Yousuf Mohamed-Ahmed

Non-contact heart rate estimation from video

Computer Science Tripos - Part II

Gonville & Caius College

Declaration

I, Yousuf Mohamed-Ahmed of Gonville & Caius College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed *Yousuf Mohamed-Ahmed*

Date May 2, 2020

Proforma

Candidate Number:	2339B
Project Title:	**Non-contact heart rate estimation from video**
Examination:	**Computer Science Tripos - Part II, July 2020**
Word Count:	**11383**[1]
Line Count:	**2399**
Project Originator:	Dr R. Harle
Supervisor:	Dr R. Harle

Original Aims of the Project

I aim to implement a system capable of estimating the heart rate of a user purely from a video of their face and to show that this is achievable on videos recorded by a regular smartphone camera. I hope to further argue, should this be possible, that smartphones represent a viable alternative to many other sensors for measuring heart rate.

Work Completed

I have completed all core requirements and two of the three extension criteria. In doing this, I have shown that non-contact heart rate estimation is capable of accuracies similar, if not better, than that of a wearable device. As a result, showcasing that reliable measurement of heart rate is not necessarily the remit exclusively of dedicated sensors.

Special Difficulties

None.

[1]This word count was computed by the `TeXcount` script

Contents

Chapter 1
Introduction

Optical heart-rate monitors, increasingly standard in modern wearable devices, measure the absorption of light by the skin as a proxy for the blood flow beneath. The most common light-sensing device of all is the camera and so, naturally, the question arises as to whether a camera can be used to measure heart rate. Research has shown that such a system can be implemented using a standard camera pointed at the face of the user [25][30][31]. I refer to this as *non-contact heart rate estimation* and this project is concerned with the implementation of a system capable of this.

In general, sensing technology has progressed, in recent times, from high to low fidelity but with costs decreasing. Often, low cost sensors are developed that attempt to measure the same phenomenon as a high cost alternative, but with a compromise on accuracy. As a result, newly developed low cost sensors arrive in a wider array of devices and hence access to their capabilities increases. Heart rate sensing used to be the remit of expensive electrocardiogram devices in hospitals, but is now commonplace in relatively low cost wearable devices. Although the latter cannot be used to diagnose medical conditions, in general, they are adequate for the majority of users. This phenomenon is a key pattern in the field and is of great relevance to the project.

This trend allows increasing numbers of users to gain access to their own health data, albeit at the cost of decreased fidelity. Expanding the scope of sensing, and computation in general, is a key principle of ubiquitous computing and, since cameras are far more common than heart rate monitors, the potential impact of this project in this context is large. In fact, I will later argue that smartphones, with their increased computational power, could be of greater fidelity than wearables for heart rate sensing, in which case, this represents a much more significant leap forward than might initally be anticipated.

There is a large body of related research. Individual, seminal, papers are summarised in Section 2.2.1, however, in general, much of the research proceeds similarly. In order to extract the heart rate signal, the face must be tracked between frames, a subset of the face pixels are monitored, with their mean colour in each frame being recorded. Given this signal, the heart rate is discovered by careful analysis of the frequencies present. It is the aim of this project to implement each of these three stages, with original work being designed and implemented throughout.

Chapter 2
Preparation

This project, by nature, combines a variety of disciplines and technologies, including, but not limited to, computer vision, sensing and signal processing. Before being able to proceed with the project, understanding the required topics in each of these respective fields was critical. Several relevant topics are outlined and described in relation to this project.

2.1 Heart rate sensing

This project can be viewed as the development of a virtual sensor[1] that derives its data from an existing sensor — the camera. It is, nonetheless, useful to understand the functioning of existing heart rate monitors. Electrocardiography and photoplethysmography are the two of the most common techniques for measuring heart rate and each proceed by measuring different phenomena.

2.1.1 Electrocardiography

An electrocardiogram (ECG) is a recording of the electrical activity of the heart (see Figure 2.1). Electrodes that make contact with the skin measure how the voltage varies with time. At the beat of a heart there is a very specific electrical pattern that occurs and can be recognised. Specfically, when the cardiac muscles contract, the muscle cells undergo *depolarization* which causes a change in the electric charge of the cell and is measurable by the electrodes. The average number of beats in a given window, is known as the heart rate. Crucially for this project, the ECG is considered the "gold standard" means of measuring heart rate. The electrodes can perceive very minor changes in

[1] A sensor that does not derive its results from a direct physical realisation of that sensor

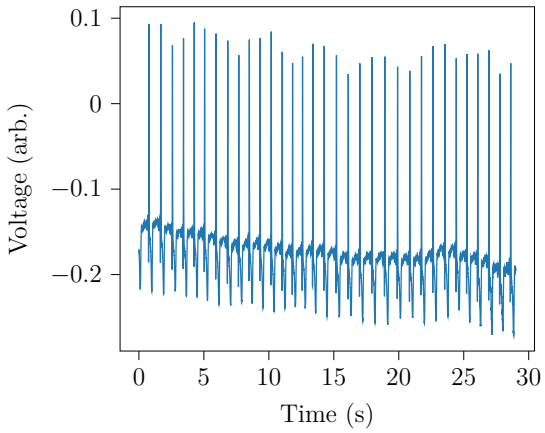


Figure 2.1: An example ECG signal

voltage and, thereby, very rarely produce false beats or miss beats of the heart. Hence, throughout this project it is considered suitable enough to behave as a ground truth for any experiments conducted. ECG sensors, however, tend to be relatively expensive and invasive, and as a result, have fueled the uptake of lower fidelity alternatives.

2.1.2 Photoplethysmography

Photoplethysmograpgy (PPG), a common alternative to an ECG, uses an optical sensor to detect changes in the volume of blood passing beneath the skin. When a heart beat occurs, blood flows outwards from the heart towards the extremities causing an increasing volume of blood in the vessels beneath. Detecting this change is, hence, an alternative to measuring electrical signals.

This can be physically realised by a *pulse oximeter*, which consists of a light that illuminates the skin and a sensor that measures the amount of light absorbed. Typically placed on the end of a finger, the LED emits light on one side and a sensor on the other measures the amount of light passing through the finger. Greater volumes of blood cause more absorption and so reduce the amount of light reaching the sensor. In this form, PPG sensing is of high fidelity and is common in medical scenarios. However, pulse oximeters require a stationary finger and so are not practical for sporting activities.

Wearable PPG sensors

Smartwatches are one of the most common classes of wearable device and are readily equipped with wrist-based PPG sensors. The above description of pulse oximetry requires many adaptations to be suitable for use on a wearable. These modifications make mobile sensing possible but at a great cost in fidelity.

Since the PPG sensor is placed on the wrist it doesn't measure the amount of light absorbed but the amount *reflected*. This is because the wrist, as opposed to the finger, contains much more cartilage so it is not feasible to measure the amount of light passing all the way through the tissue. The cartilage in the wrist, disrupts the light emitted and is the a key reason behind the decreased fidelity of wrist-based PPG.

The LEDs present on the underside of the watch will not make perfect contact with the skin. In sporting scenarios, where there is lots of movement, this will affect the light passing between the LED and the skin and acts as an additional source of noise in the signal.

Furthermore, sensors on smart devices are often subject to severe energy constraints in an attempt to increase the battery life of the device. As a result, wearable sensors use lower sampling rates than that of a medical-grade sensor.

Together these factors, and numerous others, result in a much noisier and, as a result, less accurate, PPG signal (see Figure 2.2).

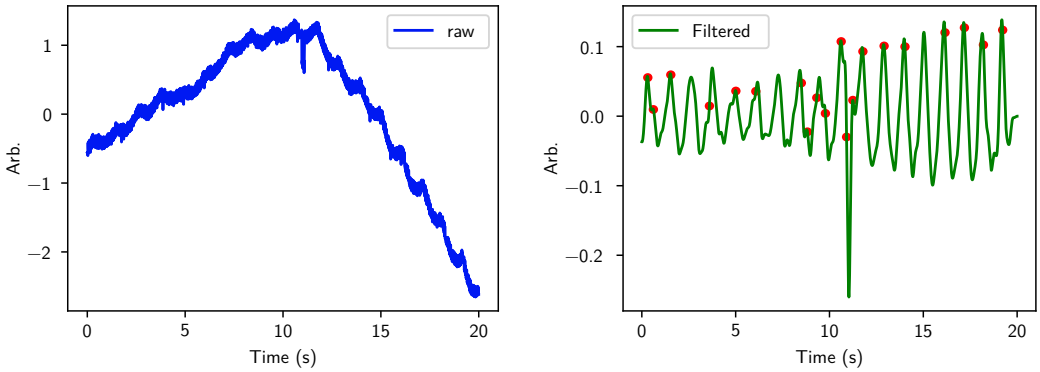


Figure 2.2: *An example PPG signal as taken from a wearable watch, with inferred heart beats on a filtered signal*

2.2 Remote photoplethysmography (rPPG)

This project, instead, is concerned with the development of a virtual PPG sensor which can infer the heart rate of a user from a camera only. That is, without an LED or a dedicated light sensor in contact with the skin. This is generally denoted in the literature as *remote photoplethysmography* (rPPG) and is an active area of research [31][24][30].

The most obvious approach to designing such a system, is to attempt to map the data returned by a camera onto one of the two methods of heart rate sensing described. In other words, to attempt to mimic an already existing, well-developed sensor. Since a camera cannot, as far as I am aware, measure electrical signals at a distance, mimicing an ECG is not a feasible approach. PPG sensors, on the other hand, are based on measuring light intensity, which is precisely the same data captured by a camera.

A camera works by recording the amount of light that reaches each light sensor in an array. Assuming that we have a camera of high enough resolution, one can measure the intensity of light at various points in the scene. The impressive results of early research into rPPG systems is that this can be achieved by a regular webcam or smartphone camera.

Clearly an rPPG system is different to both the pulse oximeter and a wrist-based PPG sensor, although it attempts to measure the same phenomenon. These differences form the basis of its trade-offs as a sensor. For example, since a camera is likely to be at a greater distance than both of the other PPG sensors described, it is even more subject to noise caused by lighting conditions. However, cameras contain much larger numbers of light sensors and so can consider more points. This trade-off between the number and reliability of measurements is difficult to evaluate without experiment and is the subject of investigation.

2.2.1 Literature review

Early research into remote heart estimation navigated the difficulties of attempting to mimic a traditional PPG sensor. Therefore, understanding the existing literature on the topic is critical to a successful implementation. Furthermore, it can reveal areas for additional experimentation and provides clarity as to a concrete implementation. For example, the description in Section 2.2 leaves several questions which must be answered before an implementation could possibly be designed.

- Which parts of the body are best to extract a PPG signal from?
- How can the recorded signal reveal the heart rate?
- What efforts must be undertaken to reduce the effect of noise?

Verkyuysse et al. In 2008, Verkyuysse et al. [31] were the first to show that remote heart rate sensing is possible. This was shown using a standard camera and no additional source of illumination. Crucially, they reported that the face, as opposed to other regions of the body, provides the strongest PPG signal and verified this experimentally. Intuitively, this is because it is believed that the tissue on the surface of the face is particularly thin and so it is easier to extract a signal from [30]. However the selection of the region of the face to consider, referred to as the *region of interest* (ROI) in the literature, was done manually for each frame. This is not desirable and, as a result, the automation of this is the subject of discussion in this project. Finally, Verkyuysse et al. [31] described the use of Fourier techniques for isolating the heart beat in the colour signal returned by the camera.

Poh et al. Poh et al. [25] introduced the idea of applying more rigourous signal processing techniques to the recorded signal, in an attempt to further reduce noise and isolate the heart rate. Specifically, they introduced the use of *blind source separation* techniques to attempt to isolate the pulse from the observed colour changes. The problem of blind source separation is outlined in Section 2.4.1 and was an important discovery in reducing the effect of noise.

2.2.2 Extensions to current literature

Remote heart rate sensing should be viewed as a means of widening access to health data. As a result, I believe that performance and robustness are of equal importance. That is, the computations involved must be achievable in a reasonable amount of time on a standard computing device. Furthermore, heart rate estimations must be robust to changing conditions or, at the very least, their failure scenarios should be well documented. Under this analysis, several extensions to the current literature were formulated and researched throughout the project:

- attempt to achieve reasonable performance on standard computing devices
- investigate the effect of motion and distance on accuracy

2.3 Relevant computer vision techniques

Remote heart rate sensing operates directly on a camera stream and understanding the composition of frames is a critical; in the same way that an ECG sensor relies on interpreting electrical signals. The field of automatically inferring knowledge from image data is generally known as computer vision and there are several relevant techniques to this project.

2.3.1 Face detection

In Section 2.2.1 it was stated that it is easiest to estimate the heart rate by considering pixels in the face of the user. This means that, given a frame from the camera, the region(s) containing a face must be ascertained. For this, many algorithms have been developed, not all of which perform equally well. For the purposes of this project, there are three main criteria by which the strength of a face detection algorithm can be evaluated.

- Tightness of bounding box: typically face detection algorithms return a bounding box within which the face is believed to be. The size of this box is of great importance since any additional background pixels contain no pulse information and will add unnecessary noise
- False positive rate: any false detections will impact accuracy by potentially considering incorrect regions of the frame
- Performance: the wider objective of this project is to increase the availability of heart rate sensing technology and so costly algorithms which cannot run effectively on standard computing devices, are of no real use

Viola-Jones algorithm The Viola-Jones face detection algorithm [33], the first of its kind to achieve real-time performance with exceptional accuracy, is implemented as a cascade of individually weak classifiers. Each classifier returns a binary outcome as to whether or not a face might be contained in the region considered. A face is 'detected' when a region passes all of the classifiers. The principle underpinning the speed of the algorithm is that most regions will fail in one of the first few classifiers and so it is rare to apply the entire sequence.

