Two-person segmentation for video based actigraph generation in a shared bed environment

Puvvula, B.

Award date:
2012

MASTER THESIS

5T746

Two-Person Segmentation for Video Based Actigraph Generation in a Shared Bed Environment

Bhargava Puvvula (0756930)
b.puvvula@student.tue.nl

Company supervisors

Ir. Adrienne Heinrich
adrienne.heinrich@philips.com

Dr. Ir. Frank Van Heesch
frank.van.heesch@philips.com

University supervisor

Prof. Dr. Ir. Gerard De Haan
g.d.haan@tue.nl

September 26, 2012

Abstract

Sleep studies assess the quality of sleep and body response to sleeping disorders. Sleep studies have gained prominence as empirical studies have shown the relation between untreated sleep disorders to e.g. high blood pressure, stroke, heart diseases and other severe medical conditions. Polysomnography (PSG) [1] and wrist actigraphy are widely used in sleep studies. These methods are neither comfortable nor a convenient way to monitor sleep because the user is confined by sensors and wires.

There is a very promising market for contactless sleep monitoring, not only for sleep clinics, but also for home applications. With the development of video processing and analysis, video based sleep analysis promises to be a suitable alternative to wrist actigraphy for home monitoring. Compared with the commonly used wrist actigraphy, the video based method is contactless and provides a possibility of measuring the movements of the entire body.

A shared bed scenario is a common case in home environment and distinguishing person of interest's (POI) movements from that of bed partner's (BP) movements is difficult as they sleep in close proximity. It becomes even more challenging when the camera is placed on the bed side table as the viewing angle would show BP to be just behind POI.

This thesis presents the development and implementation of a novel sleep monitoring application that is capable of monitoring the movements from the person of interest (POI) in a shared bed scenario using a camera and an IR emitter mounted on the bed side table. The application takes video data as input and generates a *video actigraph* that has movement signals that relate to POI movements. The POI area in the image is segmented by using an AdaBoost classifier that uses several discriminative features like brightness, focus, gradients and motion vectors. The *video actigraph* is then generated by frame differencing in the POI area. The application has been ported onto an embedded platform called *BeagleBoard* and has the potential to run in real time.

The application has been tested over diverse data sets and has been found to be robust to movements originating from under the blanket, various sleeping positions, different illumination conditions, beds and blankets. The developed sleep monitoring application can successfully replace wrist actigraphy in monitoring POI's movements. The application can detect movements from POI up to an accuracy of 92.5% compared to 70% accuracy using *video actigraph* generated without POI area segmentation and 75% using *wrist actigraph* generated by wrist actiwatch worn by POI.

Acknowledgements

I would, first of all, like to express my deepest gratitude to my graduation supervisor Prof. Dr. Gerard de Haan for giving me the opportunity to work at Philips Research Eindhoven on this project. It is during his course 'Video and Image Processing for Multimedia Systems' that I developed a great interest and passion for Video and Image Processing.

I am greatly indebted to my supervisors Ir. Adrienne Heinrich and Dr. Ir. Frank van Heesch for showing extraordinary patience and faith in me. They were always available and never hesitated to help me solve the problems that I have encountered during this project. They always stood by me and encouraged me to be innovative. Due to their mentoring, I was able to push my learning and critical thinking abilities to their limits in an inviting, fun, and energetic setting. I could not have done it without the support of my supervisors and I hope I have not disappointed them in any manner.

My special thanks to Pedro Fonseca, Els Møst, Sebastian Popa and Tom Bernards for providing the infrastructure for recording the sleep videos. I am grateful to all of them who have volunteered for the recordings. I also like to thank all my professors at Eindhoven University of Technology for their help and support during the master's program.

I thank all my colleagues Erdenay Alpay, Gerardo Santiago, Luis Alvarez, Yingying An and Yidan Guo for keeping our office cheerful. My special thanks to Ganesh Jayachandran who has always helped me with the necessary technical expertise I may have lacked and Siddharth Chunduri for the long brain storming discussions we often had. I would further like to thank my house mates for their continuous support and encouragement. Finally last but not the least I thank my parents and my sister for their unconditional love and support.

Contents

1 Introduction

One third of our day is spent on sleeping. The other two-thirds are affected by the quality of sleep during the previous night. Studies have documented the impact of poor sleep on cognitive performance and overall life quality[2]. According to Muzet[3], the alteration in the pattern of body movements might reflect the deterioration in the sleep process. Jansen and Shankar [4] made possible the classification of different sleep states e.g. deep sleep and light sleep with movement-related signals by representing the sleep-wake state as a function of body movement. So far, polysomnography (PSG)[1] and actigraphy are widely used to assess the sleep quality or to diagnose sleep-related disorders[5]. For methods like these, obstructive sensors (e.g. electrodes and accelerators) are to be worn on-body.

Existing methods for sleep monitoring are neither comfortable nor a convenient way to monitor sleep states because the user is confined by sensors and wires. Standard overnight PSG includes recording of several physiological signals such as EEG, EOG, EMG and ECG[6], for which sensors are to be worn on the head, arms, chest and legs. In wrist actigraphy, an Actiwatch[7] is worn on the wrist which measures the wrist movements.

There is a very promising market for sleep state monitoring, not only for sleep clinics, but also for home applications [8, 9]. In the home application case, the user would preferably not wear electrodes nor sleep with a wrist watch. Moreover, a shared bed scenario is a common case in home applications.

With the development of video processing and analysis, video based sleep analysis promises to be a suitable alternative to wrist actigraphy. Compared with the commonly used wrist actigraphy, the video based method is contactless and provides a possibility of measuring the movements of the entire body. Using other unobtrusive sensors (such as radar) may have difficulties in distinguishing movements between the Person of Interest (POI) and Bed Partner (BP). This thesis project is aimed at developing a video-based sleep monitoring application that uses near-infrared video to generate video based actigraph or *video actigraph*.

1.1 Existing video-based sleep monitoring systems

Traditionally, video was employed in sleep labs as an auxiliary channel for a sleep technician to monitor a subject's status during a sleep study using polysomnography (PSG). Sivan *et al.*[10] have reported that the results from PSG measurements showed high correlation with those obtained from home video recordings. The recording procedure and the subsequent analysis however were performed manually. Since then many researchers have been trying to develop automatic video-based sleep monitoring systems [8-14].

Nakajima *et al.*[11] developed an image sequence analysis method to evaluate posture change and respiratory rate of a subject in bed. Yang *et al.*[12] employed a neural-network

approach to recognize four types of sleeping position (body up, down, left, right) from image sequences captured with an IR-based night vision camera. Wang *et al.*[13] developed a machine-learning algorithm to locate the upper body of humans covered by a blanket with an intention to assist the automated sleep analysis. Several other techniques presented in [14-17] are focussed at estimating the sleeping pose of humans when covered by a blanket.

Liao and Li[18] proposed a MHI (Motion History Image) based method to generate *video actigraph*. A MHI is generated by overlaying a series of time-stamped motion templates whose intensity decays over time (the later the event, the higher the intensity). The orientation of gradients in MHI gives the direction of motion of a particular sub-region. Neighbouring local gradient orientations are combined to form a global orientation which indicates the overall trend of the movement. However the *video actigraph* generated by this method is less accurate as the gradient of large motions cannot be found.

A video based actigraph generation algorithm based on frame differencing and motion estimation is presented in[19] [20]. The *video actigraph* generated by this method has been validated by comparing it with *wrist actigraph* generated by a reference accelerometer (Actiwatch). The proposed method has been found to be robust to movements originating from under the blanket, to movements originating from body parts other than the wrist, to various sleeping positions and different illumination conditions.

1.2 Problem description and challenges

This thesis project is aimed at developing a sleep monitoring application that can monitor the movements of person of interest (POI) from a camera fitted with an IR emitter (for illumination in the dark). The application needs to distinguish the movements between the monitored person (POI) and the bed partner (BP). The output of the application will be a *video actigraph* that corresponds to the movements originating from the POI. The *video actigraph* is generated by computing the sum of absolute differences of all pixels in two consequent low-pass filtered images (frames).

To make it easier for the end user to install the application, the camera is placed on the bed side table. The placement of the camera on bed side table averts the need for cumbersome installation as in case of a ceiling mounted camera. However, the view angle of the camera makes it challenging to distinguish movements between BP and POI as they appear to be in close proximity of each other.

The following steps need to be accomplished to realize the application.

- Data acquisition and standardizing the recording protocol. This is important to replicate the physical setup (equipment installation – Section 3.1) for natural (home) and synthetic (lab) data acquisition.
- Developing an algorithm that can
  - ❖ Segment POI area from BP.
  - ❖ Generate *video actigraph* from POI movements.

- Validating the algorithm over diverse data sets.
- Hardware realization to demonstrate the feasibility of the application as a consumer product.

1.3 Video actigraph

In this section, the generation and comparison of *video actigraph* with *wrist actigraph* is presented. The *wrist actigraph* data is registered using a Texas Instruments Chronos [21] watch. The Chronos watch has an epoch length of 66 milliseconds, as opposed to the *video actigraph* which has an epoch length of 100 milliseconds (10 frames per second). Details about the physical setup and the guidelines for installation are presented in Section 3.1.

The *video actigraph* is the Sum of Absolute Differences (SAD) of all pixels between two consequent frames. The signal of the *video actigraph* is sensitive to image noise resulting in a baseline shift as shown in Figure 1.

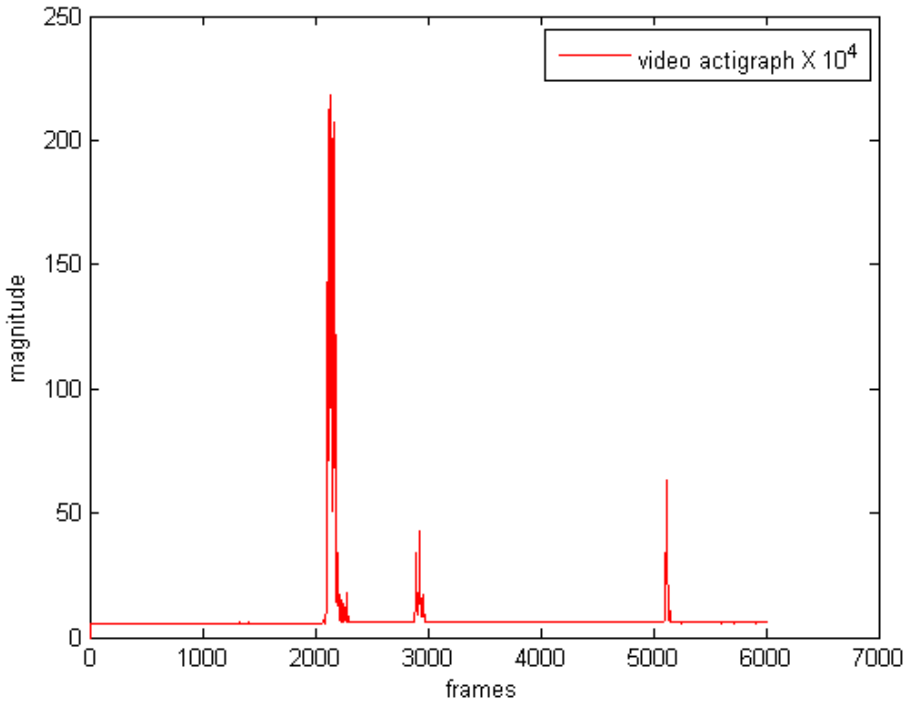


Figure 1 : *Video actigraph* generated by frame differencing. The baseline shift indicates DC component of the signal due to image noise.

Since the cameras operate in a near 0-Lux environment, the Signal to Noise Ratio (SNR) is lower. Image noise is due to electronic noise (like sensor noise, analogue to digital conversion error and quantization error) in the camera. The image noise is quite small when compared to body motions and is usually present throughout the actigraph signal. To filter such noise, a moving minimum filter is used. The impulse response of the filter is formally expressed by Equation 1.1 where $x[n]$ is the SAD sample value of n^{th} frame

and $y[n]$ is the filtered sample value of n^{th} frame. The filtered *video actigraph* is shown in Figure 2.

$$y[n] = x[n] - \min(x[n], x[n-1], \ldots x[n-N]) \qquad \textbf{Equation 1.1}$$

The filter order N is given by Equation 1.2, where TMM is the time in seconds, in which the longest body movement can be contained in. In 20 hours of video sequences observed, the longest duration of continuous motion was never greater than 60 seconds. Hence, TMM has been determined to be 60 Seconds. FR is the frame rate of the camera (10 frames per second).

$$N = \text{TMM} * \text{FR} \qquad \textbf{Equation 1.2}$$

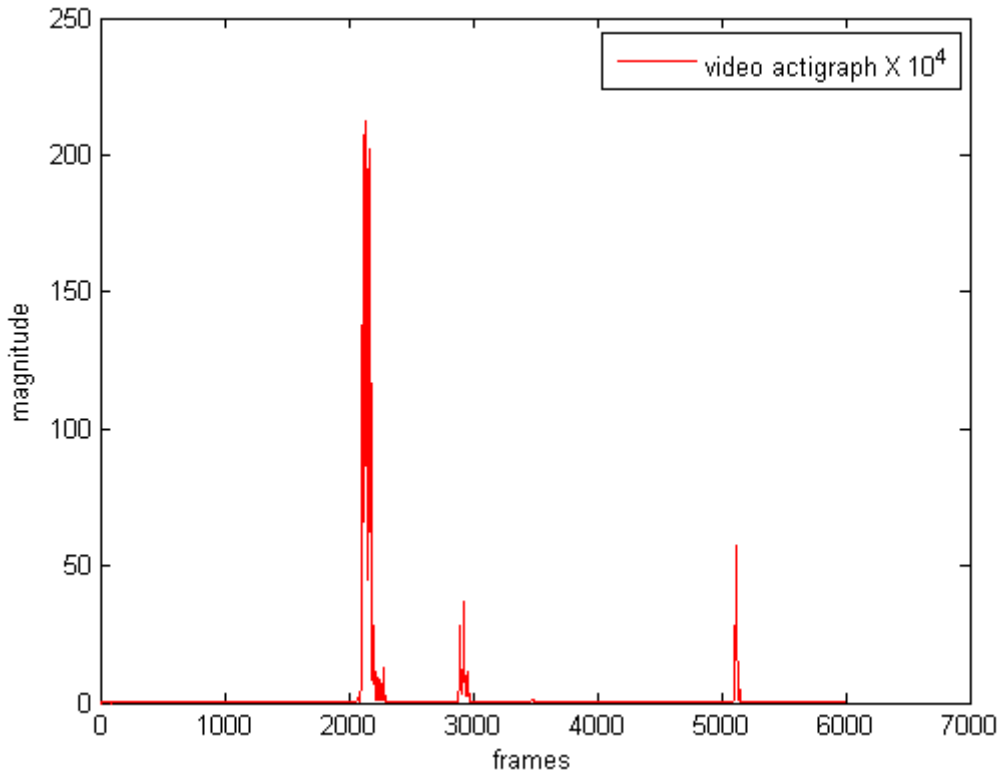


Figure 2 : *Video actigraph* after filtering image noise. The noise in the signal is filtered by using moving minimum filter. The signal no longer has a baseline shift (DC noise).

The comparison of *wrist actigraph* (blue) with *video actigraph* (red) is shown in Figure 3. Note that the *wrist actigraph* only corresponds to the POI and not the BP.

