Yousuf Mohamed-Ahmed

Non-contact heart rate estimation from video

Computer Science Tripos - Part II

Gonville & Caius College

March 23, 2020

Proforma

Name:	**Yousuf Mohamed-Ahmed**
College:	**Gonville & Caius College**
Project Title:	**Non-contact heart rate estimation from video**
Examination:	**Computer Science Tripos - Part II, July 2020**
Word Count:	**4681**[1]
Project Originator:	Dr R. Harle
Supervisor:	Dr R. Harle

Original Aims of the Project

Work Completed

Special Difficulties

None.

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

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 March 23, 2020

Contents

Chapter 1

Introduction

Overview Optical heart rate monitors, the kind of which are standard in modern wearables, work, essentially, by measuring the amount of light emitted by the surface of the skin. Given this signal, a heart rate can be extracted by analysing the predominant frequencies in this signal. In principle, this notion, can be extended to traditional cameras such as those present in smartphones, which also work by effectively measuring light intensity. This project is concerned with the implementation of such a system which can infer the heart rate of a user from a video of that user, recorded by a simple camera such as those present in a smartphone or similar device.

Motivation In general, sensing technology has progressed in recent times from high fidelity and low availability to low fidelity and high availability. Heart rate sensing used to be the remit of expensive electrocardiogram devices in hospitals, but is now commonplace in relatively low cost wearable devices. This trend allows increasing number of users to gain access to their own health data easily albeit at the cost of decreased fidelity. I argue that smartphones are in fact of greater availability than wearables and, thereby, the development of heart rate sensing capabilities for smartphones represents further progress in this wider trend. In fact, I will later argue that smartphones are of greater fidelity that wearable devices for heart rate sensing, in which case, they represent a much more significant leap forward than might be anticipated.

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

Before attempting to develop a system which can estimate the heart rate of a user without physical contact, it is useful to understand how traditional heart rate sensors work. Although it is clear that a non-contact system is constrained by different requirements, it is common when developing low fidelity sensors to begin by attempting to mimic high fidelity alternatives.

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 for this reason, that the planning of the project proceeds by, first, understanding the workings of the two main alternative sensors and then reasoning about how one might derive these results from a camera instead.

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

2.1.1 Electrocardiogram

2.1.2 Photoplethysmography

2.2 Remote photoplethysmography (rPPG)

2.2.1 Existing literature

2.2.2 Possible extensions

2.3 Relevant computer vision techniques

2.3.1 Face detection

2.3.2 Optical flow

2.3.3 Harris corner detection

2.4 Approaches to signal processing

2.4.1 Independent Component Analysis

2.4.2 Principal Component Analysis

2.4.3 Fourier transformation

2.5 Languages and tooling

2.5.1 Languages

Python

Kotlin

2.5.2 Libraries

OpenCV

Mobile Vision API

2.6 Starting Point

2.7 Requirements analysis

2.8 Professional practice

Chapter 3

Implementation

3.1 Overview

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 some region of the face, infer the heart rate

Face detection is largely a solved problem and thereby the project mostly concentrates on the latter two, for which, there is very much ongoing research.

3.2 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 some adequate number of data points is 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 from the earlier stages. Separating these tasks allows for this concurrency to be implemented safely.