Neural network approaches Neural networks for face detection have been trained since approximately the year 1998 [27], although, at the time, the performance of the Viola-Jones algorithm was considered superior, advances in computing hardware have nullified this [9]. In fact, I conducted early-stage experiments which suggested that neural networks, specifically the model provided by the OpenCV library [2] returned smaller face detection regions with similar computational costs. For this reason, the Viola-Jones algorithm, although popular, was not used in this project.

2.3.2 Optical flow

The correspondence problem, well known in computer vision, is the task of, given a pair of images of the same scene at different times or from different positions, finding the pixels corresponding to

the same locations in the scene. For example, given two images of a car that are taken a second apart, one might like to discover where the same wheel has moved to between frames. Optical flow is a variant of the correspondence problem, where we specifically consider images taken at different times but from the same position and attempt to infer motion between the observer and the scene. This can loosely be thought of as a notion of 'tracking' points in a scene between frames.

Fleet and Weiss [11] provide a formal description of the optical flow problem which has been outlined and summarised below. Suppose we have a function $I(x, y, t)$ which describes the intensity at each point (x, y) in an image taken at time t. The task is to find for each point of interest, (x_i, y_i), a pair $(\Delta x_i, \Delta y_i)$ such that for an subsequently taken at time $t + \Delta t$.

$$I(x_i, y_i, t) = I(x_i + \Delta x_i, y_i + \Delta y_i, t + \Delta t) \tag{2.1}$$

In other words, we wish to find where each point has moved to between frames. This can be achieved by taking the Taylor series expansion of equation 2.1.

$$I(x_i + \Delta x_i, y_i + \Delta y_i, t + \Delta t) = I(x_i, y_i, t) + \frac{\partial I}{\partial x_i}\Delta x_i + \frac{\partial I}{\partial y_i}\Delta y_i + \frac{\partial I}{\partial t}\Delta t + ... \tag{2.2}$$

This sets up the key equation of optical flow which a valid solution must satisfy.

$$\frac{\partial I}{\partial x_i}\Delta x_i + \frac{\partial I}{\partial y_i}\Delta y_i + \frac{\partial I}{\partial t}\Delta t = 0 \tag{2.3}$$

If we denote, the respective velocities $V_{x_i} = \frac{\Delta x_i}{\Delta t}$ and $V_{y_i} = \frac{\Delta y_i}{\Delta t}$ then the constraint in equation 2.3 can be rewritten in terms of the velocities V_{x_i} and V_{y_i}.

$$\frac{\partial I}{\partial x_i}V_{x_i} + \frac{\partial I}{\partial y_i}V_{y_i} + \frac{\partial I}{\partial t} = 0$$

Attempting to find V_{x_i} and V_{y_i} from this equation directly is not feasible since it is a single equation in two unknowns. Therefore, large bodies of work have been dedicated to investigating further sets of assumptions which can make this tractable. Typically, these additional assumptions are used to generate further equations in the variables V_{x_i} and V_{y_i} to overcome the problem described. A widely used technique that achieves this is known as the Lucas-Kanade method [19].

Lucas-Kanade Instead of considering an individual pixel, one might instead assume that the movement in a small enough region of the frame is constant. That is, we assume that a region of n pixels all move in the same direction between frames. In this scenario, we can construct n equations in two variables and so for any $n > 1$, we can attempt to solve for V_{x_i} and V_{y_i}. Typically, $n > 2$ is used and so the sets of equations are over-determined and thus it may be the case that no values of V_{x_i} and V_{y_i} exist that perfectly solve the set of equations. In this case, the Lucas-Kanade method computes the solution minimising the least squares error.

All optical flow techniques assumes that we can track points by only considering their brightness. Implicitly this further assumes that the illumination incident onto a surface is

constant between frames. Clearly this is a very large assumption and would fail if there is reflection within the surface as is the case with, for example, specular highlights. However, this assumption, as reported by Fleet and Weiss [11] works surprisingly well in practice and so is deemed to be acceptable. Naturally, however, these assumptions will not hold perfectly in reality and so cannot be used to reliably track points over exceedingly long time frames.

Shi-Tomasi corner detection Given the Lucas-Kanade method of optical flow, it would be plausible to track all points in the image by applying the algorithm to every pixel. This is known as dense optical flow and is very computationally intensive. As an alternative, one might use *sparse* optical flow where only some subset of all the pixels in the image are tracked. The selection of this subset is non-obvious. It is clear, however, that not all points in an image are equally easy to track. For example, a pixel surrounded by a large region of uniform colour cannot be easily followed but points found at boundaries are likely to be easier to track.

Shi and Tomasi [13] proposed a method for the selection of points with the purpose of ease of tracking which they termed *GoodFeaturesToTrack*. The approach taken is to find points which are surrounded by regions of differing illumination, such points are known as 'corners'. Corners tend to be easier to track and so are a good method for selecting points.

2.4 Relevant signal processing techniques

2.4.1 Blind source separation

In a busy restaurant with multiple conversations occurring simultaneously, humans can, peculiarly, focus on a single conversation comfortably. Given signals from both ears which contain a mixture of many different conversations, one can identify an individual signal corresponding to the dialogue of interest. This is an example of the selective attention that can be displayed by humans and is often referred to as the *cocktail party effect* [7].

The task of identifying sources of interest from multiple mixed signals is known as the *blind source separation problem* and is an important problem in the field of digital signal processing. In the example of a restaurant, each source is an audio signal representing the individual conversation occurring. The brain receives signals from each ear containing a mixture of conversations and is tasked with producing one coherent signal of the conversation of interest. Although this appears to be trivial for humans it is much more difficult to achieve computationally.

Analagously, the stream of colours being received from the camera contains a mixture of signals, one of which corresponds to the heart rate. Hence, identifying this signal can be viewed as a similar task and is explored in Section 3.4.1.

2.5 Requirements analysis

Goals:

- Primary output: develop a Python implementation capable of reasonable accuracy under good lighting and for stationary users
- Supplementary output: an Android implementation as a proof of concept that remote heart rate sensing is viable for mobile devices

Extensions:

- Evaluate performance in comparison to a smartwatch
- Achieve real-time performance, that is, the frequency of frame processing is less than or equal to the frame rate of the camera, thereby maximising the effective sampling frequency of the system.
- Investigate performance under more realistic scenarios:
  - with the camera further away from the user
  - with the user moving within the frame
- Ability of tracking multiple users simultaneously

These tasks are summarised in the table according to both their difficulty and importance to the overall project.

Objective	Difficulty	Importance	Status
Python implementation	Medium	High	Core
Android demonstration of rPPG	Medium	Medium	Core
Tracking multiple users	High	Low	Extension
Real-time performance	High	Medium	Extension
Evaluating in comparison to smartwatch	Medium	Medium	Extension

Table 2.1: A summary of the requirements of the project

2.6 Languages and tooling

2.6.1 Languages

Python The majority of the project will be developed in the Python [3] programming language. As a language with large support for both computer vision and signal processing applications, it was a natural choice for the project.

Kotlin A part of my core requirements is the development of an example application capable of rPPG that can run on the Android operating system. To such end, a language that can target the JVM[2] is required. With support for more functional syntax and with complete interoperability with Java, Kotlin [1] was a convenient choice.

2.6.2 Libraries

OpenCV Originally developed by the Intel Corporation, OpenCV [2] is an open-source library for computer vision applications and has built in support for face detection, as is required by the project.

2.7 Professional practice

Since large portions of the project were investigative, an agile model of development was utilised. This was to allow for shorter development cycles and protoyping phases, which were short enough to ensure that I was staying on track with the initial timeline outlined in the proposal. Furthermore, this aligned well with regular meetings with my supervisor.

2.8 Starting Point

As an interdisciplinary project, a wide array of background knowledge was required, some of which has not featured in Tripos. For example, I had to understand the basics of how modern heart rate sensors work, for which I used internet resources as well explanations by my supervisor.

Understanding existing literature regarding remote photoplethysmography was equally important and so during the early stages of the project, I reviewed a large number of journal papers to gain an understanding of the state of the art techniques being used.

A variety of Part II courses proved to be relevant this year, including but not limited to: Mobile and Sensor Systems, Computer Vision, Information Theory and Digital Signal Processing. Many of these courses occurred after a large body of the work was completed, however, they were, nonetheless, useful. I had no previous experience programming in Kotlin and only very little in Python, however, both have a plethora of online resources which I used during the development process.

[2]Java Virtual Machine

Chapter 3
Implementation

Abstractly, the program consists of three distinct tasks, each of which rely on the result from the previous. Together, forming a kind of pipeline:

- **Face detection**: identify a face in each supplied camera frame
- **Region selection**: given a bounding box around a face, select some set of pixels to consider which are amenable to heart rate detection
- **Heart rate isolation**: given a time series of the mean colour of each frame infer the heart rate

Face detection is largely a solved problem and thereby the project mostly concentrates on the latter two, which are the topic of ongoing research.

3.1 System design

Each of these tasks occur at different rates and, thereby, have different performance constraints which must be upheld. Face detection and region selection operate on every frame received from the camera and so must run in real-time, or risk dropping streamed frames.

Heart rate isolation, however, is only executed after an adequate number of data points are received and is recomputed after a fixed time. Since none of the prior stages rely on the estimated heart rate, its execution time requirements are less stringent. This is exploited to perform relatively expensive analyses without slowing earlier stages. Crucially, this relies on the assumption that it can be run concurrently. Separating these tasks allows for this concurrency to be implemented safely.

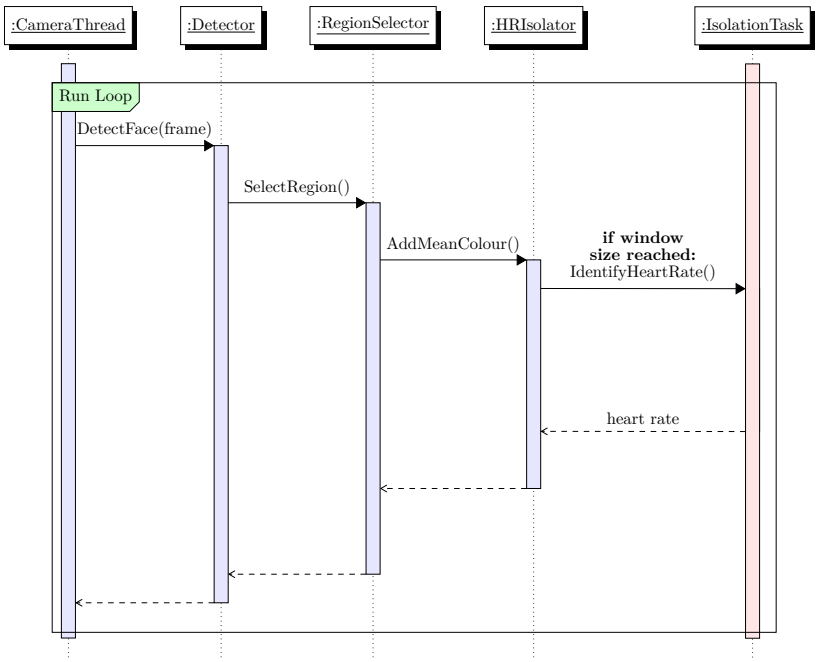


Figure 3.1: A UML sequence diagram showcasing the use of threading

The camera streams frames to the **FaceDetector**. The **RegionSelector** then takes the mean pixel colour of the region considered and adds this value to the dataset. Once enough values have been collected, as defined by the window size, the **HRIsolator** spawns a new thread, the **IsolationTask**, which attempts to infer the heart rate from the window of values.

Although the region selection could also be executed in a separate thread, the associated setup costs are likely to have an adverse impact on real-time performance. However, one might wish to use more expensive region selection algorithms which cannot run in real-time. In these scenarios, the program could copy the frame and execute the selection in a separate thread from the main face detection loop. This is fundamentally a tradeoff between memory usage and execution time. Furthermore, since there are only a finite number of threads which can be spawned, quickly the limit might be reached without providing much additional computation time to the **RegionSelector**. As a result, the **FaceDetector** and **RegionSelector** operate sequentially in the same thread.

3.2 Face detection

A face detector is expected to take a single frame and return a bounding rectangle within which a face is present. This could be extended to work on a stream of frames, naïvely, by simply repeated applying this face detector to each frame independently.

3.2.1 Face tracking

Smartphone cameras can readily stream at high frame rates, so it is unlikely that a face moves very much between a pair of consecutive frames. At thirty frames per second, frames are recorded only 0.033s apart. The position of the face in the previous frame, gives a strong indication of its subsequent position. Thus, there is opportunity for optimisation beyond simply applying a face detector to each frame individually. A speedup in this part of the pipeline provides opportunity for more expensive computation in subsequent stages which might improve accuracy. However, it is critical that any optimisations are resistant to motion. Using information from previous frames forms the distinction between face ***detection*** and ***tracking***.

Point tracking

The key principle behind face tracking is to use information from previous frames to reduce the cost of subsequent face detections. To that end, I use the Lucas-Kanade algorithm [19], as described in the Preparation chapter, to track points on the face rather than repeatedly calling the face detector. Naturally, over time, these points will diverge from their true positions. This is because images contain large numbers of pixels with similar illumination that cannot be distinguished perfectly. When this occurs, the position of the face should be redetected.

Knowing when the points have diverged is non-trivial, since the true location of the face is unknown, without calling the face detector directly. It is crucial for any implementation to act safely enough that the face is not lost track of, whilst simultaneously minimising the number of times the face detection algorithm is used. There are two obvious approaches to combat this. The algorithm could redetect the face:

- periodically
- when the tracked face changes in size significantly.

