Adaptive gain tuning for robust remote pulse rate monitoring under changing light conditions

Papageorgiou, A.

Award date:
2014

Technische Universiteit Eindhoven University of Technology

DEPARTMENT OF ELECTRICAL ENGINEER
VIDEO & IMAGE PROCESSING GROUP AT PHILIPS RESEARCH

Adaptive Gain Tuning for Robust Remote Pulse Rate Monitoring Under Changing Light Conditions

Master Thesis

Author: Andreas Papageorgiou
Email: < a.papageorgiou@student.tue.nl >
ID: 0869974

Supervisors:
Prof. Dr. Gerard de Haan (TU/e & Philips)
Dr. Sander Stuijk (TU/e)

Eindhoven, August 2014

Table of Contents

1. Introduction

Vital signs are measures of various physiological statistics such as heart rate, blood pressure and body temperature. They usually serve as an indication of someone's general health condition, yet their importance is much bigger in special cases where continuous monitoring of such parameters can significantly improve the quality of life and in extreme cases prove lifesaving. Such cases include premature infants whose vital signs, and especially their heart rate, can prove good indicators for detecting infections and identifying clinical conditions which may require surgical operation. Findings in [1] correlate the abnormal heart rate characteristics with the most common infections in baby infants while in [2] a novel risk marker for neonatal sepsis is proposed by evaluating the heart rate characteristics.

Other example cases where heart rate monitoring can improve quality of life, is in fitness and training, be it for professional sports athletes or individual hobbyists, heart rate is the primal indicator of training intensity[3] and the key to a successful and effective exercising program.

Photoplethysmogaphy (PPG) is an optical method to monitor vital signs and is described in [4]. The basic principle of operation is that a sensor detects the optical absorption variations of the human skin due to the blood volume variations during the cardiac cycle and this yields the pulse rate.

Traditionally PPG is performed using contact sensors such as pulse-oximeters. Yet in cases such as premature infant monitoring as discussed above, contact sensors have several disadvantages. Not only do they cause discomfort, stress and pain to the patient, but they can cause epidermal stripping as seen in [5 -6] and also have an impact on cognitive development [7]. For professional athletes and training enthusiasts, having a contact sensor attached constantly to the body is also not convenient.

Such needs have driven research to look for new ways to monitor vital signs in general and heart rate in particular. Results have shown that the variations of the human skin color detected by a pulse-oximeter can also be measured at a distance with a normal camera. This has has led to remote-PPG (rPPG)[8-9].

However, rPPG methods that detect minute optical absorption variations of the human skin caused by blood volume changes during the cardiac cycle are sensitive to varying ambient light conditions as well as body motion. Such conditions can occur when a cloud moves in front of the sun, when the sun sets, when a lamp is suddenly switched on or off. A recent pilot study in [10] addresses such challenges of rPPG in the neonatal intensive care unit and concludes that low ambient light level, as well as motion prevents successful measurement of heart rate from time to time.

Our goal in this paper is twofold. Initially we create a model to understand how ambient light level affects the quality of the extracted pulse rate signal. With the help of this model we confirm our hypothesis that the signal-to-noise-ratio (SNR) indeed drops under certain ambient light conditions. We try to improve the performance of the rPPG algorithm and mitigate the negative effect of varying ambient light conditions by controlling dynamically the camera's gain levels. Adapting the camera gains can minimize clipping which occurs due to very light or very dark ambient light conditions and prevents

from extracting a clear signal. Amplifying the gain levels also helps to better reveal the pulsatile component in each color channel which is caused by the heart rate and which is barely noticeable due to its very low amplitude compared to the skin color. Finally we present a control mechanism to adjust the gain levels of the camera in order to compensate for bad lighting conditions. The proposed controller is implemented in a real time control algorithm that optimizes the gain levels dynamically in accordance to the ambient light conditions.

The contributions are that (1) we create a model in Matlab which allows for experimentation and validation of different concepts and ideas with pulse extraction algorithms, (2) we introduce a novel strategy for dynamically adapting the cameras gain levels to compensate for the negative effect of changing ambient light conditions on the extracted PPG -signal, (3) we implement the proposed algorithm and integrate it with the existing state of the art chrominance based rPPG method proposed in [11].

The rest of this report is organized as follows. In Chapter 2, provide an overview of the related work is provided. In Chapter 3, the theory behind our problem is analyzed and the theoretical framework in Matlab that models it and proves our hypothesis and strategy in detail is presented. In Chapter 4 the novel algorithm to dynamically tune the gain levels is presented and the implementation and integration details with existing state of the art rPPG algorithm are described. In Chapter 5 the experiments and results that were made to measure the performance of our algorithm are described. Finally in Chapter 6 conclusions and future work are discussed.

2. Related Work

The common ground for all rPPG techniques is the fact that the variations in optical absorption of the human skin depend on the wavelength used. In 2008, Huelsbusch and Verkruijsse found that under ambient light conditions, the PPG signal has different relative strength in the three color channels, RGB, of a camera that was used to capture the image[12]. Poh et al. proposed a linear combination of the RGB channels to derive three independent signals with independent component analysis (ICA) using non-Gaussianity as the criterion for independence[13]. Lewandowska et al. used Principal Component analysis (PCA) to define the three independent linear combinations of the RGB channels and more recently de Haan et al. suggested a chrominance-based rPPG method (CHROM) in[11], which assumes a standardized skin-ton to derive the pulse as a linear combination of the three color channels. All of the above methods for pulse rate extraction are sensitive to certain ambient light conditions, be it for very dark or a very bright environments. We will use the CHROM method in order to investigate the potential of dynamic gain tuning to optimize the pulse signal in changing light environments. This state-of-art method has been proven to have a high accuracy amongst other existing rPPG methods and we will integrate our algorithm together with this one, in order to extend its functionality and robustness.

3. Theoretical Model

Our assumption is that dynamically changing light conditions can corrupt the extracted PPG signal. In order to verify that, understand how exactly the CHROM algorithm is influenced by it and see whether changing the gain levels can mitigate this problem we construct a theoretical model in Matlab which enables us to make various experiments easily. Using this framework we will ultimately define the strategy to be implemented in the real time control algorithm for dynamically adapting the gains. For our theoretical model we need to define and simulate:

- The way that the human skin color varies in time, due to the heartbeat.
- The way that a camera records frames by defining a frame rate and the random noise that is inserted into every captured frame.
- A lighting model to be able to 1) create the effect of shadows on a skin face due to its morphology and 2) simulate different lighting conditions,
- The CHROM algorithm[1]
- A metric to measure the quality of the extracted signal:

3.1 Pulse Model

The pulse rate (PPG signal) consists of two components, usually referred to as the DC and the AC component. The DC component refers to the constant offset of the signal and indicates its amplitude. This component is relative to the absorption of the light on each color channel. The AC component is the sinusoidal fraction that is added on top of the DC component. It relates to the variations in the light absorption on each color channel, caused by the fluctuations of the blood volume during the cardiac cycle. Figure 3-1 illustrates the above

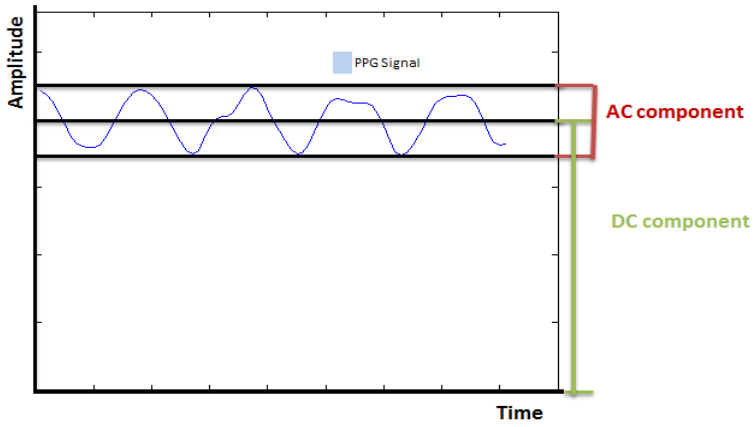

Figure 3-1. DC and AC component of a signal

[1] The CHROM method is described in [11] and its implementation is presented in detail in chapter 4.

In order to model the AC component we construct a 3xN matrix. Each row corresponds to a different color channel (R,G,B) and each column corresponds to a different frame in the temporal domain that would be captured by the video camera. The matrix is constructed out of the following functions:

$$\boldsymbol{Pulse_R = Coef_R * A \sin(2\pi f t)}$$
$$\boldsymbol{Pulse_G = Coef_G * A \sin(2\pi f t)}$$
$$\boldsymbol{Pulse_B = Coef_B * A \sin(2\pi f t)}$$

- The coefficients $Coef_{channel}$ are the normalized relative pulsatile fractions in each color channel. Experiments conducted in [11], have shown that they have the following values**:** $[Coef_R, Coef_G, Coef_B = [0.34, 0.77, 0.54]$. **I**n essence this shows that variations in the skin color caused by the heart rate are mostly seen in the green channel, and are less observable in the red and blue channels.
- The energy factor of the pulse A allows us to adjust how strong the pulse signal should be. For our model we set the energy factor to A=1.
- The frequency f of the pulse should correspond to a normal human heart rate. Typically a healthy adult has a heart rate in the range 60-80 BPM (Beats per Minute) [18]. We choose a frequency of 66 Beats Per Minute (BPM). To model that we set $f = \frac{1.1}{20}$. The reason behind this choice is that our t models discrete frames and we want to model a sampling rate of 20 fps (frames per second). Hence $ft = 1.1\ beat\ per\ sec * 60 \sec\ per\ min = 66\ BPM$.

3.2 Skin Model

In order to model the DC component we construct a rectangular matrix of $Width\ x\ Height\ x\ 3$ dimensions. Each element holds three values, one for each color channel. The initial choices for the AC values, that are identical across the whole matrix and which reflect the skin color, are defined as $[Skin_R, Skin_G, Skin_B] = [100,67,50]$. This choice is not random, as the CHROM method assumes a standardized skin tone with the following ratios $[R, \frac{2R}{3}, \frac{R}{2}]$. Choosing a skin tone that is close to these ratios will help our model to behave better.

At this point we assume for our model that the initial skin color as well as the variations to it, caused by the pulse, will happen exactly at the same time across the whole skin, and hence across all pixels.

3.3 Noise Model

To model the noise that occurs when sampling frames with a video camera we create an RGB noise image for each frame. The total noise's dimensions are $Width\ x\ Height\ x\ 3\ x\ N$. The third dimension corresponds to the random noise value of the specific color channel while N is the number of total frames sampled in the temporal domain. We draw random values which follow a normal distribution:

$$X \sim N(\mu = 0, \sigma^2 = \sqrt{10}).$$

The reason for the choice of these values is explained in Appendix A.2. By keeping the noise identical across all three channels, we model the simple case where motion affects each channel's noise equally. This assumption is true when there is a single white light source in an all-black or white environment (except from the face). In the case where a colored object is present, it acts as a secondary colored light source, (reflecting light which is not white anymore). The recorded skin region would reflect this color more or less depending on its relative distance from it. Consequently motion would change the distribution of the noise unevenly on each channel.

$$\boldsymbol{Noise_R = Noise_G = Noise_B = GaussRandom(\mu, \sigma)}$$

Having a different noise component on every channel would model more complex scenarios but this simple model is considered a good start to devise a general strategy for adapting gains dynamically.

3.4 Lighting Model

In order to model the different brightness levels observed on a human face, due to the position of the light source and the shadows from human features we create a gradient mask with the same size as our skin matrix. The mask parameters can be selected dynamically to model different light conditions and shadows. To create a mask we generate N linearly equally spaced points between Min and Max and repeat these values to create a $Width\ x\ Height$ matrix that has the same dimensions as our skin model:

$$\boldsymbol{Mask(Min, Max, N)} =$$

$$[\, Min, Min + \frac{1 * (Max - Min)}{N - 1}, Min + \frac{2 * (Max - Min)}{N - 1}, \dots,$$

$$\dots, Min + \frac{(N - 2) * (Max - Min)}{N - 1}, Max]$$

Figure 3-2 shows an example of a horizontal mask:

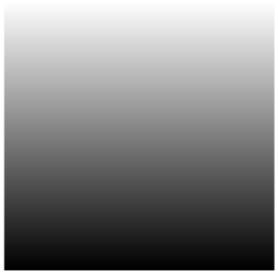

Figure 3-2. A gradient mask simulating the shadow effect.

3.5 Gain Model

In CCD imaging, the gain refers to the magnitude of amplification that a given system will produce. In essence it represents the conversion factor between electrons (e^-) recorded by the CCD sensor and the number of digital counts, or Analog-Digital Units (ADUs). It is expressed as the number of electrons that get converted into a digital number, or electrons per ADU (e^-/ADU). I.e. a gain of 1.8 e^-/count means that the camera will produce 1 count for every 1.8 recorded electrons[2]

In order to model the different analog gain levels of our camera we set up an experiment which is described in detail in Appendix A.1. This experiment allows us to derive the characteristic function of the gain of our camera[3] for the 100 different level settings. Figure 3-3 visualizes this characteristic function. We can see that the gain function is not perfectly linear.

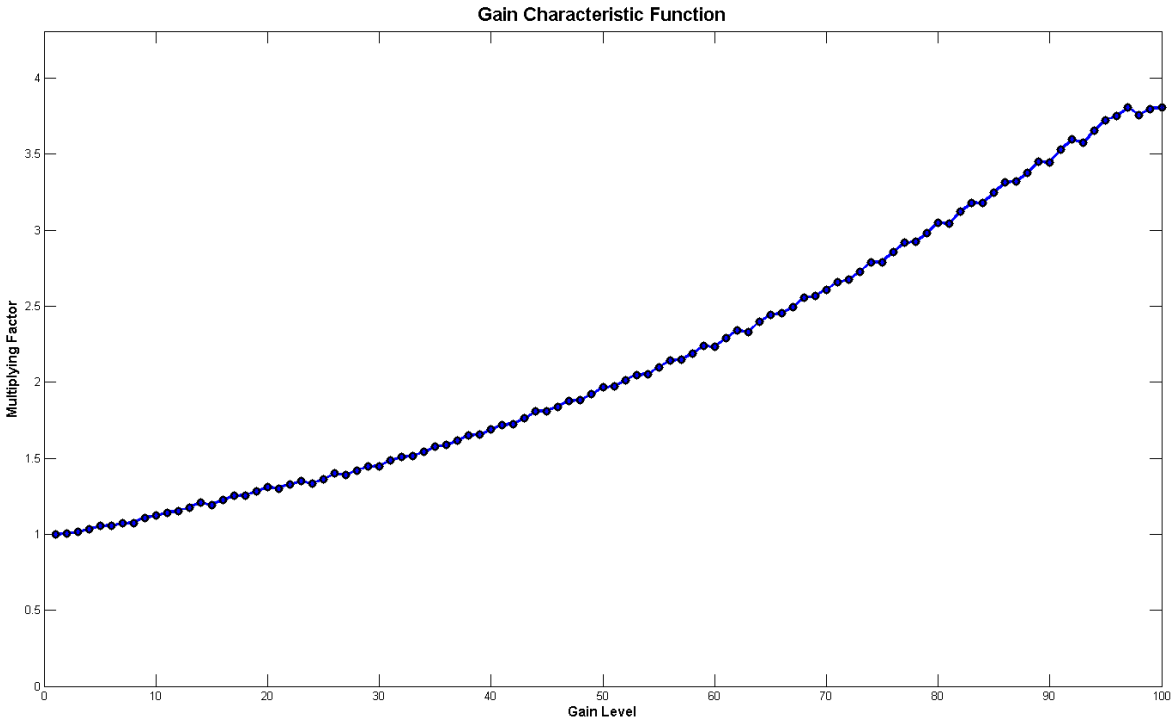

Figure 3-3. Gain Characteristic Function

[2] Naturally since electrons cannot be split, what this means is that 4/5 of the time 1 count will be produced from 2 electrons and 1/5 of the time 1 count will be produced from 1 electron.
[3] Our camera model is UI-222x/UI-622x

3.6 Complete Model – Putting it all together

Having modeled all the above factors we construct our final skin matrix as follows:

$$Image_{ch} = uint8\big(GainFactor_{ch} * (Pulse\_F_{ch} + Noise_{ch})\big),$$

$$where \quad Pulse\_F_{ch} = Mask * (Skin_{\ ch} + Skin_{ch} * Pulse_{ch})$$

- The uint8 conversion models our camera's quantization. Each pixel value in each color channel uses 8 bits for quantization meaning that each pixel can only have integer values in the range of [0-255].

- It is important to understand why we do not simply add our pulse component to the original skin color but instead multiply it and then add it. When the blood volume in the veins changes due to heartbeat, the color of the reflected light from the human skin will also change accordingly. However in total absence of ambient light we cannot observe any variations and extract a PPG signal since no light is reflected. As ambient light intensity increases, these tiny variations in the skin color will be amplified and become more easily observable as the reflected light will also be increased. This will result in a clearer PPG signal and it is the reason why the pulse component is multiplicative and not additive. The skin tone itself acts as a filter with darker skin tones reflecting less light and thus requiring more ambient light to produce a clear PPG signal. Figure 3-4 depicts the above.