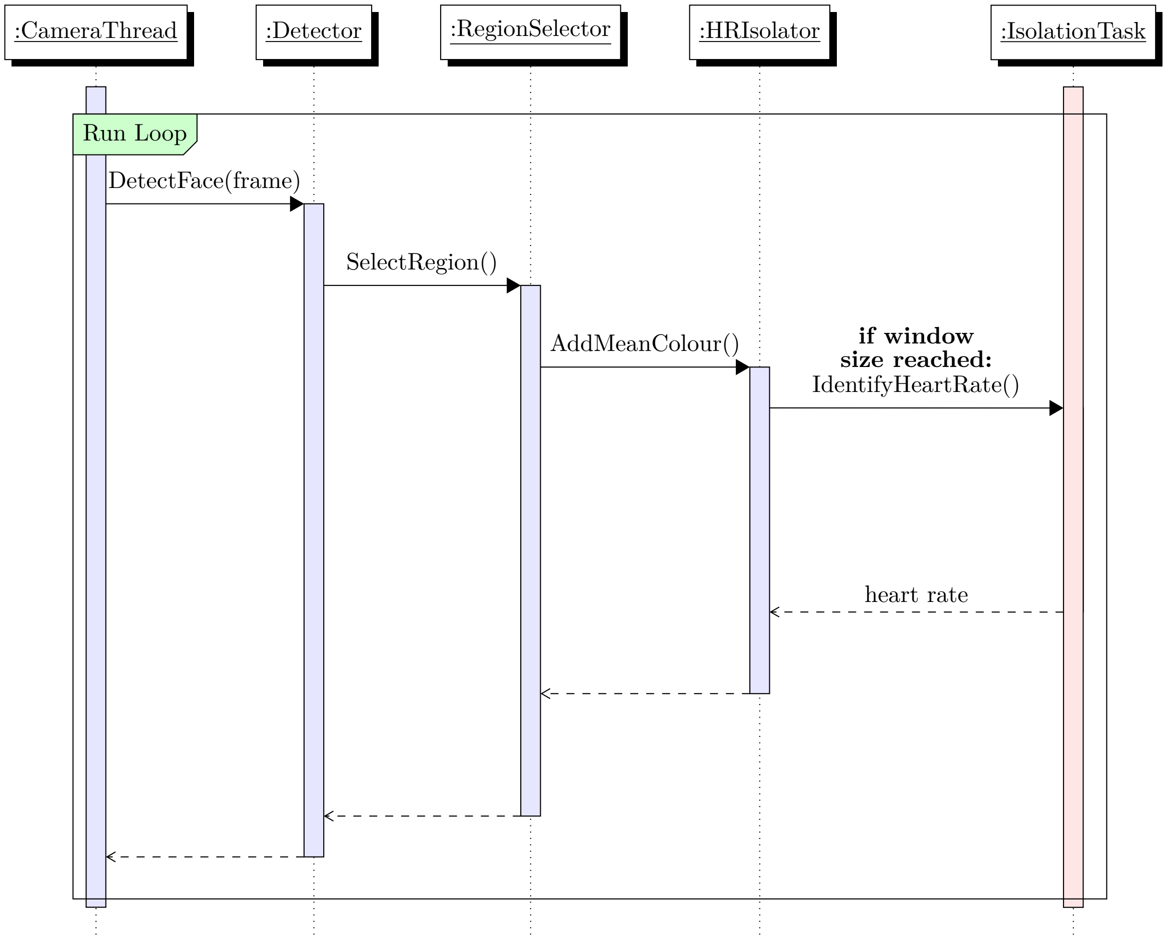

Figure 3.1: A UML sequence diagram showcasing the use of threading

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.

3.3 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.