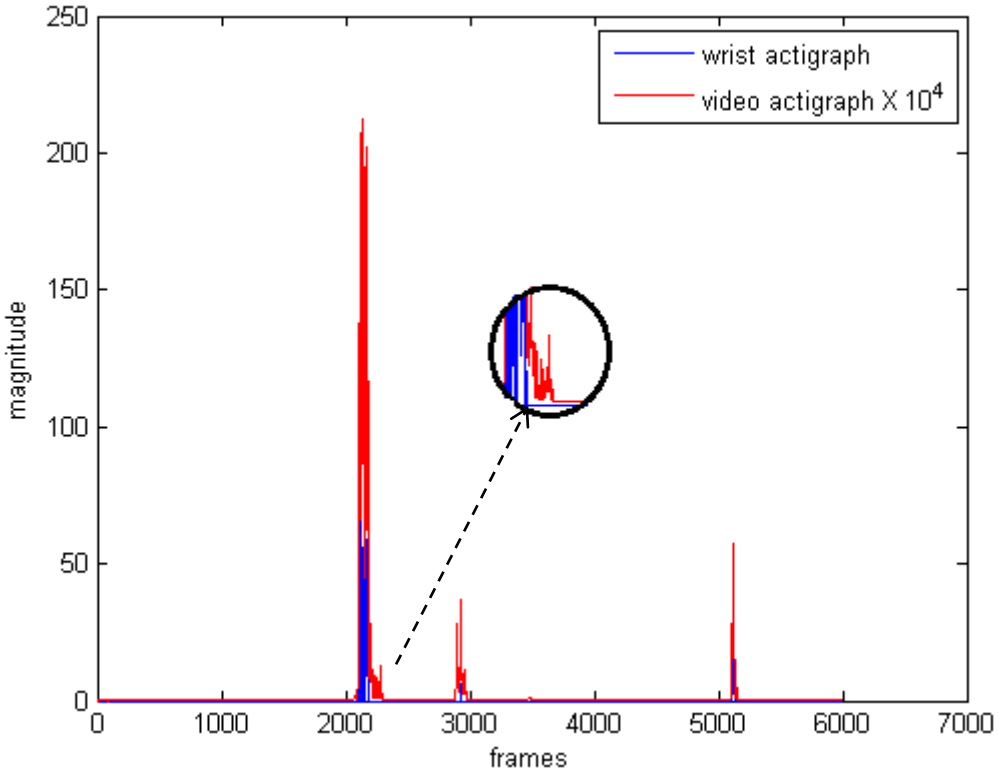


Figure 3 : Comparison of *wrist actigraph* with *video actigraph*. The circle shows the additional peaks in *video actigraph* signal. The additional peaks can be either due to movement from BP or POI body parts excluding the wrist.

It can be observed that for every peak in the *wrist actigraph*, there is corresponding signal activity in the *video actigraph* which supports our claim the *video actigraph* is a possible alternative to *wrist actigraph*. The additional peaks (highlighted in the circle in Figure 3) in the *video actigraph* (in red) that do not have a corresponding *wrist actigraph* signal (in blue) are due to the movements originating from parts of POI excluding the wrist and from BP movements.

We believe that by segmenting POI area from the image and by calculating the SAD of pixels in this area, the *video actigraph* would have signals that only correlate to POI movements. Figure 4 gives the overview of the application.

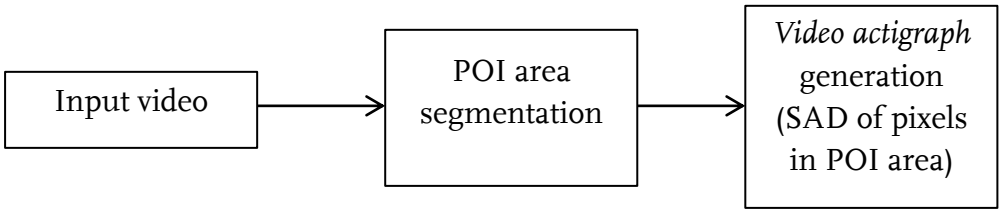


Figure 4: Application overview. The *video actigraph* is generated by calculating the SAD of pixels in POI area.

1.4 Report organization

This thesis report describes the design and development of a video based sleep monitoring application. Chapter 2 details the solution approach for realizing the video based sleep monitoring application. It explains about the various features chosen and the way they are used to develop a robust segmentation algorithm. In Chapter 3 the experimental setup for data acquisition is presented. This chapter also discusses the implementation steps and the tools used for the development of the application. In Chapter 4, the results and discussions of the application are presented. Chapter 5 illustrates the hardware realization of the application. In Chapter 6, conclusions are drawn along with summary and suggestions for future work.

2 Segmentation algorithm

In this chapter, the solution approach for the segmentation problem is presented. In Sections 2.1-2.6, various discriminative features that have been selected for use in the segmentation algorithm are introduced. In Section 2.7, the pre-processing step is illustrated. The segmentation algorithm uses several discriminative (weak) features together by means of a classifier for segmenting the POI from the image. The details of the classifier are presented in Section 2.8. In Section 2.9, an algorithm to compensate for the false POI movements caused due to blanket pulling by BP is described.

2.1 Brightness feature

Several methods have been proposed in literature to estimate depth from the brightness information in an image which can be used for POI area segmentation. One such method, to recover depth from a brightness image was proposed by Bichsel and Pentland [22] based on Oliensis and Dupuis algorithm [23-25]. The assumptions made here are that the surface is Lambertian and continuous. This method also requires the illumination plane (light source) to be perpendicular to that of the observing plane (camera). The algorithm tries to estimate the height of the surface by measuring the brightness variation (slope) in the direction of maximum descent (direction in which the brightness reduces faster, i.e. largest derivative).

In the current project, a single illumination (infrared) source, placed along with the camera, is used and the illumination of the scene is not homogeneous, hence the assumptions for the depth estimation algorithm discussed above do not hold. However, it has been observed that due to the scene being illuminated from a single source, the intensity (value of pixels) decreases from the light source to other areas in the scene. Objects closer to the light source tend to be brighter than the ones farther away.

As a result of the setup (explained in detail in Section 3.1), POI is closer to the light source. Hence, POI appears brighter than BP. This variation of illumination between POI to BP can be used to detect the POI area on a coarse scale. The POI area estimation can be improved by histogram equalization of the image which improves the global contrast in the image. By improving the global contrast, the pixel distribution would be spread throughout the intensity scale (0-255), thus aiding in improved POI area estimation.

In the rest of this section, the algorithm formulated to get the coarse estimate of POI area using brightness feature is presented. The algorithm steps include histogram equalization and POI area estimation.

2.1.1 Histogram equalization

For better estimation of POI area using brightness, the histogram is stretched to normalize the pixel distribution on the intensity scale. This would improve the global contrast in the image, which improves POI area estimation and generalizes the algorithm to video sequences recorded in varying illumination conditions.

Since the video recordings are taken in low illumination, they often have noise in them. Histogram stretching boosts the noise along with the pixel values. The limits to which the histogram can be stretched are found by estimating the noise in the image. The noise (S_i) is estimated by measuring the standard deviation (Equation 2.1) of pixels in the brightest patch (10X10- pixel window) of the image.

$$S_i = \sigma = \sqrt{\frac{1}{N}\left(\sum_{x=1}^{10}\sum_{y=1}^{10}\left(f_{x,y} - \overline{f}\right)^2\right)} \qquad \textbf{Equation 2.1}$$

Where

$$\overline{f} = \frac{1}{N}\left(\sum_{x=1}^{10}\sum_{y=1}^{10}\left(f_{x,y}\right)\right) \qquad \textbf{Equation 2.2}$$

$f_{x,y}$ represents the pixel value in the 10X10 brightest window in the image and σ is the standard deviation of all the pixels in the window.

A sample (original) image is shown in Figure 5. The distribution of intensity values in the image is shown in Figure 6. It can be seen that pixel distribution is concentrated over a part of the intensity scale.

Figure 5: Original image. The brightness decay from POI to BP can be observed.

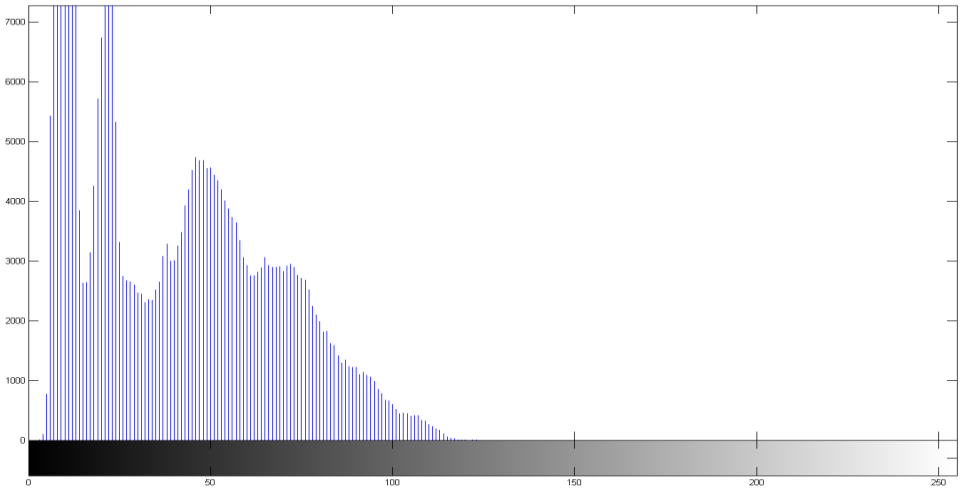


Figure 6: Histogram of a (noisy) image. The pixel distribution is limited to half the intensity scale and remains zero thereafter.

The histogram stretch limits of the original image are extended by assigning the same intensity value to the brightest *Noise Compensation pixel Percentage* (NCP) (Equation 2.3) of pixels as shown in Figure 7. The intensity value assigned is the lowest bin (intensity) value of all the pixels that fall in the brightest NCP of the image. NCP is dependent on the estimated noise in the image and is given by Equation 2.3.

$$\text{NCP} = \frac{S_i}{255}\ X\ 100$$

Equation 2.3

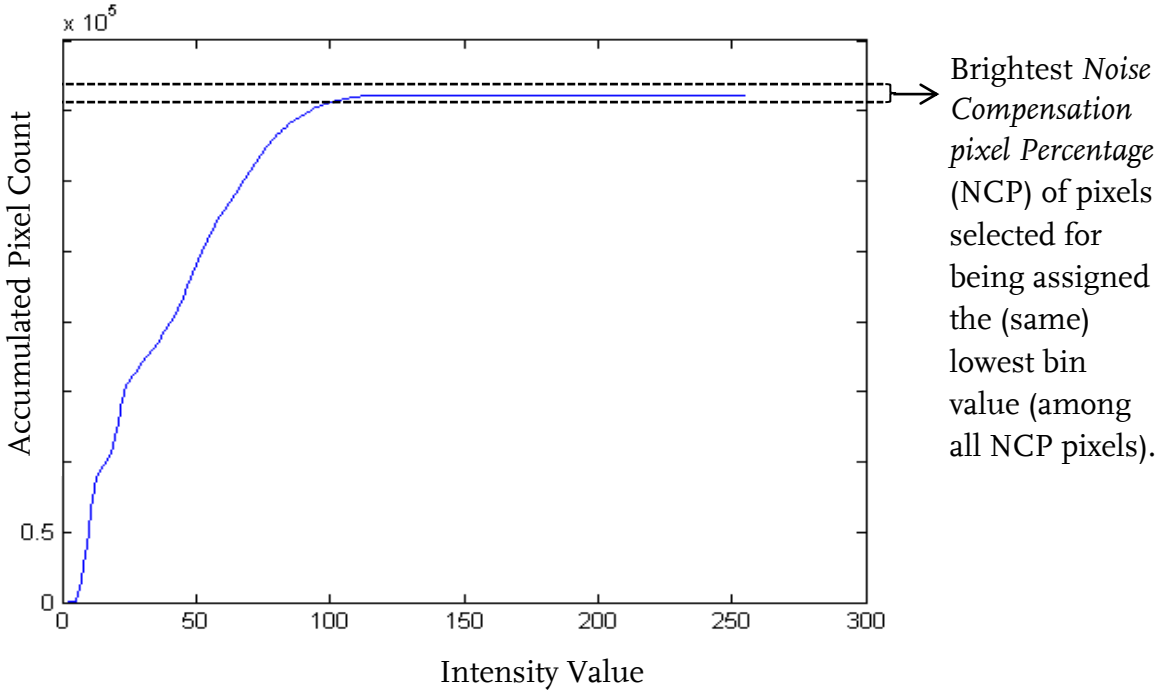


Figure 7: The accumulated histogram of a (noisy) image. The brightest N*oise Compensation pixel Percentage* (NCP) of pixels merged into a single bin.

The histogram equalization process is explained below.

Let O_u and N_u be the histogram bins of original image after merging the top NCP pixels into a single bin and histogram equalized image respectively where u represents the intensity values [0-255]. The histogram of noise compensated original image is expressed by Equation 2.4. The new bin value of the histogram equalized image is given by Equation 2.5.

$$O_u = \sum_{x=1}^{h} \sum_{y=1}^{w} \delta(c(f(x,y)) - u) \qquad \textbf{Equation 2.4}$$

$f(x,y)$ represents the intensity value of the pixel in the image where x represents the row number and y represents the column number. $\delta(X)$ is an impulse function such that $\delta(X) = 1$ if $X = 0$. $c: f(x,y) \rightarrow 1 \ldots u$; is a function that associates each pixel value to the histogram bin in to which it falls.

$$N_u = \text{NMIN} + (u * \text{factor}) \qquad \textbf{Equation 2.5}$$

$$\text{factor} = \frac{\text{NMAX} - \text{NMIN}}{\text{OMAX} - \text{OMIN}} \qquad \textbf{Equation 2.6}$$

OMIN and OMAX are the minimum and maximum intensity values in the noise corrected original image. NMIN and NMAX are the minimum and maximum intensity values in the intensity scale i.e., 0 and 255 respectively. The histogram equalized image and the pixel value distribution of it are shown in Figure 8 and Figure 9 respectively.

Figure 8: Histogram equalized image. An improvement of global contrast in the image can be observed.

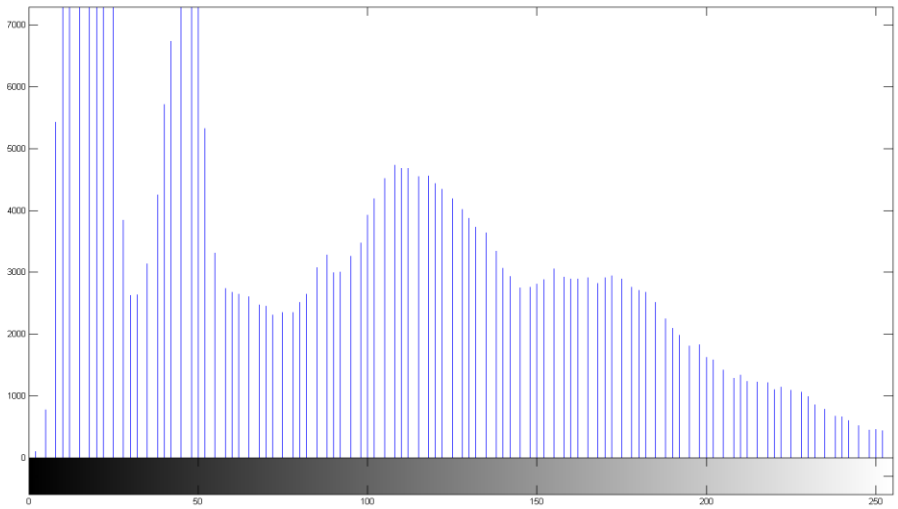


Figure 9 : Pixel distribution after histogram equalization

2.1.2 POI area estimation

After noise correction and histogram equalization, the pixel value distribution is spread across the full intensity scale as shown in Figure 9. As discussed in Section 2.1, the intensity value decreases from POI towards the BP. Based on this observation the POI area is coarsely estimated to be the brightest 50% of the pixels in the image as shown in Figure 10. This percentage has been determined by statistical analysis and the ROC curve is shown in Figure 11. The ROC curve has been plotted after calculating the TPR and FPR by comparing the ground truth (manually annotated binary image, 'True' label for pixels belonging to POI and 'False' label to all other pixels) with the output generated by using the brightness feature (all pixels that fall in the brightest x% of pixels in the image are labelled 'Positive' (pixels assigned to POI area) and the rest negative (pixels assigned to BP and background). From the ROC curve it can be seen that after 50%, the gain in TPR is very small compared to the increase in FPR, hence the brightness percentage threshold has been determined to be 50%.

The brightest 50% of the pixels are assigned a positive label (POI area) and the rest given a negative label. The result of the POI area estimation from brightness variation is shown in Figure 12. The resultant POI estimation is a binary image (label '1' for POI area and label '0' for rest of image) and these labels are used as Brightness Feature.