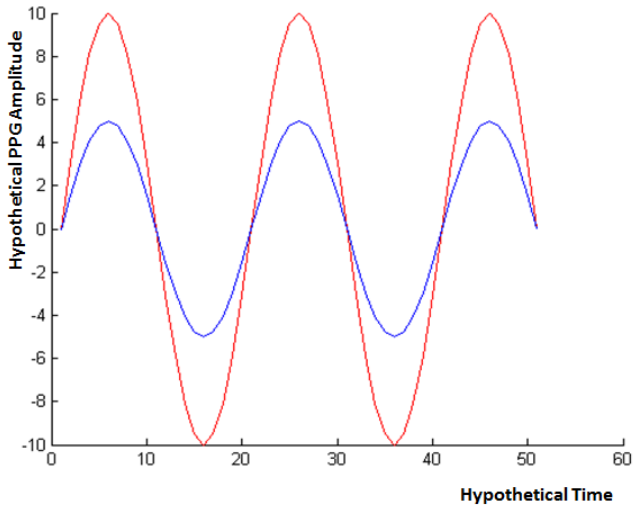

Figure 3-4. Hypothetical relative pulsatile strength of the PPG signal for two different skin tones, or two different ambient light recording scenarios. The blue line would correspond to a darker skin tone, or a dark ambient light during a recording. The red line would correspond to a lighter skin tone, or a brighter ambient light during a recording.

The variations in every skin pixel caused by the heart rate are tiny, since the AC component of the PPG signal is relatively much smaller than the DC component. For the average pink skin, experiments from

the authors of [11] have shown that the change in the signal's amplitude is in the region of ($0.001 * [\ \pm 0.34 * \boldsymbol{DC_R}, \pm 0.77 * \boldsymbol{DC_G}, \pm 0.54 * \boldsymbol{DC_B}\ ]$). In fact if it weren't for noise we would not be able to observe these variations. Consequently our strategy should be to maximize the gain in each channel, so that we can observe the AC component after the quantization and extract a more robust PPG Signal. This is explained in Figure 3-5 below:

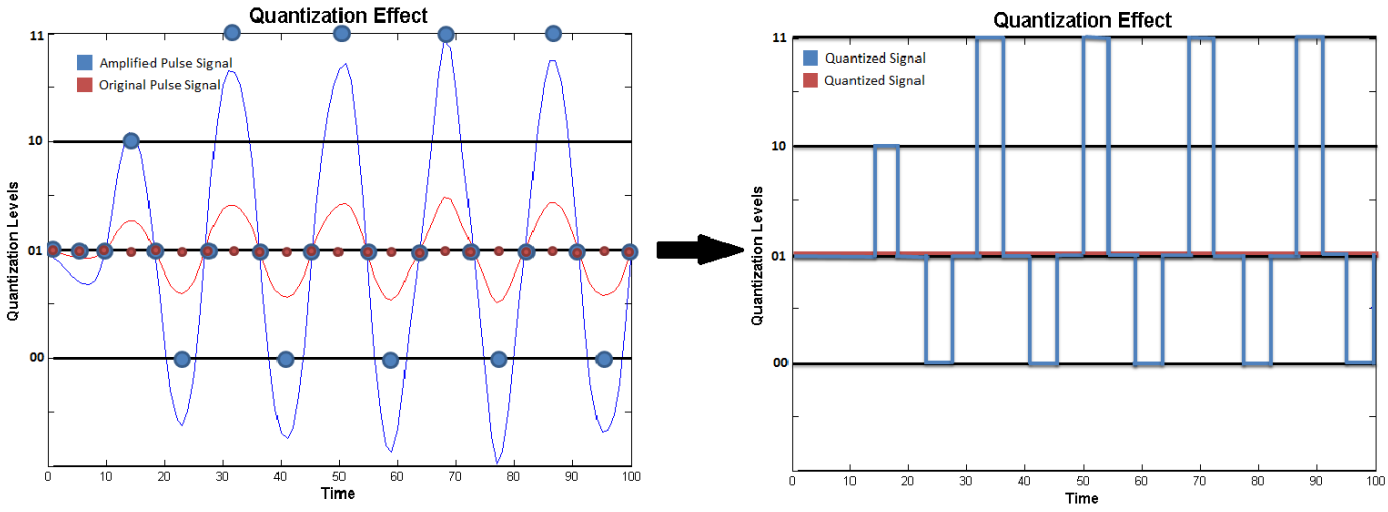

Figure 3-5. Quantization effect and the results of signal amplification.

In figure 3-5 above, the original analog pulse signal is shown in red, whereas the amplified one in blue. In this scenario we do not introduce any noise. Doing so would make the quantized PPG-signal still visible as noise's amplitude would make it exceed the quantization levels. In this hypothetical scenario, if we were to use only four quantization levels, then we would not be able to observe any pulsatility without amplification of the pulse signal. In our case we have eight bits and thus 255 quantization levels for each color channel but the concept is the same. Boosting the gains would result in a more accurate signal.

However, maxing out the gain in each channel will make the pixels of the captured frame to clip, depending on the ambient lighting conditions, when capturing a frame. Clipping is the result of capturing or processing an image where the intensity in the pixels falls outside the minimum and maximum intensity that can be represented. Since our camera uses 8-bits for quantization, we can only represent values in the range of [0-255]. Throughout this report we will refer to ***white clipping*** and ***black clipping percentage*** as the amount of pixel values that fall out of the maximum and minimum intensity that can be represented respectively.

Our hypothesis is that clipping is catastrophic for the quality of the PPG signal, as it would render pixels useless for feeding into the CHROM algorithm to calculate the PPG signal. Clipping can occur from very dark or very bright ambient light as well as very high or low gain settings. Thus we can compensate for clipping caused by the external brightness conditions using the camera gains. To prove our hypothesis with our model we first define a metric for the quality of our PPG signal, and then design an experiment to see the relation between clipping and the quality of the signal.

3.7 Quality Metric

For our experiments we run our simulation for 2400 frames that correspond to 2 minutes of recording with a frame rate of 20 fps. After applying the *CHROM* algorithm to extract our PPG signal S, we slide a window of 512 points on S with a step of 20 frames and compute the Fourier Transform of it.
I.e. we compute:

$Spec_1 = Fourier(S(1{:}\,512))$,
$Spec_2 = Fourier(S(21{:}\,532))$,
…
$Spec_{94} = Fourier(S(1881{:}\,2393))$

For each spectrum we calculate its magnitude
$M_i = Abs(\,Fourier(Spec_i, 1200)\,)$ and remove the frequencies which are beyond 300 BPM[4]. We scale our frequency range to 1200 since we have a frame rate of 20 fps and we want to calculate the pulse in BPM.

The frequency with the highest magnitude is the **dominant frequency** in our pulse signal S across the past 512 frames. This corresponds to **the average extracted heart rate** in the past 25.6 seconds.

From the magnitude of the spectrum M we calculate the signal-to-noise ratio (SNR) in the following way:

$$\mathrm{SNR_i} = 10 * \log_{10}\left(\frac{\sum_{f=i}^{i+512} U(f) * M(f)^2}{\sum_{f=i}^{i+512} |(1 - U(f))| * M(f)^2}\right),$$

$$U(f) = \begin{cases} 1, & hr * (fr - bin) < f < hr * (fr + bin,) \\ 0, & else \end{cases},$$

$1 \le hr \le 2$, number of harmonics

$$bin = 3$$

$$fr = Position\left(Max\left(Abs\left(Fourier\left(S(N{:}\,N+512)\right)\right)\right)\right), the\ dominant\ freq.$$

This is the ratio between the energy around the fundamental frequency plus the energy around the first harmonic and the energy contained in the rest of the spectrum. The SNR is also explained in the Figure 3-6.

[4] No living human being could ever have a heart rate above 300 BPM

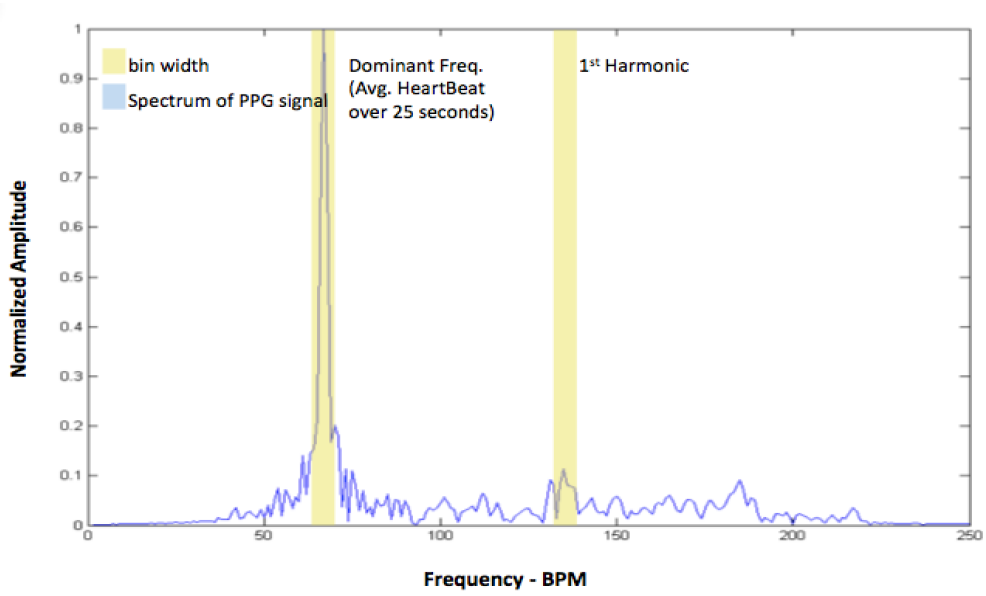

Figure 3-6. The SNR is calculated as the energy ratio of the spectral segments inside and outside the template windows.

After having calculated the SNR for each spectrum, we measure the mean SNR over all the spectrums as our final quality metric. The higher this value is, the better our signal is.

$$\text{meanSNR} = \text{mean}(\text{SNR}_\text{i}), \qquad i = 1,21,41, \dots,1881$$

3.8 SNR and Clipping

To see how clipping affects the SNR and prove our hypothesis, we run simulations of our model for 2 minutes. We start with a gain setting of 0 in one color channel, record the mean SNR and then repeat this process after increasing the gain in one color channel with a step of 1. We continue this process for all 100 gain levels in one color channel.

To measure the white and black clipping percentage, we count the number of pixels that have a value of 255 or 0 across all three color channels and divide them with the total number of pixels across all channels:

$$white\_clip\_percentage = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} val(i,j) = 255}{TotalNumberofPixels} * 100\%$$

$$black\_clip\_percentage = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} val(i,j) = 0}{TotalNumberofPixels} * 100\%$$

Figure 3-7 shows the results for increasing the red gain from 0 to 100, while keeping the green and blue to 0. We plot the mean SNR for each experiment against the white clipping percentage. The x axis corresponds to [0 - 100%] clipping, while the y axis corresponds to the mean SNR value. It is clear that once clipping occurs, the quality of the signal drops significantly.

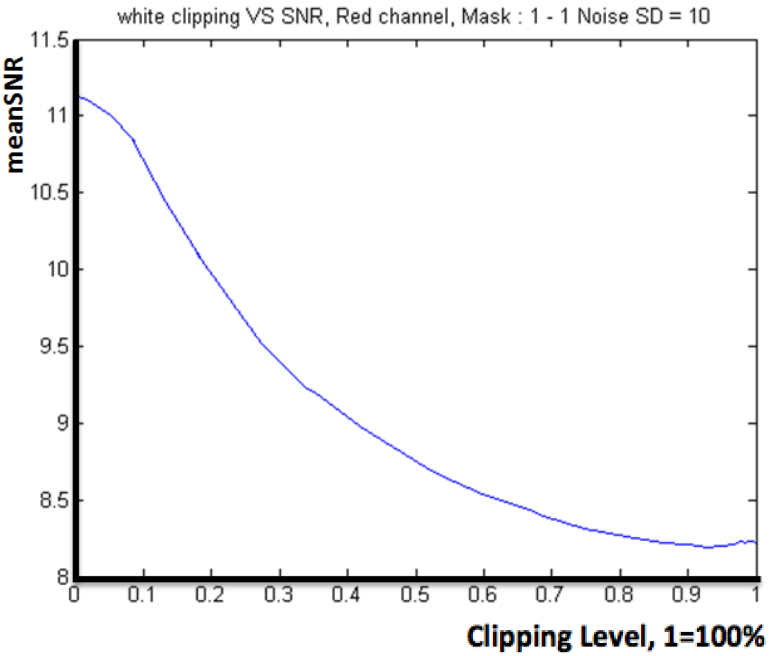

Figure 3-7. SNR vs. Clipping in the Red Channel

We repeat the same process for the green and blue gain respectively and get the following results:

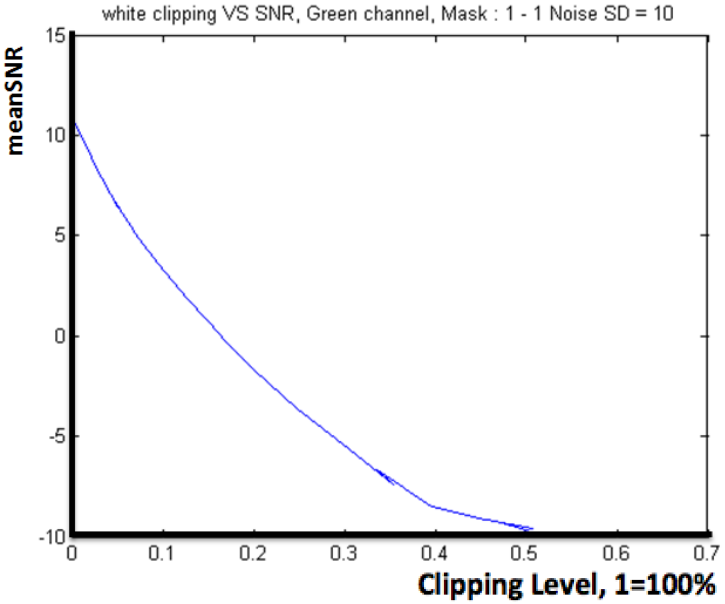

Figure 3-8. SNR vs. Clipping in the Green Channel

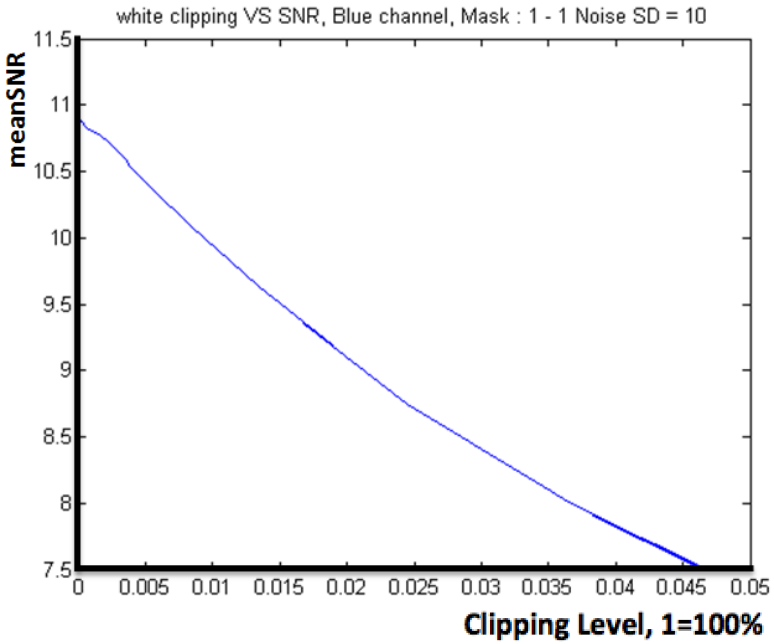

Figure 3-9. SNR vs. Clipping in the Blue Channel

Figure 3-8 shows the results for increasing the green gain from 0 to 100, while Figure 3-9 shows the results for increasing the blue gain. We observe from figures 3-8 and 3-9 that the blue and green channels are much more sensitive to clipping compared to the red channel. This is expected, as the green channel contains the most information for the extracted PPG signal. The color variations in that channel are stronger than in the other two. This relates directly to the relative pulsatile fractions in each channel.

We repeat the same process for different standard deviation (σ) values of the random noise. Figure 3-10 and 3-11 show the results that we get for σ = 30 and σ = 40 respectively. We observe that for higher standard deviation values, the SNR gets even more sensitive to clipping.

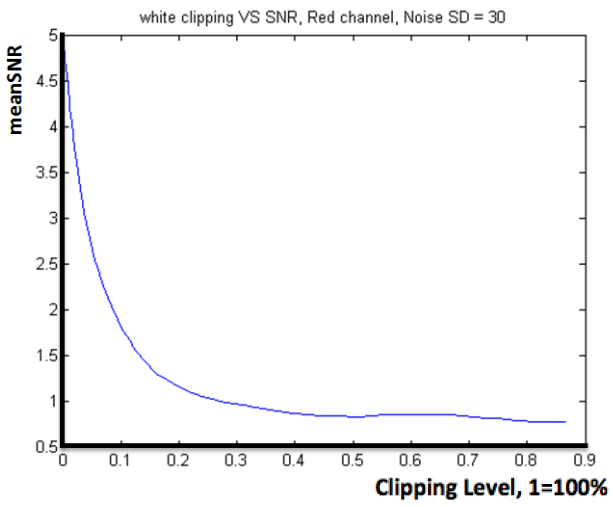

Figure 3-10. SNR vs. Clipping, Noise σ=20.