3.3.1 Face tracking

Smartphone cameras can readily stream at high framerates, so it is unlikely that a face moves very much between a pair of consecutive frames. At sixty frames per second, frames are recorded only 0.017s 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. This is important since 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 implemented an optical-flow based algorithm that tracks points on the face rather than repeatedly calling the face detector. The Lucas-Kanade algorithm [1], a particular variant of optical-flow, tracks a set of points between consecutive frames. Naturally, over time, these points will diverge from their true positions. This is because images contain large numbers of pixels with similar illumination, over time, these 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. It is crucial for any implementation to act safely enough that the face is not lost track of, whilst simultaneously minimising the number of redetections. 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. We track the face and if the size of the box surrounding it changes significantly, indicating that our confidence in the tracked points has reduced, then we recompute the true position of the face.

```
points = []
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
    previous_face = 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 4.1. 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.

Details To maximise the performance of the optical flow algorithm, selecting points randomly will not suffice. That is because, points which are particularly different to their neighbours, for example, are much easier to track. To achieve this, the Shi-Tomasi corner detection algorithm is used which returns some number of points satisfying this condition. Furthermore, for face tracking to work properly, the points selected must span the entire face of the user. This is relatively easily enforced by mandating a minimum distance between each of the points selected and by selecting enough points.

It is conservative enough that the true position of the face is not lost, whilst still providing a 3x performance boost on the naive face detection approach. This is fully evaluated in section 4.1.

3.4 Region selection

Face detection systems, typically, return a bounding box, within which it is believed a face is present. However, naturally, the box will also 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 a somewhat unsolved problem. Furthermore, it is unclear whether all skin pixels are equally useful. For example, it might be that cheeks contain greater predictive power than the nose. Several approaches are presented regarding this problem with associated algorithms that are evaluated in section 4.2.

3.4.1 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, 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. [3] and was 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.

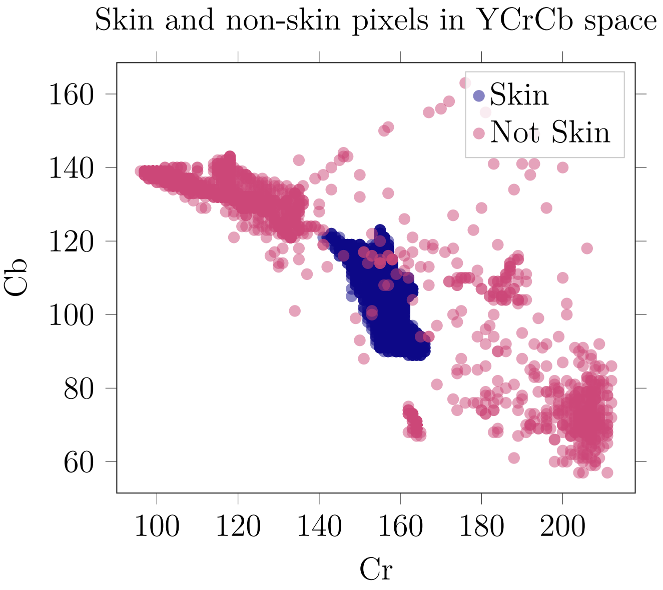

Figure 3.3: A randomly sampled subset of the skin dataset represented in the Cb-Cr axes of YCbCr space

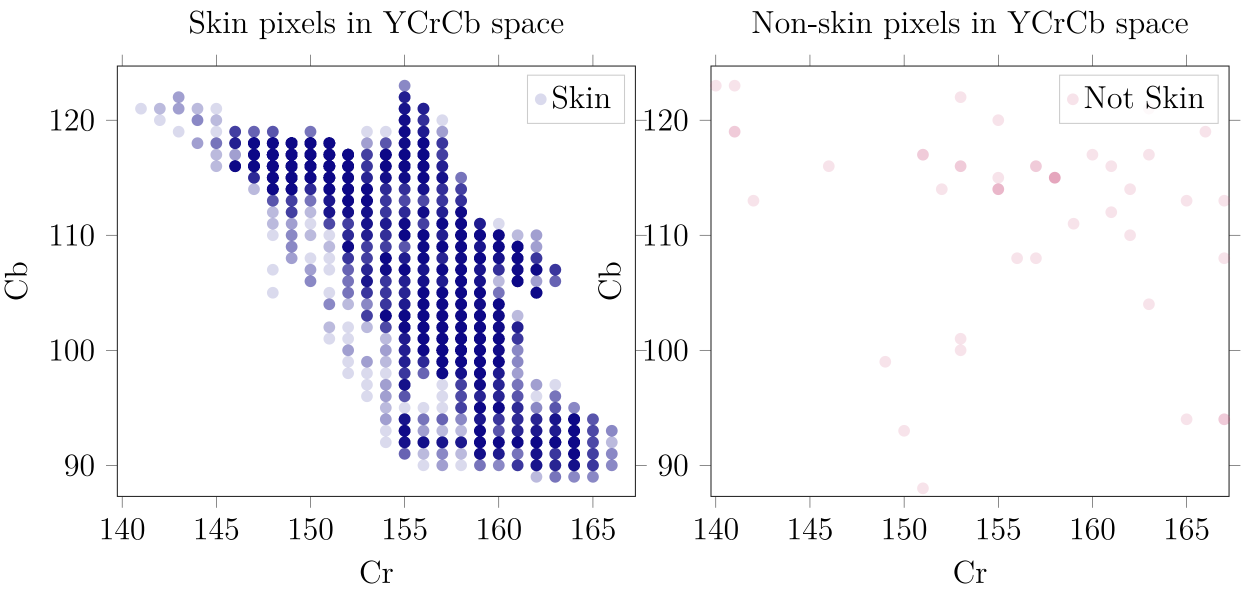

Figure 3.4: The range of skin tones present in the dataset with the skin and non-skin pixels plotted

Issues This approach takes no consideration of the particular face that it is considering. 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.

