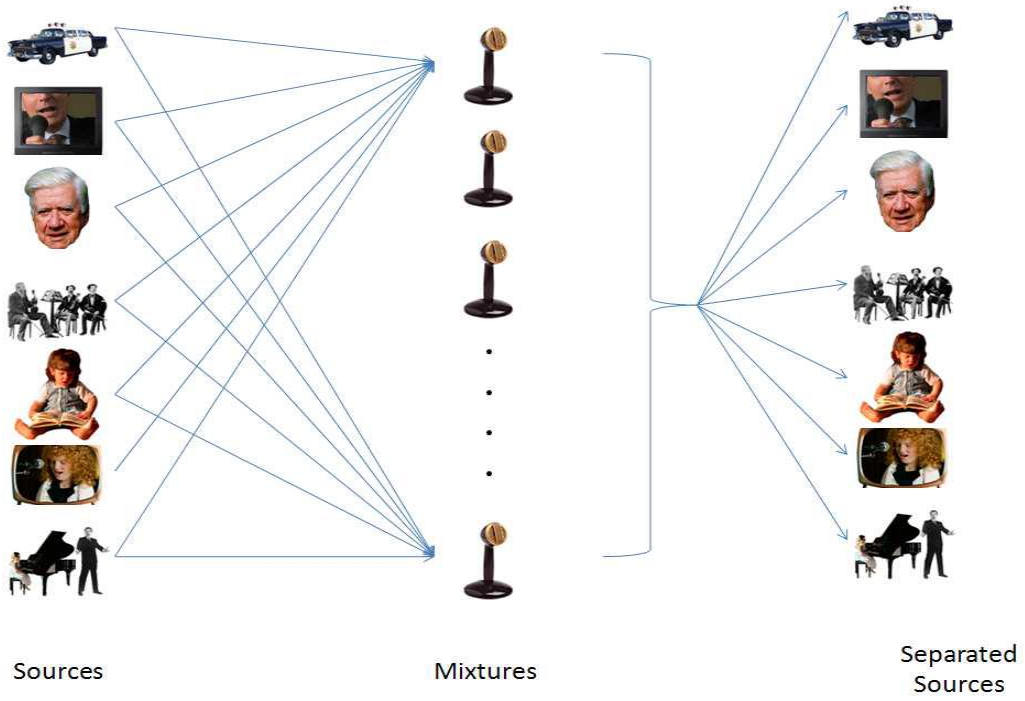

Figure 5 The cocktail part problem [18]

There are several approaches of BSS and we only focus on ICA, one simple form of BSS. In order to resolve the BSS problem, ICA assumes the original signals are stationary and non-Gaussian random variables and they are statistically independent of each other. The following equation was assumed the ICA situation:

$$X = AS \tag{1}$$

Here, the random vector A denotes the observed signals, the random vector S is representing the original source signals, and the Matrix A is the linear transformation matrix which mixes the source signal.

Then, after estimating the matrix A, the independent component can be obtained by multiplying with the inverse matrix, call M, of A:

$$\tilde{S} = WX \tag{2}$$

Where $\tilde{S}$ is an evaluation of vector S which involving the potential source signals. The distribution of a sum of independent random variables is closer to Gaussian than

any of the original variables, in accordance with the Central Limit Theorem. Therefore, in order to reveal the independent sources, W must maximize the non-Gaussian of $W^{T}X$. We focus on the JADE, one implementations of ICA [16]. In the next part, we will discuss the implementations of ICA.

3.2.1 Pre-processing: Whitening

Before the ICA processing, some pre-processing work, such as whitening the observed variables should be carried out to make the ICA estimation simpler and better conditioned. The principle of whitening is to transform a set of random variables into several new uncorrelated random variables and all with zero-mean and uniform variance. This pre-processing simplifies the statistical signals and reduces the search for W into a search for an orthogonal matrix which will maintain the irrelevance of the variables and maintain each variable has unit variance.

3.2.2 Joint Approximate Diagonalization of Eigenmatrices (JADE)

The aim of JADE is to find a transformation matrix B, which used to get the estimate source signal S transformed from the whitened observations Z [19]. The first step is using the statistics of the observation source signal to generate a set of matrices. The second step is to calculate the orthogonal transformation matrix B which can diagonalize the total set of matrices at the same time.

The join fourth-order cumulate of the whitened observation signals is the statistics used by JADE, and the joint fourth-order cumulate is composed of a set of four, zero-mean random variables, given by

$$\mathrm{cum}(Z_i, Z_j, Z_k, Z_l) = E\{Z_i Z_j Z_k Z_l\} - E\{Z_i Z_j\}E\{Z_k Z_l\} - E\{Z_i Z_k\}E\{Z_j Z_l\} - E\{Z_i Z_l\}E\{Z_k Z_j\}$$

(3)

The principle of JADE is to create a set of matrices which are symmetric. For these matrices, the form $\mathrm{cum}(x_i, x_i, x_i, x_i)$'s own joint fourth-order cumulants are treated as the diagonals. Besides, the form $\mathrm{cum}(x_i, x_j, x_k, x_l)$'s own joint fourth-order cumulants, where at least one of i, j, k, l is different from the other three, are treated as the off-diagonals. The next step is "Joint Diagonalization". A single matrix, which can simultaneously diagonalizes all of the above cumulant matrices, is built by using Jacobi rotations. Actually the diagonalization in JADE will be impossible, and the goal of JADE is to obtain the best approach theoretical result.

The foundation of JADE algorithm is illuminated by the Algorithm 3.2, which is given by Dan Brandt in [19].

Algorithm 3.2 The JADE Algorithm

Require: z is the whitened observation vector of n random variables

Ensure: s is the vector of independent source signals

1: Generate a set of cumulant matrices from the z observations

2: **repeat**

3: **for** every pair of rows i and j, $i \neq j$ **do**

4: Find the Jacobi rotation that will minimize the sum of the ij and ji elements in all cumulant matrices

5: If the rotation angle is above some threshold, perform the rotation

6: **until** no rotations **or** max number of sweeps performed

7: The unmixing matrix, B, is the product of all the performed Jacobi rotations

8: $s \leftarrow Bz$

3.3 Fast Fourier Transform (FFT)

The Fourier transform (FT) is a mathematical transformation employed to transform signals between time domain and frequency domain [28]. In practice, the

transforms often occurs within discrete samples of a signal, and the transform happens for only a discrete set of frequencies. For our project, the RGB information from frames is discrete. This computational method is referred to as Discrete Fourier transform (DFT). The Fast Fourier Transform (FFT) is the fast algorithm for implementing the DFT.

The DFT transform formula is given:

$$F[n] = \sum_{k=0}^{N-1} f[k] e^{-j\frac{2\pi}{N}nk} \quad (n = 0, \dots, N-1) \tag{4}$$

Where f[k] denoted the N samples of f(t), F[n] is the DFT of the sequence f[k]. The F[n] is respond to information about the nth frequency "bin". It corresponds to a Hertz frequency of $f = n * \Delta\omega = n * \frac{F_s}{N}$.

There are two effects introduced into the computation of the DFT aliasing and leakage [28]. The DFT values will be corrupted by aliasing, if the sample frequency is smaller than the high-frequency components in the underlying function. This aliasing effect can be reduced by sampling faster. Secondly, if we truncate a function then make it periodic, the resulting function is going to have additional frequency components in it that were not in the original function, due to the change from end to end. The only way this does not happen is if the signal is periodic with respect to the number of samples already.

3.4 Chapter Summary

This chapter discussed the ICA which is a statistical technique that attempts to uncover the independent source signals from a set of observed mixtures. After studying the main algorithm, we will discuss how the algorithm and the detecting HR method be used in our project. The next chapter the design of the application will be illuminated.

Chapter 4

Application Design

In this chapter, the design procedure of the application will be discussed. Firstly, the requirements of the application, includes functional requirements and non-functional requirements, will be analysed. Then the data flowchart, a graphical description of data through a system, of our application will be displayed. According to the data flow diagram, the architecture of the software will be designed. The structural of the program and behavioural organisation will be explained finally.

4.1 Requirements Analysis

In software engineering, requirements analysis is the process that determining the needs and user expectations for software, which is critical to the success of the project. In this procedure, the functionalities of the application, the measure process and the interface need be defined. This procedure can further divided as functional requirements and non-functional requirements. The functional requirements define the behaviour of the system, for example technical details, data manipulation and processing, calculations and other specific functionality of the system. While the non-functional requirements define how a system is supposed to be, which can be evaluated in terms of efficiency and robustness .et.ac.

4.1.1 Functional Requirements

The functional requirements of the project can be described as following:

1. The program can detect the HR of participant in a short time (30 s) using camera
2. The participant can choose which camera to be used, when the smart phone has several cameras.
3. The program can detect the face and track the face automatically.

4. The face detecting process begins automatically when starting up the application
5. The measurement HR process can be set when start and end is.
6. The value of HR should be shown in the detecting window corner.
7. The whole value of HR in the detecting time should be shown in a graph, when the detecting is end
8. The whole value of HR can be saved with a given name.