The former wastes computation time for stationary videos where the redetections might be unnecessary. Simultaneously, the time between redetections must be short enough to deal with videos with significant movement. This static approach, therefore, was not considered. For this reason, the latter was implemented. The face is tracked and if the size of the box surrounding it changes significantly, the true position of the face is recomputed.

```
points = []
redetect = True
last_detection = None
def face_tracker(frame):
    if redetect or points is empty:
        face = face_detector(frame)
        last_detection = face
        points = select_new_points(face)
    else:
        points = track_points(points)
        face = bounding_box(points)
    redetect = change_in_size(last_detection, face) > threshold
    last_detection = face
    return face
```

Figure 3.2: A simplified pseudocode representation of an optical-flow based face tracker

Impact of the rate of redetections It is important to reason about precisely when face tracking provides a performance boost. To understand this let us consider a sequence of frames which the face tracker has been applied to. The cost of tracking is compared with that of repeatedly detecting the face in each frame independently. Under the following notation:

- W: number of consecutive frames considered
- R: the total number of redetections by the face tracker
- n: size of each frame in pixels
- p: number of points tracked
- $f(n)$: cost of a face detection on a single frame of size n
- $s(p, f)$: cost of selecting p points to track in a face of size f
- $g(p, n)$: cost of tracking p points in a frame of size n

The cost of the repeated face detector is $Wf(n)$. Since, for each frame in the sequence, it detects the face independently and incurs a cost of $f(n)$.

The point tracking approach instead has a cost of $Wg(p, n) + R(f(n) + s(p, f))$. For each redetection it calls the face detector and selects a new set of points. It also tracks the set of points for every frame, hence the term $Wg(p, n)$. In the worst case, $R = W$, so there is no saving in terms of computational complexity. Instead let us investigate for what values of R there is a cost

saving.

$$\begin{aligned} Wf(n) &> Wg(p,n) + R[f(n) + s(p,f)] \\ R &< \frac{W[f(n) - g(p,n)]}{f(n) + s(p,f)} \\ \frac{R}{W} &< \frac{f(n) - g(p,n)}{f(n) + s(p,f)} \end{aligned}$$

Notice that R/W represents the percentage of frames for which a redetection occurs. There is only a performance saving when this percentage falls below the value on the right hand side. Furthermore, this value itself is based on the relative costs of face detection and of selecting and tracking points. Implicitly, the algorithm assumes that the cost of tracking and selecting points is less than that of face detection, otherwise, this endeavour would be useless. This assumption is validated and the limit is evaluated experimentally in Section 4.2. From this it is shown that even for videos with lots of movement, the value R/W falls below this limit and the inequality is satisfied. As an approach, it is conservative enough that the true position of the face is not lost, whilst still providing a 3x to 8x performance boost (see Section 4.2) depending on the amount of movement in the video. Crucially, in Section 4.2, I show that there is no statistically significant effect on the accuracy of heart rate predictions.

3.3 Region selection

The bounding box returned by face detection will contain pixels from the background of the image since faces are not, in general, perfectly rectangular. These background pixels will not contain any information as to the underlying heart rate of the user. As a result, considering the entire bounding box will add unecessary noise to the resulting signal which consists of the mean colour from each region considered in the sequence of frames. One approach might be to only consider skin pixels, however, robust, pose-invariant skin detection is non-trivial.

3.3.1 Skin detection

Suppose that we wish only to consider skin pixels. An obvious approach might be to apply an edge detection algorithm to isolate the boundary between the face and the background. However, edge detection algorithms, like the Canny edge detector [5], tend to produce large numbers of irrelevant edges and so were not implemented.

Colour-based filtering

Considering that skin tones tend to fall within a certain range of colours, one might encode this information in a primitive skin detector. For example, it is known that green is not a valid skin tone but brown might be. If a large enough number of skin tones are investigated then a range within which a skin pixel might lie could be defined.

A dataset of ∼250000 skin and non-skin pixels was collected by Bhatt et al [26] and was

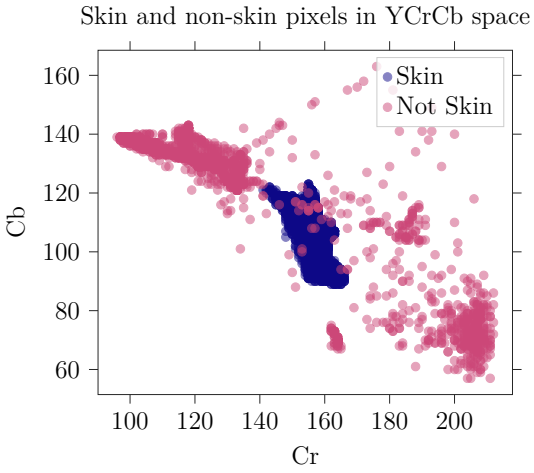


Figure 3.3: *A randomly sampled subset of the skin dataset represented in the Cb-Cr axes of YCbCr space*

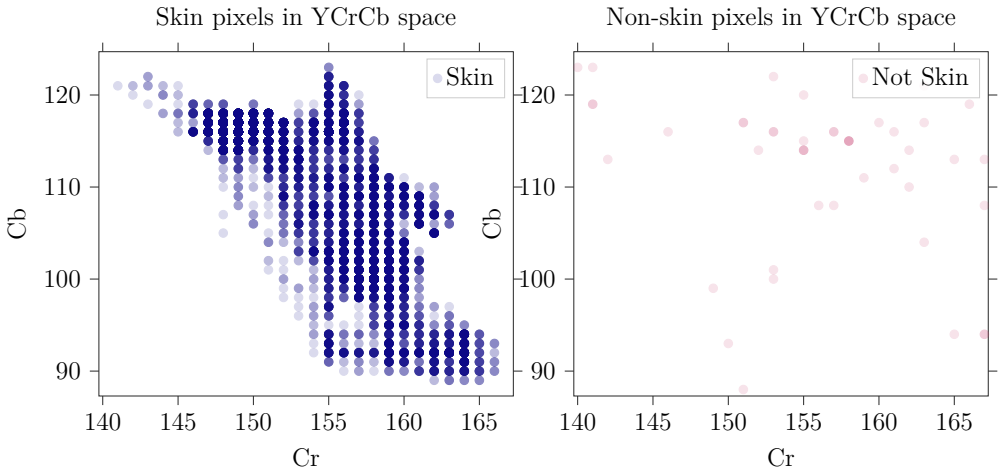


Figure 3.4: *The range of skin tones present in the dataset with the skin and non-skin pixels plotted*

sampled across a variety of skin tones, genders and ages. One could define the range of skin tones present in this dataset as a rudimentary skin detector. Clearly this will fail in many scenarios but serves as a useful baseline for comparison with more advanced techniques.

Issues This approach takes no consideration of the particular face in question. Since it is not contexualised, it can fail in several circumstances. For example, there are some hair colours which could be a skin tone on another individual but are clearly not on the particular example in question. Instead, an algorithm should attempt to identify skin on the particular face in question.

(a) Original image[1] (b) Mask indicating skin pixels

Figure 3.5: *An example of the hair failure case*

K-Means

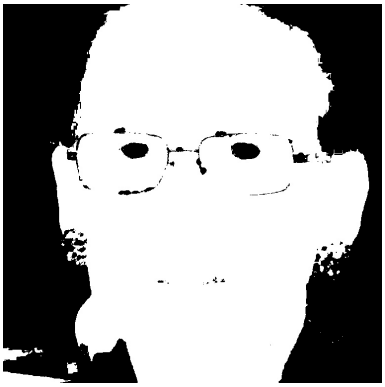
If we consider how a human might identify skin, it could involve initially identifying the skin tone of the person and then assuming all parts of the face of a similar colour are in fact skin. This reduces the problem to identifying the skin tone in a face and then measuring the colour difference between each pixel and the skin tone. If the colours are within some specified threshold, then we can consider them to be skin pixels.

We might suppose that the image consists of clusters of pixels, some of which belong to the skin and others which don't. The center of the cluster of skin pixels represents the skin tone of the individual. Implicitly, this approach assumes that the Euclidean distance between points in our colour space is representative of the perceived colour difference. This property is known as perceptual uniformity and is not a property of all colour spaces.

The colour space YCbCr is an approximation of perceptual uniformity and hence the image is converted from RGB before the application of clustering. Under the assumption that our image consists of two clusters of pixels, skin and non-skin, we could simply apply the k-means algorithm[2] [14] identify these clusters of pixels. Under the further assumption that the majority of pixels are skin pixels, the largest cluster is returned as the set of skin pixels.

[1] An image of mathematician Katherine Johnson taken from the public domain.
[2] A popular algorithm for discovering clusters in data

(a) Original image



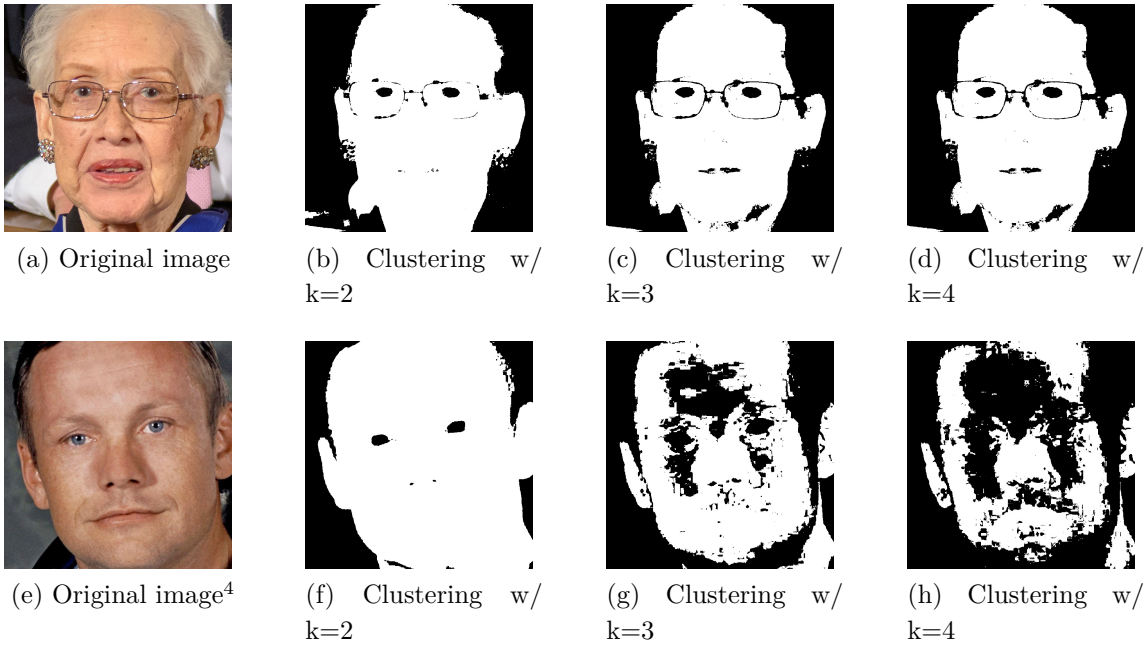
(b) Mask indicating regions considered to be skin.

Figure 3.6: *An example of the application of the k-means algorithm to skin detection*

Issues The k-means algorithm, although it improves on the results of the rudimentary approach has a plethora of pitfalls.

- **Performance**: recall from Section 3.1 that, since the region selection operates on every frame, it must run in real-time. However, in benchmarking the k-means reference implementation in the scikit-learn [4] library takes an order of magnitude longer than the minimum requirement for this constraint.[3].
- **Location**: since it encodes no notion of location with respect to other pixels, a lone pixel in the corner that has a similar colour to the skin tone is considered the same as a pixel surrounded by skin pixels.
- **Number of clusters**: there's no rigorous means for deciding the number of clusters to expect in the data and the selection of this value is critical. Too many clusters can cause unexpected behaviour, but too few can result in undesired pixels being considered.
- **Illumination resistance**: the algorithm is not resistant to differences in illumination. For example, since there is no notion of location encoded, specular reflection on the forehead may not be considered as skin.

[3]Suppose the camera streams at 30 frames per second, then each frame must be processed within 33ms, the reference implementation operates in the order of 100ms per frame.



(a) Original image (b) Clustering w/ k=2 (c) Clustering w/ k=3 (d) Clustering w/ k=4

(e) Original image[4] (f) Clustering w/ k=2 (g) Clustering w/ k=3 (h) Clustering w/ k=4

Figure 3.7: *An example of the effect of differing the number of clusters*

3.3.2 Improving the previous approaches

Recall that the purpose of the region selection algorithm is to return the mean colour of the pixels considered. The mean value from each frame is taken as a time series which is used to infer the heart rate. So rather than attempting to classify each pixel as either skin or not, I use a combination of the previous techniques to take a weighted mean which emphasises pixels which are believed to be skin pixels, in such a way as to minimise the issues encountered with the previous techniques.

Instead of applying k-means to each frame, which is not feasible with the constraint of real-time performance, we apply it once at the start of the video. One of the resulting clusters will correspond to the skin tone of the face in question. Having identified the skin tone, this colour can be used as part of a probability model which we define to try and encode the likelihood of a particular pixel being part of the skin. In this way, we amortize the cost of skin detection over the entire video rather than per frame. This reduces the problem to:

- Use k-means to identify the skin tone
- Use the skin tone to identify skin in the face for a large number of consecutive frames

Identifying the skin tone

There are two expected properties of the cluster corresponding to the skin. It is likely to contain the largest number of pixels, since we'd expect most of the face to be composed of skin. It is

[3] An image of astronaut Neil Armstrong from the public domain.

also likely to fall within the range of expected human skin tones. The difficulty of this problem, however, is that neither of these properties are guaranteed to hold. For example, the face may be dominated by hair which could become the most prominent colour in the image. Likewise, changes in illumination or the presence of specular reflection might cause the skin tone of the face to be outside the expected range.