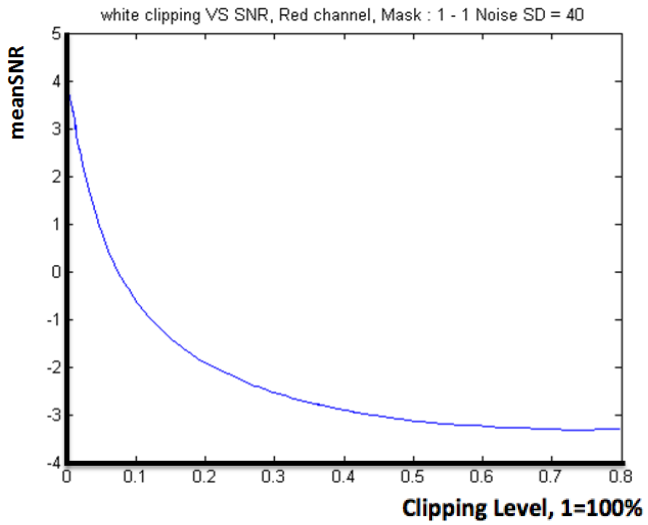

Figure 3-11. SNR vs. Clipping, Noise σ=40.

3.9 Strategic Conclusion

It is now obvious from this experiment that clipping is catastrophic for our PPG signal and it should be prevented as much as possible. At the same time the gains need to be tuned as high as possible, in order to amplify the observable variations in the skin color caused by the heartbeat, and compensate for the quantization effect of the camera. So, the strategy for our adaptive gain-tuning algorithm is now clear: Keep the gain levels, as high as possible, but at the same minimize the clipping of the skin pixels. In the next chapter we will elaborate more on this strategy, present the algorithm and see how it can be implemented alongside the existing CHROM algorithm.

4. Method and Implementation

De Haan et al proposed in [11] a method for extracting the PPG signal out of a linear combination of the three color channels assuming a standardized skin-tone (CHROM). Our proposed algorithm for adaptive gain tuning (AGT) works in combination with the CHROM method and thus we need to implement both algorithms under a common framework and explain how the choices in CHROM algorithm affect the choices in our gain-tuning algorithm. In this chapter we shall elaborate on these two algorithms and reason on the design choices of the AGT algorithm.

In order to be able to adjust the analog gains we also needed access to the drivers of our camera, which are provided by its manufacturer and written in C++. At the same time we needed access to the OpenCV libraries that empowered us with fast image processing libraries. To combine all the above we re-implemented the CHROM algorithm in C++. The original work for implementing in Java was provided by the author of [17] Wenjin Wang.

4.1 CHROM Implementation without Automatic Gain Tuning (AGT-OFF)

Figure 4-1, shows the steps for implementing the CHROM method.

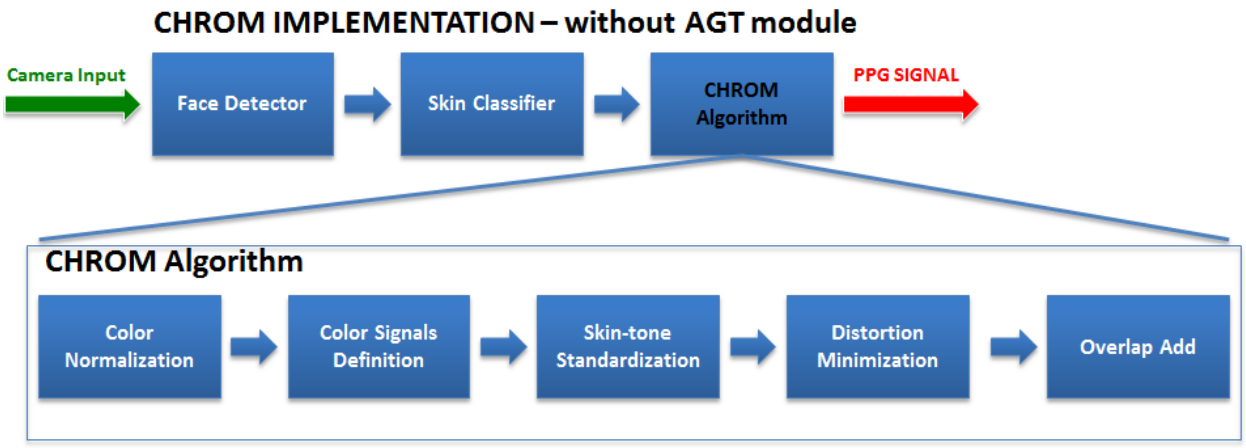

Figure 4-1. The basic modules for implementing the CHROM algorithm.

Face Detection

The CHROM algorithm works on skin pixels to extract the PPG signal. In order to detect the skin pixels of a subject which is in front of a video camera, we use a live face detection algorithm implemented in OpenCV. The output is a rectangular region of interest (ROI), which we then feed into the skin classifier. The face detector is a Haar Feature-based cascade classifier, relying on the method proposed by Paul Viola and Michael Jones in [14]. OpenCV already contains pre-trained classifiers for face.

Skin Classifier

Out of the face pixels extracted from the face detector, we need to separate the ones that are skin and can reflect the pulse rate from the rest. This means that we need to exclude the eyes, mouth or teeth, eye-brows, hair, beard and anything else that might not be a skin pixel in the face ROI. Most skin

segmentation methods use pre-defined thresholds of skin color composition or model a binary boundary between foreground and background. However, such an approach requires choosing appropriate thresholds and defining correctly the foreground/background. Assuming that most of the pixels inside a face ROI are indeed skin, we build a clustered feature-space, which detects the pixels that are further away from the cluster center as non-skin pixels. The One Class Support Vector Machine [15] is employed to estimate such a hyper-plane which encircles most of the pixel samples as a single class (skin class) without any prior skin color information. Inspired by [16] where the intensity-normalized rgb and YCrCb are used to discriminate skin and non-skin regions we train the OC-SVM by using 4 feature descriptors for each pixel, namely $r-g, r-b, Y-Cr, Y-Cb$. The reason for using the luma component Y is that we want our skin-classifier to be invariant under changing brightness conditions. The skin classifier is trained only once in the beginning of the CHROM algorithm with a frame which depicts clearly the subject's face. Experiments done in [17] show the performance of the skin classifier under different skin-types. Skin classification is an extremely important part of our algorithm and we shall further elaborate on it later in this chapter.

CHROM Algorithm

The chrominance algorithm consists of the following steps:

- **Color Normalization**
  In order to produce a pulse signal that is independent of the stationary color of the light source, as well as its brightness levels we normalize each color channel by dividing its samples with their mean over a temporal interval. This color normalization makes the algorithm robust under different color and intensity variations of the ambient light and enables us to independently fine-tune and optimize each gain level. Regarding the length of the temporal interval, we must guarantee that it contains at least one pulse period.
- **Color Signals Definition**
  The color signals X and Y are defined by the following equations:
  $$X = R - G$$
  $$Y = 0.5R + 0.5G - B$$
  The reason for using color difference, i.e. chrominance, signals as explained in [11] is that by doing so, the light component that is directly reflected from the surface of the skin can be eliminated. To eliminate motion which affects chrominance signals identically we take a ratio of the two.
- **Skin-tone Standardization**
  In order to enable correct functionality under different colored light sources, the authors of [11] did a large experiment and found out that for any skin type, normalizing the color channels with the fixed factors $[R, G, B] = [0.7682, 0.5121, 0.3841]$ yields good results.
- **Distortion Minimization**
  Since the human heart rate ranges from 40 – 240 beats per minute (BPM), we can safely remove the distortions out of this range with the use of a band-pass filter.
- **Overlap Add**
  The overlap-add procedure, explained in detail in [11], makes the algorithm more adaptive to

quick changes in the Heart Rate which can occur under different-intensity exercising intervals. It improves the PPG signal by separately optimizing partially overlapping time intervals. The resulting pieces are stitched together in an overlap-add fashion which is accommodated with the use of a Hann window. In practice, how it works is that after normalizing and minimizing the signal, we multiply it with a hanning window of a specified size **Hann_size = 32**. Then we move on the time axis with steps of size **Step_size = Hann_size/4** and repeat this procedure. To construct the final signal the partially overlapping signals are added as illustrated in Figure 4-2.

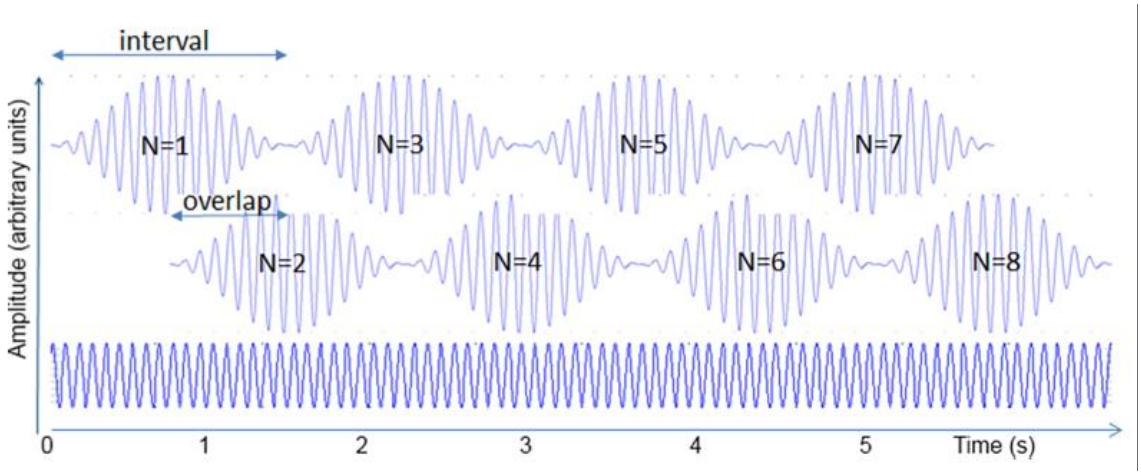

Figure 4-2. Overlap add procedure as illustrated in [11]

4.2 CHROM Implementation with Automatic Gain Tuning (AGT-ON)

We designed a real time control algorithm that is implemented alongside the CHROM method in order to maximize at any given time the SNR and dynamically compensate for very low or very bright ambient light conditions. As explained in Chapter 3, the key idea behind it is to maximize the gain levels so that we can amplify the relatively tiny pulsatility observed in the three color channels that corresponds to the heart beat, while at the same time prevent clipping. Clipping is catastrophic for the SNR of the extracted PPG signal since it wipes outs entirely any observable information about the heart rate contained in the clipped pixels. Essentially, the algorithm tries to maximize the total amount of skin pixels that are present for the extraction of the PPG signal. Figure 4-3 shows the main components of the adaptive gain-tuning algorithm together with the mandatory implementation steps that are required for it to operate in combination with the CHROM method.

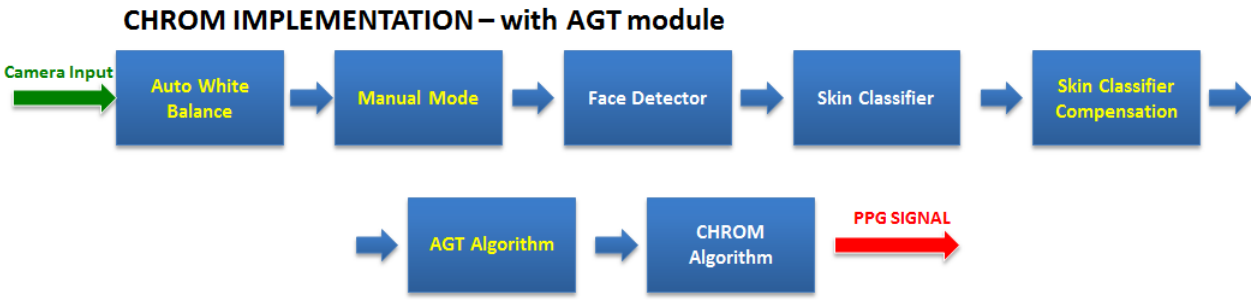

Figure 4-3. The basic modules for implementing the CHROM algorithm together with the Automatic Gain Tuning algorithm (AGT).

AWB

Initially, we begin by having the camera's built in auto white balance function set the analog gain levels individually, so that objects that appear white in person are rendered white in the captured frames, thereby removing unrealistic color casts. This does not affect directly the CHROM algorithm which is independent of the color and brightness level of the light source (*see color normalization above*), but it provides a good starting point for our dynamic gain adjustment algorithm, since the AWB would remove any significant clipping on a particular channel caused by a non-white ambient light. A weakness of the AWB algorithm is that it works solely on the complete captured frame, whereas we are interested in maintaining 0% clipping in the skin pixels of the detected face.

Manual Mode ON

After initially letting the AWB determine, sub-optimally, the gain levels, we deactivate the camera's automatic built-in functions for gain, white balance, color correction and frame rate so that we can individually set the gain parameters. The color correction matrix needs to be turned off because it tries to enhance the rendering of colors by applying a digital correction based on a color matrix which is adjusted individually for each color sensor. This results in an alteration of the true pixel values on each color channel when adjusting the gain levels which would introduce an error in calculating the number of clipped pixels. We found that the auto color correction could also affect the CHROM algorithm due to these minor changes of the pixel values and it should be turned off anyway. The frame rate is set to 15 FPS. This number is important for scaling and calculating the heart rate when going to the frequency domain to extract the dominant frequency out of the PPG signal and converting it in BPM. Also we note that using a higher frame rate of 20 fps as we did in our Matlab model was not possible because the processing time of the algorithm could not accommodate it.[5] After switching to manual mode we train the skin classifier on a single frame and enter the main part of our adaptive gain-tuning (AGT) algorithm.

[5] To achieve a higher frame rate and a better accuracy we suggest using a faster computer to run our algorithm, as well as using a dedicated GPU accelerator if present, together with the corresponding libraries from OpenCV

AGT algorithm

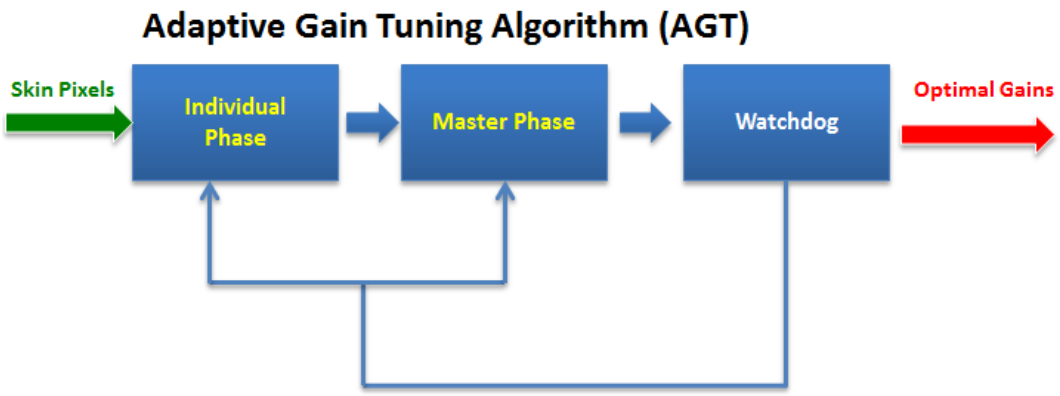

Figure 4-4. Adaptive Gain Tuning algorithm (AGT) modules.

Figure 4-4 shows the basic structure of the AGT algorithm. After every captured frame the AGT component is triggered. As an input it gets the detected skin pixels from the skin classifier and every time outputs the optimal gain settings. The adaptive gain tuning algorithm consists of two main phases, the ***Individual phase*** and the ***Master Phase***, as well as a ***watchdog component*** to monitor when to jump from the Master phase to the Individual phase. We initially start in the individual phase where each channel's gain can be adjusted separately from the others. Then the algorithm moves to the master phase where gains are adjusted through the master gain setting. This means that all three gains for the Red, Green and Blue channel are adjusted as a group, i.e. towards the same direction (increase / decrease) and for the same level (+2, -3, +8,...) as seen in Figure 4-5. Unless the watchdog triggers a change to move back to the individual phase, the algorithm will always stay in the master phase. The role of the watchdog is further discussed in a separate paragraph below.

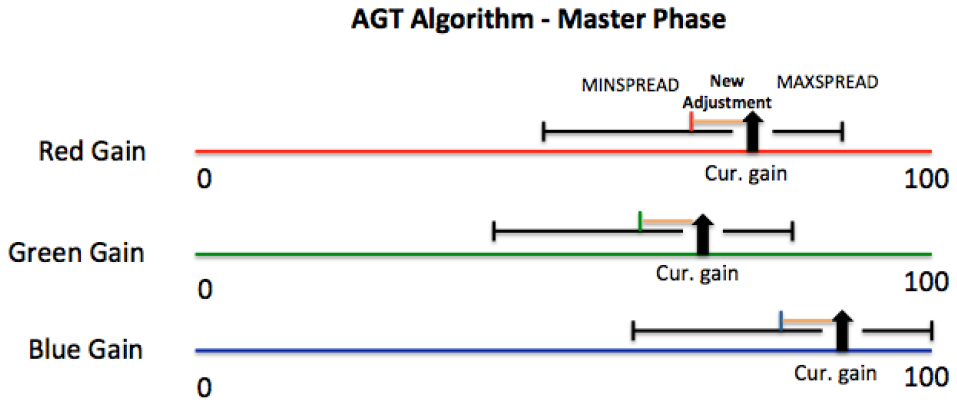

Figure 4-5. Master Phase gain movement. All three gains must change simultaneously for the same level and towards the same direction. In this figure the blue gain decreases the effective maxspread since it cannot exceed 100.