4.1.2 Non-Functional Requirements

The non-functional requirements of the project can be depicted as following:

1. The whole measure processing need run in real-time, and the rate of the frame is about fifteen frames per second.
2. The application works in normal lighting conditions.
3. The software should guarantee the accuracy, even the participant slowly moving their heads, doing some facial expression and talking.
4. The software works well, and there is not any unexpected behaviour or system error like system crashes.
5. The interface is accessible by screen, for example the outputs and the menu items can be shown and be operation in the screen.
6. The operation and the menu should be easy to understand.

4.2 Data flow diagram (DFD)

The DFD focuses on the data processing aspects, and it describes what kind of input and output information of system, where the data come from and where to go, what kind of data will be stored and where to store. They are considered as a preliminary step often employed for creating a system overview which can serve for tailor-making.

In this section we use the Microsoft Visio to build the DFD. The DFD has three levels: the top-level diagram (Level 0) and lower level diagrams (level 1 and level 2).

The main activity of the system is detecting HR. So the top-level diagram is given in figure 6. In the top-level diagram, we can see the main aim of the system is to detect HR, and the input data is frames from the camera, the output is HR of participant.

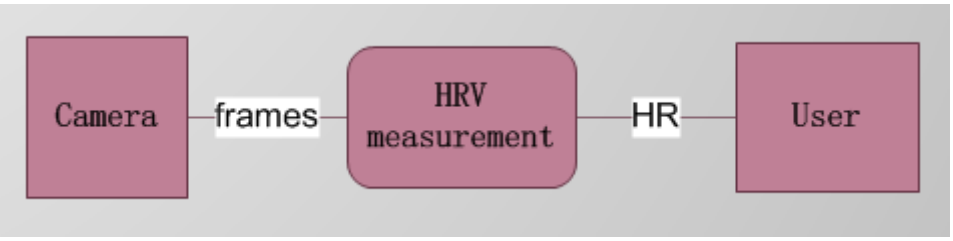

Figure 6 The top-level of DFD

In the second level diagram (figure 7), there are three main processing in our project including face detection to get the ROI, processing ROI to get RGB information and analysis the RGB traces to get HR. The input data of the first processors is frames from camera, and then the output is ROI which is the input data of the second processor. RGB information for every frame will be yielded after the second processor. Then these RGB information will be save in list to form RGB traces. Here is the import point in the system, due to the first two processor processing data frame by frame and the next processor is processing the data formed from frames in a 30 s moving window with 1 s increment. Thus the time period of the first two processing should be shorter than that of the frame capturing, and then in our system the rate of frames at least achieve 15 frames per second. While the time period of the third processing, analysis the RGB traces to get HR, should be less than one second.

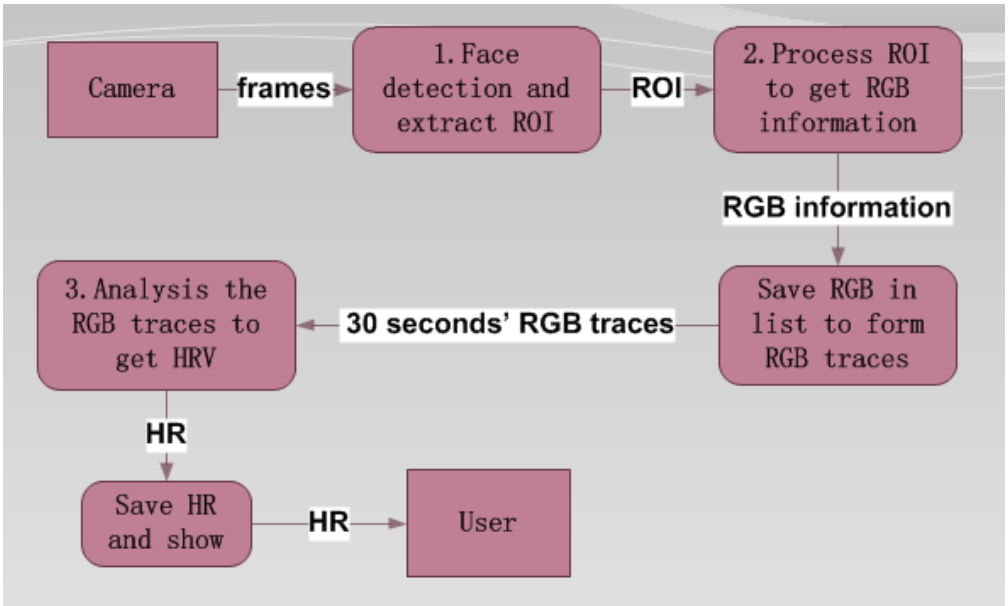

Figure 7 The second level of DFD

In the third level diagram shown in figure 8, move detail of the main three processing can be seen. The processing one can be divided into the two departments: face detected and setting the ROI. The cascade of boosted classifier we shown in the last chapter will be used to detected face. The centre 60% width and 60% height of the face area will be picked on as the ROI. We can see ROI is separated into RGB channels in the second processing first, and then the RGB information is extracted by spatially average over all pixels in the ROI. The third processing, the most important and most complex part, is separated into four steps: normalized, ICA, FFT, getting the HR. The normalization is to transforms raw RGB traces to normalized raw traces that has unit variance and is zero-mean. After normalizing, the RGB normalized traces will be decomposed to three independent signals via ICA. The ICA is employed to decrease motion artifacts and remove the noise in order to increase the accuracy. The FFT is applied on the independent source signals to get the power spectrum. A handling range to [0.75, 3.5] Hz matching to [45, 210] beat per second (bpm) is set in the power spectrum. The highest of power of the spectrum in the operational range will be the pulse frequency.

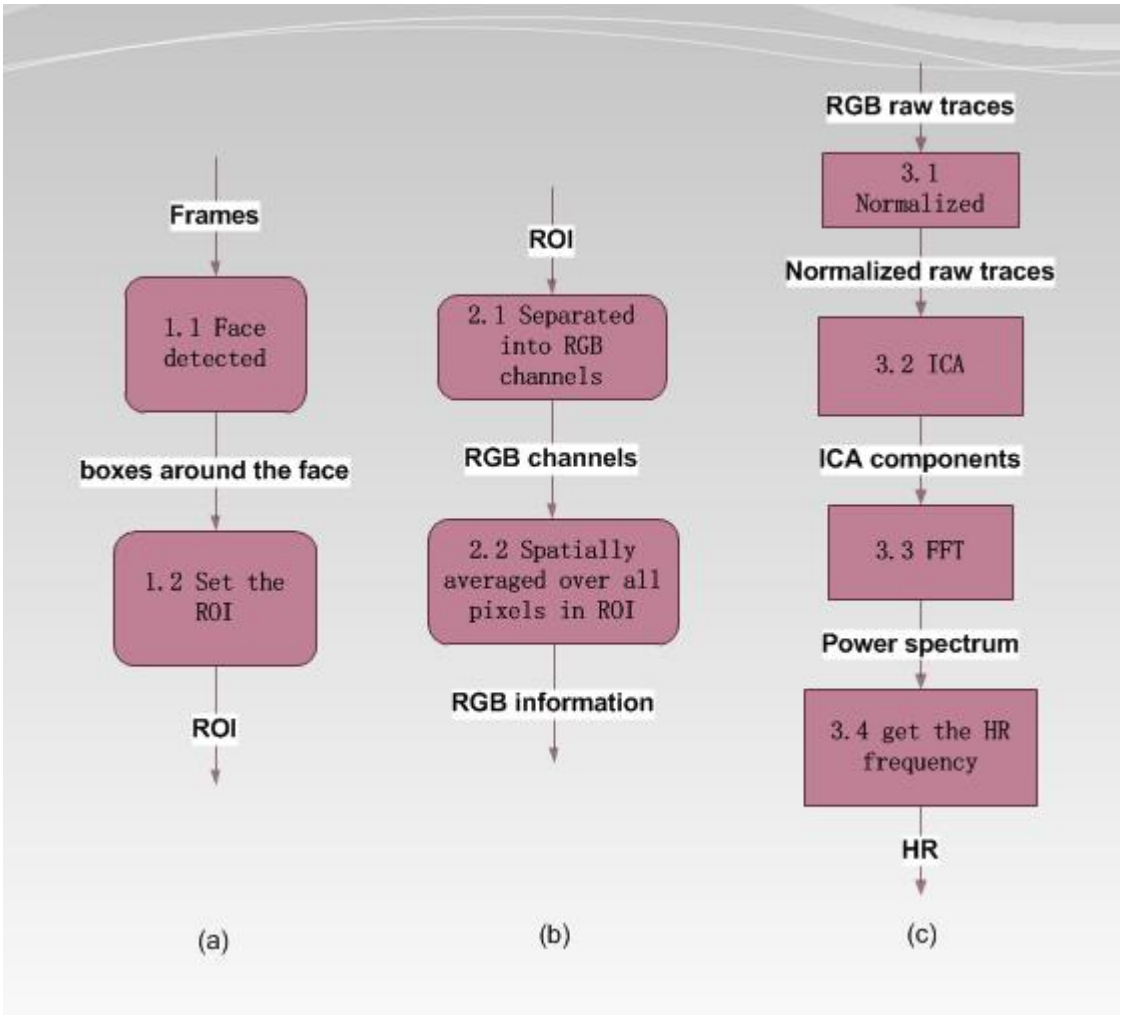

Figure 8 The third level of DFD

According to the DFD, the data processing can be organised into two main parts. One is about the PPG information collection, including face detection, set ROI, get and save RGB information to form RGB traces. The other refers to analysis the RGB traces to get HR, consist of normalized RGB traces, ICA, FFT and get the HR. Moreover, these two phases should be parallel execution. The first one processing must run following every frame capturing closely. The second processing runs when the 30s moving window increment second by second. Their periods, the beginning time and conditions are all different.