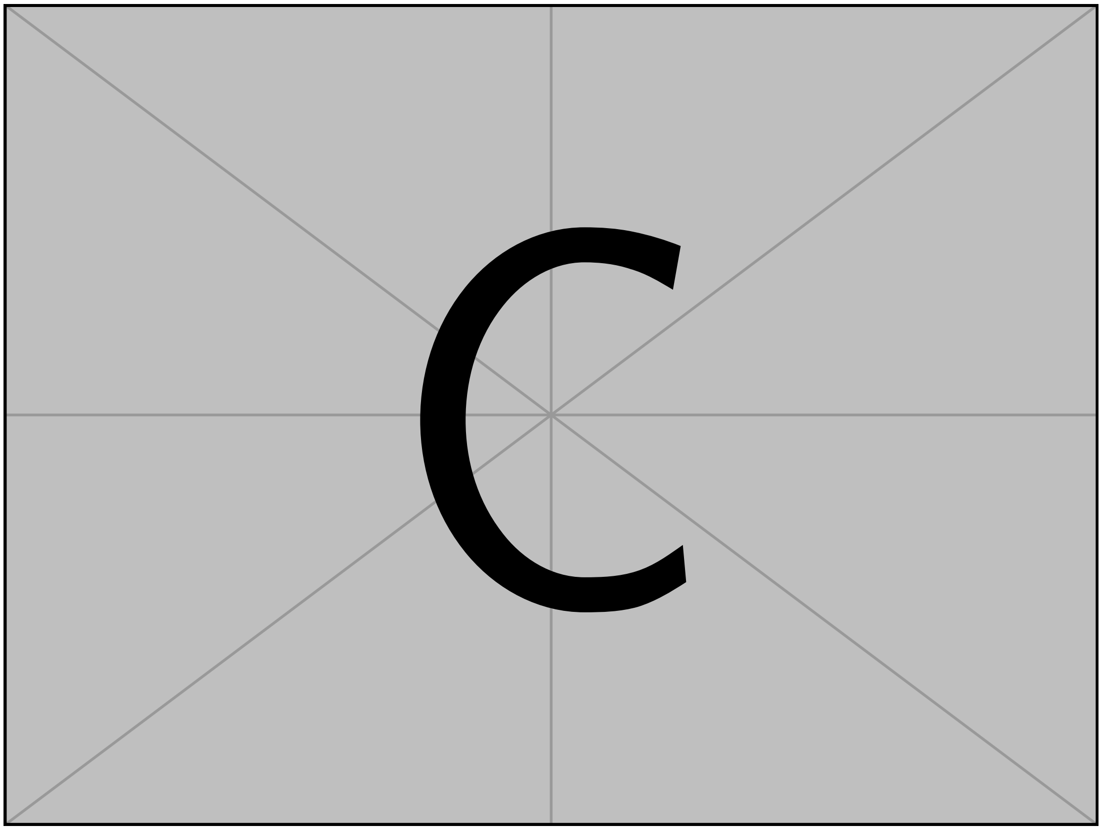

Figure 3.5: An example of the hair failure case

Instead, an algorithm should attempt to identify skin on the particular face in question.

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 to identify these clusters of pixels. Under the further assumption that the majority of pixels are skin pixels, we return the largest cluster as our set of skin pixels.

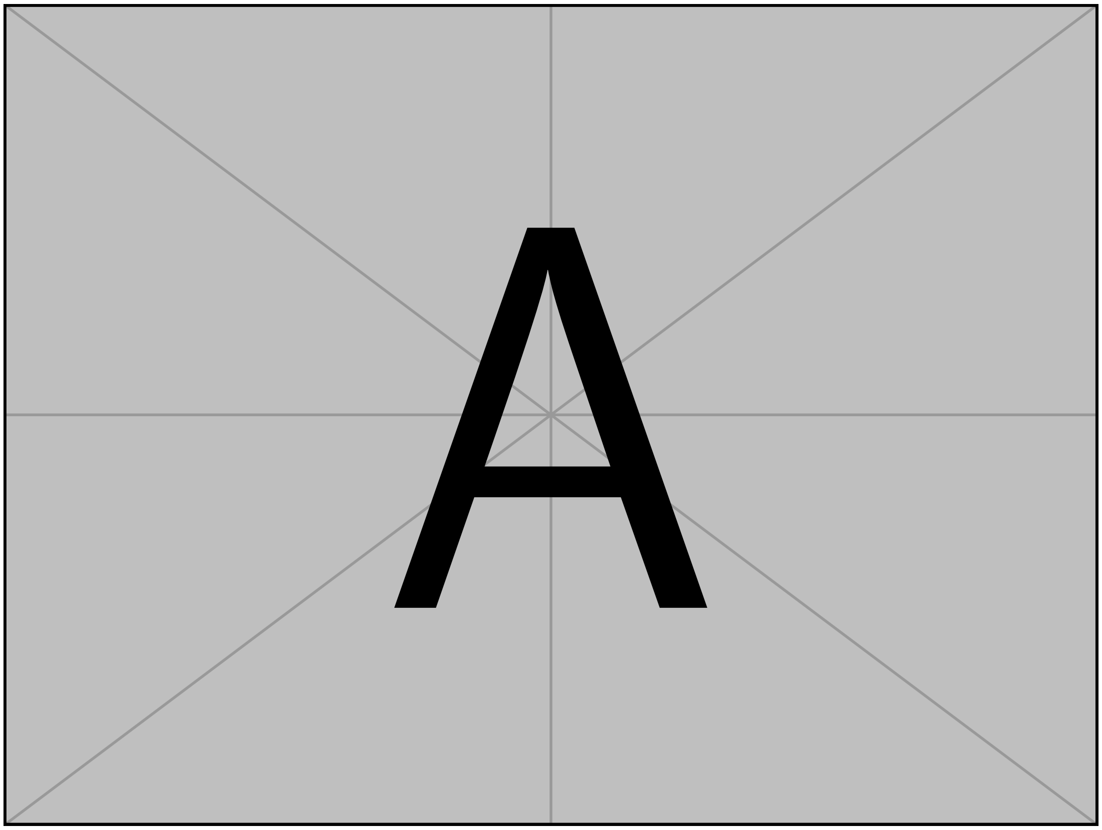

Figure 3.6: An example of the application of the k-means algorithm to skin detection

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

- 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

- Performance: recall from Section 3.2 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 SKLearn library takes an order of magnitude longer than the minimum requirement for this constraint. [1]

- 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.

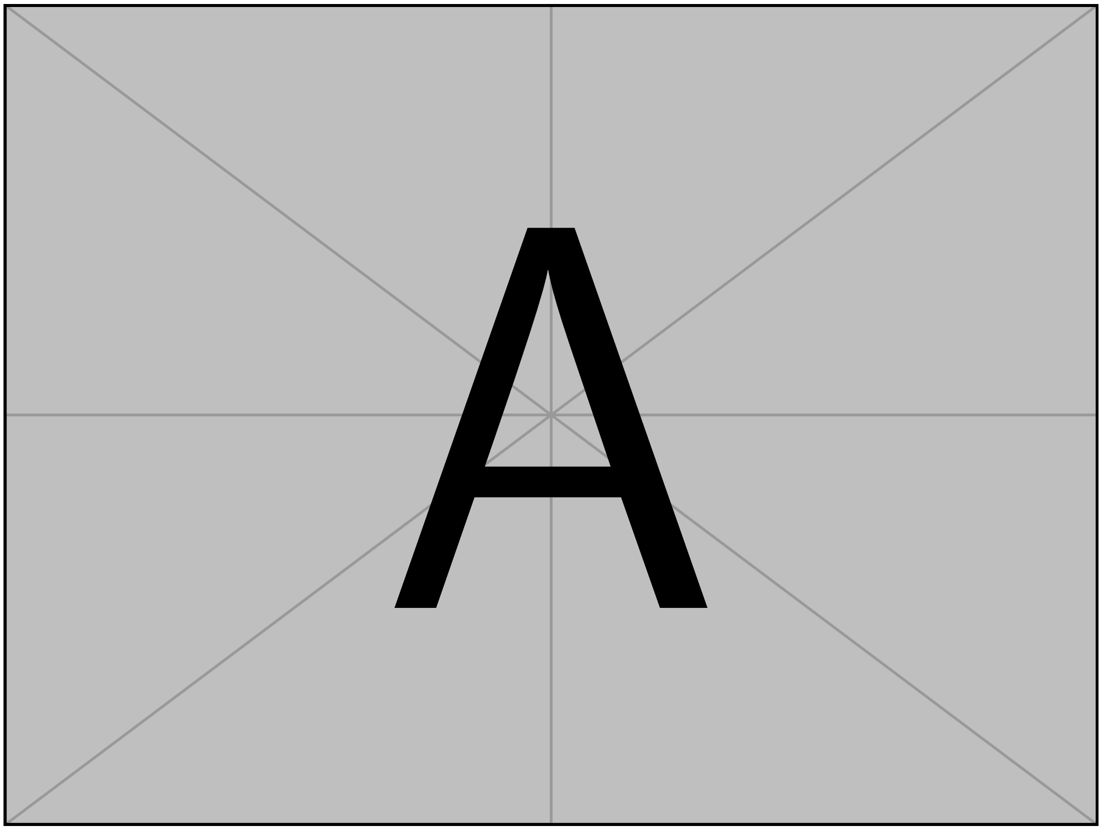

Figure 3.7: An example of the effect of differing the number of clusters

[1] 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.

3.4.2 Improving the previous approaches

Recall that the purpose of the region selection 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 periodically. 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. This reduces the problem to:

- Use k-means to identify the skin tone
- Use the skin tone to identify skin in the face for some 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 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. This is shown to improve on simply selecting the largest cluster in Section 4.2.1.

Identifying skin pixels using the skin tone

Identifying the skin tone, as described, requires the use of the k-means algorithm. However, as previously mentioned, this is not a particularly suitable algorithm for real-time applications. Thus, the skin tone can only be identified periodically rather than for every frame. So given the skin tone identified in some 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.

A possible approach could be to record the number of pixels that were in the skin cluster when the skin tone was identified. Then, our algorithm could select the n closest pixels in the colour space to the skin tone, where n was the number of pixels in the skin cluster. However, this would not be robust to rotations of the face, since, in subsequent frames, the number of pixels visible to the camera could change. Assuming a static value of n would not be robust.

Implicitly, the above solution also 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 will, clearly, 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, if the skin tone is detected frequently enough. The validity of this assumption is made clear by the evaluation of the algorithm presented in Section 4.2.1. Therefore, we proceed by classifying pixels based on their Euclidean distance from the skin tone.

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, as, 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 our prior distribution from the first approach, which indicates the likelihood of a particular colour being skin. Let us denote the following:

- C_{skin} as the classification of a pixel as skin
- 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 precisely the following:

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