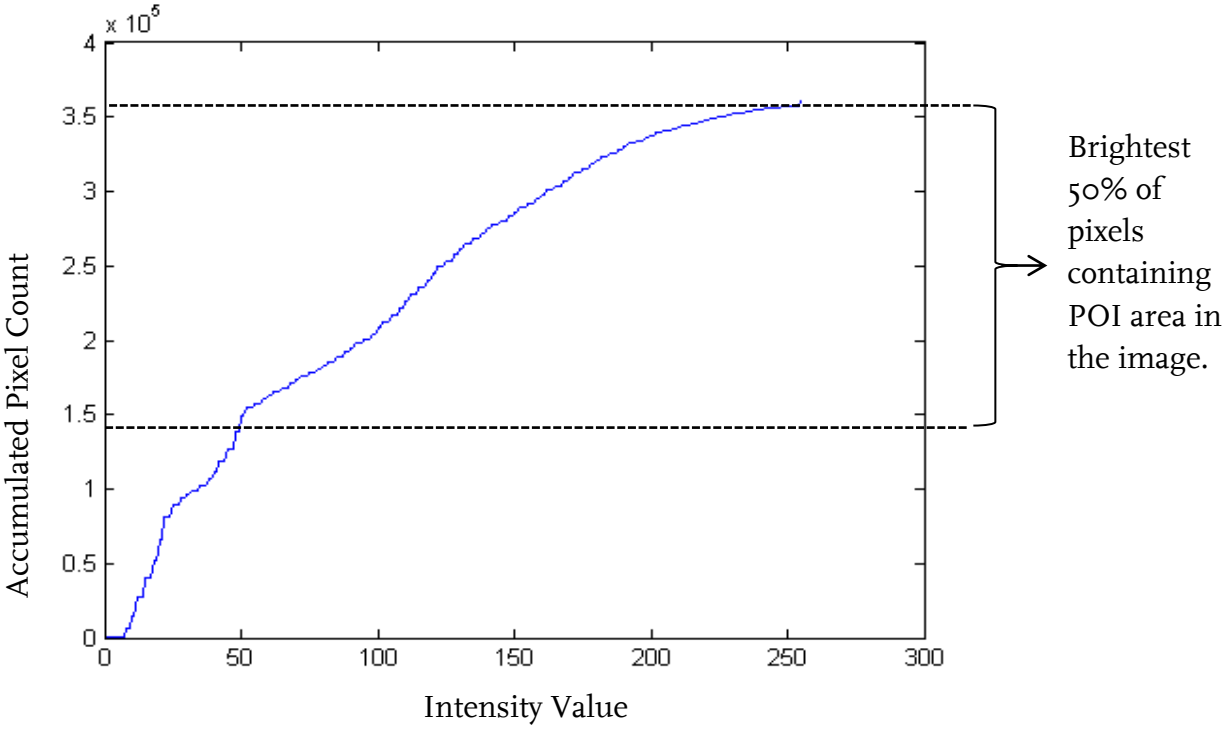


Figure 10: Accumulated histogram. POI area is estimated to be the brightest 50% of pixels in the image.

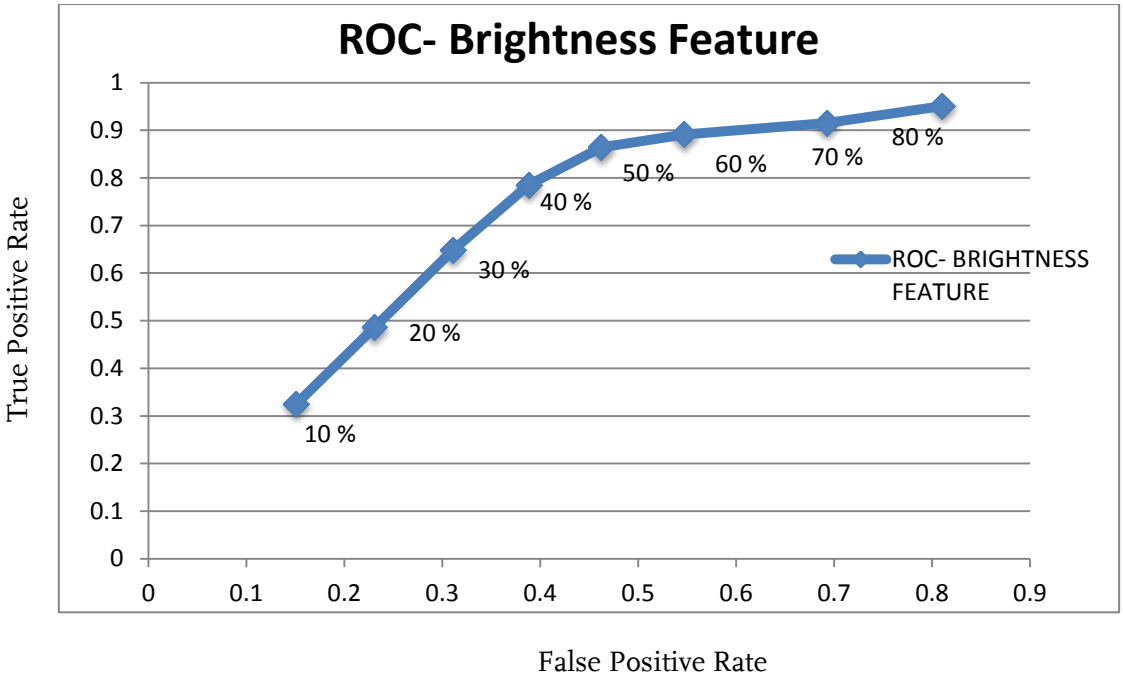


Figure 11: ROC curve showing the effect of percentage of brightest pixels chosen on TPR (True Positive Rate) and FPR (False Positive Rate). After 50% the gain in TPR is very small compared to the increase in FPR.

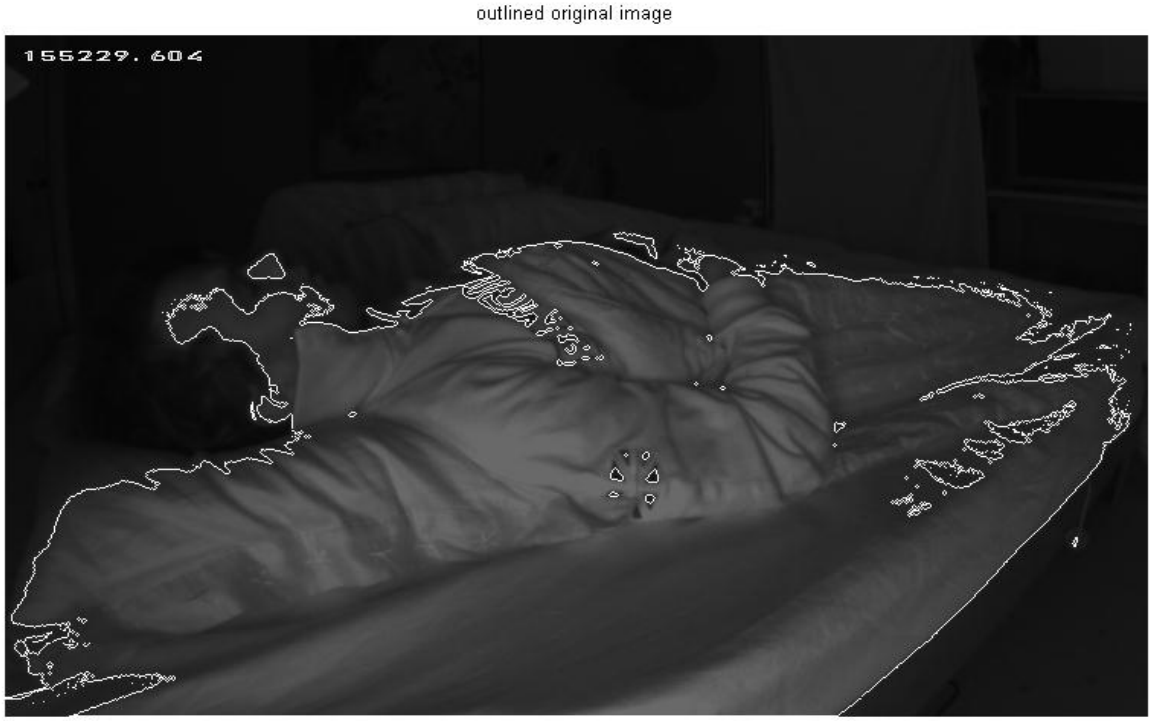


Figure 12: The result of POI estimation using Brightness Feature. The outline (white lines in the image) shows the estimated POI area.

2.2 Focus feature

In cameras, the lens can often be adjusted such that some parts in the image are focussed compared to the rest of the image parts. As a result of this, there would be parts in a focussed image which are focussed and parts which are blurred. The measure of defocus (blur) of parts in the image can be used as a feature to segment focussed objects or defocussed objects. One such segmentation algorithm based on focus feature is presented by Tsai and Wang [26]. They propose an unsupervised edge-based segmentation method which uses the measure of blur (defocus) at object edges in an image. The amount of defocus is estimated using the moment-preserving principle which is discussed in detail in the later parts of this section.

In this project, the camera is placed close to the POI on the bed side table and is adjusted such that the POI area is in focus and the background is not.

2.2.1 Defocus measure based segmentation algorithm

As discussed above, the POI area in the image closer to the lens is focussed, whereas the background area including BP is progressively blurred depending on their distances from the lens. By measuring the amount of defocus (blur) at each edge pixel in the observed image, the boundary of the focussed object can be estimated.

In typical blur estimation algorithms like in [27, 28], the blurred edge is generally modelled by convolving a focused image with a point spread function that is assumed to be Gaussian distribution with parameter σ. The parameter σ can be approximated by an iterative nonlinear search technique[27] and is used as a measure of defocus.

Tsai and Wang in [26] have proposed a new approach to model blur (measure defocus) using a moment-preserving technique. The observed image is converted into a gradient image by convolving the original image with the Sobel edge operator. In the gradient image, the edge pixels have large gradient magnitude and non-edge pixels have approximately zero gradient magnitude. For each edge point of interest, the proportion of the edge region p_e (region with high gradient magnitude) with respect to a small neighbourhood window in the gradient image is computed using the moment-preserving principle. A focused edge will result in a small p_e, whereas a defocused edge will yield a large p_e. p_e is the measure of defocus and increases as the relative distance between the imaged point and surface of exact focus increases. The algorithm is formally expressed in Equations 2.7 - 2.20 [26].

Let $f(x, y)$ be the grey-level of a pixel at (x, y) in the observed image. The gradient $\nabla f(x, y)$ image is given by Equation 2.7.

$$\nabla f(x, y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \qquad \textbf{Equation 2.7}$$

where

$$g_x = \sum_{j=-1}^{1} \sum_{i=-1}^{1} f(x+i, y+j) w_x(i,j) \qquad \textbf{Equation 2.8}$$

$$g_y = \sum_{j=-1}^{1} \sum_{i=-1}^{1} f(x+i, y+j) w_y(i,j) \qquad \textbf{Equation 2.9}$$

$w_x(i,j)$ and $w_y(i,j)$ are the 3X3 horizontal and vertical Sobel edge operators. The magnitude of the gradient is defined by Equation 2.10.

$$g_{(x,y)} = |\nabla f(x,y)| = \sqrt[2]{g_x^2 + g_y^2} \qquad \textbf{Equation 2.10}$$

An ideal gradient image would have two homogeneous regions, the bright (high gradient values) region with a uniform gradient magnitude h_e (focused region), and the dark region (low gradient values) with uniform gradient magnitude h_b. The proportions of bright region and the dark region in the gradient image can be represented by p_e(sharp area of focused image) and p_b (scattered area of defocused image) respectively. Then,

$$h_e > h_b$$

$$0 \leq p_e \leq p_b$$

$$p_e + p_b = 1$$

For a given edge point at (x,y), the first three moments real world gradient image $g(x,y)$ are given by Equation 2.11.

$$m_j = \frac{1}{n} \sum_{(s,t) \in N(x,y)} [g(s,t)]^j, \qquad j = 1,2,3 \qquad \textbf{Equation 2.11}$$

$N(x,y)$ is the neighbourhood window that consists of neighbouring points around (x,y), and n is the total number of pixels in the window. The window size is chosen to be 21X21.

By preserving the first three moments (as described in [26]) in both real-world gradient image $g(x,y)$ and the ideal gradient image, the following four equations are obtained

$$p_e h_e^1 + p_b h_b^1 = m_1, \qquad \textbf{Equation 2.12}$$

$$p_e h_e^2 + p_b h_b^2 = m_2, \quad \textbf{Equation 2.13}$$

$$p_e h_e^3 + p_b h_b^3 = m_3, \quad \textbf{Equation 2.14}$$

$$p_e + p_b = 1 \quad \textbf{Equation 2.15}$$

There exists a closed-form solution for the four unknown variables p_e, p_b, h_e and h_b, which are given by (Tsai [29])

$$h_b = \frac{1}{2}\left[-c_1 - \sqrt[2]{(c_1^2 - 4c_0)}\right], \quad \textbf{Equation 2.16}$$

$$h_e = \frac{1}{2}\left[-c_1 + \sqrt[2]{(c_1^2 - 4c_0)}\right], \quad \textbf{Equation 2.17}$$

$$p_b = \begin{vmatrix} 1 & 1 \\ m_1 & h_e \end{vmatrix} \Big/ \begin{vmatrix} 1 & 1 \\ h_b & h_e \end{vmatrix} \quad \textbf{Equation 2.18}$$

Where

$$c_0 = \begin{vmatrix} -m_2 & m_1 \\ -m_3 & m_2 \end{vmatrix} \Big/ (m_2 - m_1^2)', \quad \textbf{Equation 2.19}$$

$$c_1 = \begin{vmatrix} 1 & -m_2 \\ m_1 & -m_3 \end{vmatrix} \Big/ (m_2 - m_1^2)', \quad \textbf{Equation 2.20}$$

And

$$p_e = 1 - p_b \quad \textbf{Equation 2.21}$$

The value of p_e given by Equation 2.21, gives the proportion of edge region in the neighbourhood window and is the measure of blur. The value of p_e, $0 \leq p_e \leq 1$ increases as the amount of defocus increases. By applying a threshold to the p_e values, most blurred edges in the foreground and background are eliminated. The value of threshold chosen is 60 for scaled p_e values. The scaling factor is chosen to be 255, which is the number of possible intensity levels in a grey scale image. The edges with scaled p_e values less than the threshold are the focused edges.

2.2.2 Edge linking

The method described in Section 2.2.1 calculates the focus measure at the edge point and thresholds it to get the edge points of focussed objects. However the edge detection techniques seldom characterize a boundary completely due to noise, non-uniform illumination and other factors like texture, that can introduce spurious intensity discontinuities[30]. Due to this, the resulting focused edge does not describe the complete boundary of the focused object and hence an edge linking process is performed to link discontinuities in the resultant focus edge and to have a closed boundary that describe the focused object.

The edge linking process has three steps, dilation, thinning and line-linking. The dilation is performed on the thresholded $p_e$image with 5X2 (block) structuring element for two iterations. This ensures that small gaps in the boundaries are closed before the edge linking. The block (5X2) is chosen to have larger horizontal dimension as POI lies in horizontal direction on the bed and edges in this direction are to be connected. Choosing a block with larger vertical dimension would risk connecting the remaining edges of BP that still remain after thresholding. Following the dilation process, edge thinning is performed so that the width of edges is 1-pixel wide.

To ensure that focused objects have closed boundaries, a line-linking procedure is performed. The line-linking process finds the endpoints of the thinned edge lines and connects them to their nearest endpoint of another edge.

Let (x_o, y_o), (x_d, y_d) be two end points of the thinned edges, the point (x_o, y_o) is connected to point (x_d, y_d) as follows:

For $\theta = 360^0$ *to* 180^0 $(Lower\ half\ of\ the\ circle)$
  For $r = 1$ *to* 10 $(radius\ of\ circle)$
    If there is a focused edge point (x_d, y_d) in the search area
      Connect (x_o, y_o) with (x_d, y_d) through the shortest path.
      Terminate Search
    End
  End
End

The link process tries to find the edge point (x_d, y_d) in the search direction 360 - 180 and in the search radius 1 -10 (pixels) and connects it to (x_o, y_o) through the shortest path. The search direction has been chosen as 360^0 *to* 180^0 (lower half of the circle) to make the line linking operation faster. Since the pixel access in the image is from top-left to bottom-right which is in line with the search direction (360^0 *to* 180^0), the line linking function would need to wait only for few pixels before it starts the linking process. This makes it faster compared to choosing the search direction as 0^0 *to* 180^0 (upper half of the circle) which would need all pixels in the circle radius to be accessed before starting the linking process. The line-linking search order is shown in Figure 13. It has to be noted that during implementation, the circle has been abstracted to a square and the angles to rows in the square.