By considering both of these properties together, as a heuristic, it becomes more probable that the skin tone is identified correctly. I assign a score to each cluster, C, based on the number of elements in the cluster, $|C|$, and its distance from the expected range of skin tones, d_C.

$$\text{score}(C) = \alpha \cdot |C| - \beta \cdot d_C$$

The constants α and β were selected experimentally. The skin tone is identified as the center of the cluster with the maximum score.

Identifying skin pixels using the skin tone

Identifying the skin tone, as described, requires the use of the k-means algorithm, however it is not particularly suitable for real-time applications (see Section 3.3.1). Thus, the skin tone cannot be identified in every frame. Given the skin tone from an earlier frame, the value must be used to robustly detect skin pixels in subsequent frames. This is challenging since there will almost certainly be changing illumination conditions as well as changing positions of the skin pixels.

Implicitly, the clustering approach relies on the assumption that any change in illumination between frames affects all pixels equally. It assumes that in subsequent frames, the true skin pixels remain closer (in terms of Euclidean distance) to the skin tone than the non-skin pixels. This is a fairly strong assumption and is affected by phenomena such as shadows and specular reflection, which do not affect all pixels equally. However, it is useful enough to make the problem more tractable than it was previously, without undermining the fidelity of the algorithm.

A Bayesian approach Recall that the first approach described classifies skin based on knowledge about the range of possible human skin tones. In this sense, it acts as a prior distribution; when classifying an individual pixel, it considers nothing of the face being presented. The k-means implementation, on the other hand, considers the face presented but it has to knowledge of the prior. It instead assumes that the largest cluster must be the set of skin pixels.

A natural way to combine these two approaches would be to consider the problem of skin classification from a Bayesian perspective. Given some pixel x_i we want to discover the likelihood of it being a skin pixel, having been conditionalised on the skin tone of the person as well the colour of the pixel being classified. We have access to the prior distribution from the first approach, which indicates the likelihood of a particular colour being skin. Let us begin by denoting the following:

- C_{skin} the class of skin pixels
- x_i the colour of the pixel being considered
- s the skin tone of the face

Given this notation the probability we wish to discover is the likelihood of a pixel being skin given its colour and the skin tone of the user.

$$\Pr(C_{\text{skin}}|x_i, s)$$

From Bayes' theorem, this can be re-written to expose the prior distribution.

$$\frac{\Pr(C_{\text{skin}}, x_i, s)}{\Pr(x_i, s)} = \frac{\Pr(s|C_{\text{skin}}, x_i)\Pr(C_{\text{skin}}|x_i)\Pr(x_i)}{\Pr(x_i, s)}$$

Prior distribution Usually, one might attempt to learn the distributions $\Pr(C_{\text{skin}}|x_i)$ and $\Pr(s|C_{\text{skin}}, x_i)$ from a relevant dataset. The prior $\Pr(C_{\text{skin}}|x_i)$ was computed as the empirical distribution from the dataset discussed in Section 3.3.1. This dataset consists of randomly sampled pixels that are classified as skin or not, it does not include entire classified faces. Furthermore, to the best of my knowledge at the time of implementation, there are no publicly available, adequately sized datasets with this information. Hence, attempting to learn the distribution $\Pr(s|C_{\text{skin}}, x_i)$ (denoted as the *class conditional distribution* here) was not deemed to be feasible. Once the distribution $\Pr(C_{\text{skin}}|x_i, s)$ has been computed for a single frame, it is used as the prior for subsequent frames. This is for the purpose of providing greater resistance to sudden changes in the image. and is a benefit of the Bayesian approach. It allows a principled means of incorporationg prior knowledge about the problem space.

Class conditional distribution A reasonable approach is to define the distribution $\Pr(s|C_{\text{skin}}, x_i)$ based on the assumption that a skin pixel is likely close in colour to the overall skin tone. We want a function which returns a large probability if the Euclidean distance between the pixel and the skin tone is small. There are, however, an infinite number of functions which could encode this. The only requirement is that the probability of being a skin pixel is decreasing as a function of the distance between the colour of the pixel and the skin tone of the face.

A reasonable assumption might be that if the pixel being considered, x_i, is a skin pixel, then the Euclidean distance between the skin tone, s and the skin pixel, $d(s, x_i)$, varies as a Normal distribution with mean zero and standard deviation σ.

$$d(s, x_i) \sim N(0, \sigma)$$

However, since colours are only represented in a finite interval, for example, between zero and one (or 0 and 255), this is not strictly correct. The above model has a non-zero probability associated with impossible colour values, since there are minimum and maximum possible distances. As a

result, I use a *truncated* Normal distribution[5] instead.

The probability density function of a truncated normal distribution between the values a and b and with mean, μ, and variance, σ, is defined as:

$$f(x) = \frac{1}{\sigma} \frac{\phi(\frac{x-\mu}{\sigma})}{\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})}$$

In this case, the values of a and b represent the minimum and maximum distances respectively from the given skin tone s. Hence, $a = 0$, since the pixel being considered may have the same colour as the skin tone, in which case, $d(x_i, s) = d(s, s) = 0$. The value of b here is the distance between the skin tone and the furthest possible point in the colour space. In this approach, a three dimensional colour space is used and, hence, the value of b is the Euclidean distance between s and the furthest corner of the colourspace cube.

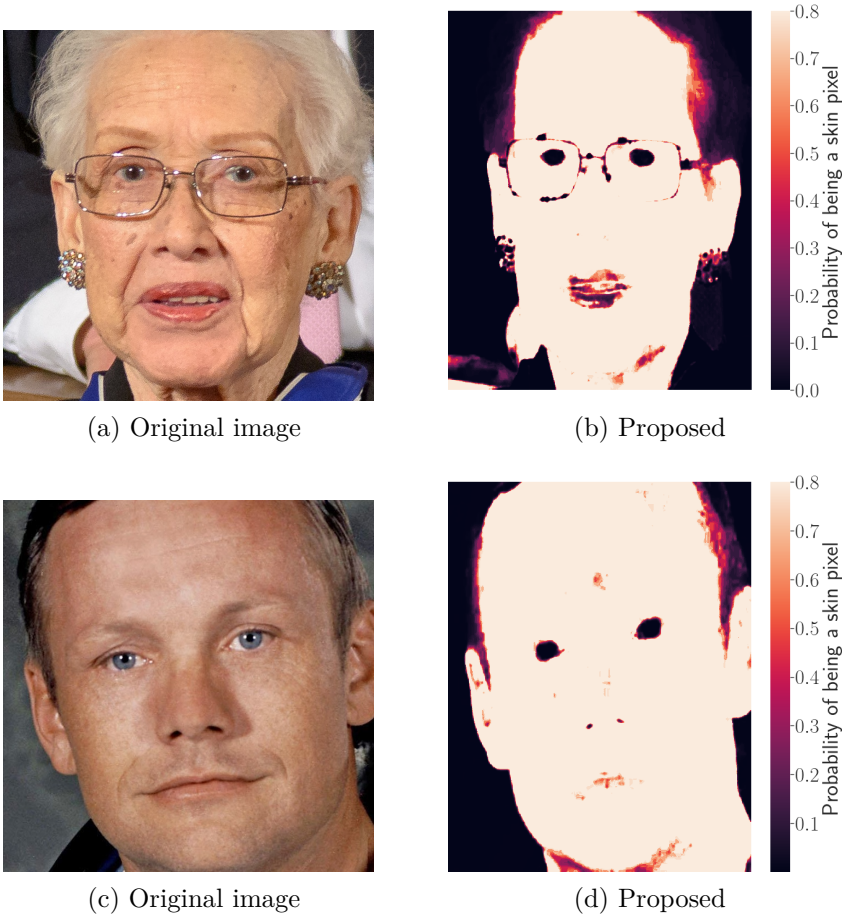
Given this, the probability is defined in proportion to the probability density function of the truncated distribution for each possible skin tone.

$$\Pr(s|C_{\text{skin}}, x_i) = \frac{f(d(s, x_i))}{\sum_{s' \in C} f(d(s', x_i))}$$

Avoiding the need for thresholding The best choice of the threshold value is not obvious, but in this context, classification is not the goal. The aim of the algorithm is to minimise the effect of non-skin pixels on the mean, so instead I proceed by taking a weighted mean colour based on the probability of each pixel being a skin pixel.

Accelerating class conditional computation Computing the class-conditional distribution for each pixel in the bounding box of the face is a costly computation. The distribution defined is continuous in nature and so maintaining floating point accuracy means the distribution cannot be precomputed (with finite memory). Instead, by, representing each component of the skin tone as an integer and rounding the Euclidean distance to the nearest integer, the distributions are precomputed for every possible skin tone. At runtime, the class conditional probability is achieved by lookup. Since the skin detection algorithm is applied to every frame in the video, using precomputed distributions achieves a 2x speedup in the overall pipeline (see Section 4.2.2).

[5] A truncated distribution modifies the domain of a distribution by defining an updated probability density function which is zero outside a particular range $[a, b]$. This is achieved in such a way that the properties of a valid probability distribution are maintained.



(a) Original image (b) Proposed

(c) Original image (d) Proposed

Figure 3.8: *An example result from the described skin detection algorithm*

The approach described achieves similar results to applying k-means per frame, but at an amortized cost inversely proportional to the number of frames in the video. In practice, this results in a order of magnitude speedup in the time to process a frame for videos with more than a single frame (see Appendix.). This means that skin detection is plausible whilst maintaining performance.

3.4 Heart rate isolation

Given a time series of the mean colour in each frame, inferring the heart rate might, naively, be taken as the prevalent frequency. That is, the largest peak in the Fourier transform. However, it is important to consider that the colours observed are not only the result of the underlying biological phenomenon of interest. This naïve assumption is prone to returning, instead, the frequency of some other factor that impacts colour of the face. For example, respiration or movement of the face will have an impact, as well as any repetitive changes in lighting such as flickering. Isolating the heart rate signal from the observed colour of the face, is a key part of the project.

To simplify this approach the problem is broken down into two subtasks:

- Identifying the pulse signal: given the noisy time series of observed colours for each frame, identify the signal corresponding to the pulse of the user
- Identifying the heart rate: given the pulse signal, identify the heart rate

Although, as will be explained, there is some overlap between the two to aid with correctly identifying the pulse.

3.4.1 Blind-source separation

Suppose that our observed colour signal, $\mathbf{I}(t)$, a vector-valued function, consists of a mixture of several underlying signals, $x_i(t)$ one of which is the pulse, $p(t)$. Our observed signal exists in three dimensions and can be viewed as the result of the mixing of three underlying signals by some mixing matrix $\mathbf{A}$.

$$\mathbf{I}(t) = \mathbf{A} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}$$

The nature of this task is to identify and return $p(t)$ from only $\mathbf{I}(t)$ and is known as the blind-source separation problem see (Section 2.4.1). The above formulation is not enough to isolate the signal $p(t)$ from $\mathbf{I}(t)$ and so assumptions must be placed on the nature of each $x_i(t)$ and $p(t)$ in order to solve this.

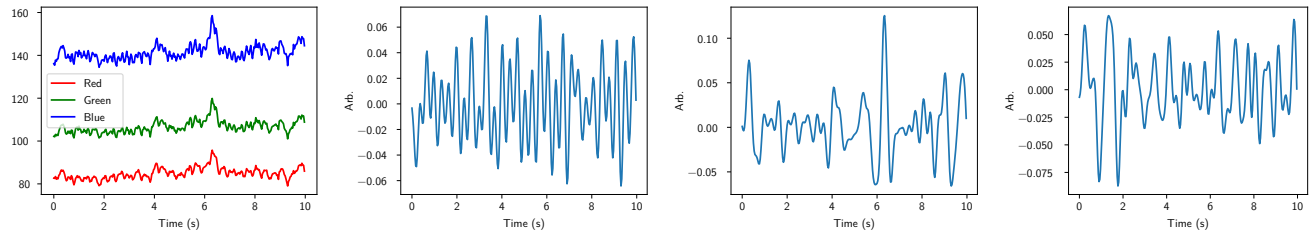
Independent component analysis (ICA)

A possible assumption might be that each of the $x_i(t)$ are statistically independent. That is, informally, one cannot gain any information about one of these signals given another. This assumption encodes the notion that the pulse should be entirely independent from the other phenomena impacting the observed signal.

For example, suppose that some of the $x_i(t)$ are the result of a physical phenomenon such as lighting conditions. In this case, it is intuitive to expect there to be no mutual information between the pulse and the other constituent signals. The ICA algorithm [8] attempts to identify these signals based on this assumption, by using non-Gaussianity as a proxy for statistical independence.

Crucially, however, this approach returns the signals $x_1(t)$, $x_2(t)$ and $x_3(t)$ but gives no indication regarding which signal corresponds to the pulse. In fact, the formulation of the ICA algorithm is such that each $x_i(t)$ are returned in a random order. Additional work must be undertaken to identify the pulse from the returned signal.

I proceed by attempting to identify the heart rate in each of these signals independently. This results in returning three different heart rate values, each of which has an associated power. This power value, which is the magnitude of the term in the Fourier transform associated with a given frequency, acts as a natural way of quantifying the importance of a particular frequency on the overall signal.



(a) *Raw video signal* (b) *ICA Component 1* (c) *ICA Component 2* (d) *ICA Component 3*

3.4.2 Identifying the heart rate

Given a pulse signal, the heart rate corresponds to the average number of beats of the heart (seen as spikes in the signal) in a minute. A possible approach to identifying this is to simply count the number of peaks in the signal. However, the Fourier transform provides a particularly rich representation from which to identify the heart rate and so is used instead. For example, the Fourier transform enables easy analysis of the relative powers of different peaks present in the power spectrum, which would not be easily possible with the former approach. Crucially, it is not always the case that the most prevalent frequency is the true heart rate. There are two cases where this may not be true.