Which, from Bayes' theorem, can be re-written as:

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

By neglecting constants across both classes, classification can proceed using $\Pr(s|C_{\text{skin}}, x_i)\Pr(C_{\text{skin}}|x_i)$. We proceed with classification as follows, for some threshold probability t that defines the decision boundary:

$$\text{isSkin}(x_i) = \begin{cases} \texttt{True}, & \text{if} \Pr(s|C_{\text{skin}}, x_i)\Pr(C_{\text{skin}}|x_i) > t \\ \texttt{False}, & \text{otherwise} \end{cases}$$

Defining the threshold The most obvious selection would be to classify the pixel based on which class has the maximum posterior probability. That is the threshold value t would be $\Pr(C_{\text{not-skin}}|x_i, s)$, where $C_{\text{not-skin}}$ is the class of pixels classified as not skin. Since this is a binary classification task, this is equivalent to $t = 1/2$.

Defining probability distributions Usually, one might attempt to learn these distributions from some relevant dataset. That is the prior distribution $\Pr(C_{\text{skin}}|x_i)$ and the distribution $\Pr(s|C_{\text{skin}}, x_i)$. The prior $\Pr(C_{\text{skin}}|x_i)$ was computed as the empirical distribution from the dataset discussed in Section 3.4.1. 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 such adequately sized datasets. Hence, attempting to learn the distribution $\Pr(s|C_{\text{skin}}, x_i)$ was not deemed to be feasible.

As a result, 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 requirements 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.

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

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 skin-tones, since there are minimum and maximum possible distances As a result, I use a *truncated* Normal distribution[2] instead.

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

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

We define the probability in proportion to the probability density function of the truncated distribution for each possible skin tone.

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

[2] 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.

3.5 Heart rate isolation

Given a time series of mean colours in the region of the frame considered, 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. Thereby this naive 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 aide with correctly identifying the pulse.

3.5.1 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 (i.e. the particular colour space in use), is 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.

The above formulation is not enough to isolate the signal $p(t)$. Thus assumptions must be placed on the nature of each $x_i(t)$ and $p(t)$ in order to turn this into a tractable problem.

Independent components 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 an other. This assumption encodes the notion that the pulse, should be entirely independent from the other phenomena impacting the observed signal.

For example, suppose that the some of the $x_i(t)$ are the result of some 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 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. Thus some additional work must be undertaken to identify the pulse from the returned signal.