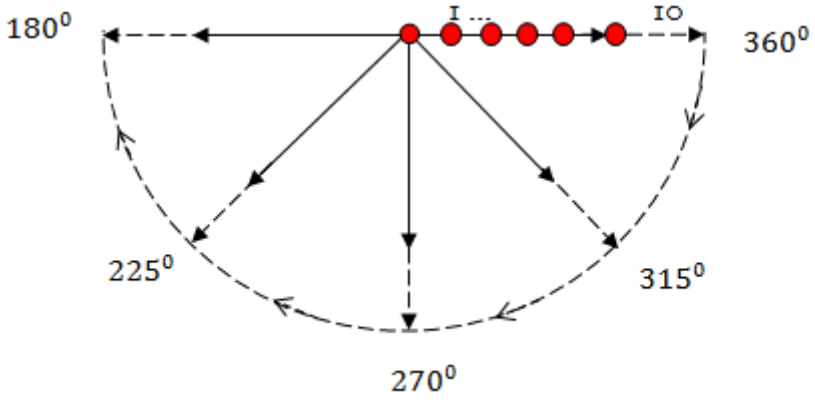


Figure 13: Line-Linking search process. The red dot represents(x_d, y_d). (x_o, y_o) can be found anywhere within the search radius of the semicircle.

The closed boundaries are filled using simple region filling (flood-fill) techniques and the largest closed region is taken as the segment containing focused object. The intermediate output with all closed regions is shown in Figure 14. The resulting POI segmentation of the algorithm is shown in Figure 15. The resultant POI segmentation is a binary image (label '1' for POI area and label '0' for rest of image) and these labels are used as Focus Feature. It has to be noted that the line-linking process would not always guarantee a closed boundary and sometimes may link the remaining edges (after thresholding) from BP with POI resulting in erroneous POI segmentation.

Figure 14: Intermediate output with all closed regions as a result of region filling.

Figure 15: The (final) POI area segmentation result using defocus measure and edge linking. The region inside the boundary marked by white lines show the segmented POI region.

2.2.3 Defocus measure value as input to classifier

The segmentation algorithm based on defocus measure is highly dependent on the edge detection techniques and often fails to determine the complete boundary of the objects due to noise and non-uniform illumination of the scene as shown in Figure 16.

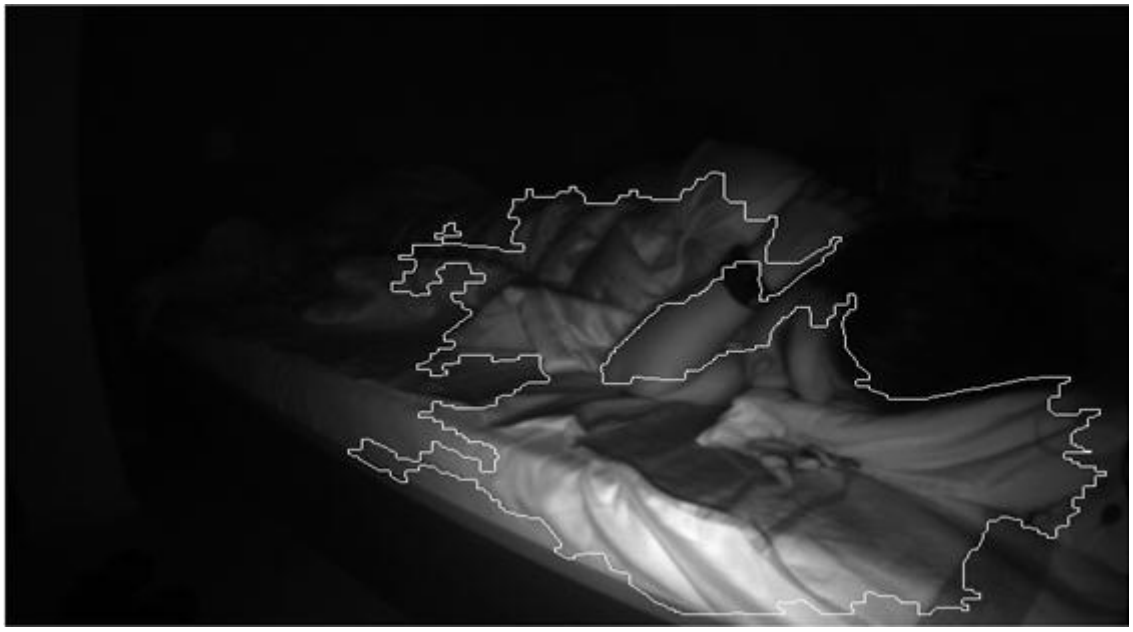


Figure 16: Erroneous output of focus based segmentation algorithm. The complete boundary of POI is not determined.

However, just the measure of defocus can serve as a weak discriminative feature that can be used to distinguish POI area from BP area. As such, the defocus measure is calculated as described in Section 2.2.1 at every pixel in the image and is given as an input to the classifier.

2.3 Region based motion feature

In the current project, the presence of stationary background in the scene suggests it suitable to use region-based segmentation methods (based on motion) to segment POI area from the image. The region-based methods (based on motion) define a group of connected pixels as belonging to single object that is moving with a different motion from its neighbouring region. One such region based method based on motion was proposed by Badenas *et al.*[31] in which a group of connected pixels with similar feature (motion vector) values are assumed to be belonging to a single object. The image is then segmented into regions based on grey scale values and motion information. The regions are tracked over multiple frames to improve the segmentation result.

In this project, we use motion estimation to find motion vectors based on best matching blocks in subsequent frames. Motion information is usually represented by a motion vector which indicates the displacement of a block between subsequent frames. Over the years, several robust motion estimation techniques have been developed which produce reliable motion vectors efficiently[32]. The motion vectors generated are robust to illumination changes and describe the true motion of the object.

Once motion has been estimated for a region, this information can be used to separate it from other regions which are stationary or move differently. In the rest of this section the approach to use motion vectors to distinguish movements of POI and BP is presented.

2.3.1 Motion vector generation

A modified version [20] of the block based motion estimation technique called 3DRS (3D Recursive Search)[32] is used in the current project. In block based motion estimation techniques, a displacement vector $\vec{X}$ is assigned to the centre of a block of pixels $B(\overrightarrow{X})$ in the current frame n by searching for a similar block within a search area $SA(\overrightarrow{X})$ around the same centre $\vec{X}$, but in previous frame $n-1$. The matching block in the previous frame would have a centre that is shifted by a displacement vector $\vec{D}(\vec{X}, n)$ with respect to the centre $\vec{X}$. The displacement vector $\vec{D}(\vec{X}, n)$ which gives the displacement of block $B(\overrightarrow{X})$ in the current frame n with respect to previous frame $n-1$ is the motion vector of the block.

To find the displacement vector $\vec{D}(\vec{X}, n)$, a number of candidate vectors $\vec{C}$ are evaluated by an error measure $\varepsilon(\overrightarrow{C}, \vec{X}, n)$ (Equation 2.22) to find the best block match. The candidate vector with the lowest error measure is taken as the best block match. In exhaustive block matching methods, all the candidate vectors in the search area evaluated for finding the best block match which is computationally intensive.

In 3DRS the candidate set is limited to the 3-D neighbourhood of the current block and is based on the idea that for objects larger than block size and with inertia, the best motion vector candidates can be found in the spatio-temporal neighbourhood.

$$\varepsilon(\overrightarrow{C}, \vec{X}, n) = \mathrm{SAD}(\overrightarrow{C}, \vec{X}, n) = \sum_{x \in B(x)} \left| F(\vec{x}, n) - F(\vec{x} - \vec{C}, n - p) \right| \qquad \textbf{Equation 2.22}$$

Where $\vec{x}$ is the pixel position in block $B(\vec{x})$ and p is a variable which usually takes the value of 1 if the signal is interlaced or 2 if not. 3DRS has been shown to outperforms other motion algorithms in M2SE (Mean Squared Prediction Error) evaluation and also to have a low operation count reflected in a very efficient implementation[33].

2.3.2 Motion vector filtering

The motion estimation algorithm may cause a propagation problem in the homogenous background, as a result of which motion vectors may appear in parts of the image where there is no motion. Background propagation elimination is critical since otherwise the region based algorithm would falsely classify the background as part of (region) a moving person. The output of the motion estimation with propagation is shown in Figure 17.

Figure 17: Motion vector propagation to background. The colours indicate the strength and direction of motion vectors. The motion vectors (purple and green) can be seen propagating to background where there is no motion.

The overview of the propagation elimination process is shown in Figure 18. Based on analysis of motion vector propagation in the background, the propagation can be classified into two types:

- First type of propagation (Type 1) exists in every frame, irrespective of the presence of true motion in the frame. This type of propagation is characterized by presence in large area, has small (magnitude) value and lasts long (duration), seen in green colour in Figure 17.
- The second type (Type 2) exists in the background around the moving person. This type of propagation is present in small area, has greater value and lasts shorter than the first type, seen in purple colour close to POI's head.

To eliminate Type 1 propagation, the luminance (SAD-Sum of Absolute Difference) difference of pixels within the same blocks in two consecutive frames is used. The motivation is that, when there is motion, the blocks are displaced resulting in change of pixel (intensity) values within the block in consecutive frames. The SAD threshold value is set to 64 and has been empirically determined.

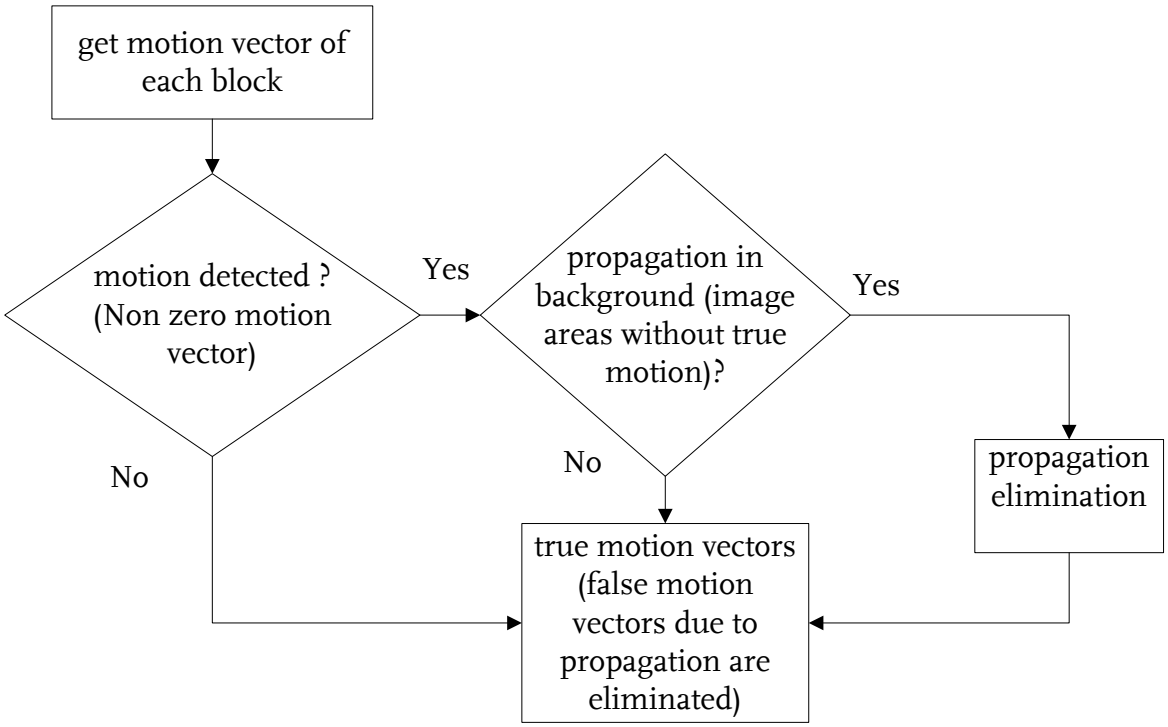


Figure 18: Overview background propagation elimination

$$\text{SAD} = \sum_{x=1}^{\text{bh}} \sum_{y=1}^{\text{bw}} \left| f_c(x,y) - f_p(x,y) \right| \qquad \textbf{Equation 2.23}$$

$$\text{If SAD} < 64, \text{MV(B)} \; is\ propagation\ in\ background$$

$f_c(x,y)$ and $f_p(x,y)$ are the pixel values in the current and previous frames respectively. bh (8) and bw (8) are the block height and width respectively. MV(B) is the motion vector of block B.

From observation, it has been noted that Type 2 propagation mostly lasts for 1 or 2 frames. The presence of motion of the current and last two frames of each block is stored as a binary value. Only if the motion exists for at least 3 frames, it can be determined as true motion, or it will be eliminated as propagation. Motion vectors generated after eliminating propagation in the background is shown in Figure 19.

Figure 19: Motion vectors after propagation elimination. The detected motion vectors specify the true motion with higher accuracy.

2.3.3 Motion vector clustering

Blocks are too small to represent the moving object, and cluster of blocks with motion by spatial similarity (adjacent or close) would better correspond to a moving area of an object in our case POI. The cluster is defined as the smallest bounding box (motion rectangle) to include all blocks of spatial similarity (connected blocks), and the number of blocks per cluster is dependent on motion (magnitude).

Each block with motion is assigned a label which indicates the motion rectangle it belongs to, and all adjacent blocks with motion have the same label. The process of label assignment starts from top-left block to bottom right in each frame, and rules are shown below[34]. In the formula, MVL is the length of motion vector, m and n indicate the column and row number of the blocks respectively. The pseudo code of the clustering process is given below,

If MVL(m,n) = 0
    Label(m,n) = 0
If MVL(m,n) ≠ 0 and Label (m-1,n) = 0
    Label(m,n) = current_label + 1
If MVL(m,n) ≠ 0 and Label (m-1,n) = a and Label(m,n-1) = 0
    Label(m,n) = a
If MVL(m,n) ≠ 0 Label (m-1,n) = a and Label(m,n-1) = a
    Label(m,n) = a

If MVL(m,n) ≠ 0 Label (m-1,n) = a and Label(m,n-1) = b

Label all blocks with Label b with Label a and Label(m,n) = a

End

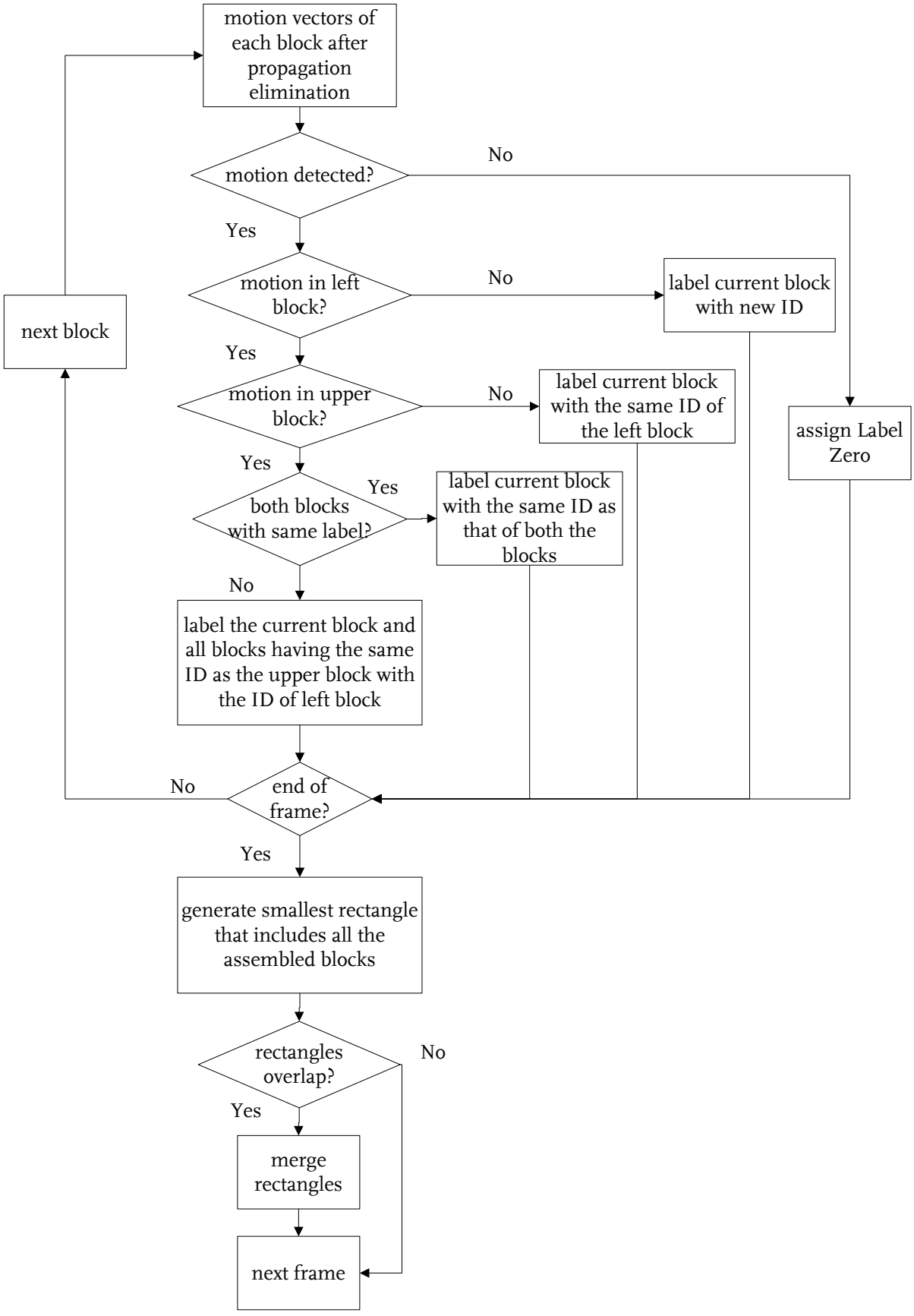


Figure 20: Overview motion rectangle generation

After the label assignment, a bounding box (motion rectangle) is created to include all blocks with the same label. If rectangles overlap, the overlapping rectangles are considered to be highly correlated and hence are merged into a bigger one. The process of motion rectangle generation is shown in the flow chart in Figure 20.