- ***Individual Phase***
  In this phase we adjust each gain level individually. We only want to do this once, since

repeatedly changing the individual gain levels would have a negative impact on the CHROM algorithm. That is true because the CHROM algorithm looks into the ratio of the chrominance signals which in turn are constructed as a linear combination of the color channels. Alternating the values in any channel individually over time would introduce unnecessary noise and corrupt the constructed PPG signal.

On the other hand, optimizing each gain level individually is made possible since the CHROM algorithm works robustly under any ambient-light hue and brightness. I.e. it does not matter if the resulting image is very reddish or greenish etc., as long as the ratios between the red, green and blue are kept constant and there is no clipping. Brightness invariance is achieved by normalizing each color channel C by dividing its samples by their mean over a temporal interval.

Consequently following our strategy to maximize each gain while at the same time minimize clipping, we try to find the optimal value as follows:

We compute for each frame the actual white and black clipping percentage on each channel as the total number of pixels whose value exceed the clipping threshold:

$$white\_clip_{Ch} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} val(i,j)_{ch} > WHITE\_CLIP\_THRSH}{TotalNumberofPixels}, (1)$$

$$black\_clip_{Ch} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} val(i,j)_{ch} < BLACK\_CLIP\_THRSH}{TotalNumberofPixels}, (2)$$

$$Ch = Red, Green, Blue$$

$$N, M = \textit{total number of rows and columns of the captured frame,}$$

$$val(i,j)_{ch}, the\ value\ of\ each\ pixel\ at\ the\ ith\ row, jth\ column\ on\ the\ ch\ channel,$$

$$WHITE\_CLIP\_THRSH\ =\ 245,$$

$$BLACK\_CLIP\_THRSH\ =\ 10,$$

We then compute the total clipping on each channel as follows:

$$total\_clip_{Ch} = total\_white\_clip_{Ch} + total\_black\_clip_{Ch} \ (3)$$

After computing the actual total clipping on each channel we can compute the hypothetical total clipping for the different gain levels[6]. Having computed our camera's gain characteristic function as explained in Appendix-A enables us to do so, as we have estimated all different gain multipliers for each gain level. After this computation we end up with a table of 100 entries for each color channel. Each entry contains the estimated clipping percentage that corresponds to the different gain levels. It is now straightforward to determine the optimal gain setting, by picking the maximum gain which has the minimum amount of total clipping.

[6] We remind to the reader that we can set the analog gain to 100 different levels.

Since the gain is analog and its characteristic function is derived using reverse engineering, the hypothetical clipping estimation is not completely accurate. This is why we need to verify in the following frames that indeed, the actual clipping is reduced. If this is not the case we further adjust the gains, but this time simultaneously on all channels, since we do not want to corrupt the PPG signal as explained previously.

- ***Master Phase***

  In this phase, the gain levels can only be adjusted as a group, as seen in Figure 4-5. Doing that does not affect the CHROM algorithm, as the extracted PPG signal is independent of the brightness level. Adjusting the gains as a group is in effect similar to adjusting the brightness level as the ratio between the color channels is kept constant. To adjust the gain levels we follow a similar process as in the individual phase.

  Initially we measure the actual white and black clipping percentage on each channel as defined in (1) and (2).

  Following that and since we can now only adjust the gains as a group, we compute the total white and total black clipping as the mean of the white and black clipping across all three color channels.

$$total\_white\_clip = mean(white\_clip_R,\ white\_clip_G, white\_clip_B) \quad (4)$$
$$total\_black\_clip = mean(black\_clip_R, black\_clip_G, black\_clip_B) \quad (5)$$

  The total clipping is again the sum of the total white and total black clipping, but notice that this time it is the same value, as opposed to before where it was measured for each channel individually:

$$total\_clip = total\_white\_clip\ + total\_black\_clip \quad (6)$$

  After measuring the actual total clipping for the current gain setting, we can again compute the hypothetical total clipping for different gain settings. Since we already have a good estimation of the optimal gains and this phase serves for fine tune them there is no reason to compute the hypothetical total clipping across all possible different gain settings. Thus we define a spread of different gain settings for which we calculate the hypothetical total clipping. After this calculation we can again choose the optimal gain following our strategy: maximum gain with minimum total clipping.

- **Watchdog functionality**

  The watchdog's functionality is to monitor changes in the color of the ambient light and not merely its brightness. Such a change could occur for example when switching on/off a colored lamp or during sundown/sunrise. When this happens, the AGT algorithm should return back to the Individual Phase as the color channel that has been affected disproportionately to the others should be readjusted individually to its optimal gain level. In addition to that, having a

significant change in the color of the light source will also affect the skin classifier. The reason is that the skin classifier is trained only once in the beginning and under certain ambient light conditions. Should these conditions change, then the same pixels that were originally considered skin will no longer be considered skin as their values will have shifted in the RGB as well as YCrCb space. Consequently we use the watchdog also as an indication to retrain the skin classifier. More can be read in the conclusions and future work chapter.

In order to detect such a change we define the following vector for input frame fr:

$$\overrightarrow{V_{fr}} = \frac{[mean(R),\ mean(G),\ mean(B)]}{norm([mean(R),\ mean(G),\ mean(B)])} \quad (7)$$

This vector corresponds to the normalized mean value of each color channel and its direction in the RGB space shows how reddish, greenish or bluish a frame is. Changes in the brightness will only change the magnitude of this vector, while changes in the color of the ambient light will change the direction of this vector.

Consequently, to detect a change in the color of the ambient light, we need to detect the angle between the two vectors of the corresponding frames. From Euclidean geometry we know that:

$$\text{angle} = ArcCos\left(\frac{\overrightarrow{V_{fr_{cur}}} \cdot \overrightarrow{V_{fr_{prev}}}}{||\overrightarrow{V_{fr_{cur}}}||\ ||\overrightarrow{V_{fr_{prev}}}||}\right) \quad (8)$$

If this angle is bigger than a threshold, then the AGT algorithm rolls back to the *Individual Phase*. This threshold is furtherly explained in design choices sector below as well as in Appendix-A.

- **Design Choices**

CLIPPING THRESHOLDS

In the Appendix-A we explain the experiment to derive the clipping thresholds.
The reason for choosing these thresholds is that we also need to consider the random error introduced by noise, which might shift a pixel's value and make it clip. Also it is preferable to calculate the hypothetical clipping with a safety margin rather than being on the edge of introducing clipping when selecting a maximum gain setting. We find that a good threshold is:

$$WHITE\_CLIP\_THRSH = 245,$$
$$BLACK\_CLIP\_THRSH = 10,$$

ANGLE THRESHOLD

In Appendix-A we explain the experiment conducted to derive the angle threshold and indicate that there has been a significant change in the color of the light source, hence our algorithm should get back to the Individual Phase. We find this angle threshold to be:

$$ANGLE\_THRSH = \mathbf{0.015}^{\circ}$$

FRAME DELAY

After each captured frame the automatic gain tuning algorithm will produce a guess for the optimal gains setting. However, when calling the camera's internal function to change the analog gains, we do not always observe the changes on the successive captured frame, even though the manufacturer states so in the manual. Changes might have a one or two frame delay. So we add a frame delay counter in our AGT algorithm that enables its functionality only every third frames, thus giving time for the new changes to take effect.

SPREAD THRESHOLD

After experimentations we have seen that the hypothetical clipping estimation does not vary significantly from the reality and that the algorithm converges after three iterations to an optimal gain setting. Thus we do not need to calculate many different hypothetical clippings in the Main Phase and introduce extra processing time. So we set the spread size to:

$$SPREAD_{THRSH} = 10$$

Figure 4-5 visualizes the idea of the spread and also includes the scenario where an individual gain is on the border, thus reducing the effective spread size.

EFFECTIVE GAIN CHANGE THRESHOLD

Since the standard deviation of noise on a color channel depends on the corresponding gain setting, having a single big clipping threshold is not entirely accurate. At the same time having a varying clipping threshold would be complex to estimate and implement, especially since the gain characteristic function is itself not entirely accurate. Observing the behavior of our algorithm after running several experiments we can see that after stabilizing on an optimal gain setting, and without further changes in the ambient light conditions our algorithm swings between two optimal thresholds with a distance of 2 levels. I.e. from [60,66,97]->[62,68,99]->[60,66,97]… and so on. This is also explained due to the inaccuracy between the real and the estimated characteristic function of the gain which is used to measure the hypothetical clipping with different gain levels. We can fix this unnecessary constant swinging by simply introducing an effective gain threshold. If the change in the master gain level is smaller than this threshold, then it shall be ignored. Data gathered from our experiments have shown that 2 is a reasonable value for this threshold.

$$EFFECTIVE\_GAIN_{THRSH} = 2$$

EXTREME CLIPPING HISTORY TRACE

We noticed that when reaching extreme clipping conditions of 100% the algorithm could not differentiate beyond that level, whereas in reality it can still get brighter/darker. This is of course due to the quantization effect and the limited accuracy that it imposes. When the AGT algorithm is in the INDIVIDUAL_PHASE, it will predict that a single step of increase/decrease in the gains is enough to minimize clipping. After this prediction it will move on to the MASTER_PHASE. However this might not be true as in reality we might need to move several more steps lower/higher to effectively reduce the actual clipping. This is why after this observation we introduced a clipping history trace which allows the algorithm to verify whether its previous guess had been correct or not. If not, it will further reduce/increase the gains individually on each channel before moving on to the MASTER_PHASE.

Skin Classifier Compensation

Changing the gain levels in each channel also affects the skin classifier. The explanation is similar as the one described in watchdog's functionality where different colors of the ambient light also affect the performance of the skin classifier. When adapting the gain in a color channel, each pixel's value on that channel changes. The classifier however is only trained once using the four descriptors r-g, r-b, Y-Cr, Y-Cb under specific gain settings. Thus changing significantly a gain level would shift skin-pixel's values far from the skin-cluster center and make the OC-SVM classify them wrongly as non-skin ones.

Luckily, we can mitigate that effect and leave the skin classifier intact by compensating for the changes that occur in each color channel when modifying its gain level. To do that, we simply apply the reverse effect every time we alter the gain. Initially we store the gain settings that were used when the classifier was trained. Subsequently, for any change of the gain in each channel we compute the reverse effect using the gain's characteristic function that is explained in the Appendix-A appendix and apply it to the captured frame before feeding it to the skin classifier.
I.e. for the training frame we store the gain settings:

$$\overrightarrow{gain_{train}} = [gain\_r_{train}, gain\_g_{train}, gain\_b_{train}]$$

The training frame's values on each color channel are derived as the element-by-element multiplication of the captured values with the initial gain:

$$\overrightarrow{frame_{train}} = [r_{train}, g_{train}, b_{train}] * \overrightarrow{gain_{train}}$$

For a new frame, captured under modified gain settings we have:

$$\overrightarrow{gain_{new}} = [gain\_r_{new}, gain\_g_{new}, gain\_b_{new}]$$

$$\overrightarrow{frame_{new}} = [r_{new}, g_{new}, b_{new}] * \overrightarrow{gain_{new}}$$

So to apply the compensation, we need to divide (element-by element) with the correction factor[7]:

$$\overrightarrow{frame_{compensated}} = [r_{new}, g_{new}, b_{new}] * \overrightarrow{gain_{new}} * \frac{\overrightarrow{gain_{train}}}{\overrightarrow{gain_{new}}}$$

[7] To derive the correction factor we look at our derived gain characteristic function. This is why the skin compensation is not 100% accurate since the actual analog gain is only approximated with this function.

5. Experimental Evaluation

To evaluate the performance of the AGT-Algorithm we compare two different implementations of the CHROM algorithm under various varying ambient light scenarios. The first implementation has the AGT component switched on while the second does not use the AGT component. We will refer throughout this rest of this chapter to the two different implementations as AGT-ON and AGT-OFF. We designed and run a series of experiments that are presented in this chapter. These experiments are used to show the strengths as well as the weaknesses of the AGT-ON by means of improving the overall SNR when compared to the AGT-OFF implementation and thereby resulting in a clearer PPG signal.

Recall that to measure the quality of the PPG signal, we use the SNR metric, as defined in Chapter 3.7. However, using the pure SNR values which are computed over a window of 512 samples and every 20 frames would not suffice. The reason is that we need to record two identical videos, one with the AGT-ON and on with the AGT-OFF versions. Having them perfectly synchronized would be practically impossible. Instead, we calculate the mean of the SNR values over a fixed temporal window of 60 seconds, as well as its standard deviation. Naturally we aim to see a higher mean value, as well as a smaller standard deviation when running the CHROM implementation with the AGT module on.

Before presenting the experiments, it is important to note the difficulties in testing and evaluating this real time control algorithm, as well as the actions that were taken in order to guarantee that the experiments were as reliable and reproducible as possible.

The key components that make the AGT algorithm hard to evaluate are the following:

1. The skin classifier, used in the implementation of the CHROM method is trained only once in the beginning of the algorithm. Even though we use the luminance component to train the classifier as explained in Chapter 4.1, we cannot have a static classifier which is invariant to varying ambient light brightness. In practice we see that if we change the brightness level of the ambient light, the skin classifier discards pixels whose value is close to clipping before feeding the rest to the AGT algorithm. Consequently the AGT algorithm cannot take a correct decision, and even if it does, it has no effect since the pixels that are already excluded from the skin classifier remain so and cannot be used to extract the PPG signal.

   In order to tackle this problem, for the purpose of our experiments, we defined a static region of interest (ROI) and turned the face detection as well as skin classifier components entirely off. In this way we only needed to guarantee that all the pixels inside our statically defined ROI are indeed skin. Figure 5-1 is a snapshot from our experiments where we can see this concept in practice. Switching the face detector and skin classifier however is not practical for commercial use and further discussion on this matter can be found in the future work chapter.

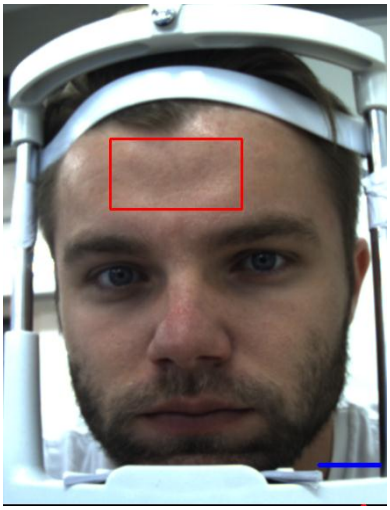

Figure 5-1. Static RoI used for experimentation.

2. Changing the analog gains of the camera can only happen in real time while a video stream is being captured. Recording a video and using it twice to run the two different implementations of the CHROM algorithm on it and compare the results would require further research and time. Therefore it is discussed in the conclusions and future work chapter. For our experiments we relied on trying to make the exact same recording twice.

   To make identical recordings we used a stopwatch to make the same changes at the same relative points in time, as well as a face stand shown in Figure 5-2 to keep the position of the face fixed throughout both recordings. A better solution would have been to run the two different algorithms in parallel using two computers and two identical cameras but unfortunately we did not have a second camera.

Figure 5-2. Face stand for experiments

3. The computations performed by the CHROM algorithm, as well as the AGT module are resource-intensive. To make sure that both algorithms do not drop any frames when extracting the PPG signal we had to implement a multi-threaded version of the algorithm, as well as reduce the frame rate to 15FPS. In order to get more accurate results and support a higher frame rate we suggest using a

faster machine with on board GPU and tweaking the algorithm by making use of the GPU libraries from OpenCV.

EXPERIMENTS

All the experiments were performed using a HP Compaq 8200 Elite CMT PC. The processor is an Intel Core i5-2400 CPU clocked at 3.1 GHz. The RAM memory is 4096MB while the operating system is Windows 7 Enterprise 64-bit. We run the experiments with Microsoft Visual Studio Professional 2012, version 11.0.61030, Update 4. The camera model that was used was uEYE 222x/UI-622x with a CCD sensor.

Due to time constraints, all of time the experiments were performed on one subject, myself.

The ROI's size used to extract the PPG signal was a 150 x 80 pixels rectangle and was centered on the forehead of the subject as shown in Figure 5-1.

5.1 Experiment – Constant Ambient Light Conditions

The purpose of this experiment is to show that under constant ambient light conditions the two different implementations of the CHROM algorithm, AGT-ON and AGT-OFF performs similarly. That is, adjusting the gains and boosting them such that they are close to their respective channel's clipping thresholds does not harm the PPG signal but on the contrary it slightly improves it. This happens due to an increase of the strength of the pulsatile component.

For this experiment we used a single light source, i.e., the lights on the ceiling of the room.

We record the stationary subject for 2 minutes, under the same ambient light conditions, twice. Initially we run having the AGT-OFF version and then we switch to the AGT-ON version. The results can be seen in Figure 5-3.