In this scenario, 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. Hence, I assume that the correct heart rate corresponds to whichever has the largest power. This particular assumption is evaluated in Section 4.3.2.

Principal components analysis (PCA)

Alternatively, one might reframe the problem as finding a new orthogonal basis $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ to represent each vector $\mathbf{I}(t)$ in. The basis vectors are in the directions of maximal variance of the observed signal $\mathbf{I}(t)$. Finding these basis vectors is known as Principal Components Analysis.

In this scenario, we assume that the pulse occurs in the direction of most variance. Under the assumption that $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are in order from most to least variance, we return $p(t) = \mathbf{I}(t) \cdot \mathbf{v}_1$. Since we are only interested in the first basis vector, this is equivalent to fitting a plane to the three dimensional signal $\mathbf{I}(t)$

that minimises the squared error. The resulting pulse signal is the projection of $\mathbf{I}(t)$ onto this plane.

Conclusion The evaluation of these two algorithms reflects whether the assumption of statistical independence or maximal variance better reflects the issue of isolating the heart rate. The performance of these two approaches are evaluated in Section 4.3.1, from which it is concluded that ICA better serves the requirements of heart rate isolation. Hence, it is concluded that the assumption of statistical independence is more valid.

3.5.2 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. The heart rate could also be taken as the most prevalent frequency in the Fourier transform, that is, specifically, the frequency with the largest associated power. Since the Fourier transform provides a particularly rich representation from which to identify the heart rate, the latter approach is used. 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.

The split peak issue

In practice, however, 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 with smaller powers as is reported by van der Kooij et al. [4]. 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. [4].

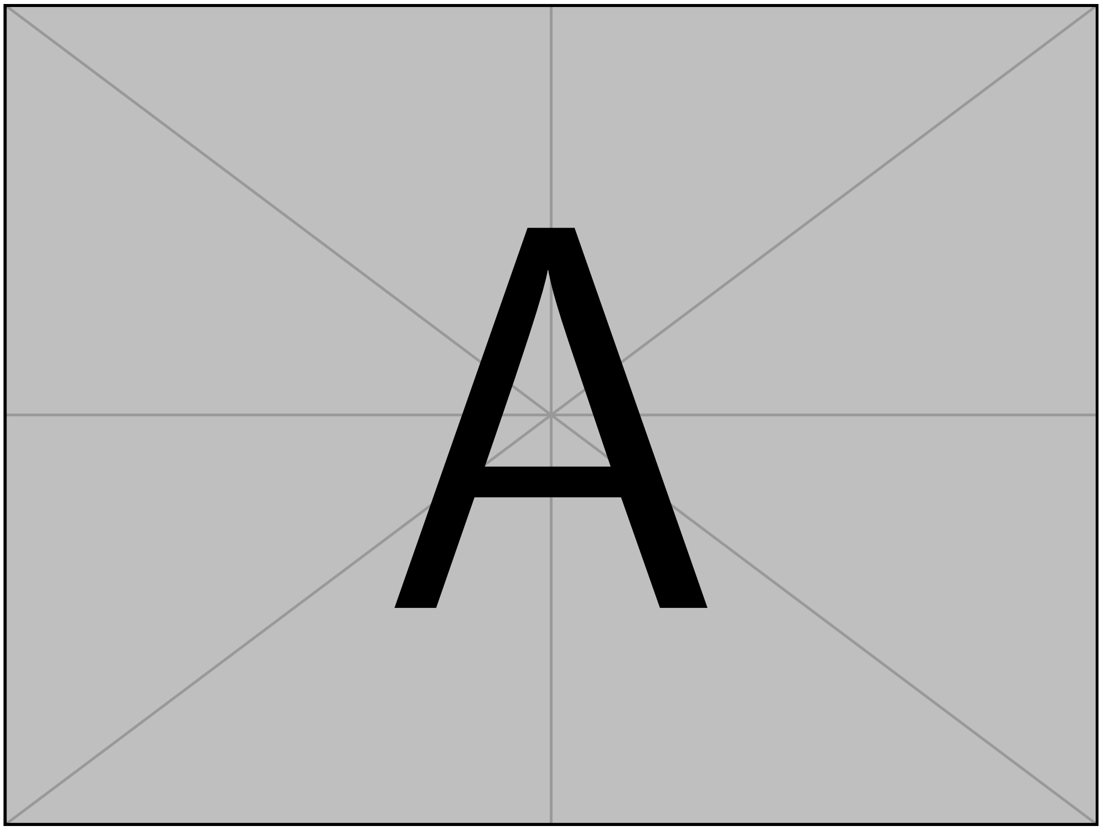

Figure 3.8: An example of the split peak issue with a Butterworth filter applied

3.6 Repository overview

Chapter 4

Evaluation

4.1 Face tracking

Face tracking was proposed in Section 3.3.1 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 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 `threshold` take?[1]

Dataset In order to better understand the performance of face tracking, a test set of several videos with varying amounts of movement was used. Some of the videos were taken from the MAHNOB dataset [2], which forms one of the key sources of data for the evaluation of this project. Initially, generated as a dataset

[1] As defined in the pseudocode in Section 3.3.1 the `threshold` value defines when the face tracker redetects the face

for study into affective computing, it consists of several thousand videos of participants as they are exposed to stimuli, with their ECG responses being recorded.