4.3 Software Architecture

The basic structure of our application will be outlined in this section. The software will be organised as a hierarchical system. The general hierarchies is given by figure 9, including the user interface (UI) layer, processing logic layer (include RGBExtraction and RGBProcessor), data structure layer, data server layer and library layer.

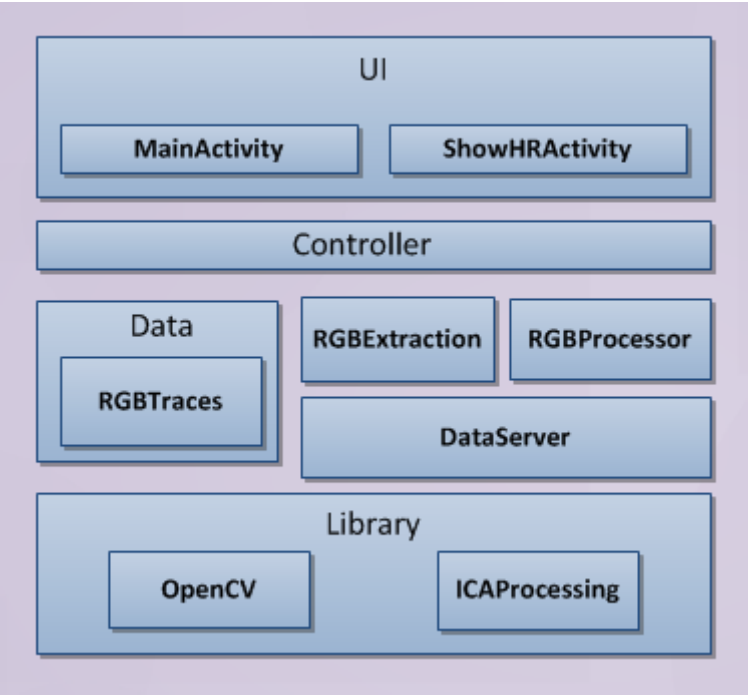

Figure 9 The structure of the software

In the UI interaction layer, two activities and the controller will be involved. In the android system, the Activity class deals with the task of creating a window in which the UI can be displayed. The MainActivity and his controller take care of camera managing, frames capturing, displaying, and the menu managing. The show-HR-Activity deals with the task of showing the whole HR value in the last time detection and the interaction of save data.

According to the DFD the data processing can be divided into two relatively independent parts. Thus the processing logic layer consists of RGBExtraction that deals with PPG information collection and RGBProcessor that handles the PPG

information processing

The data structure layer is related to data definition storage format, like RGB traces and HR values. The DataServer layer offers some common public operation of data, such as save data to SD card, change data type and some interface of library. The library layer including OpenCV library, ICAProcessing , some fundamental function can be called through this layer.

4.4 Chapter Summary

In this chapter, the design procedure of the application was discussed. According to the DFD the data processing can divide into two relatively independent parts: PPG information collection and PPG information processing which is running parallel due to their periods and conditions are all different. Moreover, the general structure of system can make up the UI interaction layer, processing logic layer, data structure layer, data processing services layer and library. In the following chapter, we will discussed how the program to implement.

Chapter 5

System implementation

In the previous chapter, we discussed the procedure of designing this system. In this chapter, we will explore the implementation of our system. Firstly, we will outline the equipment and the development environment. Then we will give an overview of different module developments. This will include the implementation of the arithmetic, the multiprocessing development and so on.

5.1 The Device of the Project

The smart phone we used in our project is a Sony Xperia Z1s that has a great camera and powerful processor. The sensor size of camera is 1/2.3’’, as same as the common sensor size that are used in bridge camera. This smartphone is equipped with Qualcomm’s quad-core Snapdragon 800 processor with the clock frequency of 2.2 GHz and 2GB RAM. For better image and video displaying, X-Reality Engine and a 5.0 inch Sony Triluminos are used this smart phone.

5.2 Development Environment

The development platform of our application is android system, which is the world’s most popular operating system for mobile devices. Android is a Linux-based and open source operating system developed by Google. The Android software development kit (SDK) is also offered by google which not only provides the application programming interface (API) libraries to developer, but also developer tools to build, test, and debug apps for android.

For some imaging processing like face detection and FFT, the Open Computer Vision (OpenCV) library will be used. OpenCV, which is widely used in both academia and industry, provides building blocks for computer vison experiments and

applications [21]. It is an open-source, cross-platform library that offers high-level interfaces for capturing, processing, and presenting imaging data.

Our development environment includes the following components:

- Java Development Kit 7 which includes tools for java programming
- Cygwin 1.7.32 that provides Unix-like programming tools on Windows system.
- Android SDK that provides tools for development Android application in Java.
- Android Native Development Kit (NDK) that offers tools for programming Android application in C++.
- Eclipase that is an integrated development environment
- Some Eclipse plugins include Java Development Tools for Java programming, C/C++ Development Tools for C/C++ programming and Android Development Tools for Android programming
- OpenCV for Android SDK that provides OpenCV's Android version for Java and C++ libraries.

5.3 Image Capturing

OpenCV abstracts away details about camera hardware and memory allocation. CameraBridgeViewBase, an abstract class which represents a live camera feed, is provided by the OpenCV. It can dispatch events to the listener which implements one of two interfaces: CvCameraViewListener or CvCameraViewListener2. In our project the listener is an activity, the MainActibity. The listener provide call-backs for handing the capture of each frame, and for handing the start and stop stream of cream input. In our project the CvCameraViewListener2 is used, which receives each frame as an instance that includes image in either RGB colour or Grayscale format.

The number of frames per second (FPS) is fluctuate in our project. It is affected by the environment like light condition, additionally it depends on the speed of processing image. In our project, the FPS is guaranteed in rough 15 fps when using the rear camera, while the FPS can reach about 30 fps use the camera in front.

5.4 RGB Traces Collection

After the image is captured, the image is operated as input to get the RGB information. This phase includes face detection, ROI setting, RGB information extraction and add the RGB information to RGB traces (figure 10). These processing mainly refer to two java classes: RGBExtraction and RGBTraces (Data). The RGBEXtraction class regards to the logic of the RGB information extraction, while the RGBTraces class refers to RGB traces management.

For the face detection, the CascadeClassifier class in OpenCV library is used to get the face location. The algorithm of the OpenCV library is a cascade of boosted classifier which can be either a Haar or a LBP classifer. The pre-trained of the front face classifier applies the OpenCV trained Data. For face recognition, the classifier returns the rectangles, which were defined by the coordinates of left-top point with the height and width, around faces. From this output, one of the rectangles will be selected and the center 60% width and height of the rectangle will be picked on as the ROI.

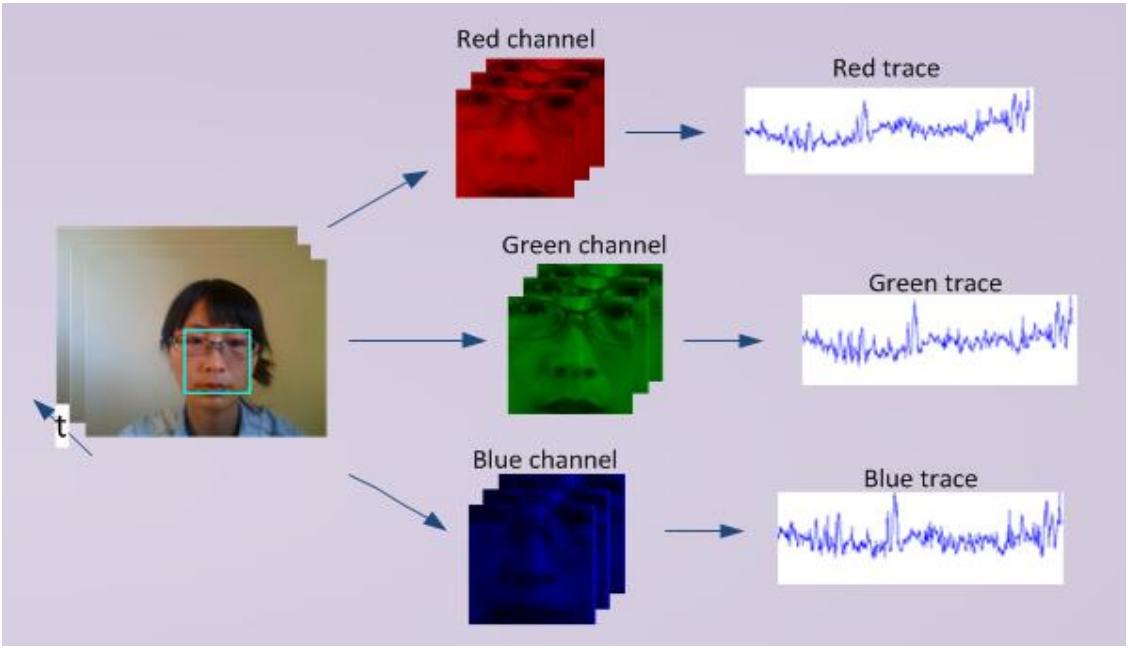

Figure 10 RGB traces collection method

However, there are some errors about face segmentation that will affect the performance of the algorithm. For example no faces were detecting in some frames, sometimes many faces were detecting in one frame, different faces were detecting in adjacent frames and multiply boxes were detecting for the same face. Some correction is necessary to improve the performance of our application. If no faces were detected the coordinates of face from the preview frame will be used. If one face or multiple faces were detected, the top-left point distance and the central point distance between boxes from this frame and previous frame will be compared. If the smallest distance was smallest than the threshold value, the face coordinates which were the closest to the coordinates of forward frame were be selected; if not the face coordinates of the previous frame will be used. The following table display the errors and the correction methods.