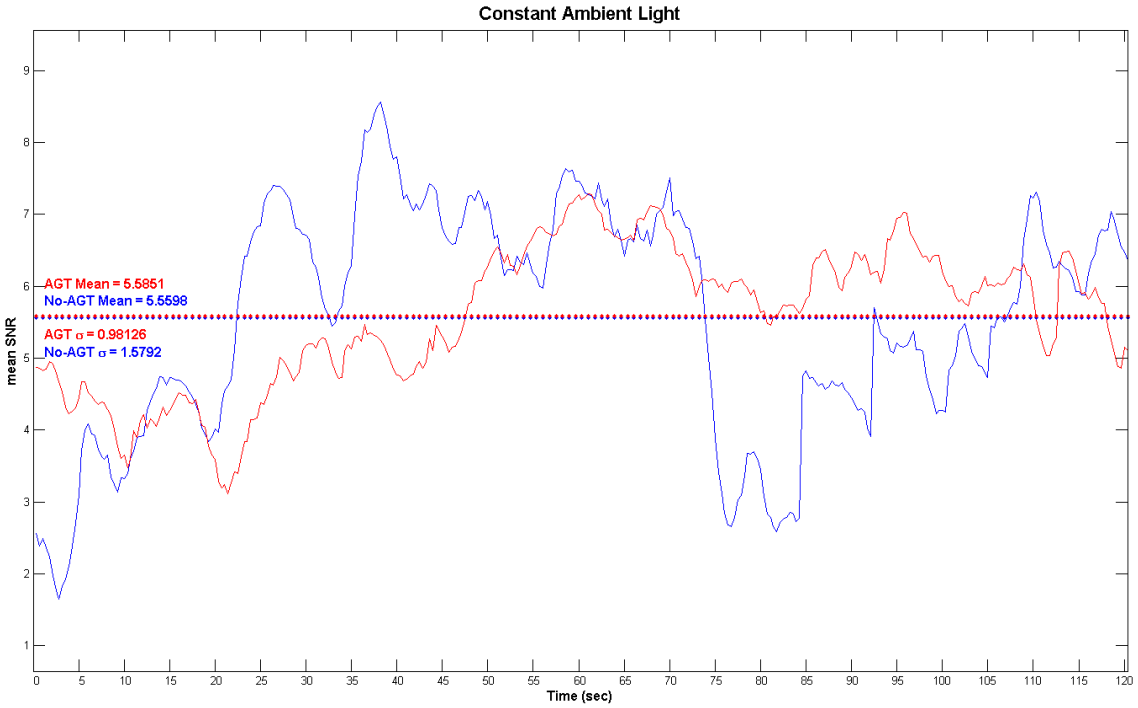

Figure 5-3. The blue line depicts the mean SNR trace for the AGT-OFF implementation while the red line depicts the mean SNR trace for the AGT-ON implementation. The dotted lines represent the mean of the both traces over the 2 minutes period.

In the Figure 5-3 above we can indeed see that the mean SNR of both implementations does not differ under static ambient light conditions while the standard deviation is a bit lower in the AGT enabled version. From this we can safely conclude that the AGT algorithm does not harm the PPG signal and interfere with the CHROM method. On the contrary it can slightly increase its performance.

5.2 Experiment – Extreme Ambient Light Conditions

The purpose of this experiment is to show that under very high or very low brightness conditions, the CHROM method suffers from clipping which results in a decreased SNR. Having the AGT-ON implementation should compensate for such extreme conditions and increase the SNR. We expect to verify that by performing two experiments. In the first experiment we first measure the PPG signal for one minute under normal brightness conditions (0% clipping). Then we increase the overall brightness until we reach 70% total white clipping on the ROI without using the AGT module and measure the SNR again for one minute. Finally we switch on the AGT module and measure again for one minute the SNR. In the second experiment we follow the exact same procedure. We decrease the overall brightness until we reach 70% black clipping.

In order to increase/decrease the overall brightness on each frame we manually adjust the shutter of the camera. In this way we can guarantee that the intensity in each color channel is changed for the same amount. When we tried to do the same experiments by using a secondary light source and adjusting its intensity, we noticed that the change in each color channel was not proportional, since the light from that source was not truly white-light.

In Figures 5-4 and 5-5 we can see the results for white clipping.

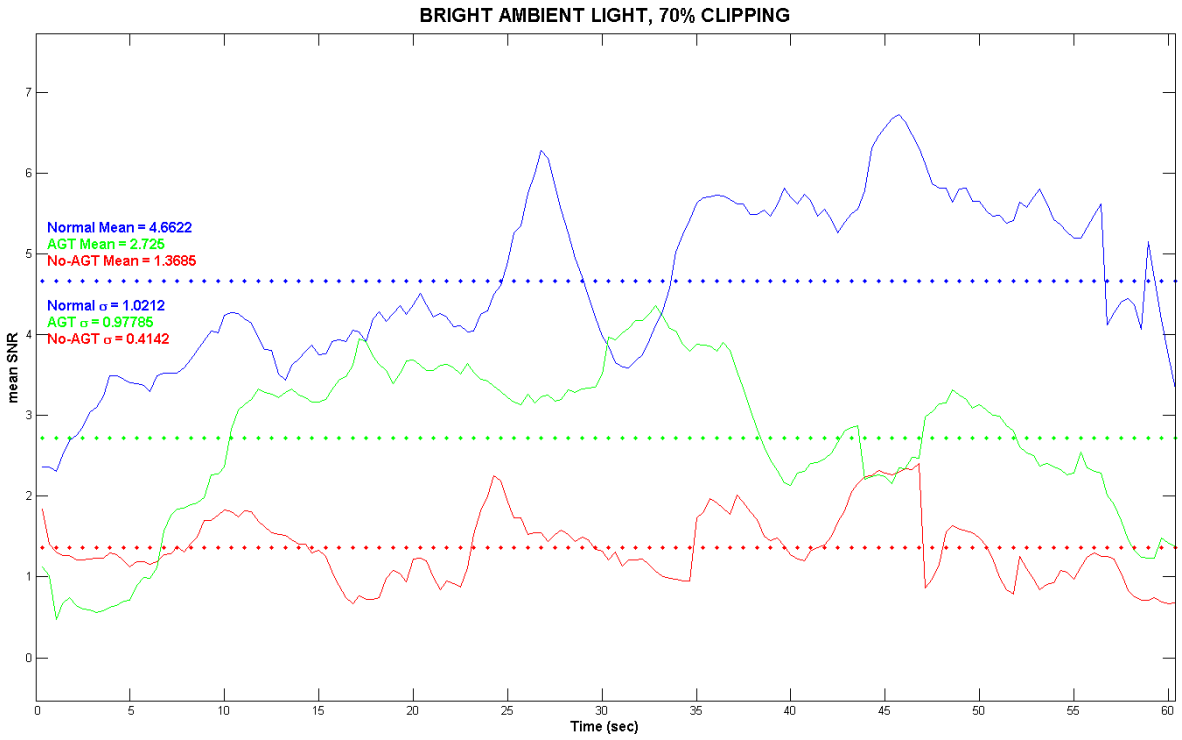

Figure 5-4. The blue line depicts the mean SNR value without any clipping. The red line shows the mean SNR value with the AGT-OFF implementation for high white clipping conditions (70%). The green line shows the mean SNR value with the AGT-ON implementation. The dotted lines represent the mean of each respective trace.

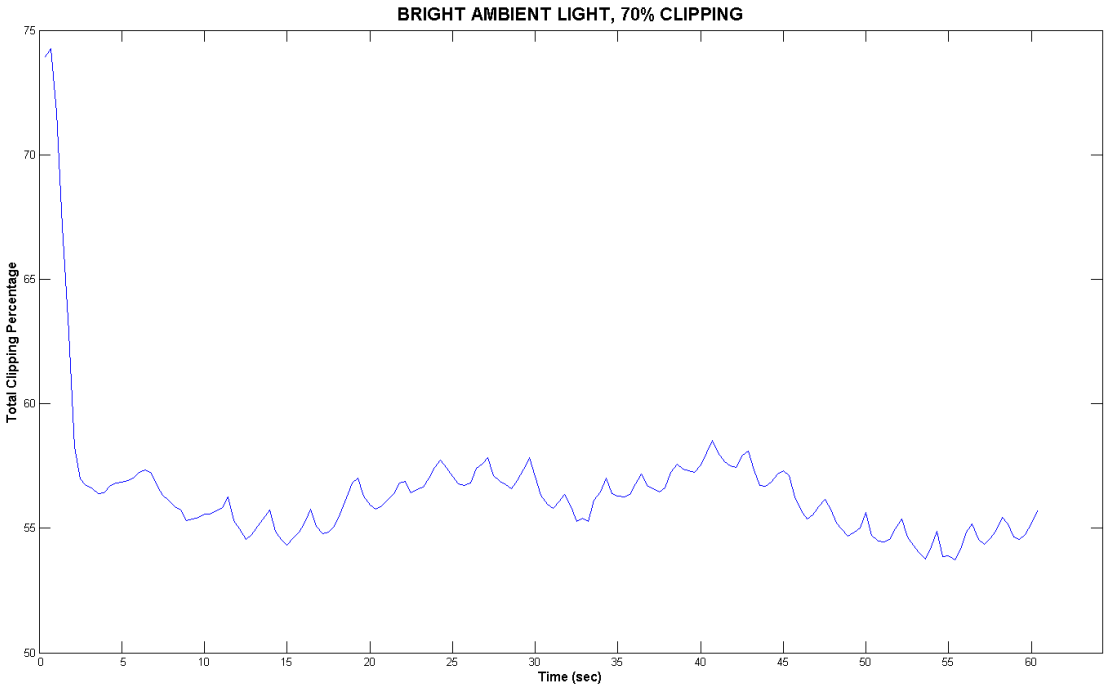

Figure 5-5 shows how clipping percentage is decreased and maintained at a low level when using the AGT-ON implementation.

We can verify from the results above that the AGT implementation indeed improves the overall SNR under high white clipping conditions. Looking at Figure 5-5 we see that the AGT algorithm decreased clipping by as much as 28%. The fluctuation in the clipping percentage that is shown is a result of the noise. We also see from Figure 5-4 that the mean SNR value over one minute increased from 1.36dB to 2.72dB when we turned the AGT module on.

In Figures 5-6 and Figure 5-7 we see the results for the second experiment (black clipping).

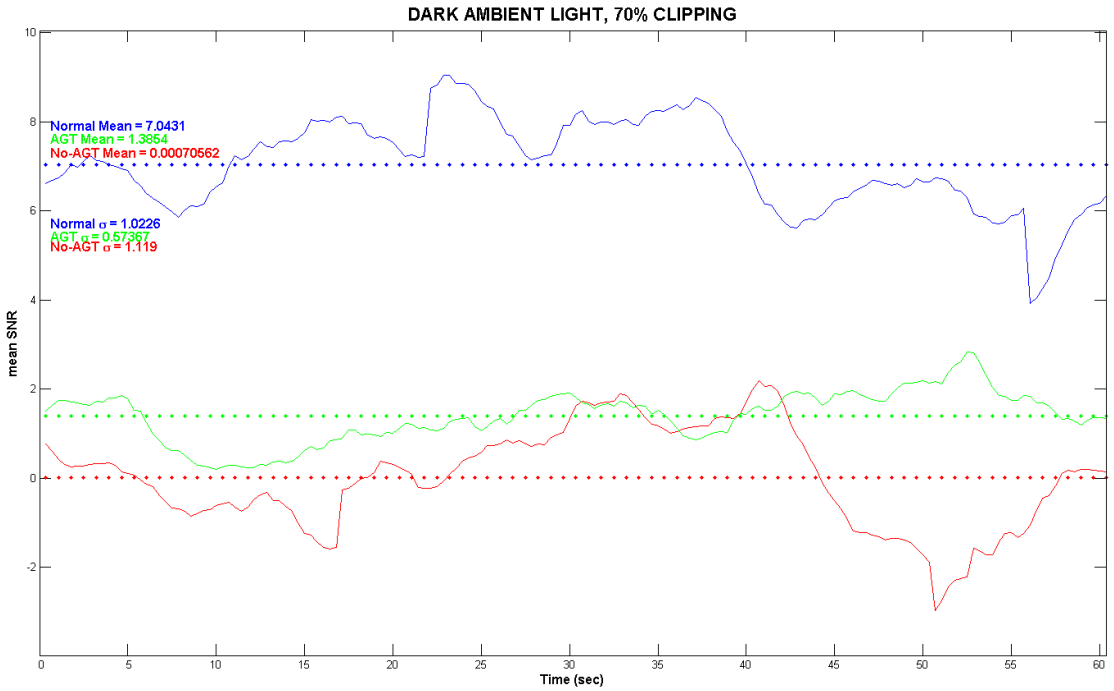

Figure 5-6. The blue line depicts the mean SNR value without any clipping. The red line shows the mean SNR value with the AGT-OFF implementation for high black clipping conditions (70%). The green line shows the mean SNR value with the AGT-ON implementation. The dotted lines represent the mean of each respective trace.

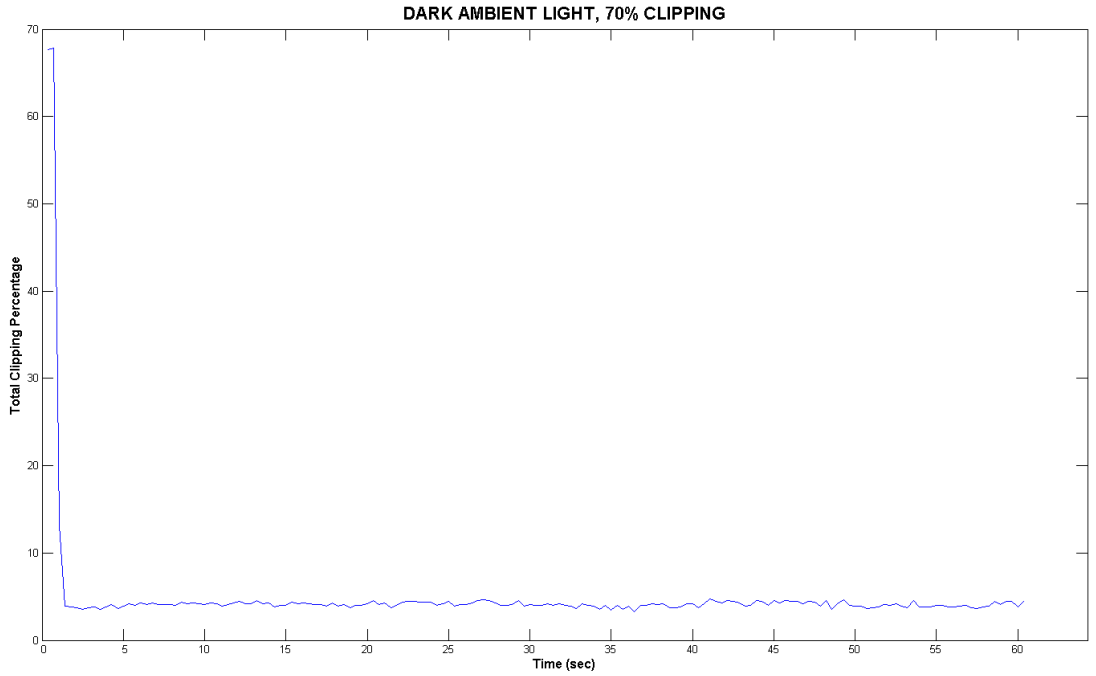

Figure 5-7 shows how clipping percentage is decreased and maintained at a low level when using the AGT-ON implementation.

In the case of dark clipping, we see that the black total clipping is decreased from 70% to 5%. Accordingly the mean SNR over one minute is increased from 0.0007dB to 1.385dB. The fact that we have a higher decrease of clipping in the dark versus the bright clipping experiment lies in that the AWB algorithm of the camera had preset the gains to the levels [10, 0, 65] before we manually increased the clipping. Thus the AGT-ON algorithm could compensate more in the case of dark clipping versus the case of white clipping.

These two experiments show the tangible added value of the AGT algorithm, since it can dynamically compensate for high clipping caused by a significant change in the ambient light conditions (i.e. morning versus evening brightness conditions, lights on/off in a room).

Additionally the results of these experiments indicated the extreme clipping history trace improvement in the AGT algorithm described in Chapter 4.2.

5.3 Experiment - Varying Ambient Light Conditions

In this set of experiments we tried to simulate real life situations where clouds or tree leaves might be constantly moving due to wind, causing frequent variations in the brightness conditions. To see how the AGT-algorithm behaves under such conditions we designed a series of experiments.

Initially we simulated a slow movement of a cloud, by periodically changing the brightness level from dark to bright. We did this by adjusting the shutter of the camera. The period where we gradually adjust the shutter is 30 seconds. Within this period we went from *normal->high->normal->low->normal* brightness levels. This procedure lasted 1 minute in total. We repeated it two times for both the AGT-OFF and AGT-ON implementations. We performed three experiments of this slow brightness adjustment, each one of them for different levels of high and low brightness. In the first experiment we will swing the brightness without introducing white or black clipping. In the second experiment we will reach 30% of white and 30% of black clipping in the ROI when using the AGT-OFF, and then reach repeat the same adjustment for the AGT-ON implementation. In the third experiment we will reach 70% of clipping on both ends.

The results for each experiment are in Figures 5-8, 5-9 and 5-10 respectively. In every figure we can see with the blue line, the mean SNR value of the extracted PPG signal without any brightness fluctuations. This serves as a reference to compare and see how the fluctuations in brightness affect the SNR. With the red line we adjust the brightness as described above and having the AGT-OFF implementation and with the green line we adjust the brightness and have the AGT-ON implementation. The dotted lines show the respective mean for each trace over one minute.