Figure 21 shows some examples of motion rectangle formulation in a video sequence. The black rectangles in the figure are the motion rectangles, which include the motion vectors (seen as blocks with colour).

Figure 21: Generated motion rectangles containing motion blocks (seen in colours).

2.3.4 Labelling of motion vectors (feature value generation)

The motion rectangles generated only specify connected motion blocks. There can be instances where there are two motion rectangles originating from the same person (POI), for example from the head and the legs. The blocks within the same motion rectangle would have similar motion (magnitude and direction) and could be represented by a single (MRV- Motion Rectangle Vector) motion vector, which has the strongest accumulated magnitude in a direction.

The MRV's of motion rectangles originating from the POI and BP would have different or at times, the same magnitude and directions. Hence, it would not be possible to assign the motion rectangles to a person (POI or BP) based on the MRV magnitude and direction. To overcome this, a spatial constraint that specifies the rough location of POI in the image is used.

The setup specifies the height of the camera from the ground, the distance of the camera from the bed (Section 3.1). Therefore it would be possible to roughly know where the POI lies in the image. Statistical analysis shows that POI area lies below row 150 of the image (image size 752X480). The resulting ROC curve of the statistical analysis is shown in Figure 22. It can be seen that above 150, the gain in TPR is small than the loss in FPR. The ROC curve has been plotted after determining the TPR and FPR by comparing the ground truth (manually annotated binary image, 'True' label for pixels belonging to POI and 'False' label to all other pixels) with the output generated by using row number (all pixels below the row number are labelled 'Positive' (pixels assigned to POI area) and the rest negative (pixels assigned to BP and background).

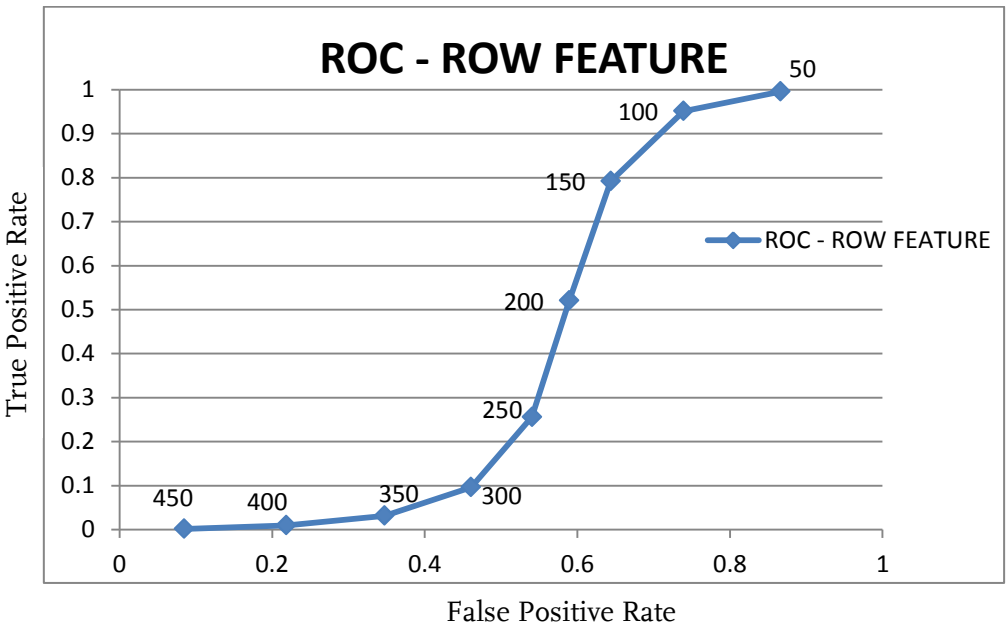


Figure 22: ROC curve showing the statistical analysis of POI location in the image using row numbers.

This spatial constraint is used to assign the motion rectangles to either POI or BP. If the spatial location of motion rectangle in the image is below or overlaps row 150 of the image, it is assigned to POI (label '1'). If the motion rectangle falls above row 150 of the image it is assigned to BP (label '2') and to background if there is no motion (label '0'). These labels are used as a feature values and are given as an input to the classifier. The assignment is shown in Figure 23 and Figure 24.

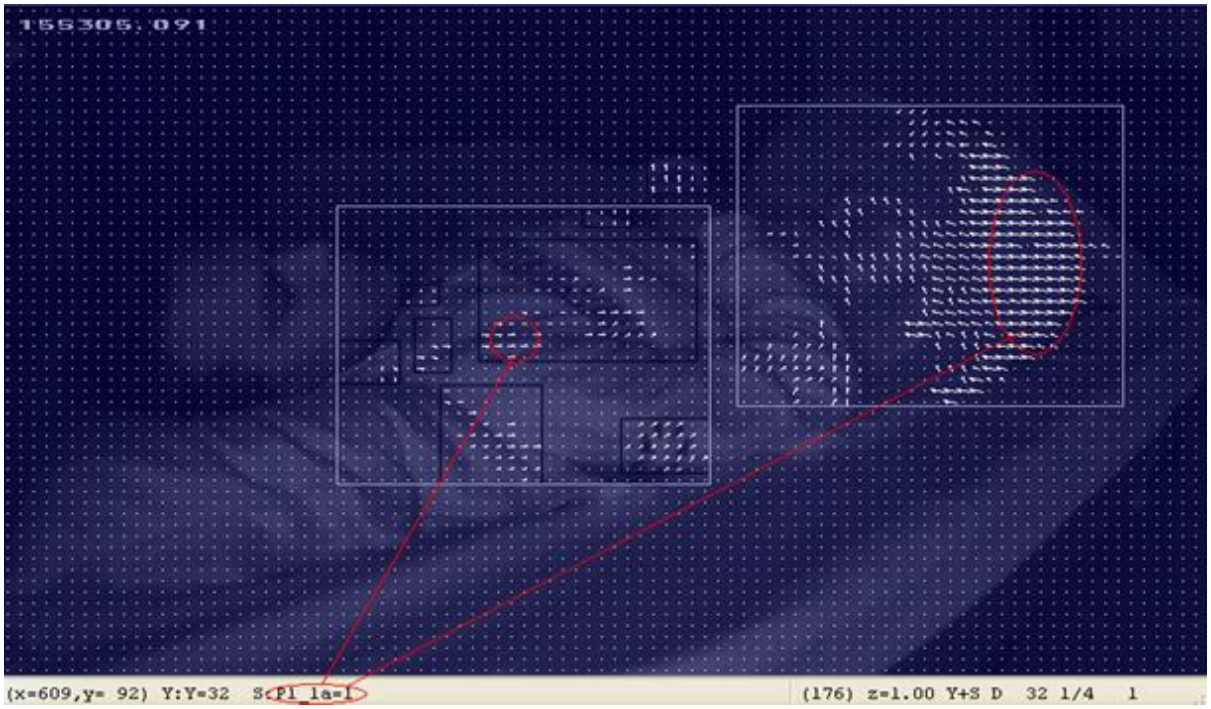


Figure 23: The assignment of motion blocks to POI and BP after adding the spatial constraint. Label 1 (seen as P1_la=1 in the image) indicates that the motion rectangles (seen as white rectangles) are assigned to POI.

Figure 24: Label 2 (seen as P1_la=2 in the image) indicates that the motion rectangles (marked in red circle) are assigned to BP.

2.4 Histogram of Oriented Gradients (HOG) feature

Histogram of Oriented Gradients (HOG) descriptors were (first) introduced by Dalal and Triggs [35]. The idea behind HOG is that the local object appearance and shape are often characterized by the distribution of local intensity gradients or edge directions, even without precise knowledge of corresponding gradients or edge positions[35].

HOG is calculated by accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels in a small window (cells). For better invariance to illumination changes, the HOG is contrast normalized by accumulating a measure of local histogram 'energy' over larger spatial regions (blocks) and using the results to normalize all the cells in the block. Histogram of Oriented Gradients (HOG) descriptors in a dense overlapping grid have been shown to outperform the best Haar wavelet based descriptors in person detection[35].

Because the scene is illuminated by a single IR light source placed on the bed side table (close to POI head), the entire scene is not illuminated uniformly. The parts of the image that are away from the IR source such as the legs of POI are weakly illuminated and the gradients are low. Edge detectors would fail in these areas as they characterize an edge in limited directions (horizontal, vertical or diagonal). However, HOG has an advantage of local contrast normalization, making it capable of capturing the edge or gradient structure that is very characteristic of a local shape, hence is more robust than the usual edge detectors. HOG can find maximum gradient direction in a local window in any direction (based on the number of bins) and thus can describe an edge more robustly. Hence, HOG bin values are used as a descriptor of the boundary between POI and BP. The HOG image is shown in Figure 25.

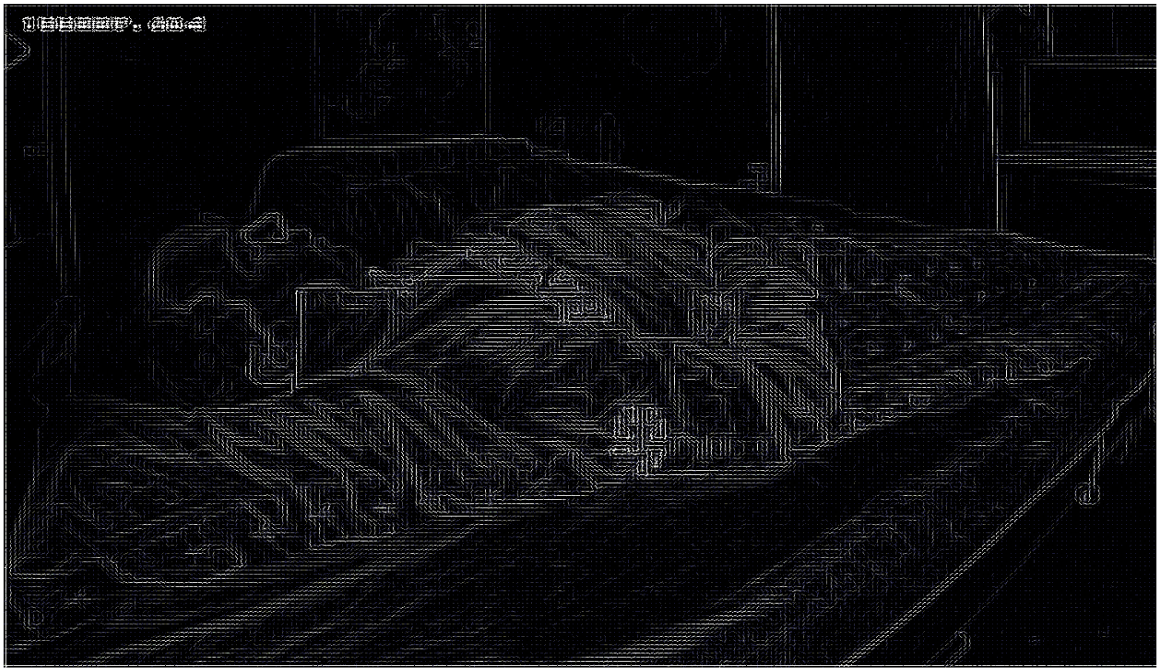


Figure 25: Image showing HOG descriptors. The edges are highlighted (described) throughout the image although the illumination is not uniform.

2.5 Location based features

Due to conditional (camera height, distance between bed and camera) setup, POI area can be assumed concentrated in the center of the image. This concentration of POI area in the image has been observed to hold in multiple data sets (video sequences). The features specifying the location of an object (POI area) in an image have great prominence in object (POI area) classification; provided the object movement is spatially constrained (POI area is always concentrated in a certain part of the image).

Based on the above discussion, the following features have been selected for use in classifier for POI segmentation:

- **Euclidean distance Feature:** This feature specifies the Euclidean distance of pixels from the centre pixel in the image. There are two features Euclidean Horizontal Split (EHS) Feature and Euclidean Vertical Split (EVS) Feature. The EHS feature specifies the Euclidean distance between the rows of top and bottom half pixels to the middle row in the image with a sign. The values are assigned positive value for top half and negative value to bottom half of the pixels in the image. EVS feature specifies the Euclidean distance between columns of left and right side pixels to the middle column in the image with a sign. The values are assigned positive value for right half and negative value to left half of the pixels in the image.

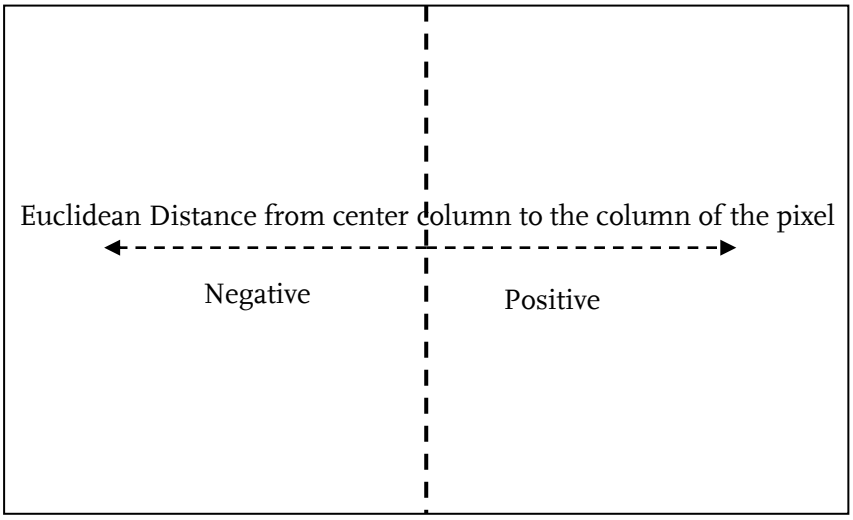


Figure 26: Schematic representation of Euclidean Vertical Split (EVS) feature. The feature specifies the Euclidean distance of pixels from the centre column with a sign. Pixels on the left side and right side of the centre column are assigned negative value and positive sign respectively.