Face Segmentation Error	**Correction Method**
No faces were detecting	The face coordinates from the forward frame will be used
Many faces were detecting	The face coordinates, which were the closest to the coordinates from the preview frame, were selected
Different faces were detecting in adjacent frames	If the distance between the face coordinates and the face coordinates from the previous was larger than the threshold, the face coordinates of the forward frame will be used. If not, it will be considered the faces were detecting in adjacent frames were the same face, and the face coordinates in this will be used in normal.
Multiple boxes were detecting for the same face	Compare the top-left point distance and the central point distance between boxes from this frame and previous frame, the box which has smallest distance will be selected

Table 2 Correction methods of face segmentation errors

Then the ROI is divided into the RGB channels, and the RGB information values for each frame are acquired by averaging over all pixels in the ROI. Then the RGB information and the time of frame capturing will be added the RGB traces list and the time information list.

5.5 RGB Traces Processing

According to our designing, the phase one and the phase two must be parallel due to their periods and conditions are all different. Moreover, the second phase processing is time-consuming. If the parallel execution technology is not used, the rate of collection data in the first phase cannot be guaranteed. Multithreading is a technology enabled multiple threads to be within the context of a single process, which is a widespread programming and execution model. These threads share the resources of process, but are capable of executing independently. The phase one is executed in the main thread, while the phase two executed in anther thread which will be notified to process when the condition are met. The input data of the phase two is RGB traces formed from RGB information for 30 s. Thus at the beginning, this thread will be in waiting state until the RGB traces of 30 s formed. Additionally, the 30 s moving window that perform the phase two increases in 1 second, therefore when the thread finished processing data and the new data in 1 second did not collection, the thread will change their state to waiting until the new data finished collection and notified the thread. Here we use a task queue to save which period of data have been finished collection and wait to be processed. When new data formed, the time information will add to the queue and notify the thread to work. Then the thread can get the time information from the queue, and when the queue is empty the thread will be waiting.

The RGB traces processing is shown in figure 11, including normalized the raw RGB traces, decompose the RGB traces into three independent components using ICA and the fast Fourier transform (FFT) was used on the three independent

components. We normalize the raw RGB traces using the following equation:

$$x_i'(\mathrm{t}) = \frac{x_i(\mathrm{t}) - \mu_i}{\sigma_i} \qquad (i = 1,2,3) \tag{5}$$

Where $x_i(t)$ is the raw RGB traces, i = 1, 2, 3 means red, blue and green traces respectively, μ_i is the mean of $x_i(t)$ and σ_i is the standard deviation of $x_i(t)$. The normalization makes sure that $x_i'(t)$ has unit variance and zero-mean.

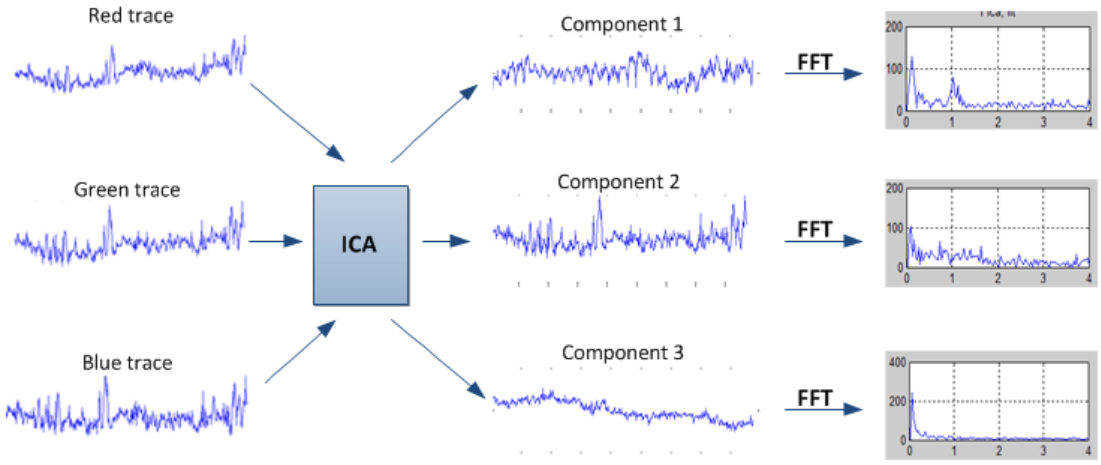

Figure 11 Analysis RGB traces

After normalizing, the data will be decomposed into three independent signals via ICA. We implement the ICA algorithm in C, and then it is compiled to dynamic library. The Application calls the ICA code via Java Native Interface (JNI). JNI defines a method for managed code written in the Java to interact with native code written in C/C++. It supports for loading code from dynamic shared libraries. This will be a little cumbersome, but it is reasonably efficient. We found the ICA implement in C is more efficient and time-saving than implement in Java. When the application calls the ICA written in Java, the period of phase two of processing is far more than 1 second. However, according the DFD the period of the phase two must be less than 1 second for the aim of real time. Thus we implement ICA in C for more efficient.

After ICA, the power spectrum is obtained by using the FFT on the independent signals. An operational scope to [0.75, 3.5] Hz matching to [45, 210] beat per second (bpm) is set in the power spectrum. The highest of power of the spectrum in the operational range will be the pulse frequency. A threshold for maximum change in HR between continuous measurements is used to reject artifacts to improve the performance of system. In our project the threshold is 12 bpm, and if the different between the result of current HR and the value of last HR went beyond the threshold, we will rejected the current HR value and search the next highest power frequency in the operational range. The last HR value will be retained if no frequency peaks were located.

5.6 Chapter Summary

The application development process and the development environment were explored in this chapter. Firstly, we outlined the equipment and development environment. Secondly, we described the development process of our project, including image capturing, RGB information collection and RGB traces processing. After completion of the implementation, the allocation will be tested and evaluate in the following chapter.

Chapter 6

Results and Discussion

In this chapter, we will analysis the correlation by suing Bland-Altman plots. First, we will outline our test method and introduce the analysis method. Then the result of HR measurements at rest and the result of HR measurements during motion will be outlined.

6.1 Testing Method

We invited 20 participants (17 males, 3 females) for our experimental. Our samples of participants involve different genders, ages and skin colours (Asians, Africans and European). For all experiments, a pulse oximeter, which can be used for measuring the pulse rate and the oxygen concentration in the blood by putting the figure in the device (figure 12), was applied to be used as the reference Then we, compared the result from our application with the reference from the pulse oximeter. The experimented were carried out in the laboratory, and the light source covers different amounts of sunlight lighting and is stable. The figure 14 displayed how the experimental executed. The participants were asked to seat at a table in front of a smart phone at a distance of rough half meter from the camera. The measure operation was carrying out in two phases. During the first detecting, participants were requested to sit statically, while for the second phases, participants were requested to move naturally but to avoid large or fast movement.

Figure 12 The pulse oximeter. That was used to measurement pulse rate in our project for validation

Figure 13 The application of our project.