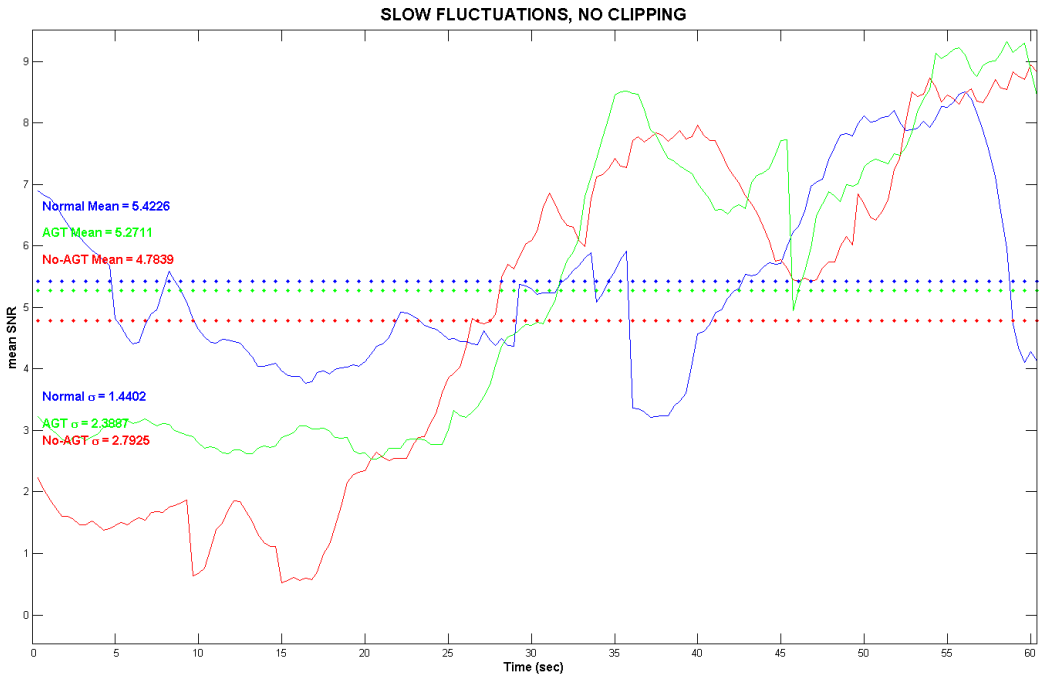

Figure 5-8. In this experiment we alternate the brightness without introducing any clipping.

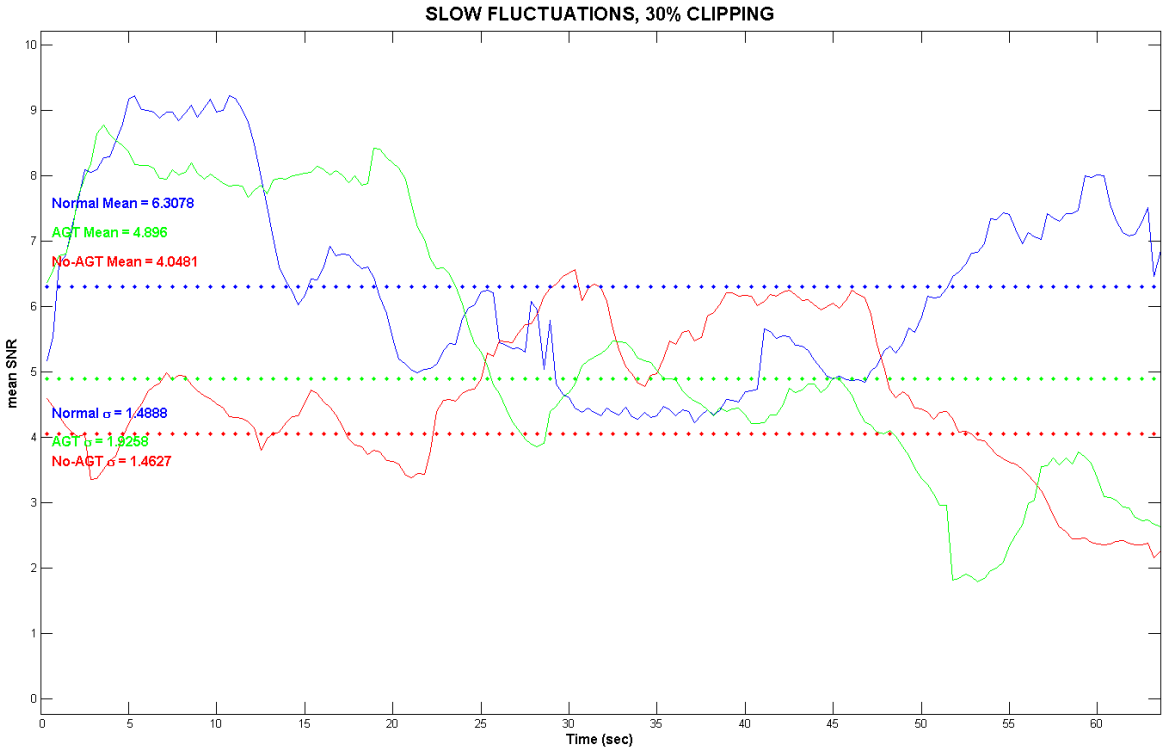

Figure 5-9. In this experiment we alternate the brightness and reach a 30% clipping on both ends.

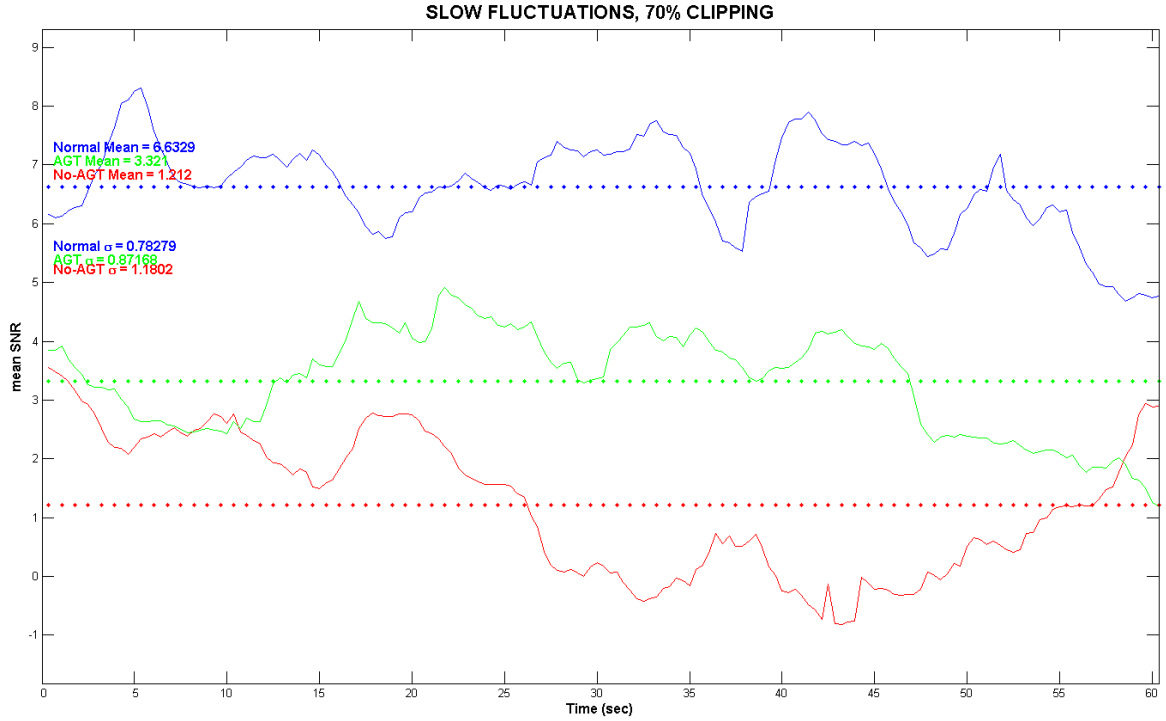

Figure 5-10. In this experiment we alternate the brightness and reach a 70% clipping on both ends.

We can see from the results above that the figures that:

- In the case where we do not introduce any clipping, the AGT-OFF performs slightly worse compared to the AGT-ON having a lower mean SNR. Both their respective mean SNR values were close to the reference one, while their respective standard deviation is bigger.
- In the case of small clipping, both mean SNR values drop lower when compared to the reference signal. The AGT-ON mean SNR is 4.89 while the AGT-OFF mean SNR is 4.04. Compensating for clipping with gains helps get a clearer signal in this case.
- In the case of large clipping, we can see a clearer improvement, looking at the mean SNR value. It is increased from 1.21db to 3.32db when using the AGT-ON implementation. In this experiment we can also see that the SNR in both cases is significantly reduced when compared to the SNR without any clipping at all.

The next set of experiments is similar to the one above. The difference is that this time we will increase the frequency of brightness change by having a period of 5 seconds. We again repeat the same three experiments for no, low, and high clipping. The results can be seen in Figures 5-11, 5-12 and 5-13.

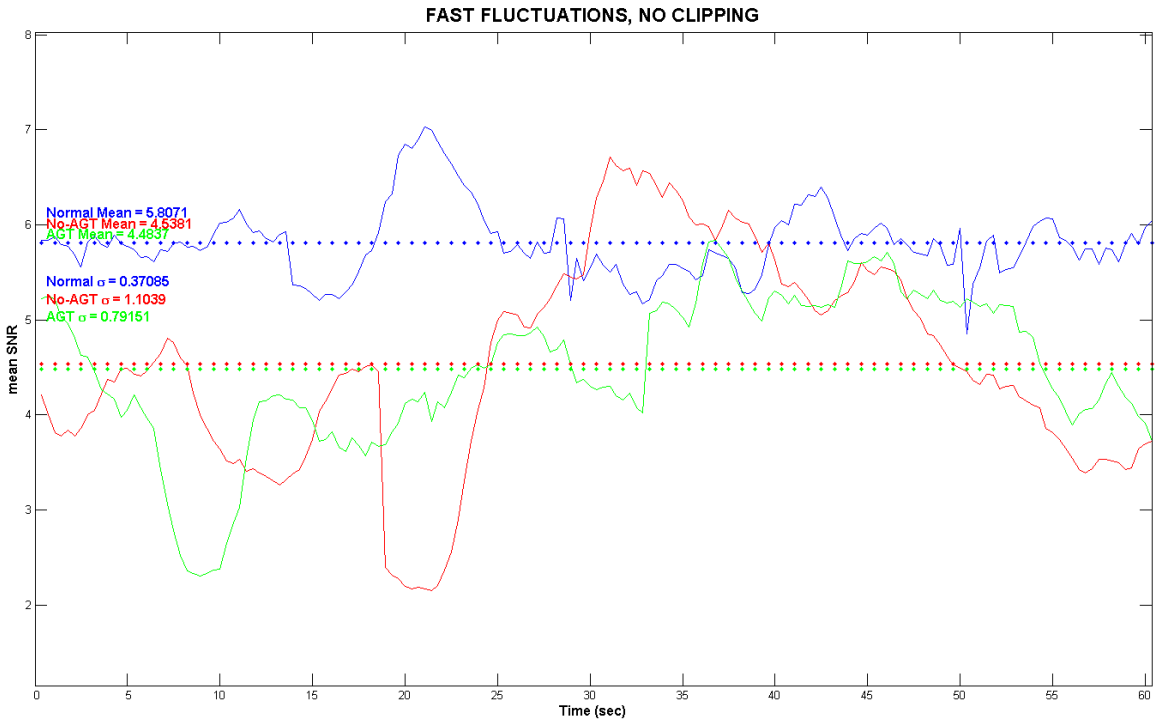

Figure 5-11. In this experiment we alternate the brightness without introducing any clipping.

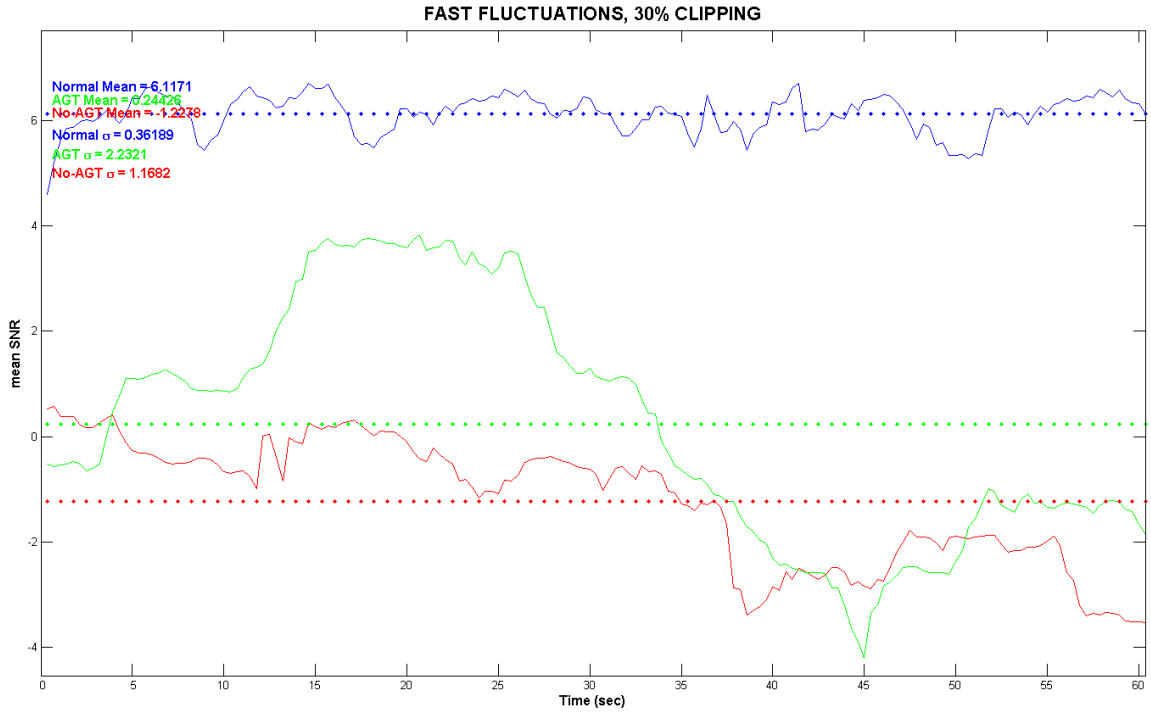

Figure 5-12. In this experiment we alternate the brightness and reach a 30% clipping on both ends.

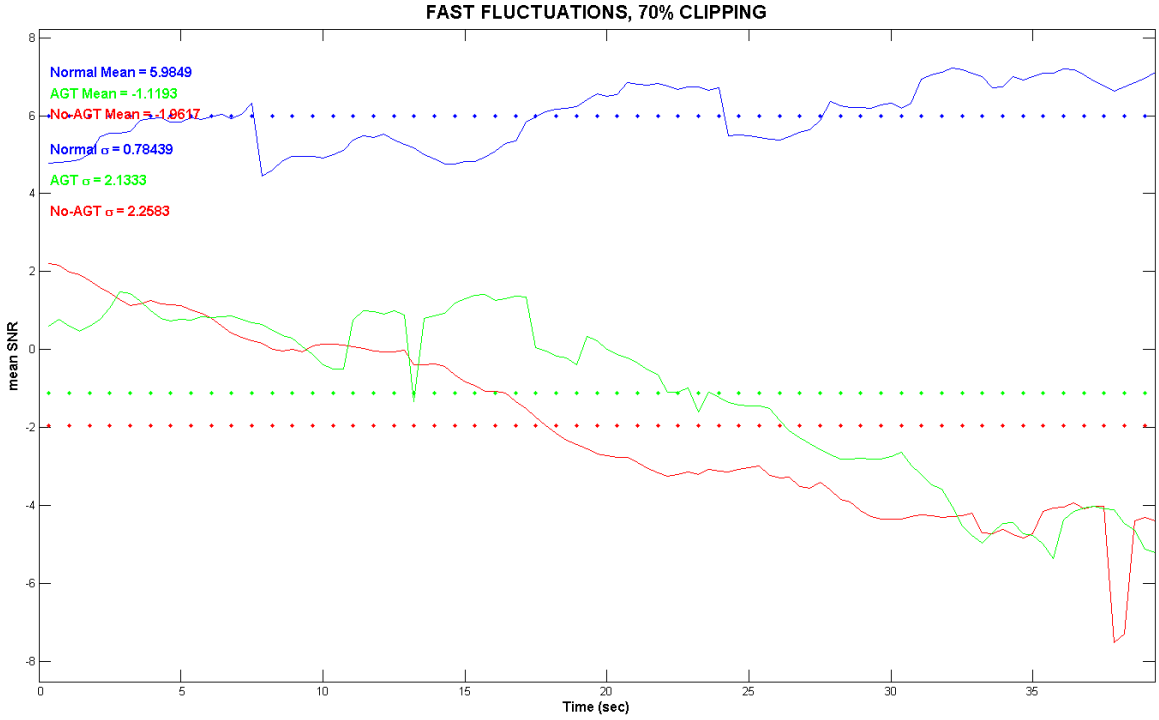

Figure 5-13. In this experiment we alternate the brightness and reach a 70% clipping on both ends.

We can see from the figures above that:

- In the case where we do not introduce clipping, the mean SNR value of both AGT-OFF and AGT-ON implementations does not vary. Also they do not differ from the mean SNR value of the signal without any brightness adjustments. This shows that the CHROM method is indeed brightness invariant.
- In the case where we introduce small clipping –with a high frequency this time- the mean SNR value drops significantly. The AGT-has a higher mean SNR value than the AGT-OFF version, but it also has a higher standard deviation, which makes it hard to derive any safe conclusions.
- In the case where we introduce high clipping with a high frequency, the PPG signal gets completely destroyed. We present only the trace for the first half-minute, since the SNR continued dropping and behaved completely irregularly. The AGT-ON implementation was unable to compensate for high clipping with a high frequency.