The split peak issue

In practice the true heart rate is not present in the resulting Fourier transform as a clear single peak. Instead, often, it will correspond to several peaks that are close by each with smaller powers. A larger peak unrelated to the heart rate might be present in the resulting transform which is unrelated to the heart rate. This phenomenon was reported by van der Kooij et al. [30] and is denoted here as the *split peak issue*. In order to avoid this, a Butterworth filter is applied which removes isolated peaks and increases the power of peaks close together as is recommended by van der Kooij et al. [30].

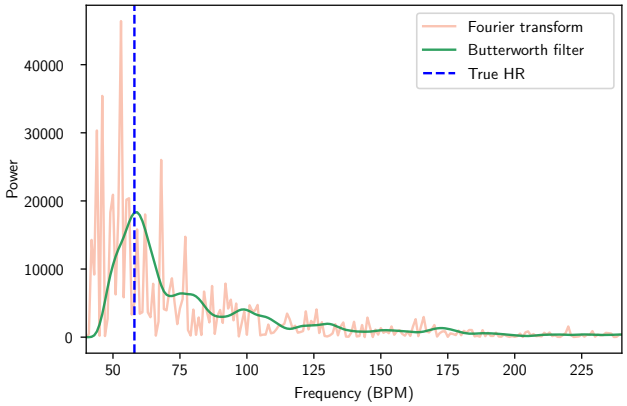


Figure 3.9: *An example of the split-peak issue*

Majority voting heuristic

In scenarios where there are large amounts of movement in the video, the most prevalent frequency may not correspond to the heart rate. This is problematic, if the movement occurs at a frequency which could be otherwise interpreted as a heart rate, that is between 0.6Hz (36 beats per minute) and 4Hz (240 beats per minute). As a result, selecting the highest peak in this case could be incorrect. For these scenarios, I have developed a heuristic denoted here as the *majority voting heuristic*.

Given the three independent signals returned by ICA, we identify the largest peak in each signal. In most scenarios, these three heart rates will be relatively close together, but they tend not to be in scenarios with lots of movement, where a single large peak at a very different frequency is common. If this is the case, then we discard the frequency of maximum power and return the frequency with greatest power of the two remaining signals. This helps to guard against a common pitfall of PPG-based systems where the frequency of movement is returned instead of the heart rate [29].

3.4.3 Summary

The work presented, largly, falls into two categories: improving the performance of the system and improving its fidelity. As two overarching goals of the project, it is in this vein it will be evaluated. All the described work is presented in the Python implementation and can be found in the supplied source code with documentation included.

3.5 Repository overview

- `dissertation`
  - `Python`
    - `tooling` Related tooling
    - `evaluation` Evaluation scripts
    - `pipeline.py` Implementation of the overall pipeline
    - `pre_compute_distributions.py` Script for precomputing the distributions required for skin detection
    - `visualisation.py` Code for visualising region selection outputs
    - `region_selection.py` Techniques implemented for region of interest selection
    - `hr_isolator.py` Code to identify the heart rate in a received signal
    - `face_det.py` Face detection and tracking implementations
  - `AndroidRPPG` An Android based demonstration of rPPG
  - `PPGLogger` Smartwatch application for recording PPG sensor values

Chapter 4
Evaluation

This project aims to increase access to heart rate sensing technology. Since sensors, in general, behave in an online way and return measurements as and when they become available, the project should be expected to behave in the same way. This is referred to as 'real-time' performance and is an extension goal of the project. In other words, this means that it should not rely on lengthy computations which a standard sensor would not be able to perform. Furthermore, the fidelity of its outputs should be in line with similar alternative sensors. The two overarching goals form the basis of this evaluation.

4.1 Data collection

In order to be able to evaluate the project, a dataset consisting of videos with an associated ground truth heart rate is required. For this, I used the *MAHNOB* dataset [20], initially collected by Imperial College London for research into affective computing, it is widely used in remote photoplethmysography literature for performance benchmarking [12][17][21]. MAHNOB, however, only consists of videos where the participants are stationary and at a fixed, close distance to the camera (see Appendix for examples), this alone, would not be adeqeuate for testing many aspects of the project. As a result, I augmented the dataset with my own experiments that contained varying amounts of movement and were recorded at different distances to the camera. The final dataset, as a result, contains a range of genders and skin tones to ensure robustness. Through this, the limitations of the project can be documented more effectively.

4.1.1 Methodology

The data collected has been used for two distinct reasons: the justification of implementation decisions and the profiling of the overall success of the project. Since this project involves the measurement of health data, it is critical to understand in what scenarios its outputs are unreliable. With this in mind, experiments were designed to discover any fundamental limitations. The data was collected with an aim to answer the following questions:

- Can the results of remote heart rate sensing be trusted?
- To what extent does the user's distance from the camera affect the fidelity of the outputs?
- Does the user have to be stationary with respect to the camera for the results to be accurate?
- How does the project compare to a PPG sensor on a consumer watch?

Experimental setup In these additional experiments, videos were recorded of myself at distances of 1m, 1.5m and 2m. At each of these distances, three different activities were recorded, each for one minute with three repeats being conducted, for a total of 27 minutes worth of video and associated heart rate data. A ground truth was taken in the form of a chest-based ECG sensor and a smartwatch was used for comparison with the video-based heart rate. The data taken from the smartwatch was the raw PPG sensor values, that is, not the estimated heart rate. This is so that the same heart rate isolation algorithms can be applied to both the remote PPG signal and the wrist based PPG signal making for a fair comparison. Video recordings were conducted on a Pixel 3A mobile device at a framerate of 30 frames per second, a resolution of 1920 by 1080 pixels and under ambient lighting conditions.

The activities used are as follows with the latter two having been selected to test both raised heart rates and significant movement, thereby providing a test case not present in the MAHNOB database. These exercises were selected to be representative of the kind of scenarios in which rPPG might be used.

- Stationary for sixty seconds
- Jogging on the spot for forty seconds followed by twenty seconds of rest
- Star jumps for forty seconds followed by twenty seconds of rest

Hypothesis testing The nature of evaluation is comparative. We would like to be able to compare results from different algorithms and be able to draw reliable conclusions. However, there is an element of chance associated with all of the measurements we would like to compare. To such end, hypothesis testing is used throughout to compare values and discern differences which may or may not be significant. Specifically, Welch's t-test [35][1] is used to test, given two sets of samples from random distributions, the hypothesis that they have equal expected values.

4.2 Analysis of performance

The main technique introduced for improving runtime performance is *face tracking*. It is not clear, initially, the extent of the performance improvement and what error, if any, it might have on heart rate measurements. In this section, it is shown to give a performance boost across all tested videos with a minimum speedup of 3x and a maximum of 8x, with no statistically significant negative effect on measurement accuracy.

4.2.1 Face tracking

Face tracking was proposed in Section 3.2.1 as an alternative to detecting the face in each frame independently. From the implementation alone, it is unclear as to whether or not it is beneficial. To answer this, several separate aspects of the algorithm must be evaluated. Specifically, any

[1] Welch's t-test is considered a more reliable alternative to the Student's t-test when samples have unequal variance [28]

described performance gains must be shown clearly as working across stationary scenarios and situations with movement of the face being tracked. Furthermore, it must be ensured that face tracking is not less accurate than simply repeatedly detecting the face in each frame. To evaluate these properties several metrics are defined and are measured across a variety of test videos.

Research questions

- Does face tracking provide a performance boost over face detection?
- Does face tracking have the same fidelity as face detection?
- Does face tracking maintain its accuracy under increasing motion?
- What value should the redetection threshold take?[2]

Metrics Face detection is a binary classification task. As an algorithm, it defines a boundary within which all pixels are defined as belonging to a face or not. As a result, if we consider face detection as a ground truth, the results of face tracking can be compared using standard classification metrics. In this case, I proceed by considering the following outcomes from the face tracker.

- False negative (FN): pixel is incorrectly classified as not belonging to a face
- False positive (FP): pixel is incorrectly classified as belonging to a face
- True negative (TN): pixel is correctly classified as not belonging to a face
- True positive (TP): pixel is correctly classified as belonging to a face

These metrics can be combined to define the recall and precision of the output of the face tracker over a given frame. These respectively represent the rate at which true skin pixels are correctly identified as such and the likelihood that a skin prediction is correct.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \text{ and } \text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

The face detector being used as a ground truth here, is in itself, imperfect. This is because, in general, face detection returns a bounding box that will contain some pixels from the background of the image but contain the vast majority face pixels. In this sense, the face detector itself has almost perfect recall but imperfect precision. Since the face tracker was designed, as described in Section 3.2.1, to mimic face detection with better performance, recall is the most important metric. However, perfect recall could be achieved by returning the entire frame as the bounding box of the face. As a result, the overall goal is to achieve perfect recall with a minimal false positive rate and so these are the metrics investigated.

[2] As defined in the pseudocode in Section 3.2.1 the `threshold` value defines when the face tracker redetects the face, specificially, it defines the percentage change in the size of the face tracked before redetection.

	Stationary		Star jumps		Jogging	
Threshold	**Recall**	**FPR**	**Recall**	**FPR**	**Recall**	**FPR**
0.10	0.938756	0.000411	0.934786	0.000702	0.948687	0.000572
0.15	0.911294	0.000506	0.902089	0.001008	0.907298	0.000794
0.20	0.873086	0.000639	0.878286	0.001326	0.877068	0.000897
0.25	0.838909	0.000611	0.860806	0.001369	0.859240	0.000982
0.30	0.829122	0.000611	0.852494	0.001472	0.850538	0.001010
0.35	0.825344	0.000714	0.846642	0.001687	0.848516	0.001064
0.40	0.824945	0.000673	0.842027	0.001867	0.844379	0.001185
0.45	0.819709	0.000720	0.844673	0.002008	0.836196	0.001140
0.50	0.825645	0.000786	0.838628	0.002211	0.837360	0.001196

Table 4.1: *A table showing the recall and false positive rate (FPR) of the face tracker across the test suite and at with different threshold values*

Since face tracking is being proposed as an optimisation, the time to process each frame, with both implementations being executed on the same hardware, is also recorded as a means of measuring the relative performance of each algorithm.

Finally, the entire test suite is run using both face detection and tracking to ensure there are no statistically significant increases or decreases in the error of the heart rate measurements.

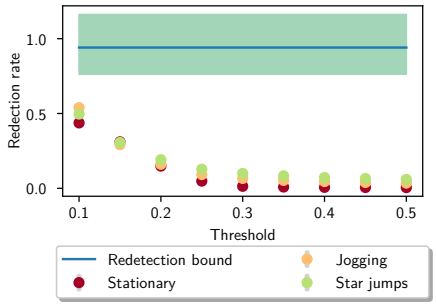
Robustness Lower threshold values, as expected, tend to result in better recall by the tracker, at the cost of more redetection. This is more or less constant across all the classes of videos tested.

Performance cost Recall from Section 3.2.1 that the following inequality was introduced that describes the performance of face tracking.[3]

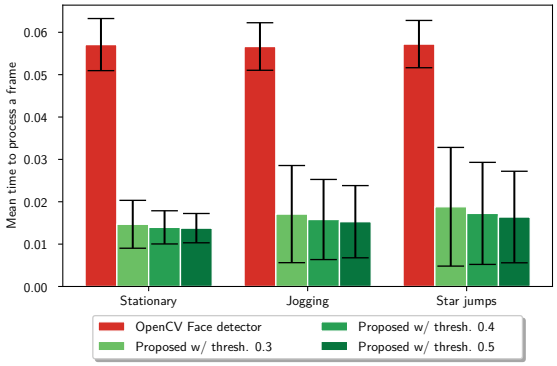
$$\frac{R}{W} < \frac{f(n) - g(p,n)}{f(n) + s(p,f)}$$

The above inequality relates the rate of redetections (R/W) and the costs of several required algorithms. It was shown in Section 3.2.1 that when this inequality holds, face tracking provides a performance benefit. In order to justify this, the value of the right hand side is evaluated experimentally on a variety of videos and is shown to exceed the value R/W across all videos and thresholds tested, as shown below.

[3]Notation: R: number of redetections, W: number of frames considered, $f(n)$: time to detect a face in a frame of size n, $g(p,n)$: time to track p points on a face of size n, $s(p,f)$: time to select p points on a face of size f



(a) *The experimental redetection rate, R/W and theoretical upper bound*



(b) *The mean time to a process a single frame for both face detection and face tracking.*

As is expected, as the threshold increases the number of redetections, decreases. However, crucially, across all the thresholds tested, the redetection rate is beneath the upper bound specified.

Results Across all the videos tested and all the threshold values, face tracking is shown to give performance benefits over face detection. Ranging from an average of 8x in the stationary case, to 3x in exercise videos.

Using a null hypothesis that both face tracking and face detection have an identical expected error, an independent two sample t-test is conducted. The p-values over each type of video tested are reported. Given a significance threshold of 0.05, the null hypothesis is rejected for only the stationary videos in favour of the face tracker. It is not rejected for the other two cases, hence, there is no statistically significant evidence to suggest face tracking introduces unexpected error in the outputs of the program.

	OpenCV		**Proposed**			
Type of video	**MAE (bpm)**	**std.**	**MAE (bpm)**	**std.**	**t-statistic**	**p**
Stationary	9.007	9.309	7.959	9.391	2.172	0.030
Jogging	65.551	15.146	64.128	16.655	1.732	0.084
Star jumps	20.035	15.314	19.752	14.989	0.366	0.714

Figure 4.1: *A table reporting the mean absolute error (MAE) in estimating the heart rate when using the OpenCV face detector on each frame vs the proposed face tracker, with independent two-sample t-tests shown.*

Summary Face tracking, as described, provides a performance benefit across all the videos tested. Critically, this comes without any evidence of a detrimental effect on fidelity. As an optimisation, this is important since, although the videos in the test suite have a fixed frame rate it allows future work to use higher frame rate cameras whilst still maintaining real-time behaviour.