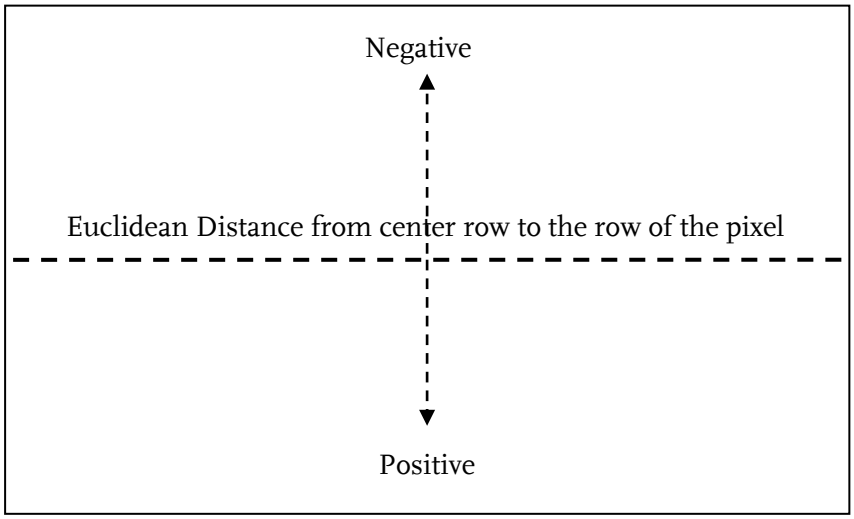


Figure 27: Schematic representation of Euclidean Horizontal Split (EHS) feature. The feature specifies the Euclidean distance of pixels from the centre row with a sign. Pixels on the top and bottom side of the centre row are assigned negative value and positive sign respectively.

2.6 Edge detectors

Edge detection is the process of identifying and locating sharp discontinuities in an image. The discontinuities are abrupt changes in pixel intensity which can characterize boundaries of objects in a scene. Edge detection involves convolving the image with an operator (a 2D filter), which is constructed to be sensitive to large gradients in the image, returning low values in uniform regions. Each edge detection operator is designed to be sensitive to certain type of edges. The geometry of the operator determines a characteristic direction in which it is sensitive to edges. Gradient based methods of edge detection detect the edges by looking for a maximum and minimum in the first or

second derivative of the image. Some of the widely used edge detection operators are Laplacaian, Sobel, Robert, Prewitt, Laplacian of Gaussian (LOG), Canny operators[30].

Edges are not scale invariant and hence the size of the operator (kernel size) is important for edge detection. Using kernel size smaller than the edge width would result in the edge being undetected as the local gradient value would be close to zero as in a smooth surface. Similarly using a large kernel size would also result in edge being undetected due to averaging over a large window.

The influence of kernel size on boundary detection between POI and BP has been experimented for different gradient operators. For the evaluation, instead of using different kernel sizes, the downscaled versions of the image have been used with the same kernel size. The evaluation has been done for 10 images having different POI sleeping postures, all from the same recording.

The evaluation is based on two parameters ED (Edge Deviation) and EM (Edge Match). ED is the metric that reflects the deviation between the true edge and the detected edge. EM signifies the number of detected edge pixels that match with the true edge. The evaluation criterion is to have low ED and high EM.

Let P_t be the true (annotated ground truth) edge made of number of pixels and the location of each pixel is given by (x_t, y_t). (x_d, y_d) is the location of a pixel on the detected edge that is closest to pixel at location (x_t, y_t) on the true edge. $d((x_t, y_t), (x_d, y_d))$ gives the Euclidean distance between (x_t, y_t) and (x_d, y_d).

$$\mathrm{ED} = \sum_{(x_t, y_t) \in P_t} d\big((x_t, y_t), (x_d, y_d)\big) \qquad \textbf{Equation 2.24}$$

$$\mathrm{EM} = \sum m\big((x_t, y_t), (x_d, y_d)\big) \qquad \textbf{Equation 2.25}$$

$m\big((x_t, y_t), (x_d, y_d)\big)$ is a function that evaluates to 1 only if $(x_t, y_t) = (x_d, y_d)$, i.e., the true edge pixel at location $(x_t, y_t) = (x_d, y_d)$ is detected by the edge detector. Table 1 gives the evaluation results of one of the images tested.

Operator	Scale (Horizontal and vertical)	ED	EM
Sobel	1	278.29	743.00
	0.75	280.30	603.00
	0.5	283.40	598.00
Canny	1	231.50	685.00
	0.75	273.80	401.00
	0.5	272.60	420.00
Robert	1	313.80	559.00
	0.75	316.60	509.00
	0.5	321.10	395.00
Prewitt	1	277.10	689.00
	0.75	280.60	578.00
	0.5	284.80	553.00
LOG	1	291.10	495.00
	0.75	293.90	317.00
	0.5	274.40	301.00

Table 1: ED and EM metric values of a test image. The metrics are used to determine the best edge detector that describes the boundary between POI and BP.

From the evaluation, it can be noted that the use of smallest possible kernel size i.e., 3X3 for Sobel, Canny, Prewitt, LOG operators and 2X2 for Robert operators give the better gradient descriptors of the boundary between POI and BP compared to using operators with bigger kernel size. Since the evaluation has been for images from the same recording, we train the classifier with gradient features computed by Sobel, Canny, Prewitt, LOG operators with 3X3 kernel size and Robert operator with 2X2 kernel size and leave the classifier to choose the best gradient features that helps in segmentation.

2.7 Pre-processing

Due to the use of a single IR source, the illumination in the image is not uniform and the signal strength is low in image areas near the leg and head of POI. The features in these areas like the gradients (edge operators) would not be as discriminative as in other brighter areas in the image and may result in (POI) misclassification.

By pre-processing, the intensity values in these darker areas can be boosted, thus increasing the contrast and in turn increasing the gradients. A suitable contrast enhancement technique that is capable of increasing the contrast in these areas without boosting the pixel values in the back ground has to be used for pre-processing. Boosting the background pixels may affect features like brightness (the boosted intensity values may fall in the brightest 50% of pixels, affecting the coarse estimate of POI), edge detectors, HOG (structures in the background may get highlighted, increasing the gradients in the background) and Focus feature (as it is dependent on edges) and may have an impact on the classification.

Figure 28 shows the transfer curve that has been used for contrast enhancement. The transfer function is expressed mathematically by Equation 2.26. The blue line indicates the original intensity values and the red line indicates the boosted intensity values. The pixels in an original image have intensity values ranging from 0 - 150 among which the background pixels fall into intensity range of 10-15 and pixels near POI leg into intensity range of 25-30. From the transfer curve it can observed that the background pixels (intensity range 10-15) are boosted less than the pixel near POI leg (intensity range 25-30).

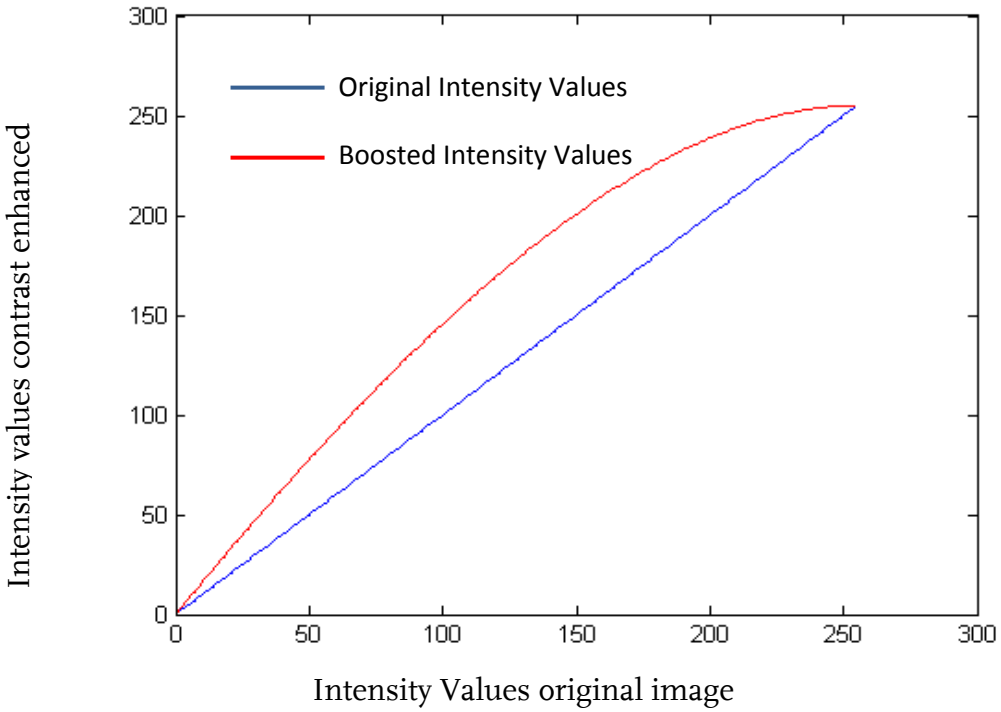


Figure 28: Transfer curve for contrast enhancement

$$f'_{x,y} = f_{x,y} * \left(1 + G\left(f_{x,y}\right)\right)$$ **Equation 2.26**

Where,

$$G\left(f_{x,y}\right) = 1 - e^{\frac{f_{x,y}-255}{255}}$$ **Equation 2.27**

$f_{x,y}$ is the original pixel value at location (x,y) in the image. $f'_{x,y}$ is the new pixel value after pre-processing. The original image without pre-processing is shown in Figure 29 and Figure 30 shows the image after pre-processing. It can be seen that in Figure 30, the edges at the image corners close to POI head and foot are highlighted. Hence, we believe that pre-processing would improve the segmentation in these areas. The influence of pre-processing step on segmentation results are experimented and documented in Section 4.1.6.

Figure 29: Original image without pre-processing. The image area near POI's foot is dark and cannot be distinguished from background.

Figure 30 : Image after pre-processing. The pixels in image area near POI's foot are boosted and are distinguishable from background.

2.8 AdaBoost classifier

As discussed in Section 2, several discriminative (weak) features together can be combined by using a classifier for segmenting the POI area from the image. In Sections 2.1 – 2.6, various (discriminative) features that can be used by a classifier for segmenting POI and BP have been presented. Using a classifier, the best discriminative features can be selected and used together to obtain at an accurate and robust segmentation result. The segmentation (using a classifier) of POI and rest of the image (BP and background) can be expressed as a binary classification problem. A '*True*' label to POI area and a '*False*' label to the rest (BP and background) of the image area.

For this, a supervised machine learning algorithm called AdaBoost is used which has been found to be fast and robust in similar applications (e.g. face detection, gesture recognition) which use multiple weak features[36]. AdaBoost is the acronym for Adaptive Boosting and was formulated by Freud and Schapire [37]. The algorithm takes as input a labelled training set of feature values. By selecting and linearly combining the features, a classifier is built that mimics the labelling. AdaBoost creates a function H that maps the inputs to desired outputs using a labelled training set. More formally, the training data consists of feature values ($\overrightarrow{x_i}$) and desired output or labels (y_i),

Where

$$\overrightarrow{x_i} \in X^M, X \in \boldsymbol{R},$$

$$y_i \in Y = \{0,1\},$$

$$0 \leq i \leq N,$$

N represents the number of samples and M the number of feature values per sample. AdaBoost finds a function H,

$$H = X^M \rightarrow Y \qquad \textbf{Equation 2.28}$$

that minimizes the error E [38] with respect to distribution $D(i)$ (the weight of each sample).

$$E = \sum_{i=0}^{N-1} D(i)[y_i \neq H(\overrightarrow{x_i})], \qquad \textbf{Equation 2.29}$$

The algorithm starts with initializing uniform distribution to all samples. Then for every iteration, the distribution is normalized. A weak classifier is selected using the normalized distribution. This weak classifier is used to update the distribution. The weight of the correctly classified samples is decreased, such that in the next iteration, the emphasis on the incorrect classified samples is higher. This process is repeated several times for a fixed number of iterations or till an acceptable detection rate and false alarm rate is achieved.

A significant improvement to the AdaBoost algorithm was made by Viola and Jones [36]. They described a methodology for constructing a cascade of classifiers which achieves higher performance while reducing the computation time significantly. The cascaded AdaBoost configuration can be seen in Figure 31. Each classifier represents one layer of the strong cascaded classifier.

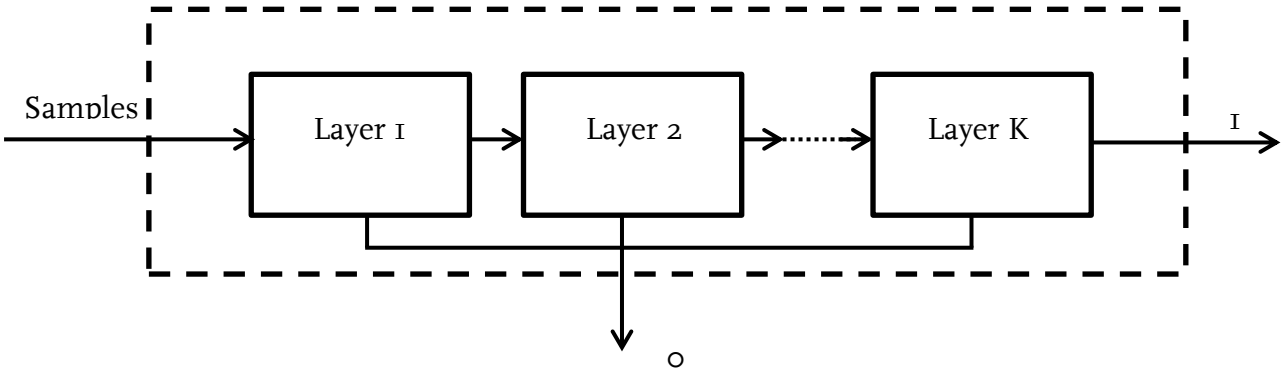


Figure 31: Cascaded AdaBoost

Cascaded AdaBoost gets two sets as input, one for training and one for validation. The first set is used by AdaBoost to select the best weak classifier. The second set is used to determine if the desired Pfa (false alarm rate per layer) and Pfa_{end} (false alarm rate stop criteria) are reached.

$$\text{Pfa} = \frac{(\#\text{FP})}{(\#\text{FP}) + (\#\text{TN})}$$

Equation 2.30

Where $\#FP, \#TN$ are the total numbers of classified false positives and true negatives respectively.

Adaboost is configured (trained) in the following way:

1. In case that Pfa_i (false alarm rate of layer i) is not low enough to finalize the layer (i.e. not lower than Pfa), another weak classifier is added to the current layer of strong cascaded classifier.
2. In case that Pfa_i is low enough to finalize the layer (i.e. lower than Pfa), but not lower than Pfa_{end} , then another layer is added to the strong cascaded classifier.
3. In case that Pfa_i is low enough to finalize the layer (i.e. lower than Pfa and lower than Pfa_{end}), the AdaBoost configuration (number of cascaded layers, number of weak classifiers per layer) is finalized (AdaBoost training is completed).

At the end of every cascade layer, only the positive labelled (ground truth-annotations) samples and the positive classified samples of the training set are passed to the next cascade layers.

At the input of every cascade layer, AdaBoost initializes the distribution using $D(i) = \frac{1}{N}$ for all samples i. At the output of each weak classifier within a layer, the distribution is updated and normalized such that, the emphasis on the incorrect classified samples is higher as discussed earlier.

The classifier is trained with frames from different data sets and each pixel in the image acts as a sample. For each pixel sample, there is a corresponding feature value and a ground truth. In case of a block based feature (one feature value per block), the pixels belonging to a block are assigned the same feature values as that of the block. The details of classifier training and ground truth generation are explained in detail in Section 3.3.

The output of the training phase is a classifier configuration file containing the details of features used per cascade layer of classifier and their corresponding thresholds. The final AdaBoost configuration, as a result of training has 13 cascade levels (weak classifiers). It uses Brightness feature, Focus feature (defocus measure), Region based motion feature, HOG - Histogram of Oriented Gradients feature (bins 1, 2, 3, 6, 8, 12, 14, 15, 16, 17, 18), Location based features (EVS - Euclidean Vertical Split, EHS- Euclidean Horizontal Split), Sobel feature (gradients in vertical direction calculated by Sobel operator), Laplacian of Gaussian feature.