The above experiments were not completely accurate, since changing the brightness by tuning the manual shutter is a procedure that is prone to human error. The reason is that firstly it can introduce noise due to a slight movement of the camera when adjusting the manual shutter. Furthermore the frequency as well as the magnitude of alternation of the shutter is not completely identical for all experiments since it also relies to the human factor. However this set of experiments can serve as an initial indication that the AGT-ON implementation can indeed improve the quality of the PPG signal under certain ambient light conditions. The most prominent situation where AGT-ON proves beneficial is when the brightness levels change to either very bright or very dark. In such cases we observed an increase of almost 2dB in the mean SNR. When alternating the brightness levels, if such alternations do not introduce high clipping (>70%) and do not have a high frequency (>0.03Hz) then the AGT-ON implementation can compensate for that clipping and improve the SNR of the extracted signal.

6. Conclusions and Future Work

In this thesis, we studied how changes in the ambient light conditions can decrease the quality of the extracted PPG signal and how adjusting the analog camera gains can mitigate this problem. To do that we initially designed a theoretical framework in Matlab. In this framework we modelled 1) the color behavior of the human skin due to heartbeat, 2) varying ambient light conditions, 3) the CHROM algorithm presented in [11], 4) the gain effect of our camera model. Having all these components in place we could simulate real life scenarios and confirm our initial hypothesis that clipping caused by extreme lighting conditions –either very bright or very dark- decreases dramatically the quality of the PPG signal. Based on that observation, we devise the strategy for our real time control algorithm that adjusts the camera gains dynamically.

The logic of the Adaptive Gain Tuning (AGT) algorithm is to boost the analog gains in each color channel, while at the same time prevent clipping on the skin pixels that are used to extract the PPG signal. In this way we can amplify the relatively weak pulsatile component that can be observed due to the skin color variations during the cardiac cycle while at the same time compensate for extreme brightness conditions that cause pixels to clip.

To implement the AGT algorithm we needed to have access to the camera's drivers programmatically. Consequently we integrated in a single solution the CHROM algorithm together with the camera's functions as well as OpenCV libraries for image processing. The solution was written in C++ due to the fact that the camera drivers are also written in C++. The CHROM algorithm was re-implemented based on an existing implementation in Java by Wenjin Wang, author of [17]. The AGT algorithm component works independently of the CHROM method used to derive the pulse signal and can be integrated with different state-of-the-art algorithms for remote PPG. We also created experiments to derive the gain's characteristic function as well as set appropriate thresholds for the AGT's implementation which are all described in Appendix-A.

To test the performance of the AGT algorithm we designed a series of experiments to highlight both its strengths and weaknesses. The metric we used was the signal-to-noise ratio (SNR) of the PPG signal as described in chapter 3.7. Due to limited amount of time we did not test exhaustively the AGT algorithm but the results are a good first indication for future steps. In the few experiments conducted we saw that the AGT algorithm increased the SNR as much as 30% when very bright or very dark ambient light conditions cause most of the skin pixels to clip. In the case of slow periodic changes in the ambient light conditions we observed an increase of 17% in the SNR when small clipping (30%) was introduced and an 170% in the case where large clipping (70%) was introduced. However in the case where fast brightness fluctuations caused clipping, the SNR was too low to derive any safe conclusions and the AGT algorithm could not compensate for this.

The results above were expected. The biggest strength of the AGT algorithm is when it can compensate for extreme clipping conditions. Increasing the amount of pixels that are used to derive the PPG signal increases the SNR of the PPG signal. But, when the changes in the brightness happen with a high

frequency, similar to the one of the heart rate, then the AGT fails. The reason lies in the accuracy of the gain characteristic function and the predicted optimal gains, as well as the CHROM algorithm itself. The mean values in the three color channels change abruptly due to ambient light changes and with a frequency similar to the human heart beat. This effect cannot be completely and accurately reversed by adjusting the gains and thus we get a very low SNR.

The experiments above were conducted on only one subject and are hardly reproducible because they lie on the human factor for adjusting the brightness levels through the camera's shutter. Even though the experiments happened in a controlled environment the results can only serve as an early stage indication of the potential of this or similar algorithms that will tune dynamically the camera settings. Further testing is definitely required. Furthermore an idea worth investigating in the future is to create a phantom so that experiments can become more reproducible. One could make recordings of subjects and then play them on a screen having the camera pointed to it to extract the PPG signal. For such a scenario to work the original recordings need to be made with the highest accuracy possible, since we would suffer from quantization error twice, once in the recording phase and once when extracting the PPG signal.

Also the AGT algorithm was not tested against cases of different color changes in the environment. Should such a change happen only once, then the algorithm is designed to detect it and go back to Individual Phase where each gain can be again boosted. However under constant different-color changes of the environment, we observed that the PPG signal gets completely destroyed and the AGT algorithm cannot compensate for that. The reason lies in that the CHROM algorithm is not invariant to periodic color changes. A possible solution to mitigate this problem and something to consider for the future would be to decorrelate the luminance induced color changes from the pulse-induced color variations in the input signal (3D-matrix representation of the skin pixels). To do that one would need to decompose the input data into a set of orthogonal vectors which are linearly-independent with the use of some method of blind source separation. After successfully doing that, the gains could be individually adjusted to keep the angle of the luminance-induced color changes constant.

Another aspect of the AGT algorithm that needs further testing is its quick adaptation to changing brightness conditions. From our experiments we have noticed that within two predictions of gain levels, when in the MASTER PHASE, the AGT stabilizes. The CHROM algorithm itself is brightness invariant so we believe that the AGT component will not introduce any oscillations that can be wrongly interpreted as pulse-signal. However, if the color of the ambient light changes periodically, then the AGT algorithm will move to the INDIVIDUAL phase and set the gain levels individually. This could introduce noise and it is why we suggest above compensating for the luminance induced color changes.

We also notice that the skin classifier component in the existing CHROM algorithm implementation made it hard to measure AGT's performance as it already removed pixels that were clipped in particular, and pixels whose value had changed due to a gain adjustment or ambient light variation. The reason is that this skin classifier is static and trained only once in the beginning of the algorithm. This is why we used a static RoI to carry out our experiments. In real case scenarios we would need both the face detector as well as the skin classifier components and this is why we believe that making a dynamic,

real-time skin classifier is a necessary future step. Implementing one who can work in parallel with the AGT algorithm as well as the CHROM method is a challenging task since it needs to run fast enough to support a frame rate that can be used for robust heart rate extraction.

Last but not least, we see a potential future improvement on the SNR of the PPG signal if the camera allows for dynamic adjustment of the exposure. In case of extreme clipping conditions for example the algorithm knows whether tuning the gains has an effect or not. If we still get a 100% clipping with the gains maxed out, then we can further try to compensate for such conditions by adjusting the exposure time, should the camera-type allow for such functionality. In the case of extreme white clipping one could reduce the amount of clipped pixels by simply minimizing the exposure time. Similarly in the case of very dark environments -but still with the presence of some light- one could increase the exposure time to get a potentially better signal. Of course having a long exposure time and a moving subject could potentially prove harmful for the CHROM method but an adaptive exposure time is definitely something worth researching in the future.

To sum up, in this thesis we presented a dynamic gain tuning algorithm which under certain conditions can increase the SNR of the extracted PPG signal. Further experimentation to evaluate this algorithm as well as a novel ideas to improve it in the future are suggested but during the limited amount of time that we had, we proved that this is a topic worth further researching and that will enable more robust rPPG.

Appendix - A (Experimental Chapter)

A.1 Reverse engineering the gain factor of the camera.

To find how our camera's[8] analog gains affect the captured image we conducted a small experiment with our camera. From the camera manual we read that "*we can set the gain factor in increments from 0 to 100, but these increments are not graduated linearly throughout the range due to the sensor. The increments will typically be greater in the upper range than in the lower range*"[19]. In order to verify that statement, as well as measure and model the exact multiplier on each gain level, we set up a small experiment where we point our camera in a dark region and also make sure that there is no clipping[9] in the image. Then we increase the gain by steps of 1 and divide the mean of that image with the mean of the first image (which had a gain level of 0). We repeat this process until we reach the maximum gain level of 100. The gain factor for every step is given by the formula:

$$GainFactor(step) = \frac{mean\big(Image(step)\big)}{mean\big(Image(1)\big)},$$

$$where\ 1 \leq step \leq 100\ \wedge step\ \in \mathbb{N}$$

$$and\ Image(step)\ is\ the\ image\ captured\ with\ the\ respective\ gain\ setting$$

The results for gain levels from 0-100 give us the following function:

```
gain_factor = [1.00000,1.00478 ,1.01551 ,1.03252 ,1.05448 ,1.05603 ,1.0711
,1.07405 ,1.10964 ,1.12242 , 1.14208, 1.15271, 1.17757, 1.2085 , 1.19374,
1.22417, 1.25412, 1.25458, 1.28201, 1.31069, 1.30029, 1.3292 , 1.35209,
1.33468, 1.36011, 1.39888, 1.38943, 1.41968, 1.4451 , 1.44792, 1.48692,
1.50771, 1.51717, 1.54296, 1.57528, 1.58545, 1.61544, 1.6517 , 1.65397,
1.69003, 1.72091, 1.72545, 1.76187, 1.80931, 1.81081, 1.83937, 1.87833,
1.88111, 1.92301, 1.96711, 1.97061, 2.01368, 2.04836, 2.05402, 2.10064,
2.1449 , 2.14697, 2.18987, 2.23943, 2.23211, 2.29021, 2.34042, 2.33031,
2.39961, 2.44443, 2.45216, 2.49325, 2.55389, 2.5672 , 2.60934, 2.65991,
2.6771 , 2.72804, 2.79006, 2.791  , 2.85893, 2.9172 , 2.9233 , 2.98237,
3.05175, 3.04112, 3.12366, 3.18021, 3.17848, 3.24807, 3.31583, 3.31964,
3.3761 , 3.44881, 3.44759, 3.53075, 3.59815, 3.57295, 3.65349, 3.72554,
3.74844, 3.80727, 3.75949, 3.79711, 3.80898, 3.80898];
```

In Figure A1 we plot our camera's gain characteristic function which confirms the manual:

[8] Our camera model is UI-222x/UI-622x

[9] It is important to have 0 percent clipping, as this would affect our calculations in deriving the multiplying factor for each gain level. In order to measure clipping, we measure the amount of pixels whose value is above 254 or below 1. Keep in mind that a pixel value can only be between 0-255 as the camera quantizes in 8 bits.

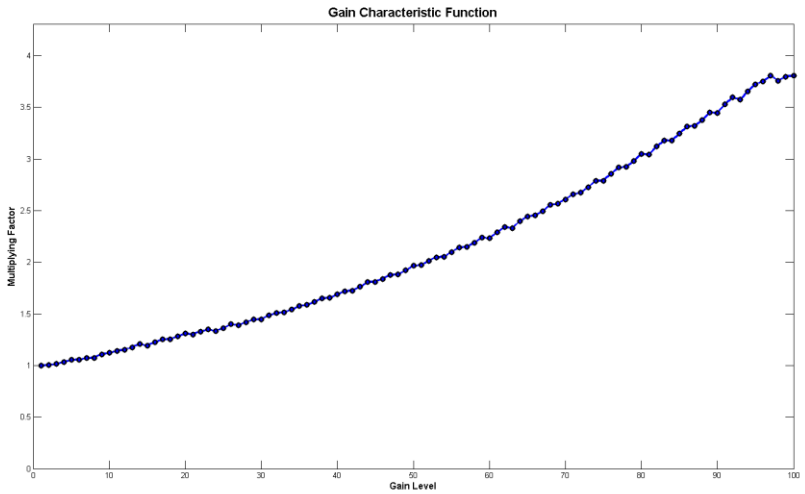

Figure A1. Gain's Characteristic Function

A.2 Experiment for the standard deviation of our model's noise.

In order to determine the standard deviation of our noise and see how the mean SNR of the extracted heart rate signal throughout 2400 frames behaves for different values of σ we run a small test.

Initially we increase σ in the range of **[0.100 - 0.500]** with a step of **0.001.**
Figure B1. shows the results of this test:

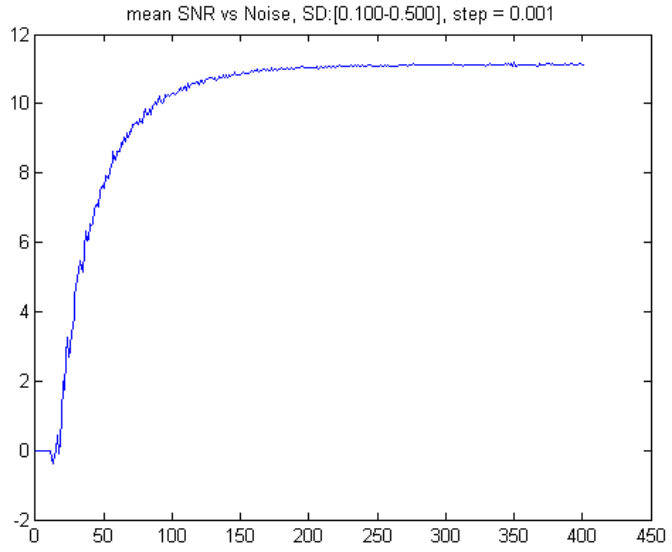

Figure B1. Mean SNR against Noise for different standard deviation values, σ = [0.1 – 0.5] step 0.001

From the above we can see that:

- Without noise we cannot extract a pulse signal.
- For a noise with **σ < 0.111** we still cannot extract a pulse signal, and **SNR = 0**

- For a noise with **0.111<σ<0.115**, **SNR<0**
- For a noise with **0,115< σ, SNR increases**
- **SNR** reaches a maximum threshold of **[11.00 - 11.2],** when σ > 0,353

Figure B2 shows the SNR trace for the corresponding values of σ. The rainbow colors indicate that the closer we are to purple, the bigger the σ is and the larger the SNR.

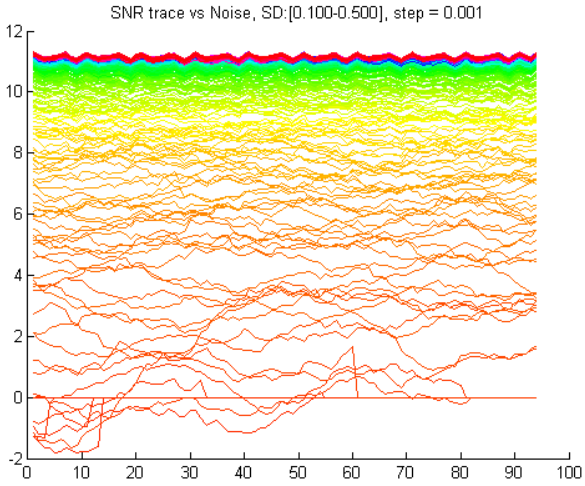

Figure B2. SNR trace against Noise for different standard deviation values, σ = [0.1 – 0.5] step 0.001

In order to see for how long the SNR is maintained in the maximum threshold with respect to noise's standard deviation(i.e. measure our signals resilience to noise) we run more tests to see at which value of σ the SNR starts dropping.

Figure B3 shows where this happens:

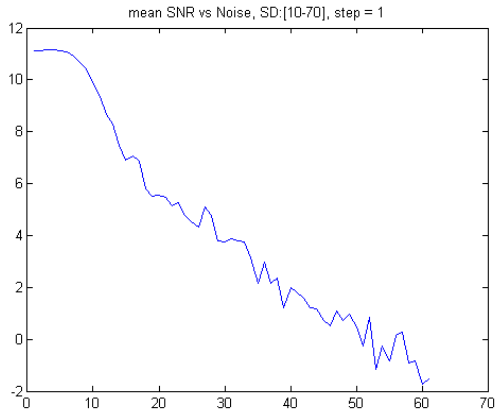

Figure B3. Mean SNR against Noise for different standard deviation values, σ = [10 – 70] step 1

In this experiment we let σ be in the range [10 – 70] and step = 1. From the figure B3 above we can see that:

- **SNR** drops from the highest threshold for 15<**σ**<**50**
- **SNR** becomes negative for **50**<**σ**

The results can also be seen in the SNR trace diagram in figure B4 below:

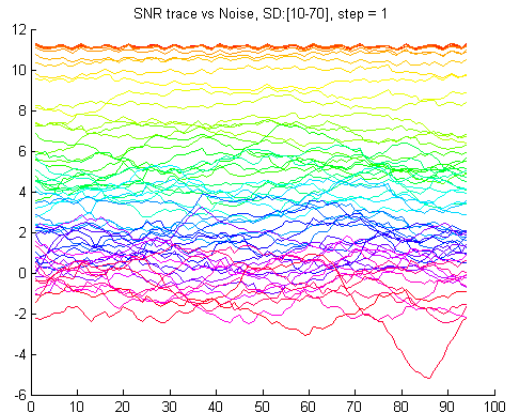

Figure B4. SNR trace against Noise for different standard deviation values, σ = [10 – 70] step 1

The reason that low SD noise increases the SNR of our extracted signal lies in the fact that the noise reveals the pulsatility in all three channels which is otherwise not visible due to its tiny amplitude with respect and the fact that we use just 8 bits for quantization. We can verify that by plotting our pulse's spectrum in all channels. Figure B5 has no noise at all, Figure B6 has noise with σ=10 and Figure B7 has noise with σ = 60. We see that in the first case we cannot see any pulsatility, in the second case we clearly see pulsatility and in the third case we see many irregularities which are caused due to the high amplitude of the noise in comparison with the pulse signal.