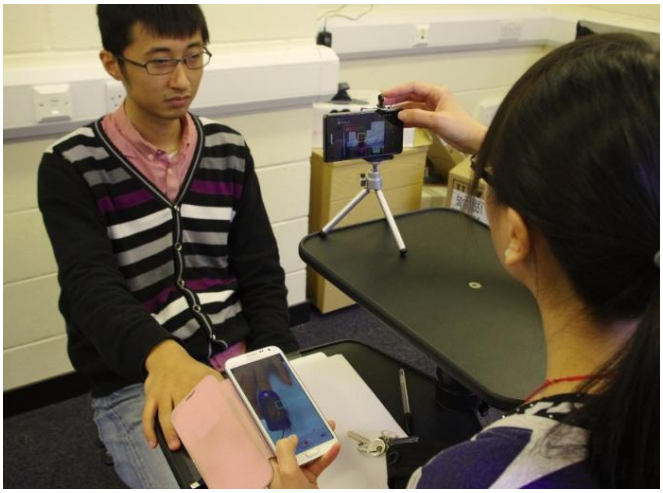

Figure 14 Experimental Setup

6.2 Analysis Method

A Bland-Altman plot, which is a method of data plotting, is used in analysing the agreement between two different measurements [22]. This approach indicates that any

two different approaches that are designed to measure the same parameter should have good correlation when a set of samples are chosen such that the property to be determined varies considerably. A high correlation for any two methods designed to measure the same property could thus in itself just be a sign that one has chosen a widespread sample. High correlation does not automatically mean that there is good agreement between the two methods. The Bland–Altman method figures the mean and standard deviation (SD) of the difference between two measurement methods, and 95% limits of agreement as the mean difference (1.96 SD). It is expected that the 95% limits include 95% of differences between the two approaches of measurement.

A scatter plot uses Cartesian coordinates to display values for two variables for a set of data. It is often used to show whether a relationship exists between two sets of data [23]. The data is displayed as a collection of points, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis.

In our project, the correlation and the differences between measure result from our application and the pulse oximeter device were plotted against the averages of both systems. We calculated the mean and SD of the differences, and 95% limits of agreement (± 1.96 SD). Additionally, the root mean squared error (RMSE), correlation coefficients and spearman rho value were calculated for both systems.

6.3 Testing Result

When the participants were sitting still, 763 pairs of measurements from 20 participants were tested by Bland-Altman analysis (figure 15). The left plot of figure 15 is a scatter plot. The measurement pairs were display as collection of points. If the values of pair are the same, the points will locate on the red straight line. The points from our test lie along a straight line ($y = 0.928x + 4.54$) that indicated a high correlation between the two measurements. The correlation coefficient represents the strength of a relation between two variables [23]. The value range of r is from +1 to -1,

where 1 is total positive correlation, 0 means no correlation, and -1 represents total negative correlation. The plot shows that the correlation coefficient r of our project highly reaches to 0.9313, and the RMSE only 2.8 bpm. The Bland-Altman plot is shown on the right of figure 15. The mean bias $\bar{d}$ is -0.78 bpm with 95% limits of agreement -6.3 to 4.8.

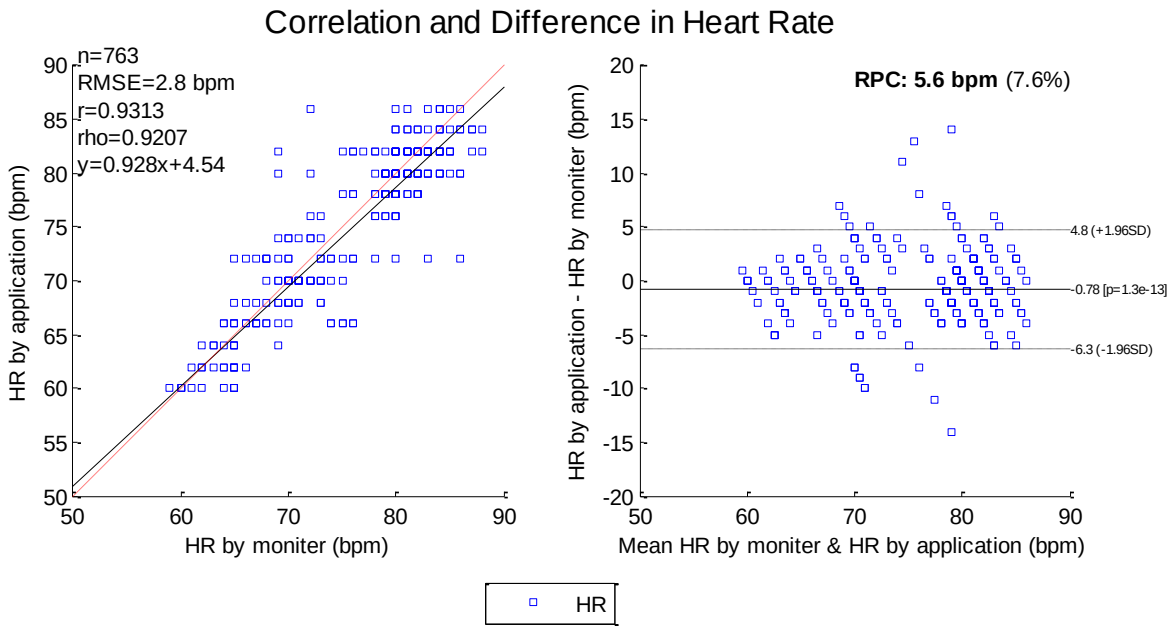

Figure 15 Correlation and Differences between HR by monitor and HR by application, for participants sitting still

The robustness of our system for HR measurements in the presence of motion artifacts was also evaluated. During these experiments, the participants can free to move their head including nodding the head, looking up and down, and tilting the head sideways, but to avoid large or fast movement. Moreover, the participants can make various facial expressions, talk or laugh during testing.

From the Bland-Altman analysis of 628 pairs of measurements from 20 participants (figure 16), we can see the accuracy still good. The RMSE is a little bit higher than the still case, reaching 3.4 bpm. The correlation coefficient r slightly falls, but it is still high, reaching 0.911. The mean bias $\bar{d}$ increases to -1.4 bpm with 95%

limits of agreement -8.5 to 5.7. High correlation coefficient values were obtained for bath participants sitting still and move naturally.

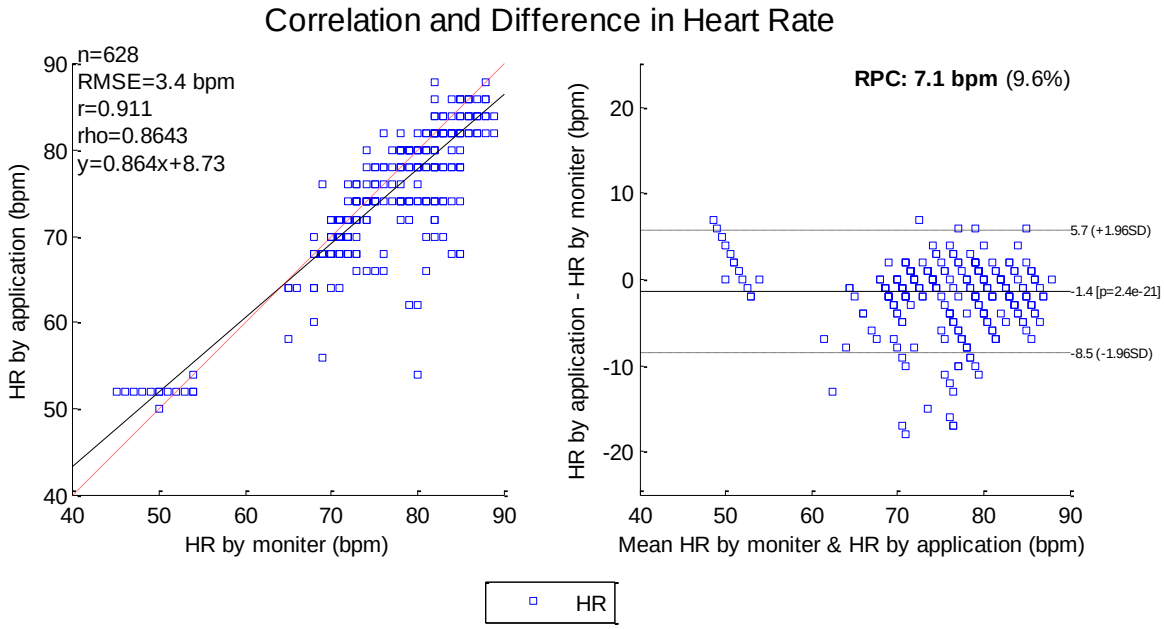

Figure 16 Correlation and Differences between HR by monitor and HR by application, for participants move naturally

From the Bland-Altman analysis, we can see the accuracy of measurements of HR is good. The mean ± standard deviation (SD) of the differences between heart rate measurements from our application and sensor is -0.78 ± 2.83 when the participants were sitting still and -1.4 ± 3.62 in the presence of motion artifacts. Linear regression indicated a high correlation between the two measurements across the two evaluated conditions (r = 0.9313, 0.911, respectively). Table 3 shows the accuracy rate from the reference [7] and our system. The mean of difference from our application is larger than that from the reference [7], but the SD of difference from our application is better than that from the reference [7].

	Reference [7]	Our application
$\bar{d}$ when Still	-0.05 ± 2.29	-0.78 ± 2.83
$\bar{d}$ when motion	0.64 ± 4.59	-1.4 ± 3.62

Table 3 The accuracy rate of system

6.4 Chapter Summary

In this chapter, the application was testing and the measurement result was compared with the result of a monitor using Bland-Altman. The robustness of the proposed methodology for HR measurements in the presence of motion artifacts was also evaluated. The accuracy of our application in measuring HR is good.