MAHNOB [2] only consists of videos where the participants are stationary and at a fixed distance from the camera, this alone, would not be adeqeuate for the testing of face tracking. Hence, it was augmented with several videos which I recorded with varying amounts and types of movement. I recorded a video with strictly 'translational' movement, that is, the size of the face of the user remains largely constant as they move from one side of the frame to the other. Another more challenging kind of movement, denoted here as 'rotational', is where the user rotates their face so that different parts of the face are in a view in different frames. Intuitively, one would expect this to be more difficult for the face tracker to deal with, since points will come in and out of view regularly.

Metrics Face detection is a binary classification task. As an algorithm, it defines a boundary within which all pixels are defined as belonging to 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.

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

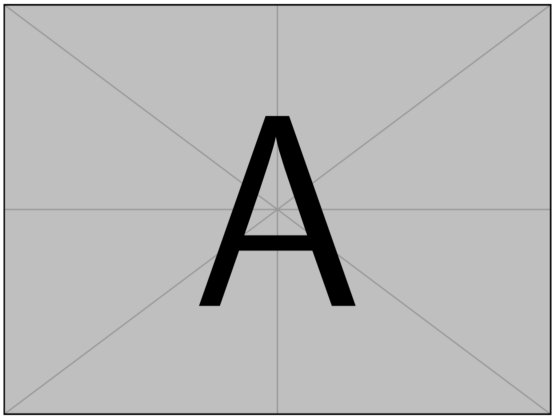

Figure 4.1: A labelled diagram outlining the four metrics defined above.

Computing the classification outcomes The result of each face detection is represented as a matrix of zeros and ones over the size of the entire frame. Pixels within the bounding box of the face have value one and pixels outside have value zero in this matrix. Using this representation, for a given frame, the output from the face detector, **D** and the output of the face tracker **T**, we compute the metrics as follows:

$$\begin{aligned}
\text{FP} &= H(\neg\mathbf{D} \wedge \mathbf{T}) \\
\text{FN} &= H(\mathbf{D} \wedge \neg\mathbf{T}) \\
\text{TP} &= H(\mathbf{D} \wedge \mathbf{T}) \\
\text{TN} &= H(\neg\mathbf{D} \wedge \neg\mathbf{T})
\end{aligned}$$

Under the following notation:

- The function $H(\mathbf{A})$ returns the number of ones present in the matrix $\mathbf{A}$
- $\wedge$ is an elementwise, bitwise AND
- $\vee$ is an elementwise bitwise OR operation
- $\neg$ is a NOT operator on each element in the matrix

The time to process each frame is also recorded as a means of measuring the relative performance of each algorithm.

Results

Accuracy

Motion resistance

Performance cost Recall from Section 3.3.1 that the following inequality was introduced that describes the performance of face tracking.

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

[2] The above inequality relates the rate of redetections (R/W) and the costs of several required algorithms. It was shown in Section 3.3.1 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 the exceed the value R/W.

Thresholds

Summary

4.2 Region selection

The problem of region selection does not, at an initial glance, lend itself to easy evaluation. It is unclear, through reasoning exclusively, whether increasingly complex algorithms for skin detection provide any benefit to the overall accuracy of the heart rate prediction. In Section 3.4 it was weakly hypothesised that minimising the number of non-skin pixels included in the mean colour would decrease the noise in the resulting signal. This was made under the further assumption that a less noisy signal would result in more accurate heart rate predictions. Underpinning these assumptions is the notion that any pixels in the background, that is within the bounding box of the face but not true skin pixels, will contain no predictive power regarding the heart rate of the user.

Crucially, however, it is unclear as to whether or not there exists some tradeoff between the number of pixels considered and their individual fidelities. It could be reasonably argued that by considering more pixels, the resulting