In the classification stage, the features are calculated for every pixel in the test frame and are given as input to the classifier along with the classifier configuration file. The classifier outputs a binary label (0, 1), the 'Positive (1)' label indicates that the pixel belongs to POI and 'Negative (0)'indicates that the pixel belongs to rest of the image (BP and background).

An example of the output classification result of AdaBoost classifier is shown in Figure 32. The classifier results are presented and evaluated in Section 4.1.

Figure 32: AdaBoost classified output of POI segmentation result. The closed regions given by the white line show the classified POI area in the image.

2.9 Suggested blanket compensation

Video actigraph is generated by calculating the sum of absolute differences of pixels that belong to POI area in two consequent frames. However, in a shared bed it is common for POI and BP to share a common blanket and there can be instances where the blanket on top of POI moves when it is pulled by BP. This would cause a false signal in *video actigraph* even after segmentation. In this section, we describe an algorithm by which this type of false detections of POI movements caused due to blanket pulling by BP might be compensated. The algorithm is based on the idea that when there is motion in POI area due to blanket pulling by BP, the number of blocks having motion in POI area would be less than the number of blocks that would have motion when POI moves. The flow chart is shown in Figure 33. The results of algorithm validation are present in Section 4.2.3.

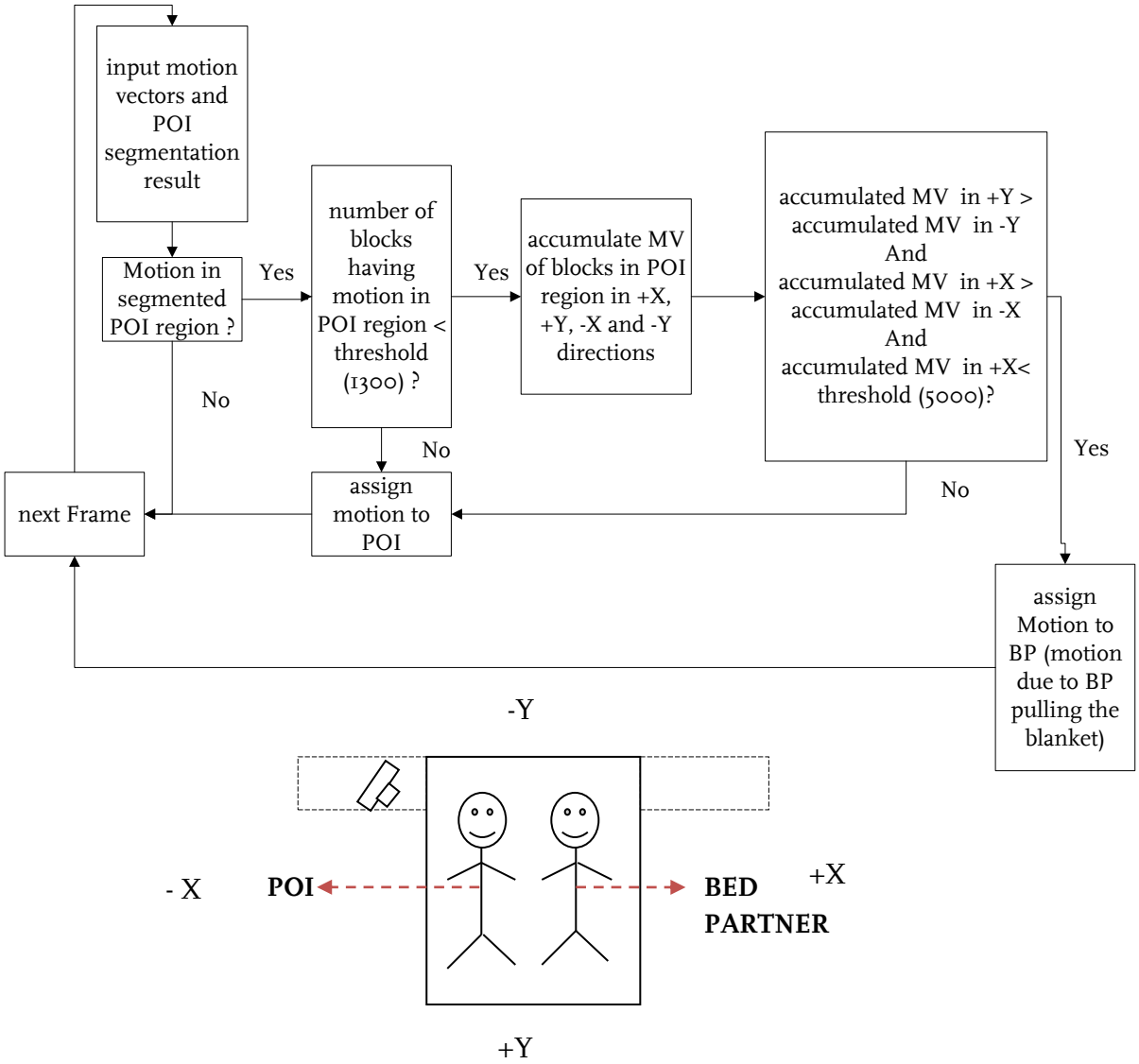


Figure 33: Flow chart describing blanket compensation method to detect false POI movements caused by blanket pulling by BP. MV is the motion vector magnitude of a block.

3 Experimental setup

Data acquisition is a crucial step in developing and testing the algorithm. In this chapter, the data set collection is described. In Section 3.1, the equipment and the installation details for collecting the data are introduced. In Section 3.2, the characteristics of data sets like the recording environment, the motion (POI and BP movements) types are explained. The details about ground truth generation and classifier training are given in Section 3.3.

3.1 Equipment and installation

The end consumer application would be a standalone product where the illumination source and the camera would be together. This is replicated in the development of the project by placing the illumination source just below the camera. The camera and the light source are then together placed on the bedside table close to the head of the POI. However during development, to avoid the influence of varying the height of the bed side tables used for recording different data sets, a tripod is used in place of a bed side table. The scheme of the camera installation is shown in Figure 34, h_c is the height of the camera from the ground and h_b is the height of the bed. The distance between the camera and the bed (h_{bc}) is set to be between 30-35 cms. The camera height is fixed to be 25 cms from the top of the bed, i.e., $h_c - h_b$ =25 cms. This height would ensure BP to be occluded by POI for most of the time. The viewing angle of the camera is adjusted in such a way that POI is completely visible from head to foot. The image recorded from the camera as a result of the installation is shown in Figure 35.

For the video recording, a µEye CMOS camera USB UI-1220SE-M[39] with Fujinon FE185C086HA-1[40] fish eye lens is used. For illumination in the dark, an infrared light source with maximum intensity at wavelength 825nm is used. The resolution of the image is 752X480 and the frame rate is 10 fps. To compare the *video actigraph* generated by the application with the *wrist actigraph, t*he *wrist actigraph* data of POI is registered using a Texas Instruments Chronos [21] watch. The recordings have been done by using Philips - Home Sleep Monitoring application, the screen shot of the application window is shown in Figure 36. The features of the application are 1) encryption of recorded data which addresses the privacy concerns of participants 2) Synchronization of video and *wrist actigraph* samples from chronos watch and 3) Fragmenting the video recordings into small chunks for easy processing. The recording protocol supplied to the participants can be found in *Appendix –A*.

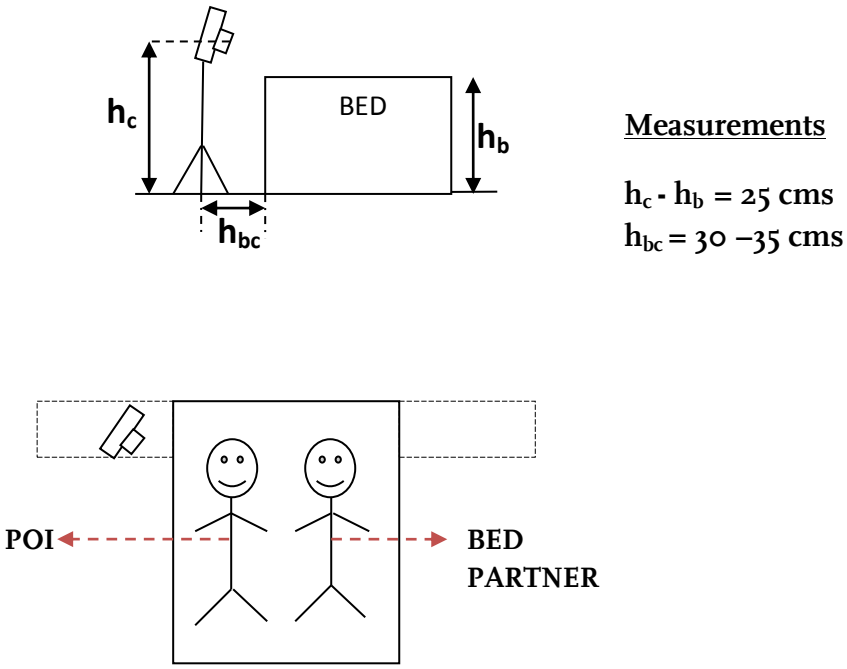


Figure 34 : Scheme for camera installation.

Figure 35 : Image recorded from the camera after the installation. POI is visible from head to foot and is focussed. Most of the BP region is occluded by POI.

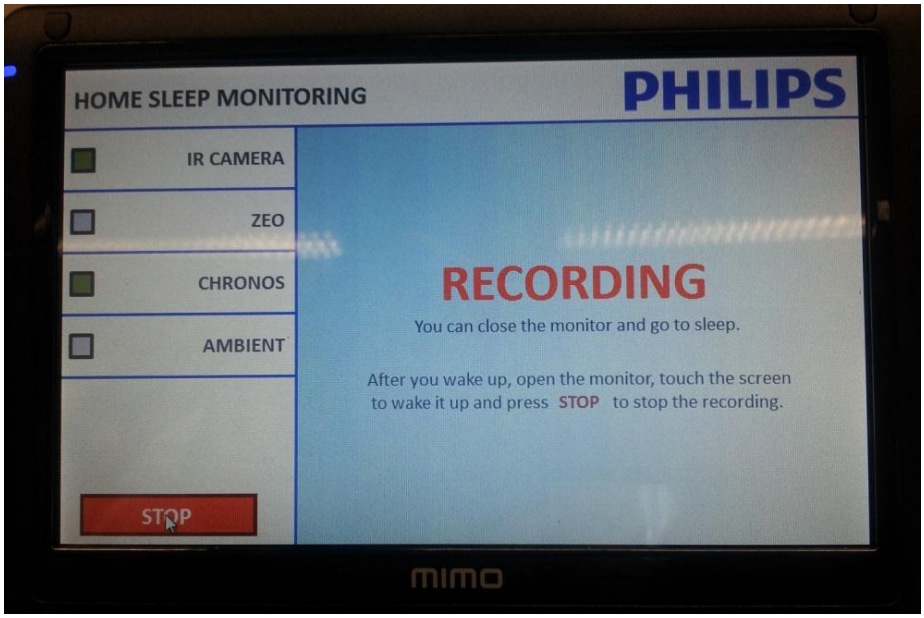


Figure 36: Philips - Home Sleep Monitoring Application used for recording data sets

3.2 Data sets

The data sets recorded can be classified into two categories, *Natural* and *Synthetic*. The *Natural* data sets are recordings acquired in a natural sleeping environment at home. The movements of persons in the *Natural* data sets are not restricted and reflect the real sleep situation. There are two *Natural* data sets recorded, each recorded with different participants in their respective homes. The two Natural data sets are referred to as *DATA_Home_1* and *DATA_Home_2* henceforth.

The *Synthetic* data sets are acquired both at a lab (home lab at Philips) and a home with restrictions on movements of persons. There are two *Synthetic* data sets, referred as *DATA_Synt_1* and *DATA_Synt_2* henceforth. In the *Synthetic* data sets, both small (single arm, single leg and head movements) and large movements (turning around, moving both arms and legs together) were performed by two persons, one at a time and also together.

The properties of both *Natural* and *Synthetic* data sets are listed in Table 2. A sample image from each of the data sets can be seen in Figures 37-40. The recordings have been taken in both dark and bright illumination conditions to test the application for its robustness against varying illumination conditions. To validate the application against movements originating from under the blanket, different types of blankets (thin and thick) have been used.

DATA Set	Lighting Condition	Blanket	Restrictions	Duration in hours (frame rate 10)
DATA_Home_1	Dark	1 shared (Thick) blanket	N.A	7
DATA_Home_2	Bright	1 shared (Thick) blanket	N.A	7.5
DATA_Synt_1	Bright	1 shared (Thick) blanket	• Both Persons moving • One person moving at a time	2
DATA_Synt_2	Dark	1 shared (Thin) blanket	• Both Persons moving • One person moving at a time • Blanket movement by POI • Blanket movement by BP	2

Table 2: Overview of lighting conditions, blankets used and movement restriction in recorded data sets.

Figure 37: Image from data set: DATA_Home_1

Figure 38: Image from data set: DATA_Home_2

Figure 39: Image from data set: DATA_Synt_1

Figure 40: Image from data set: DATA_Synt_2

3.3 Classifier training

As discussed earlier in Section 2.8, a supervised machine learning algorithm (AdaBoost) is used for POI area segmentation from the image. Any supervised (machine) learning would need training and the training samples are the various feature values (used for classification) along with the ground truth.

For the classifier training 240 frames were selected from data sets *DATA_Home_1* (80 frames), *DATA_Home_2* (80 frames), *DATA_Synt_1* (40 frames) *and DATA_Synt_2* (40 frames). The frames were selected such that they reflect most of the poses of a person on a bed. The selected frames are temporally separated by a large extent (at least by 100 frames).

As discussed in Section 1.3, the *video actigraph* is generated by taking the sum of absolute differences of pixels in two consecutive frames. Hence, POI segmentation result has direct influence on generated *video actigraph*. The accurate the segmentation result, the higher will be the correlation of generated *video actigraph* with POI's movements. Therefore, to have the segmentation result accurate to a pixel level, it has been chosen to use pixel samples, i.e. every pixel in the image is a sample. Pixel samples means that every pixel has its corresponding feature values and ground truth. In case of a block based feature (one feature value per block), the pixels belonging to a block were assigned the same feature values as that of the block. Generating the ground truth with pixel accuracy poses a difficult problem with existing image annotation tools. An open source annotation tool called ITK-SNAP[41] is used for annotating the images with pixel accuracy. With modifications (enabling it to write the annotated image into PNG format), the tool has been adapted to annotate the selected frames. The tool window is shown in Figure 41.

Figure 41: ITK-Snap annotation tool window

The generated annotations (ground truth) for some of the selected frames can be seen in Figure 42.