Chapter 7

Conclusions and Future Work

7.1 Project Conclusions

In this project we have developed a dedicated application on the smart phone platform, which can be used for non-contact, motion-tolerant and automated HR detecting. This application bases on PPG, face tracking and Blind source separation technology. The application process images captured from camera to collect the optic information of face skin. Then we analysed the optic information to obtain the HR. In our project we use the ICA to reduce the efficient of motion artefact. The ICA assumed that the observed signals were linear mixtures. However, the optic information reflected from face varies nonlinearly due to the motion of head. We give a short time (30s) window for ICA, thus a liner model should provide a reasonable local approximation. The result of the testing shown that this approach can achieve high accuracy, The mean ± standard deviation (SD) of the differences between heart rate measurements from our application and sensor along with the 95% limits of agreement -0.78 ± 2.83 when the participants were sitting still and -1.4 ± 3.62 in the presence of motion artifacts. Linear regression indicated a high correlation between the two measurements across the two evaluated conditions (r = 0.9313, 0.911, respectively).

However, there are several limitations of this project. Firstly, the motion artifacts in our test were slow and small movements, for instance, nodding the head, looking up and down, tilting the head sideways, talking and laughing. When the motion occurred suddenly or quickly, the result will be affected seriously due to large change of the baseline of the RGB signal cause by the motion. Additionally, the automatic face tracker can bring movement artifact. Even the participants sited still, the localized ROI also fluctuates that is another source of artifacts. In our application, we

have a choice to set the ROI localized fixed. Then the artifacts caused by face tracker can be removed, but the participants should keep still during the testing to remain the face in the ROI localized. Secondly, our experiments were executed at the library with the illumination relatively strictly and moderation. Thirdly, the performance of the application is influenced by the faces detecting. In the face detecting stage, the application identified a number of false positives in the image clips due to background complicated. Finally, the pulse frequency information can be match in one of ICA component, but there is no ordering of the ICA components. Our project captures the highest power of the spectrum in all three ICA components to obtain the HR. However, when the power of other ICA components is larger than the pulse power, the HR detecting will be failed.

7.2 Future Work

Our project has shown that mobile phone cameras have the potential to capture the PPG signal and measurement HR. The PPG signal provides the worthy information regarding to the cardiovascular system, for example the blood pressure, blood oxygen saturation, HR, HRV and cardiac output. Thus the mobile phone can be developed as monitor for several physiological variables. It brings great convenience to users. It is easy for users to carry a physiological monitor anywhere.

The performance of the application is influenced by the faces detecting. The face tracker of our application was trained with the OpenCV frontal face classifier. When the background is too complicated, the detecting of face may be false. Skin detection can be used to improve the detection rates.

The application need to test in different lighting condition to reveal the relationship between the illuminations and the performance of our technique. Then the following research can focus on improving the robustness of our technique in different illuminations.

The application can be improved by mobile phone architecture such as multi core and GPU. The application contains some complex and time-consuming process, for example some image processing and some complicated algorithm ICA. Image processing and ICA are all large amount of data operation, which can be accelerated by using GPU technology.

Bibliography

1. S. Cook, M. Togni, M. C. Schaub, P. Wenaweser, and O. M. Hess, “High heart rate: a cardiovascular risk factor?” Eur. Heart J. 27(20), 2387–2393 (2006).
2. Arsenos P, Gatzoulis K, Dilaveris P, et al. “Arrhythmic sudden cardiac death: Substrate, mechanisms and current risk stratification strategies for the post-myocardial infarction patient”. Hellenic J Cardiol, 2013, 54: 301-315.
3. Böhm M, Swedberg K, Komajda M, et al. “Heart rate as a risk factor in chronic heart failure (SHIFT): the association between heart rate and outcomes in a randomised placebo-controlled trial”. The Lancet, 2010, 376(9744): 886-894.
4. Vasu V, Heneghan C, Arumugam T, et al. “Signal processing methods for non-contact cardiac detection using Doppler radar”. Signal Processing Systems (SIPS), 2010 IEEE Workshop on. IEEE, 2010: 368-373.
5. Gu C, Li R, Li C, et al. “Doppler radar respiration measurement for gated lung cancer radiotherapy”. Biomedical Wireless Technologies, Networks, and Sensing Systems (BioWireleSS), 2011 IEEE Topical Conference on. IEEE, 2011: 91-94.
6. Yuen P, Hong K, Chen T, et al. Emotional & physical stress detection and classification using thermal imaging technique. 2009.
7. Poh M Z, McDuff D J, Picard R W. “Non-contact, automated cardiac pulse measurements using video imaging and blind source separation”. Optics Express, 2010, 18(10): 10762-10774.
8. Bousefsaf F, Maaoui C, Pruski A. Remote assessment of the heart rate variability to detect mental stress. Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013: 348-351.
9. Scully C, Lee J, Meyer J, et al. Physiological parameter monitoring from optical recordings with a mobile phone. Biomedical Engineering, IEEE Transactions on, 2012, 59(2): 303-306.
10. Jonathan E, Leahy M. Investigating a smartphone imaging unit for photoplethysmography. Physiological measurement, 2010, 31(11): N79.
11. Allen J. Photoplethysmography and its application in clinical physiological measurement. Physiological measurement, 2007, 28(3): R1.

12. Takano C, Ohta Y. Heart rate measurement based on a time-lapse image. Medical engineering & physics, 2007, 29(8): 853-857.
13. Poh M Z, Swenson N C, Picard R W. Motion-tolerant magnetic earring sensor and wireless earpiece for wearable photoplethysmography. Information Technology in Biomedicine, IEEE Transactions on, 2010, 14(3): 786-794.
14. Cardoso J F. Blind signal separation: statistical principles. Proceedings of the IEEE, 1998, 86(10): 2009-2025.
15. Comon P. Independent component analysis, a new concept?. Signal processing, 1994, 36(3): 287-314.
16. Cardoso J F. High-order contrasts for independent component analysis. Neural computation, 1999, 11(1): 157-192.
17. What is independent component analysis: A demo. Online at http://www.cis.hut.fi/projects/ica/icademo.
18. A. Hyv¨arinen and Shubhendu Trivedi. Blind source separation in magnetic resonance images. Online at http://onionesquereality.wordpress.com/2010/01/30/blind-source-separation-in-magnetic-resonance-images/.
19. Brandt D. Investigation of GPGPU for Use in Processing of EEG in Real-time. Rochester Institute of Technology, 2010.
20. Lienhart R, Maydt J. An extended set of haar-like features for rapid object detection. Image Processing. 2002. Proceedings. 2002 International Conference on. IEEE, 2002, 1: I-900-I-903 vol. 1.
21. Howse J. Android Application Programming with OpenCV. Packt Publishing Ltd, 2013.
22. Martin Bland J, Altman D G. Statistical methods for assessing agreement between two methods of clinical measurement. The lancet, 1986, 327(8476): 307-310.
23. Lee Rodgers J, Nicewander W A. Thirteen ways to look at the correlation coefficient. The American Statistician, 1988, 42(1): 59-66.

24. GO-Gulf.com (2012). Smartphone Users Around the World – Statistics and Facts [online]. Available from: http://www.go-gulf.com/blog/smartphone/ (Accessed 20 March 2014)
25. ABI Research(2013). 45 Million Windows phone and 20 million BlackBerry 10 Smartphones in Active Use at Year-end; Enough to Keep Developers Interested [online]. Available from: https://www.abiresearch.com/press/45-million-windows-phone-and-20-million-blackberry (Accessed 20 March 2014)
26. Viola P, Jones M. Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. IEEE, 2001, 1: I-511-I-518 vol. 1.
27. Freund Y, Schapire R E. Experiments with a new boosting algorithm. ICML. 1996, 96: 148-156.
28. Burrus C S S, Parks T W. DFT/FFT and Convolution Algorithms: theory and Implementation John Wiley & Sons, Inc., 1991.