4.2.2 Accelerating region selection

In Section 3.3 it was proposed that instead of computing the values of the distributions at runtime, precomputing them offline and using a lookup table would be much less costly. This change alone,

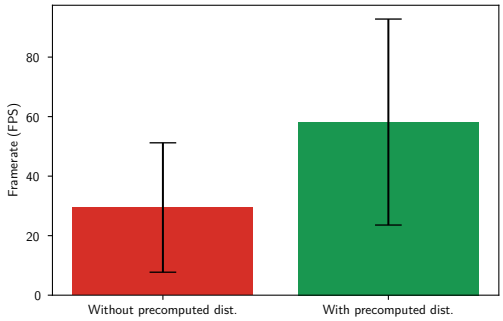


Figure 4.2: *A graph showing the effective framerate of the entire pipeline, with and without precomputing the distributions used by the skin detector*

causes an approximately 2x speedup in the effective framerate of the entire system.

4.3 Analysis of sensing fidelity

The fidelity of a sensor forms the key basis upon which it is evaluated. It is critical to be able to reason about how trustworthy its outputs are and, more importantly, to understand when its results cannot be trusted. To such end, evaluating the project, in this vein, is paramount. I show that over the test suite used, rPPG is comparable in fidelity to a smartwatch, although, crucially, both are detrimentally affected by motion artifacts.

Research questions

- Is remote heart rate sensing viable for stationary videos?
- To what extent does accuracy depend on distance and movement?
- Can it replicate or exceed the performance of a wearable device?

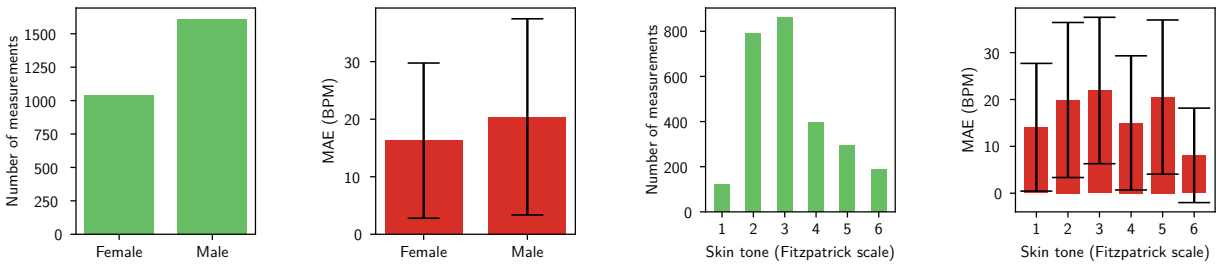
Metrics In this context, we wish to particularly penalise large differences between the true heart rate and the predicted heart rate. Hence, the *root mean squared error* is reported in the overall evaluation of the system (Section 4.3.3). The *mean absolute error* is also reported to give an intuitive understanding about the accuracy of the system.

4.3.1 Sources of error

There are four characteristics that are investigated as potential sources of error: the gender of the user, the skin tone, the amount of movement relative to the camera and the distance between the user and the camera. To quantify skin tone, the Fitzpatrick scale [10] is used, which broadly classifies skin tones into six numbered categories ranging from light to dark. For each of these, the number of measurements per category is presented with the mean absolute error.

Impact of gender

At a significance level of 0.05, there is no statistically significant difference in expected mean error for female and male users. In the experiments conducted a value of $p = 0.06642$ is reported by the significance test.



(a) *Frequency of measurements across genders* (b) *Mean absolute error by gender* (c) *Frequency of measurements per skin tone* (d) *Mean absolute error by skin tone*

Figure 4.3: *A summary of the impact of gender and skin tone on sensing fidelity*

Impact of skin tone

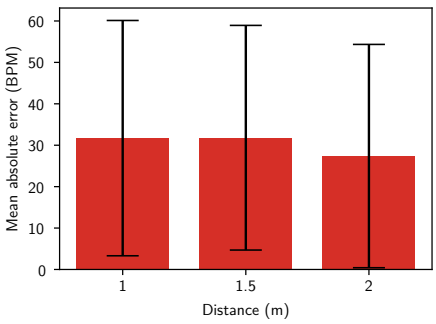
There are several statistically significant differences between individual skin tone categories, as highlighted in Table 4.2, however this arrives with two important caveats. There is no systematic trend from lighter skin tones to darker tones or in the opposite direction and there are relatively few measurements for several of the categories of skin tones. As a result, I conclude it would not be appropriate to suggest a bias against any particular skin tones.

		Skin tone					
	p-value	**1**	**2**	**3**	**4**	**5**	**6**
Skin tone	**1**		**3.555e-05**	**2.949e-08**	0.5149	**5.245e-05**	**4.655e-05**
	2			0.0106	**1.757e-07**	0.5768	**2.976e-31**
	3				**2.959e-14**	0.2031	**6.745e-42**
	4					**5.296e-06**	**4.221e-11**
	5						**1.542e-22**
	6						

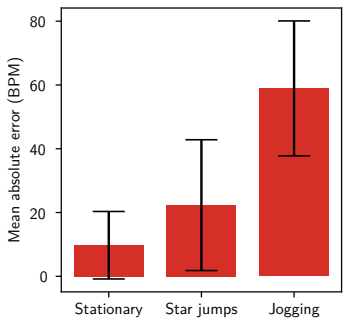
Table 4.2: *Pairwise significance testing between MAEs for different skin tones*

Impact of distance

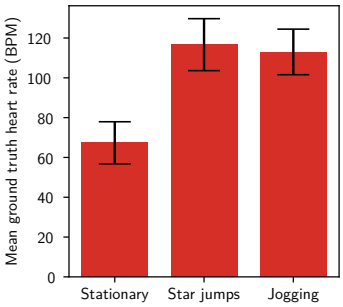
Under the experiments conducted, increasing distance shows no statistically significant detrimental impact on accuracy. Crucially, this is likely to be as a result of the nature of the experiments, rather than a trend expected to continue indefinitely. In the experiments the illuminant was behind the camera at all distances and, hence, increasing the distance had limited impact on the observed colour signal. If, for example, the illuminant was between the user and the camera then the scene might become too bright to gain a clear signal, in which case we might expect error to increase. To understand this fully, further data must be collected.



(a) *Mean absolute error per distance*



(b) *Mean absolute error per class*



(c) *Mean ground truth heart rate per class*

Impact of movement

Movement, or the lack of it, proves to have the greatest impact on accuracy. Stationary videos show much smaller errors. Furthermore, there exists a large discrepancy between the two classes of non-stationary videos. The cause of this is the subject of subsequent investigation.

Impact of class of movement

It is, initially, unclear the exact nature of movement that causes an increase in error. Concretely, what is it that separates the case of star jumping and jogging? One might hypothesise that the system produces larger errors as heart rate increases, but this can be safely ruled out since the mean heart rate in both classes is almost identical.

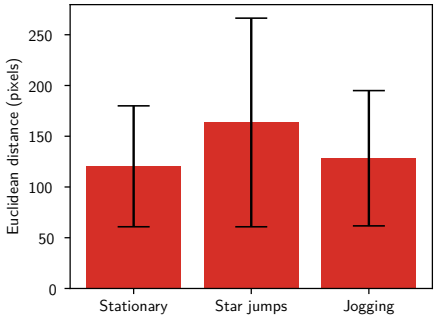


Figure 4.4: *Mean Euclidean distance moved by tracked points between frames*

Instead I propose the hypothesis that the *frequency* of movement rather than the amount is the key factor determining increasing error. Specifically that error is caused when the frequency of movement falls in the range of possible heart rate frequencies. It is clear that the amount of movement is not the determining factor since, on average, tracked points in the videos of star jumping moved *more* than in the jogging case, but showed smaller errors (Figure 4.4). In order to provide evidence for the proposed hypothesis, I define a metric to quantify the amount of noise unrelated to the true heart rate, present in the range of possible heart rate frequencies. Given this, I showcase a strong correlation with large errors in the output of the program.

Quantifiying noise The metric defined is the ratio of the sum of all the components in the Fourier transform, $F(\omega)$, and the components corresponding to the true heart rate, R, with a range δ being included to discount for errors in the ground truth. A perfect signal, that is, a signal

where the only components present in $F(\omega)$ correspond to the heart rate, will have a noise value of 1, with the noise metric increasing beyond this for imperfect signals.

$$\text{Noise} = \frac{\sum_{\omega} F(\omega)}{\sum_{\omega \in [\mathrm{R}-\delta, \mathrm{R}+\delta]} F(\omega)}$$

This is equivalent to the ratio of the sum of the components in the highlighed regions in Figure 4.5.

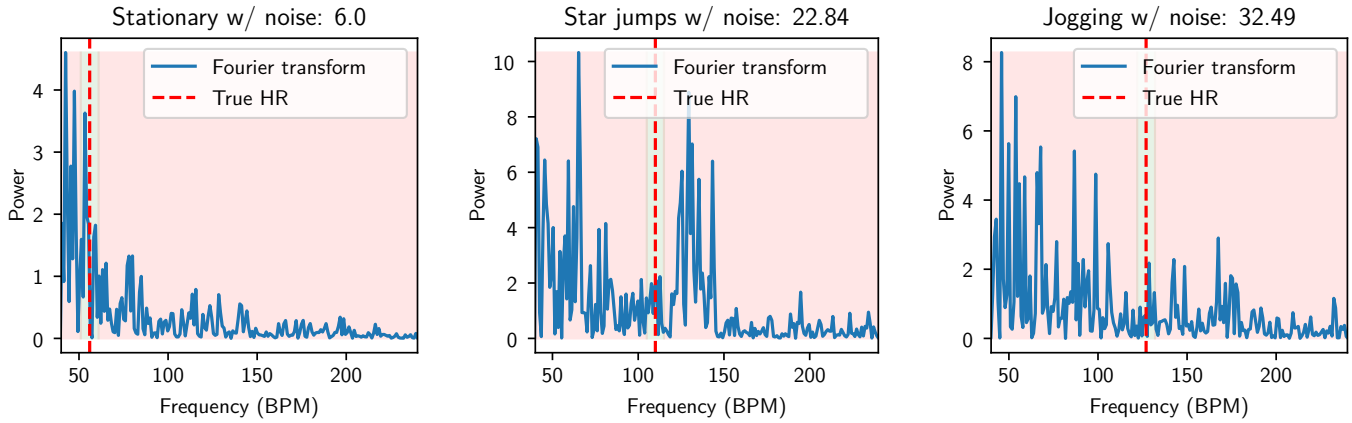


Figure 4.5: *An example of the defined noise metric*

The defined noise metric shows a strong correlation with absolute error, root mean squared error and percentage error, with all three error metrics showing a Pearson correlation coefficient of over 0.8.

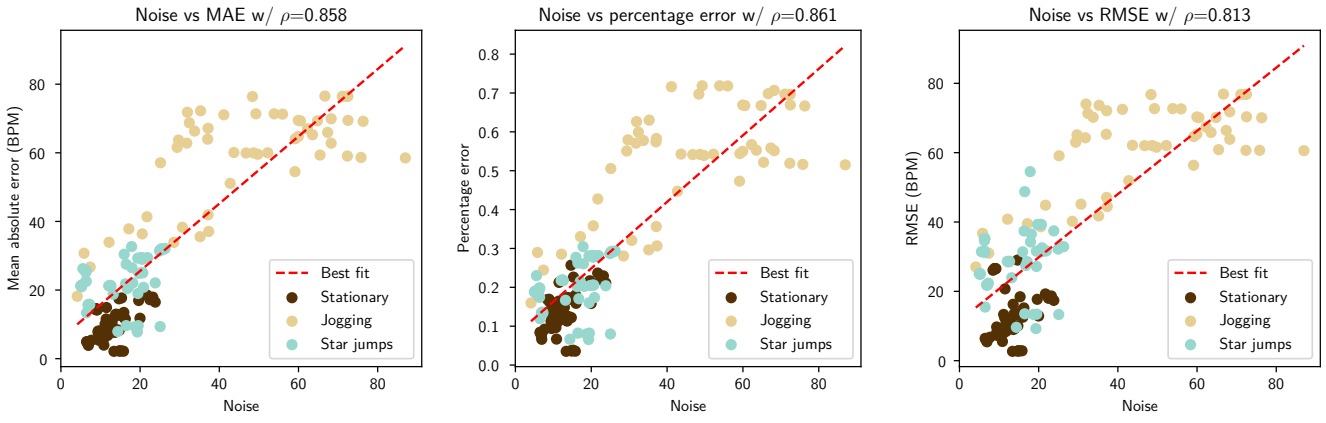


Figure 4.6: *Scatter plots showing the defined noise metric vs various error metrics with correlation coefficients*

This correlation does not, in itself, prove that the cause of the increasing error is frequency of movement but, rather, it indicates that large values of the defined noise metric is a common

property of signals which exhibit larger errors. Proving causality is not, in general, possible and hence the evidence showcased is the strongest possible proof of the hypothesis proposed.

4.3.2 Predicting unreliability

The above metric is useful for understanding the source of errors in the output of the system, however, it relies on access to a ground truth heart rate, which, in general, we would not have access to. By considering each component outputted by ICA independently and returning a heart rate for each signal, I show that the standard deviation of these outputs also shows a relatively strong correlation ($\rho = 0.707$) with error producing scenarios. Although it is a less strong correlation, it does not rely on access to a ground truth and so could be used to report potential error scenarios to users.