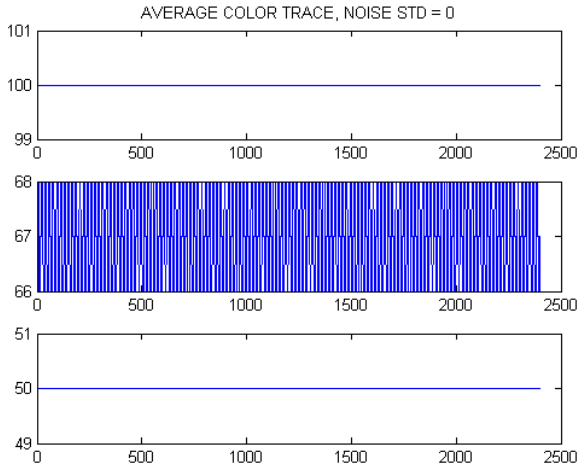

Figure B5. Pulse Trace in RGB-Channels, Noise σ = 0.

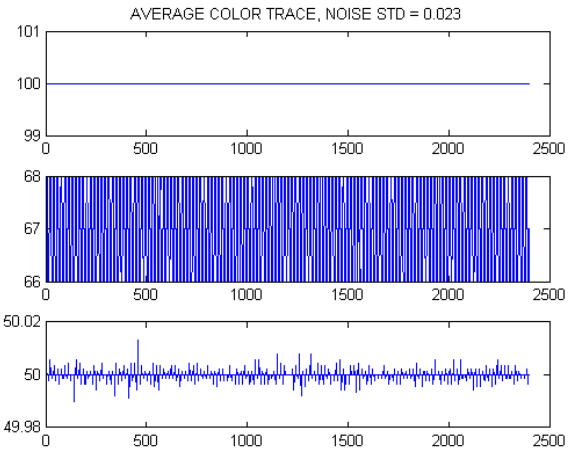

Figure B6. Pulse Trace in RGB-Channels, Noise σ = 0.023

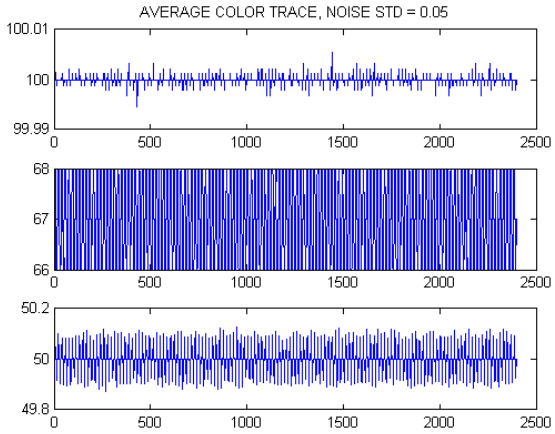

Figure B7. Pulse Trace in RGB-Channels, Noise σ = 0.05

The table below summarizes our findings:

Noise Standard Deviation	Mean SNR
0 (no noise)	no pulse
σ<0.11	no pulse
0,111<σ<0.115	Negative SNR
0,115<σ<0.353	SNR increases
0.353<σ<15	Max threshold SNR = [11 -11.2]
15<σ<50	SNR decrease
50<σ	Negative SNR

From the above we choose for our noise model to use noise with standard deviation of σ = 10 which gives a mean SNR of 9.911 which is close to the SNR we observe in the reality.

A.3 Angle Threshold Experiment

In order to detect a significant change in the color of the ambient light source and notify our AGT algorithm to move to the individual phase, as well as indicate a retraining of the skin classifier we create the following experiment:

We sit in a dark room with low ambient light conditions and the only two light sources being:

- The computer screen which shows on fullscreen a single colored picture
- Distant white lights of the room, which are kept on throughout the whole experiment

The gains settings for each color channel are kept constant at the values:

$$gain\_settings = [1, 9, 89]$$

For each set of consecutive frames we calculate the angle between the two vectors $\overrightarrow{\mathrm{V}_{fr_{cur}}} . \overrightarrow{\mathrm{V}_{fr_{prev}}}$:

$$\text{angle} = ArcCos\left(\frac{\overrightarrow{\mathrm{V}_{fr_{cur}}} . \overrightarrow{\mathrm{V}_{fr_{prev}}}}{||\overrightarrow{\mathrm{V}_{fr_{cur}}}|| \; ||\overrightarrow{\mathrm{V}_{fr_{prev}}}||}\right),$$

$$where \; \overrightarrow{\mathrm{V}_{fr_i}} = \frac{[mean(R_i), \; mean(G_i), \; mean(B_i)]}{norm([mean(R_i), \; mean(G_i), \; mean(B_i)])},$$

$$where \; \mathrm{R_i}, G_i, B_i \; are \; the \; pixel \; values \; on \; each \; color \; channel \; of \; the \; i-th \; frame$$

We change the color of the image shown from the computer screen to red, green, blue, white, black with the following sequence:

Black->red->Black->red->
Black->green->Black->green->
Black->blue->Black->blue->
Black->red->green->blue-> red->green-> blue->
Black->white-> Black->white-> Black->white-> Black->white.

In the figure C1 we plot the angle between consecutive frames, as well see the corresponding intensity of the vector of the current frame computed by:

$$\text{intensity}_{\mathrm{fr_i}} = norm([mean(R_i), \; mean(G_i), \; mean(B_i)])$$

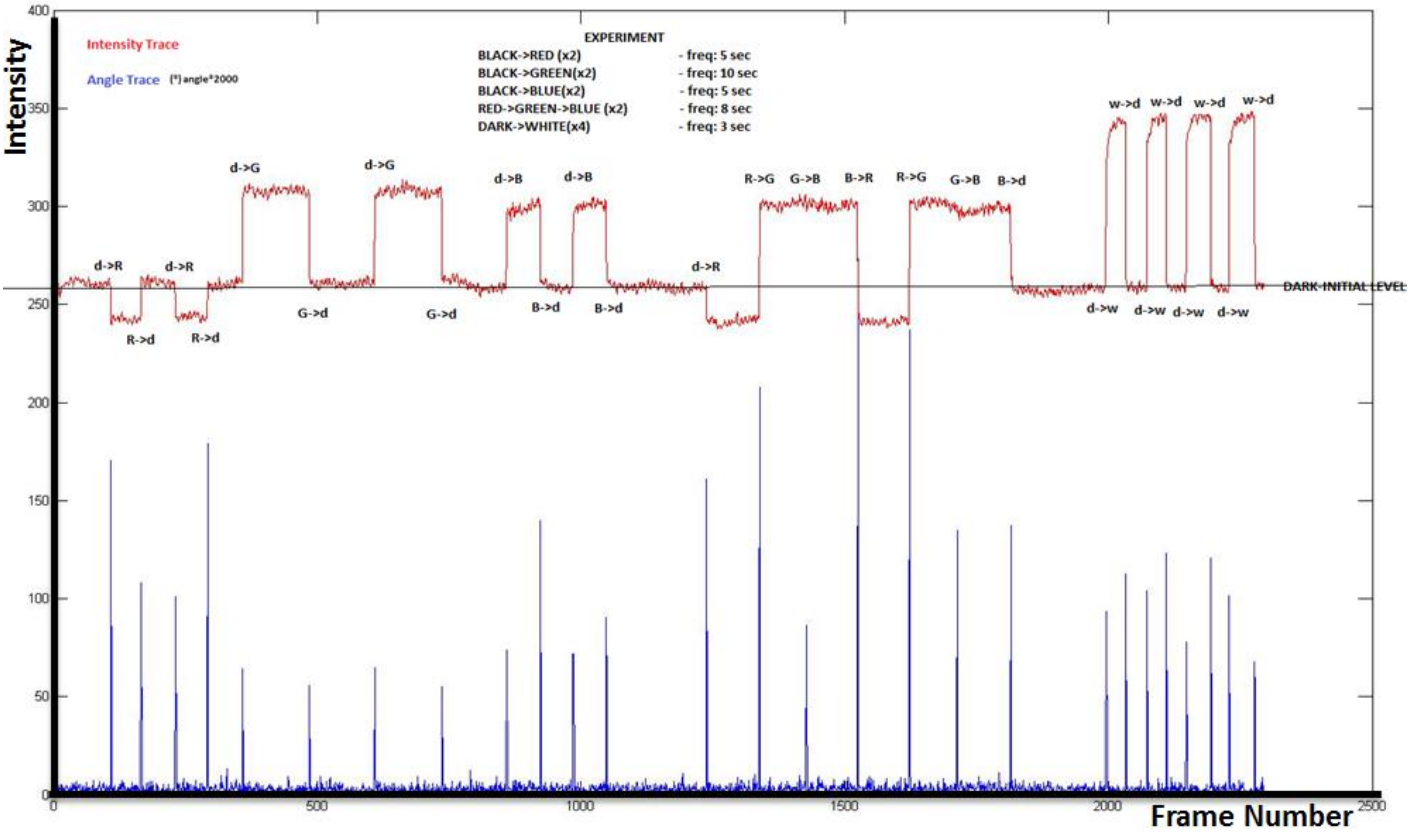

Figure C1. Intensity and Angle Trace for varying ambient light conditions.

The blue peaks indicate a change in the angle which corresponds to a change in the color of the light source. The values are scaled with a factor of 2000 for demonstration purposes.

Removing the scaling factor, we see that an angle threshold that signifies a change in the color channel is:

$$ANGLE\_THRSH = \mathbf{0.015}^{\circ}$$

A.4 Clipping Threshold Experiment

To determine a safe clipping threshold we need to consider that some pixels that may indeed be clipped could have their values shifted due to random noise, and thus not be counted in. Of course the same argument can hold for pixels which are originally not clipped but due to noise still appear as clipped. However since we want to avoid clipping at any cost, it is better to have a safe margin for measuring clipping than being on the edge and measuring less clipping than there actually is. To determine a safe threshold we create an experiment to estimate the actual standard deviation of the noise:

Initially we point our camera to a still object and capture 10 consecutive frames under a fixed gain setting of 0. (We process only the red channel, because the computation is heavy and also because we assume the noise to have the same standard deviation across all channels. Even if this is not entirely true, we still need to determine one clipping threshold for all the three color channels).

After acquiring the 10 frames we end up with 10 NxM matrices, where N=768 and M=576:

$$Fr_i[N, M], i: 1, \dots, 10$$

We then take the pixel-by-pixel average to arrive to a noise-free matrix, since the noise should be zero-mean.

$$noiseFreeM(i,j) = \frac{\sum_{k=1}^{10} Fr_k(i,j)}{10}, \qquad i = 1, \dots, N, \qquad j = 1, \dots, M$$

Then for each of the initial 10 frames we subtract the noise-free matrix and compute the standard deviation of the values. As a final touch, we take the average of those values. This should be an accurate indication of the standard deviation of the noise.

$$noiseStD = \frac{\sum_{i=1}^{10} std(Fr_i - noiseFreeM)}{10}$$

We repeat the same process by adjusting the gain in steps of 10 until 50. From there on clipping occurred in our setting, and the estimation would be corrupted. Figure D1 shows the results of this experiment:

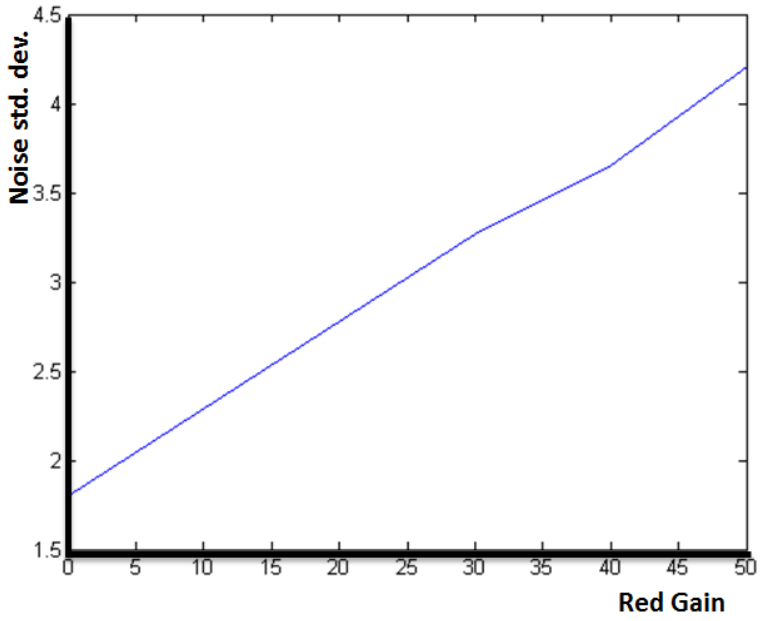

Figure D1.

We see that the standard deviation of the noise increases linearly with the gain levels. This then must mean that the highest standard deviation of the noise would be lower than 10. So a safe threshold for measuring white and black clipping would be, since our dynamic range is [0-255]:

$$WHITE\_CLIP\_THRSH = 245,$$
$$BLACK\_CLIP\_THRSH = 10,$$

References

[1] Brynne A. Sullivan, MD1, Stephanie M. Grice, MD1, Douglas E. Lake, PhD2, J. Randall Moorman, MD2,and Karen D. Fairchild, MD. Infection and Other Clinical Correlates of Abnormal Heart Rate Characteristics in Preterm Infants. The Journal of Pediatrics, Vol. 164, is. 4, pp775-780, April 2014

[2] Douglas E. Lake • Karen D. Fairchild •J. Randall Moorman, Complex signals bioinformatics: evaluation of heart rate characteristics monitoring as a novel risk marker for neonatal sepsis. Springer Science and Business Media, New York 2013

[3] Vincent Pichot, Frederic Roche, Jean-Michel Gaspoz, Franck Enjolras, Anestis Antoniadis, Pascal Minini, Frederic Costes, Thierry Busso, Jean-Rene Lacour, Jean Claude Barthelemy, Relation between heart rate variability and training load in middle-distance runners, Medicine and Science in Sports & Exercise 32, no. 10, January 2000.

[4] A.B. Hertzman, "Photoelectric plethysmography of the fingers and toes in man," Exp.Biol. Med., vol. 37, no. 3, pp. 529-534, 1937

[5] F.S. Afsar, Skin care for preterm and term neonates, Clin Exp Dermatol, 34 (2009), pp. 855–858

[6] kuller J. McManus, Skin breakdown: risk factors, prevention, and treatment, Newborn Infant Nurs Rev, 1 (2001), pp. 35–42

[7] K.J. Anand, F.M. Scalzo, Can adverse neonatal experiences alter brain development and subsequent behavior, Biol Neonate, 77 (2000), pp. 69–82

[8] M. Huelsbusch and V. Blazek, "Contactless mapping of rhymthical phenomena in tissue perfusion using PPGI," Proc. SPIE, vol. 4683, pp. 110-117, 2002.

[9] C. Takano and Y. Ohta, "Heart rate measurement based on a time-lapse image," Med. Eng. Phys., vol 29, pp. 853-857, 2007.

[10] Lonneke A.M. Aarts, Vincent Jeanne, John P. Cleary, C. Lieber, J. Stuart Nelson, Sidarto Bambang Oetomo, Wim Verkruijsse, "Non-contact heart rate monitoring utilizing camera photoplethysmography in the neonatal intensive care unit – A pilot study", Early Human Dev., Vol 89, Is. 12, Dec. 2013, pp. 943-948.

[11] G. de Haan and V. Jeanne, "Robust pulse rate from chrominance-based rppg," Biomedical Engineering, IEEE Transactions on, vol. 60, no. 10, pp. 2878–2886, 2013

[12] W. Verkruijsse, L. O. Svaasand, and J. S. Nelson, "Remote plethysmographic imaging using ambient light," Optics express, vol. 16, no. 26, pp. 21 434–21 445, 2008.

[13] M. Lewandowska, J. Ruminski, T. Kocejko, and J. Nowak, “measuring pulse rate with a webcam – A non-contact method for evaluating cardiac activity,” in Proc. Federated Conf. Copmut. Sci. Inform. Syst., 2011, pp. 405-410.

[14] P. Viola, M. Jones, “Rapid object detection using a boosted cascade of simple features”, Computer Vision and Pattern Recognition, 2001, Vol. 1, pp. 511-518.

[15] Y. Chen, X. S. Zhou, and T. Huang, “One-class svm for learning in image retrieval,” bol. 1, pp. 34-37 vol.1, 2001

[16] K. N. Plataniotis and A. N. Venetsanopoulos, Color image processing and appllications. Springer, 2000.

[17] Wenjin Wang, Sander Stuijk, and Gerard de Haan, Exploiting Spatial-redundancy of Image Sensor for Motion Robust rPPG, IEEE transactions on Biomedical Engineering, 2014.

[18] Yechiam Ostchega, Kathryn S. Porter, Jeffery Hughes, Charles F. Dillon, Tatiana Nwankwo, Resting Pulse Rate Reference Data for Children, Adolescents, and Adults: United States, 1999-2008, National Health Statistics Reports, Number 41, August 24, 2001.

[19] uEye Camera Manual, Function Descriptions, is_SetHardwareGain,
Retrieved from http://en.ids-imaging.com/manuals-ueye.html