Appendix A

Java Code

1. Class of ROIExtraction

```
/**
* The ROIExtraction used to extraction RGB information form the inputFrame.
* This phase includes face detection, ROI setting, RGB information extraction
* It contains a face detection class for detecting face
* @author Weihua Lin
*/
public class ROIExtraction {
    private static final String    TAG                = "ROI_Extraction";
    private static final Scalar    FACE_RECT_COLOR    = new Scalar(0, 255,
0, 255);
    private Mat                    mRgba;
    private Mat                    mGray;
    private Mat                    mFace;
    private Mat                    mCell;
    private Size                   mShowSize; //show ROI size
    private File                   mCascadeFile;//face detector trained file
    private CascadeClassifier      mJavaDetector;//face detector

    private float                  mRelativeFaceSize   = 0.2f;
    private int                    mAbsoluteFaceSize   = 0;
    private Rect       mPreviousRect;//save the last frame face box position
    private boolean                mIsPreviousable;
    private boolean                mIsStart;
    private boolean                mIsBoxFix;
    private Activity               mActivity;

    public ROIExtraction(Activity activity) {
        this.mActivity = activity;
        mPreviousRect = new Rect();
        mIsPreviousable = false;
        mIsBoxFix = false;
        mIsStart = false;
    }
  /**
  * This function initialize the face detector
  * and load training file to train the detector
*/
```

```
public void initDetector(){
        try{   // load cascade file from application resources
            InputStream is =
mActivity.getResources().openRawResource(R.raw.lbpcascade_frontalface);
            File cascadeDir = mActivity.getDir("cascade",
Context.MODE_PRIVATE);
            mCascadeFile = new File(cascadeDir,
"lbpcascade_frontalface.xml");
            FileOutputStream os = new FileOutputStream(mCascadeFile);

            byte[] buffer = new byte[4096];
            int bytesRead;
            while ((bytesRead = is.read(buffer)) != -1) {
                os.write(buffer, 0, bytesRead);
            }
            is.close();
            os.close();

            mJavaDetector = new
CascadeClassifier(mCascadeFile.getAbsolutePath());
            if (mJavaDetector.empty()) {
                Log.e(TAG, "Failed to load cascade classifier");
                mJavaDetector = null;
            } else
                Log.i(TAG, "Loaded cascade classifier from " +
mCascadeFile.getAbsolutePath());

            cascadeDir.delete();
        }catch (IOException e) {
            e.printStackTrace();
            Log.e(TAG, "Failed to load cascade. Exception thrown: " + e);
        }
    }

    public void prepareStar(final int width,final int height){
        mGray = new Mat();
        mRgba = new Mat();
        mFace = new Mat();
        mCell = new Mat();
        int size = Math.min(width, height);
        size = size/4;
        mShowSize = new Size(size,size);
    }
```

```
    public void stopFrame(){
        mGray.release();
        mRgba.release();
        mFace.release();
        mCell.release();
    }

  /**
  * This function extract RGB information from image
  * @param inputFrame the image captured from camera contains RGB and gray
image
  * @param currentTime is the time of capturing the image
*/
    public synchronized Mat roiExtract(CvCameraViewFrame inputFrame, long
currentTime) {
        mRgba = inputFrame.rgba();
        mGray = inputFrame.gray();
        int hr = 0; // hear rate

        if (mAbsoluteFaceSize == 0) {
            int height = mGray.rows();
            if (Math.round(height * mRelativeFaceSize) > 0) {
                mAbsoluteFaceSize = Math.round(height * mRelativeFaceSize);
            }
        }

       MatOfRect faces = new MatOfRect(); // to save the face detecting result
       if (mJavaDetector != null){
           mJavaDetector.detectMultiScale(mGray, faces, 1.1, 2, 2,
               new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new Size());
       }
        Rect[] facesArray = faces.toArray();
        //choose and correct the ROI POS, save the coordinate in the
mPreviousRect
        setTheFaceRect(facesArray);

        //ROI extract
       if(mIsPreviousable){
            int widthAdjust =(int) mPreviousRect.width / 5;
            int heightAdjst = (int) mPreviousRect.height/5;
           Rect roi = new Rect(mPreviousRect.x + widthAdjust,mPreviousRect.y
           + heightAdjst, mPreviousRect.width - widthAdjust *2,
           mPreviousRect.height - heightAdjst * 2);
```

```
            //the center 60% width and full height of the rectangle will be
picked on as the ROI
            mCell= mRgba.submat(roi.y ,roi.y + roi.height, roi.x,roi.x +
roi.width );
            //record the RGB traces
            if(mIsStart){
                final RGBTraces rgbTraces =
            (RGBTraces)mActivity.getApplication();
                rgbTraces.AddROI(mCell, currentTime);
                hr = rgbTraces.getlastHR();//get HR
            }
            //show the ROI face in the screen
            mFace =
mRgba.submat(0,(int)mShowSize.width,0,(int)mShowSize.height);
            Imgproc.resize(mCell, mFace, mShowSize);

            Vector<Mat> rgb = new Vector<Mat>();
            Core.split(mFace, rgb);
            //to show R, G, B channel
            Vector<Mat> temp = new Vector<Mat>();
            for(int i = 0; i < 3 ;++i){
                temp.add(Mat.zeros(mShowSize, CvType.CV_8UC1));
            }
            temp.add(3, rgb.get(3));

            for(int i = 0; i < 3; ++i){
                mFace = mRgba.submat(0,(int)mShowSize.width,
(int)mShowSize.height*(i+1),(int)mShowSize.height * (i + 2));
                temp.set(i, rgb.get(i));
                Core.merge(temp,mFace);
                temp.set(i, Mat.zeros(mShowSize, CvType.CV_8UC1));
            }
        //plot the roi
            Core.rectangle(mRgba,roi.tl(), roi.br(), new Scalar( 255, 255,
0 ), 3);
        }
//        //plot the box
//        for (int i = 0; i < facesArray.length; i++){
//         Core.rectangle(mRgba, facesArray[i].tl(), facesArray[i].br(),
FACE_RECT_COLOR, 3);
//        }
        //show HR
        if(hr != 0){
            Point pos = new Point( mRgba.size().width -
```

```
200 ,mRgba.size().height -20);
            Core.putText(mRgba, String.format("HR: %d",hr),pos,
                    Core.FONT_HERSHEY_PLAIN, 2, FACE_RECT_COLOR, 4);
        }
        return mRgba;
    }

    //choose and correct the ROI POS
    private void setTheFaceRect(Rect[] facesArray) {
        if(facesArray.length == 0)  return;
        if(mIsPreviousable && mIsBoxFix) return;

        if(mIsPreviousable == false){
            for(int i = 0; i < facesArray.length; ++i){
                if(facesArray[0].area() >0) {
                    mPreviousRect = facesArray[0].clone();
                    mIsPreviousable = true;
                    return;
                }
            }
        } else {
            //save the min distance and the index
            double minValue = Double.MAX_VALUE;
            int minIndex = 0;

            //the coordinate of the central point in previous face
            Point preCentral = new Point(mPreviousRect.x + 0.5 *
mPreviousRect.width, mPreviousRect.height  * 0.5 + mPreviousRect.y);

            for(int i = 0; i < facesArray.length; i++){
                Point central =new Point (facesArray[i].width * 0.5 +
facesArray[i].x, facesArray[i].height * 0.5 + facesArray[i].y);

                //the distance of central point
                double distance =Math.pow((preCentral.x - central.x), 2)
                    + Math.pow((preCentral.y - central.y), 2);

                //the distance of left top point
                double tpDistance = Math.pow(mPreviousRect.x
-facesArray[i].x,  2 ) + Math.pow(mPreviousRect.y -facesArray[i].y,  2 ) ;

                distance = distance + tpDistance/2;
                if(minValue > distance) {
                    minValue = distance;
```

```
                    minIndex = i;
                }
            }//for

            if(minValue < 20000) {
                mPreviousRect = facesArray[minIndex].clone();
            }
        }//else
    }
....
}
```