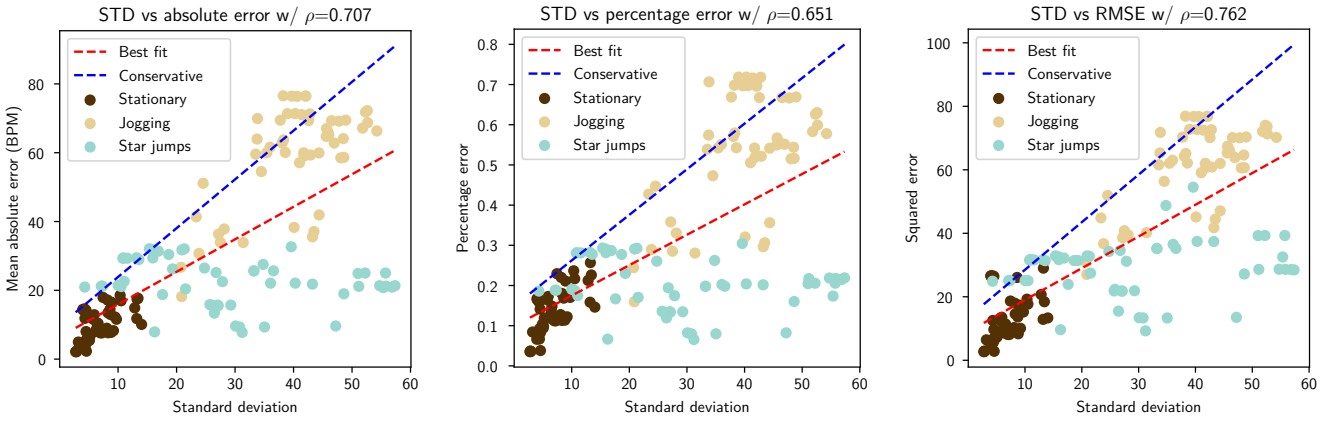


Figure 4.7: *A scatter plot showing the standard deviation of HR predictions by ICA components vs. MAE, percentage error and RMSE*

4.3.3 Overview

MAHNOB The benefit of using the MAHNOB dataset for testing, is that it allows for a direct comparison with existing literature on rPPG. Techniques proposed by Poh et al. [24], Li et al. [18] and Osman et al. [22] have all been tested on the database, with scores being reported in a comparative survey by Wang et al. [34]. Although, the precise subset of the database is not specified and nor is the size of each window being considered before estimating a heart rate. To such end, it is not a perfectly fair comparison, however, is included as an indicator of state of the art performance, in comparison with my own implementation.

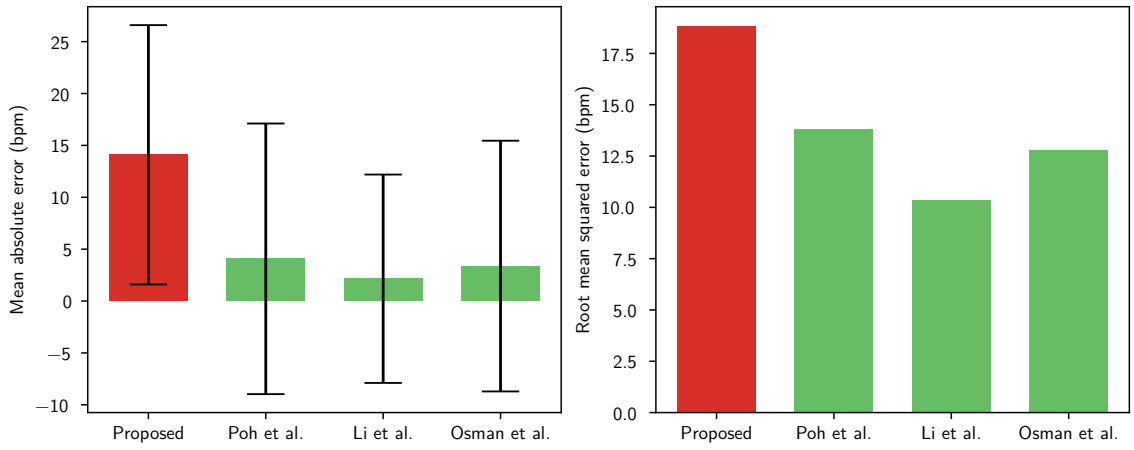


Figure 4.8: *A graph comparing the mean absolute error achieved on the MAHNOB database of the proposed implementation with comparisons to state of the art techniques.*

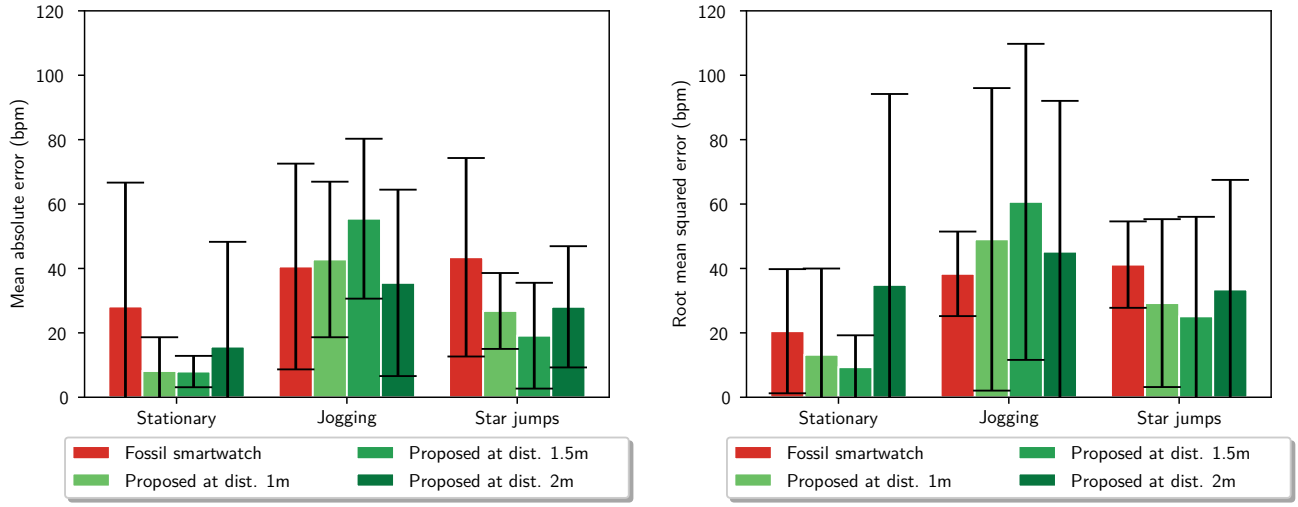


Figure 4.9: *A graph showcasing the mean absolute error and root mean squared error, in beats per minute, achieved on the collected data*

		Smartwatch		Proposed					
Exercise	**Distance**	**MAE**	**RMSE**	**MAE**	**RMSE**	t_{MAE}	p_{MAE}	t_{RMSE}	p_{RMSE}
Stationary	**1.0**	**24.3271**	**31.101**	**8.1443**	**13.1871**	**-5.8566**	**4.7376e-07**	**-3.8488**	**0.0003967**
Stationary	**1.5**	**12.1434**	**13.3007**	**7.981**	**9.3286**	**-4.2034**	**5.0180e-05**	**-3.5903**	**0.0005042**
Stationary	2.0	25.0661	40.1543	15.734	34.8932	-0.9185	0.3683	-0.3244	0.7488
Jogging	*1.0*	*46.5215*	*47.796*	*42.7873*	*49.0403*	*6.2848*	*9.1124e-09*	*7.7032*	*2.0095e-11*
Jogging	*1.5*	*35.5262*	*37.394*	*55.4532*	*60.6769*	*10.9089*	*4.3155e-19*	*10.6435*	*1.8475e-17*
Jogging	*2.0*	*32.9423*	*36.6214*	*35.5231*	*45.2086*	*3.8566*	*0.0007285*	*4.6909*	*0.0001218*
Star jumps	**1.0**	**43.3262**	**46.2494**	**26.7984**	**29.2383**	**-6.6244**	**1.2608e-09**	**-7.4695**	**3.4237e-11**
Star jumps	**1.5**	**51.2919**	**52.2412**	**19.1177**	**25.1191**	**-14.5253**	**6.3123e-27**	**-12.6585**	**4.3824e-24**
Star jumps	2.0	28.9574	31.9423	28.0886	33.4622	-1.3990	0.1735	-0.7828	0.4415

Table 4.3: *A table showing the mean absolute error and root mean squared error for both the Fossil smartwatch and the proposed system, with stastically significant results in shaded rows. Italics indicates in favour of the smartwatch and bold in favour of the proposed system.*

Results The effect of movement on PPG-based heart rate sensing is evident. Both the smartwatch and the proposed system observe much larger errors in the non-stationary cases, although this has been widely reported before [29], it is, nonetheless, concerning given that this is a common use case. In such scenarios, both systems showed mean absolute errors around 40BPM, which is, not particularly useful as a measurement, given that the range of possible heart rates is around 200BPM.

However, a key success of the project is the comparative performance with the smartwatch over the test suite. Over the nine classes of exercises and distances tested, seven results were statistically significant at a threshold of 0.01, all of which are over an order of magnitude beneath the threshold (see Table 4.3). In this scenario, there are two null hypotheses, that the absolute and squared errors, respectively, have the same expected value for the smartwatch and proposed system. In the seven cases highlighted in Table 4.3, both null hypotheses were rejected, with four in favour of the proposed system and three in favour of the smartwatch. It would be inappropriate to draw overarching conclusions about the relative fidelity of smartwatch heart rate sensing and the proposed system, however, over the test suite used, it is fair to conclude that, the performance of rPPG is, certainly, comparable to that of a smartwatch. Given that smartwatches are already widely accepted as means of heart rate sensing, these results pave the way for the uptake of rPPG.

Chapter 5
Conclusion

Non-contact heart rate estimation, as a technology, shows huge promise. Although, it might never achieve the medical-grade fidelity of an ECG sensor, it has shown that it is certainly a viable alternative to the use of smartwatches. In this project, all video data was recorded on a standard smartphone and the main software base was shown to be easily implementable as an Android application. Given that there are 2.5 billion active Android devices across the world[1], a vast number of people could gain access to their own heart rate data who might not be able to otherwise. This might have large implications for regions of the world where phones could become the most readily available heart rate sensing device. As one of several key indicators of physical wellbeing, this has large scope to benefit those who might otherwise not be able to measure these values.

5.1 Successes and failures

The project was a resounding success. Having achieved all of my core criteria and two of my three extensions, I am very much pleased with the end result. There were several statistically significant differences in accuracy, some in favour of the Fossil smartwatch and some in favour of my implementation. This indicates the feasibility of my project as a means of heart rate sensing and was both surprising and a major success of the project. Furthermore, the optimisations introduced for the face detection stage were a substantial technical undertaking that ending up being hugely beneficial, by allowing more complicated region selection algorithms to become feasible.

However, since the project contained large amounts of research work, where the undertakings may not necessarily be fruitful, it was difficult at times to stay on course. Although, this was the nature of the project, I should have allocated more time to these research avenues, and iterated more quickly. This being said, the project managed to finish on time, largely due to the conservative initial timeline set out in the proposal.

5.2 Personal remarks

Having self taught two new programming languages as well as learning about Android programming, this project has hugely benefitted my software development skills. The key concepts underpinning this project, specifically in the fields of sensing, computer vision and signal processing were also new to me and have been very rewarding to learn about and implement ideas from.

[1] Google I/O Conference 2019

5.3 Future work

As an active area of research, there are always further aspects of remote heart rate sensing to investigate. Some of which, I intended to pursue but did not manage due to time constraints.

- **Illumination correction**: the main source of error in videos with increasing amounts of movement, is that the illumination incident onto the user is constantly changing as they move with respect to the light source. There is lots of research in computer vision as to how we can correct for these effects, such as the presence of specular highlights [37][36][15]. By implementing this, it might be possible to improve the performance of rPPG in scenarios with lots of movement.
- **Multiple cameras**: being able to integrate signals from multiple camera streams of the same user may help to overcome the issues caused by increasing amounts of movement.
- **Energy cost**: if we view rPPG as a kind of sensor, then its energy consumption is a critical factor that has not been considered here. If we wish to use it for continuous heart rate measurement over long periods of time, then this might become a stumbling block for wider uptake.
- **Further testing**: it was beyond the scope of this project to conduct very large scale testing across, say, hundreds of users, however, before it can be reliably used across entire populations, it should be tested across a wider variety of people.

Bibliography

[1] Kotlin programming language. `kotlinlang.org`. Accessed: 2019-10.

[2] OpenCV library. `opencv.org`. Accessed: 2019-10.

[3] Python programming language. `python.org`. Accessed: 2019-10.

[4] scikit-learn library. `scikit-learning.org`. Accessed: 2019-10.

[5] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[6] Weixuan Chen and Daniel McDuff. Deepphys: Video-based physiological measurement using convolutional attention networks. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 356–373, Cham, 2018. Springer International Publishing.

[7] E. Colin Cherry. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the Acoustical Society of America*, 25(5):975–979, 1953.

[8] Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

[9] M. Da'san, A. Alqudah, and O. Debeir. Face detection using viola and jones method and neural networks. In *2015 International Conference on Information and Communication Technology Research (ICTRC)*, pages 40–43, 2015.

[10] Thomas B. Fitzpatrick. The Validity and Practicality of Sun-Reactive Skin Types I Through VI. *Archives of Dermatology*, 124(6):869–871, 06 1988.

[11] D. Fleet and Y. Weiss. *Optical Flow Estimation*, pages 237–257. Springer US, Boston, MA, 2006.