[2] 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

average becomes more resistant to noise at individual pixel level, say that caused by fluctuations in lighting, for example. It is not immediately clear whether considering a small set of guaranteed skin pixels is better than considering a large number of noisy pixels.

This evaluation, hence, is concerned with the validation of these assumptions.

4.2.1 Skin tone detection

Research questions

- Is considering a subset of pixels within the bounding box advantageous?
- Does skin detection reduce the noise of the resulting colour signal?
- Does a less noisy signal improve accuracy of heart rate predictions?

Metrics

Results

Summary

4.3 Heart rate isolation

4.3.1 Identifying the pulse signal

Research question

Metrics

Results

Summary

4.3.2 Extracting the heart rate from the pulse signal

Independent components analysis

Research question

Metrics

Results

Summary

Principal components analysis

Research question

Metrics

Results

Summary

4.3.3 Window and stride size

Research question

Metrics

Results

Summary

4.4 Evaluation of remote heart rate sensing

4.4.1 Evaluation method

Experimental setup

Ethics

4.4.2 Accuracy

Research questions

Metrics

Results

Summary

4.5 Summary

Chapter 5

Conclusion

Bibliography

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

[2] 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.

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

[4] 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.

Appendix A

Project Proposal

Computer Science Tripos - Part II - Project Proposal
Non-contact heart rate estimation from video

Yousuf Mohamed-Ahmed

Project Originator: Robert Harle
Project Supervisor: Robert Harle
Director of Studies: Timothy Jones and Graham Titmus
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 [1]. 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 [2], 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 [2]. 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 [3], hence simple band-pass filters will no longer work.
- End to End Deep Learning Approach: Recent papers [4] 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 [5]. 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.

References

[1] N. J. Verkruysse W, Svaasand LO, "Remote plethysmographic imaging using ambient light," Jul 2009.

[2] K. M. van der Kooij and M. Naber, "An open-source remote heart rate imaging method with practical apparatus and algorithms," *Behavior Research Methods*, May 2019.

[3] F. Peng, Z. Zhang, X. Gou, H. Liu, and W. Wang, "Motion artifact removal from photoplethysmographic signals by combining temporally constrained independent component analysis and adaptive filter," *BioMedical Engineering OnLine*, vol. 13, no. 1, p. 50, 2014.

[4] W. Chen and D. McDuff, "Deepphys: Video-based physiological measurement using convolutional attention networks," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), (Cham), pp. 356–373, Springer International Publishing, 2018.

[5] T.-M. Li, M. Gharbi, A. Adams, F. Durand, and J. Ragan-Kelley, "Differentiable programming for image processing and deep learning in halide," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 139, 2018.