2. RGBTraces Class

```
/**
* The RGBTraces class used to save the information regarding RGBTraces.
* @author Weihua Lin
*/
public class RGBTraces extends Application {
    public static  final int protimelong = 30; //time of windows
    private static final String    TAG   = "RGBTraces";
    private static final String    rFilename = "r.txt";
    private static final String    gFilename = "g.txt";
    private static final String    bFilename = "b.txt";
    private static final String    hrFilename = "hr.txt";
    private static final String    ltFilename = "lt.txt";
    private static final int fEndPos = 120;

    private List<Double>      lRTrace;
    private List<Double>      lGTrace;
    private List<Double>     lBTrace;
    private List<Integer> icaHR;
    private List<Integer> gHR;
    private int    lastfreIndex;
    //Save every second start frame position
    private List<Integer>     lFramesTrace;
    private long            mStartime;
    private List<Double>      lHeartRate;
    //Save need to processing time(eg. 29 means time from 0 to 29)
    private Queue<Integer>      qWaitQueue;
    private DataProcessor     mHRThread; //do normalizing, ICA, FFT

    static {  System.loadLibrary("JadeR"); }

    public RGBTraces(){
```

```
        lRTrace = new ArrayList<Double>();
        lGTrace = new ArrayList<Double>();
        lBTrace = new ArrayList<Double>();
        icaHR = new ArrayList<Integer>();
        gHR = new ArrayList<Integer>();
        lFramesTrace = new ArrayList<Integer>();
        mStartime = 0;
        lHeartRate = new ArrayList<Double>();
        qWaitQueue = new LinkedList<Integer>();
        mHRThread = new DataProcessor();
        mHRThread.start();
        lastfreIndex = 0;
    }

    public void AddROI(final Mat roi, long currentTime){
        Scalar tracts = Core.mean(roi);
        synchronized(lBTrace) {
            lRTrace.add(tracts.val[0]);
        }
        synchronized(lRTrace)#{
            lGTrace.add(tracts.val[1]);
        }
        synchronized(lGTrace) {
            lBTrace.add(tracts.val[2]);
        }

        if(mStartime == 0){
            mStartime = currentTime;
            lFramesTrace.add(0);
        } else {
            int time = (int) (currentTime - mStartime)/1000;

            //if new second begin, record the position
            //Save every second start frame position
            //lFramesTrace.size means how many seconds has been recorded
            //if time >= lFramesTrace.size means new second begin
            if(time >= lFramesTrace.size()){
                lFramesTrace.add(lBTrace.size() - 1);
                //when the time larger then the window time, every second
increase
                // need to do notify mHRThread
                if(time >= protimelong/* && time < topTime*/){
                    Log.i(TAG, "add time to wait queue");
                    qWaitQueue.add(time);
```

```
                mHRThread.isRunning = true;
                if(mHRThread.getState() == Thread.State.WAITING) {
                    synchronized(mHRThread){
                        Log.i(TAG, "notify the mHRThread");
                        mHRThread.notify();
                    }
                }
            }
        }
    }
}

public void clear()
{
    lastfreIndex = 0;
    lRTrace.clear();
    lGTrace.clear();
    lBTrace.clear();
    icaHR.clear();
    gHR.clear();
    lFramesTrace.clear();
    qWaitQueue.clear();
    mStartime = 0;
}

public void saveRGBData() {
    //chekc the state of SD card
    String sdStatus = Environment.getExternalStorageState();
    if(!sdStatus.equals(Environment.MEDIA_MOUNTED))
    {
        Log.e(TAG, "sd card can not be used");
        return;
    }
    if(lRTrace.isEmpty()) return;

    JSONArray numbers = new JSONArray(lRTrace);
    String jsonString = numbers.toString();
    DataServer.saveData(rFilename, jsonString);

    numbers = new JSONArray(lGTrace);
    jsonString = numbers.toString();
    DataServer.saveData(gFilename, jsonString);

    numbers = new JSONArray(lBTrace);
```

```
        jsonString = numbers.toString();
        DataServer.saveData(bFilename, jsonString);

        numbers = new JSONArray(icaHR);
        jsonString = numbers.toString();
        DataServer.saveData(hrFilename, jsonString);

        numbers = new JSONArray(lFramesTrace);
        jsonString = numbers.toString();
        DataServer.saveData(ltFilename, jsonString);
    }

    public String getHRvaluse(){
        JSONArray numbers = new JSONArray(icaHR);
        return numbers.toString();
    }

    public List<Integer> getHRlist() {
        return icaHR;
    }

    public int getlastHR(){
        int hr = 0;
        if(!icaHR.isEmpty()){
            hr = icaHR.get(icaHR.size() - 1);
        }
        return hr;
    }

    public void SavaHR(String filename){
        //chekc the state of SD card
        String sdStatus = Environment.getExternalStorageState();
        if(!sdStatus.equals(Environment.MEDIA_MOUNTED)){
            Log.e(TAG, "sd card can not be used");
            return;
        }
        if(icaHR.size() <= 0 ) return;

        JSONArray numbers = new JSONArray(icaHR);
        String jsonString = numbers.toString();
        DataServer.saveData(filename + "_hr.txt", jsonString);

        numbers = new JSONArray(gHR);
        jsonString = numbers.toString();
```

```
        DataServer.saveData(filename + "_ghr.txt", jsonString);
    }
}
```

3. DataProcessor Class

```
/**
* The DataProcessor used to process RGB traces.
* It includes normalized the raw RGB traces, decompose the RGB into three
independent source signals using ICA and the fast Fourier transform (FFT)
was used on the three independent source signals.
* @author Weihua Lin
*/
public class DataProcessor extends Thread {
    public boolean          isRunning = true;
    @Override
    public void run(){
        Log.i(TAG, "run the mHRThread");
        while(isRunning) {
            Log.i(TAG, "while the mHRThread");
            try{
                if(qWaitQueue.isEmpty()){
                    isRunning = false;
                    Log.i(TAG, "mHRThread is waiting");
                    synchronized(mHRThread) {
                       try {
                           mHRThread.wait();
                       }catch(InterruptedException e) {
                           e.printStackTrace();
                       }
                    }//synchromized
                } else{
                    int endtime = qWaitQueue.poll();
                    int starttime = endtime - protimelong;
                    Log.i(TAG, "starttime is " +
Integer.toString(starttime));
                    Log.i(TAG, "endtime is " + Integer.toString(endtime));

                    if(starttime < 0 || endtime >= lFramesTrace.size())
                        continue;

                    int startFrame = lFramesTrace.get(starttime);
                    int endFrame = lFramesTrace.get(endtime) - 1;

                    Log.i(TAG, "startFrame is " +
```

```
Integer.toString(startFrame));
                Log.i(TAG, "endFrame is " + Integer.toString(endFrame));

                // normalize the raw RGB traces
                double[][] xsrc = DoNormalize(startFrame, endFrame);

            //-----------------------------------test g trace------
                List<Double> gcom = new ArrayList<Double>();
                doDft(xsrc[2], gcom);
            //-----------------------------------------------------
                //ICA
                //DataServer.doICAbyJava(xsrc, xsrc, 3);
                DataServer.doICAbyNative(xsrc, xsrc, 3);

                //fft
                List<Double> com1 = new ArrayList<Double>();
                List<Double> com2 = new ArrayList<Double>();
                List<Double> com3 = new ArrayList<Double>();

                doDft(xsrc[0], com1);
                doDft(xsrc[1], com2);
                doDft(xsrc[2], com3);
                //get HR
                float fHR = getHR(com1,com2,com3,gcom);
            }
            }catch(Exception e)   {
                e.printStackTrace();
                Log.e(TAG, "Failed to write. Exception thrown: " + e);
            }
        }
    }

    /**
    * normalized the raw RGB traces
    */
    private double[][] DoNormalize(int startFrame, int endFrame){
        //do the R
        List<Double> src;
        List<Double> rdst = new ArrayList<Double>();
        synchronized(lRTrace) {
            src = lRTrace.subList(startFrame, endFrame);
            //Log.i(TAG, "r" + src.toString());
            DataServer.doNormalize(src, rdst);
        }
```

```
        //G
        List<Double>gdst = new ArrayList<Double>();
        synchronized(lGTrace){
            src = lGTrace.subList(startFrame, endFrame);
            DataServer.doNormalize(src, gdst);
        }
        //b
        List<Double> bdst = new ArrayList<Double>();
        synchronized(lBTrace){
            src = lBTrace.subList(startFrame, endFrame);
            DataServer.doNormalize(src, bdst);
        }
        double[][] xsrc = new double[3][];
        xsrc[0] =  DataServer.convertDoubles(rdst);
        xsrc[1] = DataServer.convertDoubles(gdst);
        xsrc[2] = DataServer.convertDoubles(bdst);
        return xsrc;
    }
    /**
    *The Fourier transform was used on the three independent source signals
    */
    private void doDft(double[] ica, List<Double> fft) {
        Mat mat = DataServer.convertToMat(ica, CvType.CV_64FC1);
        Vector<Mat> templist = new Vector<Mat>();
        templist.add(mat);
        templist.add(Mat.zeros(mat.size(), CvType.CV_64FC1));

        Core.merge(templist, mat);
        Core.dft(mat, mat);
        Core.split(mat, templist);

    Core.magnitude(templist.get(0),templist.get(1),templist.get(0));

        double[] temp = new double[templist.get(0).channels()];
        fft.clear();
        for(int i = 0; i < fEndPos; ++i) {
            temp = templist.get(0).get(0,i);//(i, 0);
            fft.add(temp[0]);
        }
        mat.release();
        int size = templist.size();
        for(int i = 0; i < size; ++i)
            templist.get(i).release();
    }
```

```
    private int getHR(List<Double> com1, List<Double> com2, List<Double>
com3, List<Double> gcom){
        int index = getMax(com1,com2,com3);
        double gmax = Collections.max(gcom.subList(23, 105));
        int gindex = gcom.subList(23, 105).indexOf(gmax);

        int hr = 0;
        if(index >0) {
            hr= (index + 23) * 2;
            icaHR.add(hr);
            gHR.add((gindex + 23) * 2);
        }
        lastfreIndex = index;
        return hr;
    }
    /**
    *Get the highest power of the spectrum in all three ICA components
    */
    private int getMax(List<Double> com1, List<Double> com2, List<Double>
com3)  {
        //frequency [0.75,4] corresponding to [45,240]
        List<Double> sub1 = com1.subList(23, 105);
        List<Double> sub2 = com2.subList(23, 105);
        List<Double> sub3 = com3.subList(23, 105);

        double max1, max2,max3, max;
        int m = 20, index = 0;
        List<Double> temp;
        while(m > 0) {
            max1 = Collections.max(sub1);
            max2 = Collections.max(sub2);
            max3 = Collections.max(sub3);
            index = 0; temp = null;

            if(max1 > max2) {
                max = max1;
                temp = sub1;
            } else {
                max = max2;
                temp = sub2;
            }
            if(max < max3) {
                max = max3;
```

```
                temp = sub3;
            }
            index = temp.indexOf(max);
            if(lastfreIndex <= 0) break;
            if(index <= 0) {
                index = lastfreIndex;
                break;
            }
            if(Math.abs(lastfreIndex -index) <=6) break;
            temp.set(index, 0.0);
            m--;
        }
        if(m == 0){
            Log.i(TAG, "index is " + Integer.toString(index));
            index = lastfreIndex;
        }
        Log.i(TAG, "index is " + Integer.toString(index));
        return index;
    }
}///class DataProcessor
```