[12] M A Hassan, A S Malik, D Fofi, N Saad, and F Meriaudeau. Novel health monitoring method using an rgb camera. *Biomedical optics express*, 8(11):4838–4854, 10 2017.

[13] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[14] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, July 2002.

[15] Chen Li, Stephen Lin, Kun Zhou, and Katsushi Ikeuchi. Specular highlight removal in facial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3107–3116, 2017.

[16] Tzu-Mao Li, Michaël Gharbi, Andrew Adams, Frédo Durand, and Jonathan Ragan-Kelley. Differentiable programming for image processing and deep learning in halide. *ACM Transactions on Graphics (TOG)*, 37(4):139, 2018.

[17] X. Li, J. Chen, G. Zhao, and M. Pietikäinen. Remote heart rate measurement from face videos under realistic situations. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4264–4271, 2014.

[18] Xiaobai Li, Jie Chen, Guoying Zhao, and Matti Pietikainen. Remote heart rate measurement from face videos under realistic situations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4264–4271, 2014.

[19] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision (darpa), April 1981.

[20] T. Pun M. Soleymani, J. Lichtenauer and M. Pantic. "a multimodal database for affect recognition and implicit tagging". *IEEE Transactions on Affective Computing. 3*, pages 42–55, April 2012.

[21] X. Niu, H. Han, S. Shan, and X. Chen. Continuous heart rate measurement from face: A robust rppg approach with distribution learning. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 642–650, 2017.

[22] A. Osman, J. Turcot, and R. E. Kaliouby. Supervised learning approach to remote heart rate estimation from facial videos. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 1, pages 1–6, 2015.

[23] Fulai Peng, Zhengbo Zhang, Xiaoming Gou, Hongyun Liu, and Weidong Wang. Motion artifact removal from photoplethysmographic signals by combining temporally constrained independent component analysis and adaptive filter. *BioMedical Engineering OnLine*, 13(1):50, 2014.

[24] Ming-Zher Poh, Daniel J McDuff, and Rosalind W Picard. Advancements in noncontact, multiparameter physiological measurements using a webcam. *IEEE transactions on biomedical engineering*, 58(1):7–11, 2010.

[25] Ming-Zher Poh, Daniel J McDuff, and Rosalind W Picard. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Optics express*, 18(10):10762–10774, 2010.

[26] Abhinav Dhall Rajen Bhatt. Skin segmentation dataset. *UCI Machine Learning Repository.*

[27] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[28] Graeme D. Ruxton. The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test. *Behavioral Ecology*, 17(4):688–690, 05 2006.

[29] Thiago Toledo Souza. Heart rate estimation during physical exercise using wrist-type ppg sensors. 2019.

[30] Koen M. van der Kooij and Marnix Naber. An open-source remote heart rate imaging method with practical apparatus and algorithms. *Behavior Research Methods*, 51(5):2106–2119, 2019.

[31] Wim Verkruysse, Lars O. Svaasand, and J. Stuart Nelson. Remote plethysmographic imaging using ambient light. *Optics express*, 16(26):21434–21445, Dec 2008. 19104573[pmid].

[32] Nelson JS Verkruysse W, Svaasand LO. Remote plethysmographic imaging using ambient light. Jul 2009.

[33] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[34] Chen Wang, Thierry Pun, and Guillaume Chanel. A comparative survey of methods for remote heart rate detection from frontal face videos. *Frontiers in Bioengineering and Biotechnology*, 6:33, 2018.

[35] B. L. WELCH. The generalization of 'student's' problem when several different population varlances are involved. *Biometrika*, 34(1-2):28–35, 01 1947.

[36] Qingxiong Yang, Jinhui Tang, and Narendra Ahuja. Efficient and robust specular highlight removal. *IEEE transactions on pattern analysis and machine intelligence*, 37(6):1304–1311, 2014.

[37] Qingxiong Yang, Shengnan Wang, and Narendra Ahuja. Real-time specular highlight removal using bilateral filtering. In *European conference on computer vision*, pages 87–100. Springer, 2010.

Appendices

Appendix A
Accelerating region selection

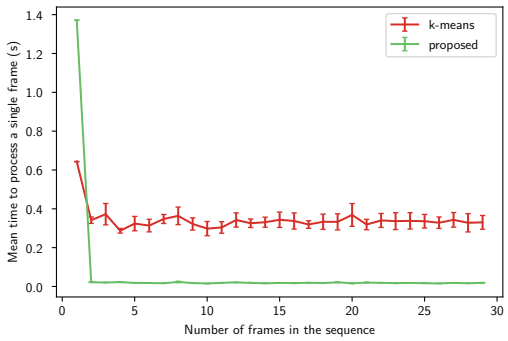


Figure A.1: *A graph showing the mean time to process a single frame when using k-means on each frame vs. skin detector described*

Appendix B
Experiment example distances

Figure B.1: *Example distance of 1m*

Figure B.2: *Example distance of 1.5m*

Figure B.3: *Example distance of 2m*

Appendix C
MAHNOB example

Figure C.1: *An example recording taken from the MAHNOB [20] database*

Appendix D
Project Proposal

Computer Science Tripos - Part II - Project Proposal

Non-contact heart rate estimation from video

Yousuf Mohamed-Ahmed, Gonville and Caius College

Originator: Dr Robert Harle

Supervisor: Dr Robert Harle

Directors of Studies: Timothy Jones, Graham Titmus and Peter Robinson

Overseers: Rafal Mantiuk and Andrew Pitts

Introduction and Description of the Work

Heart rate measurements from smartwatches are notoriously spurious and can, especially during exercise, provide inaccurate measurements. Optical heart rate monitors work by measuring the amount of light emitted by the monitor that is reflected by the surface of the skin. From this, the volume of blood flowing beneath the surface can be inferred, known as a plethysmogram. Tracking this value through time allows the heart rate of the user to be estimated. However, large amounts of movement, as typical during exercise, generate motion artifacts which degrade the accuracy of the measurements from smartwatches by decreasing the signal-to-noise ratio of the plethysmogram.

Papers have shown that an accurate heart rate value can be extracted from a video recorded by a camera [32]. This is because slight changes in the colour of the face, as well as slight movements of the face, allow a plethysmogram to be inferred. Any technique which can measure a plethysmogram optically via a non-contact method is known as remote photoplethysmography (rPPG) and is an active area of research. This project is concerned with the implementation of such a system which should be accessible through an Android application using a phone camera.

Since, as some experiments have shown, the face contains a greater PPG signal than the wrist [30], I would like to investigate the extent to which rPPG could be used as a replacement for smartwatches during exercise. This will involve investigating the effects of distance on accuracy as well as comparisons of different algorithms for computing the heart rate.

It is expected that some may perform better than others in certain scenarios such as amount of movement or lighting conditions and, as a result, there is scope for combining algorithms or

using heuristics for selecting appropriate implementations at runtime. In order to develop these heuristics, the effects on accuracy of numerous conditions should be examined such as, potentially, the effects of the resolution and framerate of the camera.

This could be critical since many current rPPG systems do not run in realtime and thus, any reductions in the amount of processing required, for example, by downsampling images or not considering every frame, could help to maintain accuracy whilst improving performance.

Furthermore, recent reductions in the price of smartphones means that high-quality cameras are much more ubiquitous than heart-rate measuring equipment. An abnormal heart rate can be indicative of wider medical problems and, as a result, the ability to measure heart rates easily could be helpful for communities without regular access to healthcare. To that end, this project may be suitable for medical uses in remote areas.

Starting Point

I have no previous experience developing Computer Vision applications and nor in Digital Signal Processing. However, I believe, Part 1A Introduction to Graphics and Part 1B Further Graphics will prove useful precursors to understanding the Computer Vision aspects of the project. Likewise, I am currently studying Part II Information Theory which I hope will help provide a grounding in some of the mathematics behind Digital Signal Processing, which in turn will help with understanding the fundamental algorithms in the field.

Work to be Done

Face detection

The face is typically the most exposed region of the body during exercise and is believed to be the easiest region to extract a reliable PPG signal from [30]. Given this idea, the system, from videos, will have to be able to identify faces within the videos. At present, this is likely to involve a sweep of the frame using the Viola-Jones algorithm to box any faces in the frame. This initial stage may then be followed by picking out the exact area of the face from within the box.

Region of interest selection

Within the region of the face itself different areas contain differing amounts of information regarding the pulse. For example, eyes give no information about the pulse and it is speculated that some regions of the face such as the forehead and cheeks give more information than other regions like the nose. Several algorithms will be developed to locate these regions and their performance will be compared.

Region tracking

Once a region is selected, it must be tracked between frames. This is because the colour of the face itself is of no real concern but the frequency at which the colour changes is the means by which we can infer the heart rate. Thus if different regions in each frame are tracked then this frequency will begin to diverge from the true value. This is likely to use some kind of Optical Flow algorithm, the exact nature of which will be investigated.

Signal processing of RGB signals

The resulting signals extracted from the region of interest (ROI), will be very noisy and will require the use of signal processing techniques before applying a Fourier transform to extract the prominent frequency. It is expected that this will require the use of a Blind Source Separation technique such as Independent Component Analysis to separate the independent sources contributing to the signal.

Android application

An Android application will be developed for easy testing of the system by allowing users to record videos of themselves and estimate their own heart rate.

Interaction between Android application and video analysis

The provision of signal processing and computer vision libraries isn't particularly strong in the JVM languages which are how Android applications are typically programmed. As a result, it is likely that the the video analysis software will be written as a separate program, most likely in Python or Julia, which will then interact with the Android application via pipes.

Evaluation

I hope to conduct my own experiments on the performance of the system relative to a smartwatch when compared to a known ground truth. As well as experiments across a variety of skin tones and lighting conditions.

Test bench application

In order to test the developed system, in comparison with a traditional smartwatch, an application for extracting the heart rates measured by the smartwatch will have to be developed. This will take the form of a logging application which will be able to run in the background on the watch.

Possible Extensions

- Tracking multiple faces in a single video
- Measuring heart rate whilst exercising

  - **Increased tracking**: the core program is only expected to deal with minor movements such as limited head movement, an increase in the stability of the tracking algorithms will almost certainly be required to deal with exercising users.
  - **Dealing with increased distance from camera**: an upscaling algorithm might be required to be able to extract heart rate from a distance, since the face region of the user might be small which would lead to a decrease signal-to-noise ratio, upscaling may help to increase this.
  - **Increasingly noisy signals**: more rigorous signal processing will be required to remove additional noise caused by movement from the signals. This is because the frequency of movement may fall within the same range as the expected frequency of the heart [23], hence simple band-pass filters will no longer work.

- End to End Deep Learning Approach: Recent papers [6] have shown that we can use models based on Convolutional Neural Networks taking a pair of frames to estimate the change in pulse. An implementation of this system could then be compared to the core program.

- Most systems outlined in the literature carry out off-line processing of the video frames, however, many provisions exist on Android for parallelized computation on images [16]. These could be utilised to develop a real-time application.

Success Criteria

- Develop an Android application that allows users to estimate their own heart rates

- Stationary users in appropriate lighting conditions should be able to measure their heart rate with reasonable accuracy

Timetable

I have created a timetable consisting of 12 2-week periods starting on 26/10/2019 and finishing on 25/4/2020.

1. **26/10/2019 - 09/11/2019**
   I will begin by conducting research into the OpenCV library for Computer Vision and assessing the provisions already present in the library. I should also be prototyping, most probably in Python, the face detection.

2. **09/11/2019 - 23/11/2019**
   The process of researching and prototyping should continue, with a rough structure of the analysis software in place, meaning that the program should be able to receive streams of frames and detect faces in each frame.

3. **23/11/2019 - 07/12/2019**
   Various Region of Interest selection algorithms should be implemented and tested, although their effect on accuracy cannot be measured yet, they should be tested for correctness.

4. **21/12/2019 - 04/01/2020**
   Several different point tracking algorithms should be selected and included in the program. At present, this is likely to include Lucas-Kanade optical flow (a sparse technique) and dense optical flow.

5. **04/01/2020 - 18/01/2020**
   Implement signal processing pipeline to allow for cleaning of RGB signal and extracting heart rate. This will complete the analysis software and I expect tweaking of the pipeline to occur at this stage based on any accuracy issues.

6. **18/01/2020 - 01/02/2020**
   Write the progress report and begin writing an Android application to allow for users to measure their own heart rate, should also allow for overlaying heart rate on the image.

7. **01/02/2020 - 15/02/2020**
   Write serialisation code to allow for communication between Android application and the analysis program. With the completion of this, the core project should be finished.

8. **15/02/2020 - 29/02/2020**
   Begin drafting the introduction chapter of the dissertation as well as beginning work on the code required to evaluate the project. This will include code for running experiments and measuring accuracy.

9. **29/02/2020 - 14/03/2020**
   Continue writing the introduction chapter and begin work on the preparation chapter, as well as beginning work on the extensions.

10. **14/03/2020 - 28/03/2020**
    Work on the extensions should be concluded and the implementation chapter finished. Feedback on the previous chapters should be sought out and appropriate modifications made.

11. **28/03/2020 - 11/04/2020**
    Finish writing the dissertation and seek final supervisor comments.

12. **11/04/2020 - 25/04/2020**
    Incorporate suggestions and hand in the final version.

Resource Declaration

I will use my own laptop, a Lenovo 510s with a 2.5 GHz Intel Core i5 CPU and 8GB of RAM. I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure. All my code for the project and the dissertation itself

will be pushed to a Git repository that will be hosted on GitHub. I will push to this regularly so that, in the case of an issue with my laptop, I will be able to resume work promptly. I own an Android phone (Google Pixel 3A XL) which will be used as part of the development process. I have also been provided on loan with a smartwatch by my supervisor, the Fossil Q smartwatch which has Wear OS installed, will be used for any evaluation requiring a smartwatch comparison.