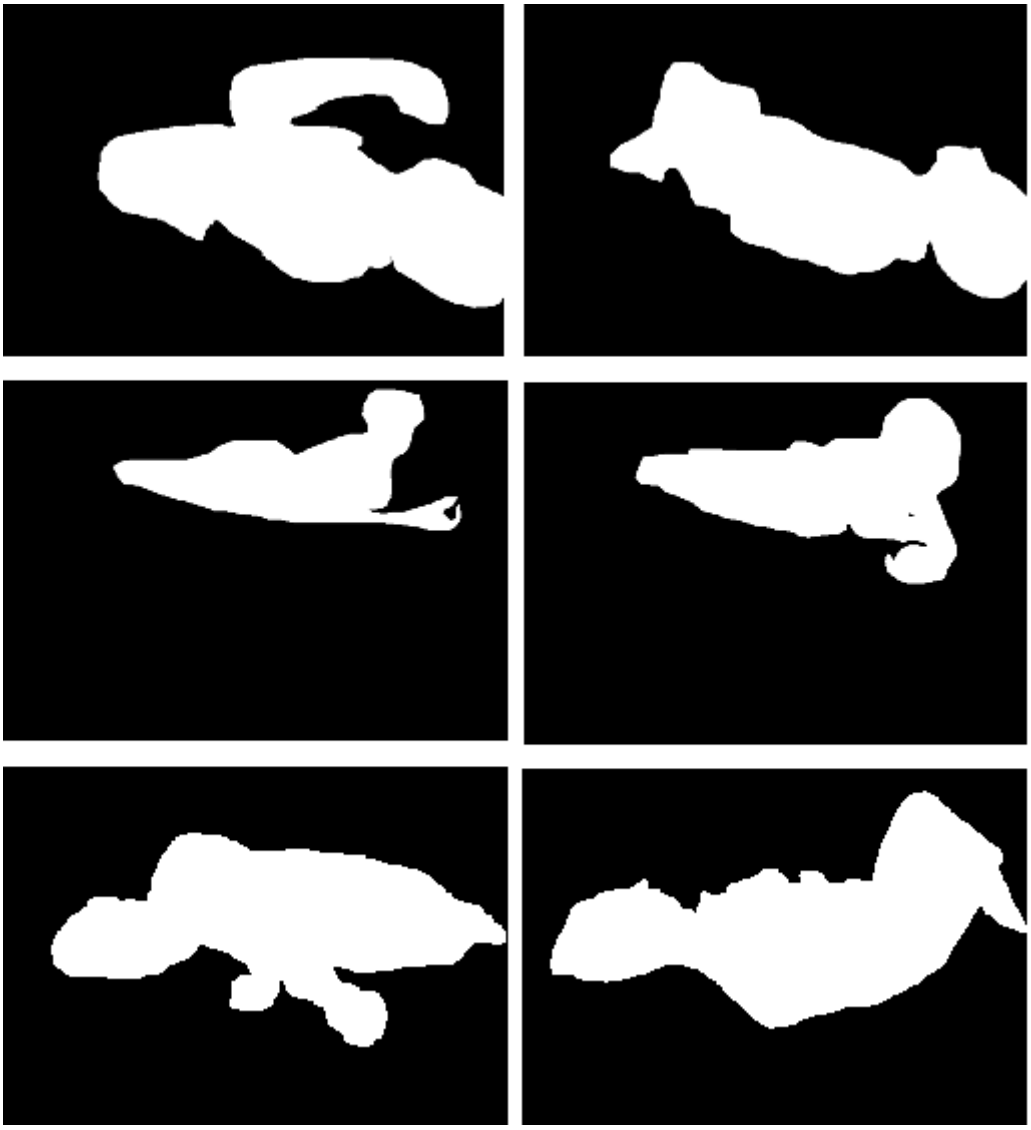


Figure 42: Generated Annotations of selected frames

Although a tool capable of annotating images at pixel accuracy, has been used, the annotations can still have small errors especially at the boundaries. These small errors are due to the fact that a human is involved in the annotation process (supervised annotations). To compensate for this error, the boundary of the annotated image is eroded by a 3X3 block. The motivation behind this is to have no false samples even at the expense of losing some true samples to avoid sample pollution which can (greatly) affect the classification. The classifier is trained with all the samples (feature values of pixels and the corresponding ground truth) and the output of the training phase is an AdaBoost configuration file which specifies the number of weak classifiers in the cascade, the number of layers per weak classifier, the feature being used at each layer, and their corresponding thresholds. This configuration file is used as input to AdaBoost during classification. The evaluation of the classifier is presented in Section 4.1.

4 Results and discussion

Using the setup as described in Section 3.1, data sets (video) have been recorded. These data sets were used to train the classifier as described in Section 3.3. The classifier uses several discriminative features to segment POI as described in Section 2. The POI segmentation result is used to generate the *video actigraph* as discussed in Section 1.3. The analysis of POI segmentation and *video actigraph* results is given in Section 4.1 and Section 4.2 respectively. In Section 4.3, a brief discussion about the features used and results obtained is presented.

4.1 Classifier evaluation (POI segmentation analysis)

In this section, the analysis of classifier (POI segmentation) is presented. In Section 4.1.1, the classifier testing infrastructure is described. The metrics for evaluating the classifier are discussed in Section 4.1.2. The validation method used for classifier evaluation is illustrated in Section 4.1.3. Sections 4.1.4 and 4.1.5 present the POI segmentation results of inter and intra data set cross validation respectively. In Section 4.1.6, a brief discussion about the influence of pre-processing on segmentation result is presented.

4.1.1 Classifier testing

The classifier as discussed in Section 3.3 is trained with the training set (selected frames). The output of the training phase is a configuration file that is used as an input to the classifier testing (classification) phase. The schema of classifier testing is given in Figure 43. The performance of the classifier is evaluated by comparing the output label generated by the classifier with the annotated ground truth.

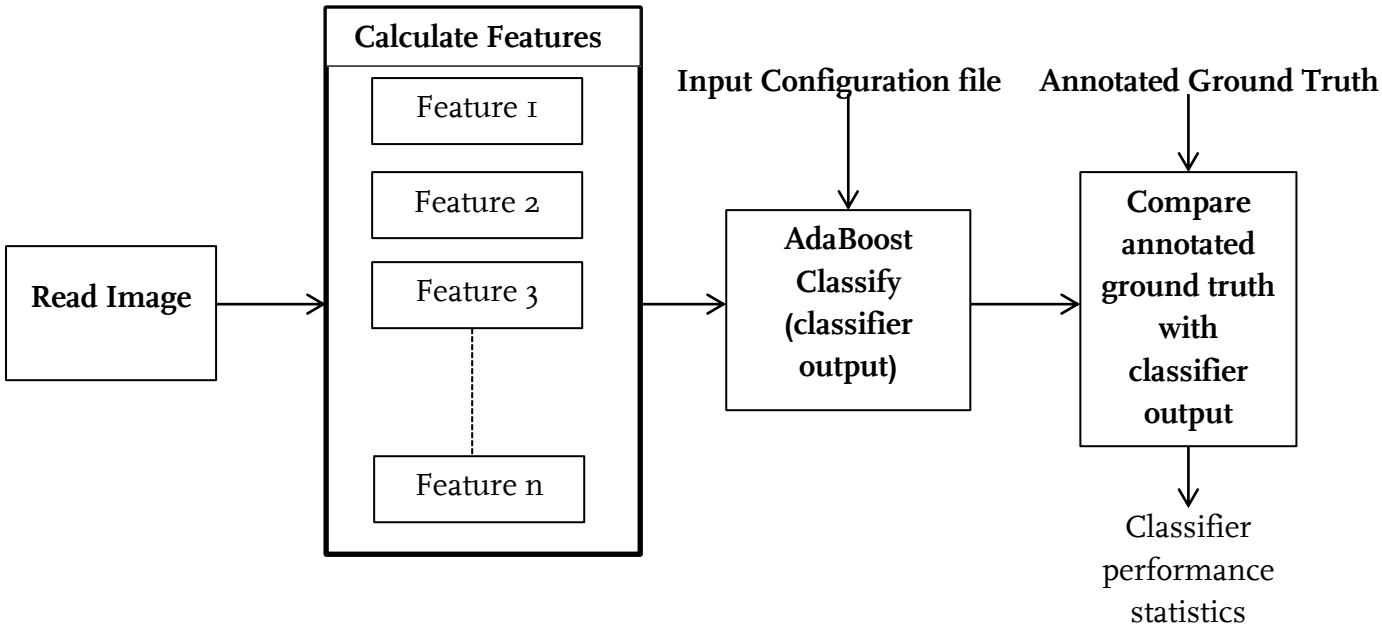


Figure 43: Schema - Classifier testing. The output of the classifier is compared with the annotated ground truth to compute the metrics. The metrics describe the classifier performance.

4.1.2 Metrics for classifier evaluation

For the evaluation, the confusion matrix for a binary classifier shown in Table 3 is used. TP (True Positives) gives the fraction of correctly classified positive samples. TN (True Negatives) gives the fraction of correctly classified negative samples. FP (False Positive) gives the fraction of negative samples misclassified as positive and FN (False Negatives) gives the fraction of positive samples classified as negative.

		Ground Truth		
		True (1) (Pixel belongs to POI)	False (0) (Pixel belongs to BP or Background)	
Classifier output	Positive (1) (Pixel assigned to POI)	TP	FP	➡ Positive Prediction Value
	Negative (0) (Pixel assigned to BP or background)	FN	TN	➡ Negative Prediction Value
		⬇ Sensitivity	⬇ Specificity	↘ Accuracy

Table 3: Confusion matrix summarizing the definitions used for evaluating the (binary) classifier[33].

Binary classifiers can be evaluated by calculating, average *sensitivity*, average *specificity*, average *accuracy* and average *precision*. *Sensitivity* or True Positive Rate is the percentage of positive labelled samples that were classified positive by the classifier.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Equation 4.1

Specificity is the percentage of samples that were classified as negative by the classifier.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Equation 4.2

Accuracy is the percentage of correct classifications, i.e. negative labelled samples classified as negative and positive labelled samples classified as positive. *Accuracy* gives the overall classification accuracy of the classifier.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Equation 4.3

Precision or *Positive Predictive Value* is the percentage of positive predictions that are correct.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Equation 4.4

The '*True'* samples in the training set are the pixels in the image that belong to POI. The classification of these 'True' samples is of greater importance than the '*False'* samples for our application. This is due to the fact that the fraction of '*True'* samples in the image is far lower than the '*False'* samples and the application generates *video actigraph* by calculating the sum of absolute differences (SAD) of pixels in POI area (pixels having positive label) in two consecutive frames. Hence, POI segmentation result has direct influence on generated *video actigraph*. The higher the precision (correctly classified 'True' samples- TP), the higher will be the correlation of generated *video actigraph* with POI's movements.

As a result, the metric to rightly evaluate the segmentation result would be the one that can quantify '*True'* samples detection rate. There are two such metrics *sensitivity* and *precision*. However, *sensitivity* is not robust against overfitting. For example, due to overfitting the classifier may classify all the pixels in image as '*True'* samples. In this case the *sensitivity* metric would evaluate to 1 i.e., 100% detection of 'True' samples. However, there would be a significant number of FP (False Positives).

The *Precision or positive predictive value* metric is robust to overfitting as it also considers False Positives (FP). *Precision* metric is ideal in cases where the number of '*False'* samples are much higher than '*True'* samples in the training set (the fraction of pixels in the image belonging to POI area is very small). *Accuracy* is also a good metric for evaluating the performance as it reflects the overall correct classifications. Thus, we chose to express classifier performance by *Precision* and *Accuracy*.

4.1.3 K-fold cross validation

The classifier performance is evaluated by means of K-fold cross-validation method. In this method the whole data set is used for both training and testing but never simultaneously. The entire data is split into K subsets and for each subset $i = 1,2,\ldots K$, the

classifier is trained with all subsets except for i^{th} subset and tested with i^{th} subset to obtain a generalization error. The average of all the estimated generalization errors across *K* folds is taken as the average *Precision* and *Accuracy* of the classifier.

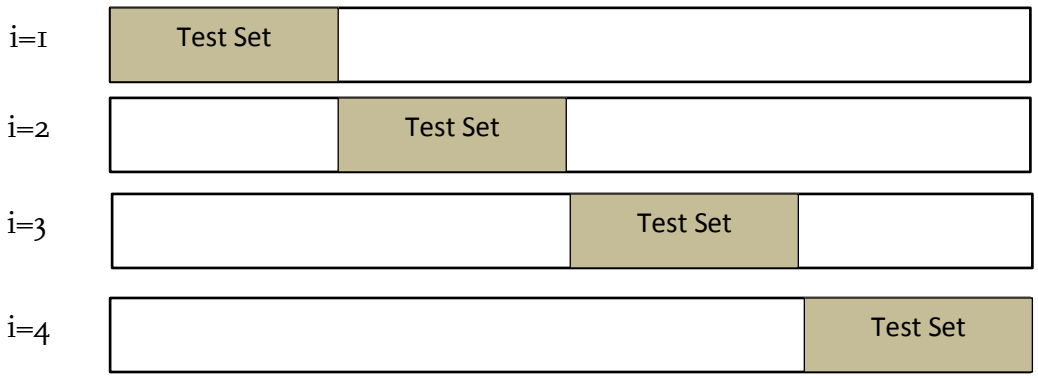


Figure 44: An illustration of K-fold Cross – Validation method

In Section 3.2, various data sets (*DATA_Home_1, DATA_Home_2, DATA_Synt_1* and *DATA_Synt_2*) and their characteristics have been discussed. The experimental setup only varies from one data set to the other. Hence the classifier has to be evaluated by *K*-fold Cross Validation method for both *intra data set* and *inter data set*.

4.1.4 Cross –validation (*intra data set*)

In this section, the classifier evaluation for *intra data set* with and without pre-processing is presented. In *intra data set* the classifier is evaluated by training and testing with frames from the same recording. The annotated frames of the recording set are split into *K* subsets and are used for both training and testing. The value of K is chosen as 4. The maximum number of annotated frames for a data set is 80 and with K=4, for every iteration the testing set would have 20 frames which is large enough to evaluate the classifier performance considering the small data set we have for both training and testing. For data sets *DATA_Synt_1 and DATA_Synt_2,* K is chosen as 2, since there are only 40 frames that are available from each of these for training and testing. This evaluation is done for all the data sets separately and the final average cross-validation (*intra data set*) *Precision* and *Accuracy* is determined. The classifier evaluation without pre-processing is presented in Table 4 and Table 5 and with pre-processing (as described in Section 2.7) in Table 6 and Table 7.

Training Data Set	Accuracy (%) – without pre-processing				
	Testing Data Set (Subset of the same training data set)				
	K=1	K=2	K=3	K=4	Average
DATA_Home_1	89.81	89.99	90.11	89.96	89.96
DATA_Home_2	94.32	92.26	91.67	93.15	92.85
DATA_Synt_1	90.19	91.19	-	-	90.69
DATA_Synt_2	87.36	86.68	-	-	87.02
Average classification accuracy without pre-processing: cross-validation(*intra data set*)					90.13

Table 4: Classifier statistics – Accuracy without pre-processing (trained and tested with same data set)

Training Data Set	Precision (%)- without pre-processing				
	Testing Data Set (Subset of the same training data set)				
	K=1	K=2	K=3	K=4	Average
DATA_Home_1	74.75	80.02	81.81	79.36	78.98
DATA_Home_2	82.22	78.10	72.33	80.94	78.39
DATA_Synt_1	83.48	90.00	-	-	86.74
DATA_Synt_2	74.57	80.36	-	-	77.46
Average Precision (Positive predictive value) without pre-processing: cross- validation(*intra data set*)					80.39

Table 5: Classifier statistics – Precision without pre-processing (trained and tested with same data set)

Training Data Set	Accuracy (%)- with pre-processing				
	Testing Data Set (Subset of the same training data set)				
	K=1	K=2	K=3	K=4	Average
DATA_Home_1	89.47	91.69	89.94	90.75	90.46
DATA_Home_2	94.03	92.39	91.50	93.48	92.85
DATA_Synt_1	90.55	90.15	-	-	90.35
DATA_Synt_2	91.7	90.34	-	-	91.02
Average classification accuracy with pre-processing: cross- validation(*intra data set*)					91.17

Table 6: Classifier statistics – Accuracy with pre-processing (trained and tested with same data set)

Training Data Set	Precision (%)- with pre-processing				
	Testing Data Set (Subset of the same training data set)				
	K=1	K=2	K=3	K=4	Average
DATA_Home_1	72.67	84.07	81.58	81.75	80.02
DATA_Home_2	81.75	77.23	71.97	80.68	77.91
DATA_Synt_1	85.46	83.20	-	-	84.33
DATA_Synt_2	82.37	79.56	-	-	80.97
Average Precision (Positive predictive value) with pre-processing: cross-validation(*intra data set*)					80.66

Table 7: Classifier statistics – Precision with pre-processing (trained and tested with same data set)

From Table 4 and Table 5, it can be seen that the average *accuracy* (without pre-processing) is lower for data sets *DATA_Home_1* and *DATA_Synt_2* as compared to the other two data sets. This is because of the weak lighting conditions (Section 3.2) in which these data sets were recorded. Due to the weak lighting conditions, the illumination at the image corners (near the head and the legs of POI) is poor. As a result, the signal strength in these regions is low making the features less discriminative, which in turn affects POI segmentation (segmentation degrades near head and legs of POI). However, with pre-processing the signal strength is improved in these areas, making the features more discriminative. This improves the average classification *accuracy* as seen in Table 6 and Table 7.

The segmentation results (with pre-processing) are shown in Figure 45. The heat maps of classification errors (with pre-processing) for all data sets are shown in Figure 46. The heat maps have been generated to see in which regions the misclassification is more. As expected, the misclassification is higher around the POI boundary than the centre of POI. The heat maps show the accumulated misclassification of POI area in the image per data set. The classification error is found by comparing the output segmentation result of the classifier with the manually annotated POI area in the image (ground truth).

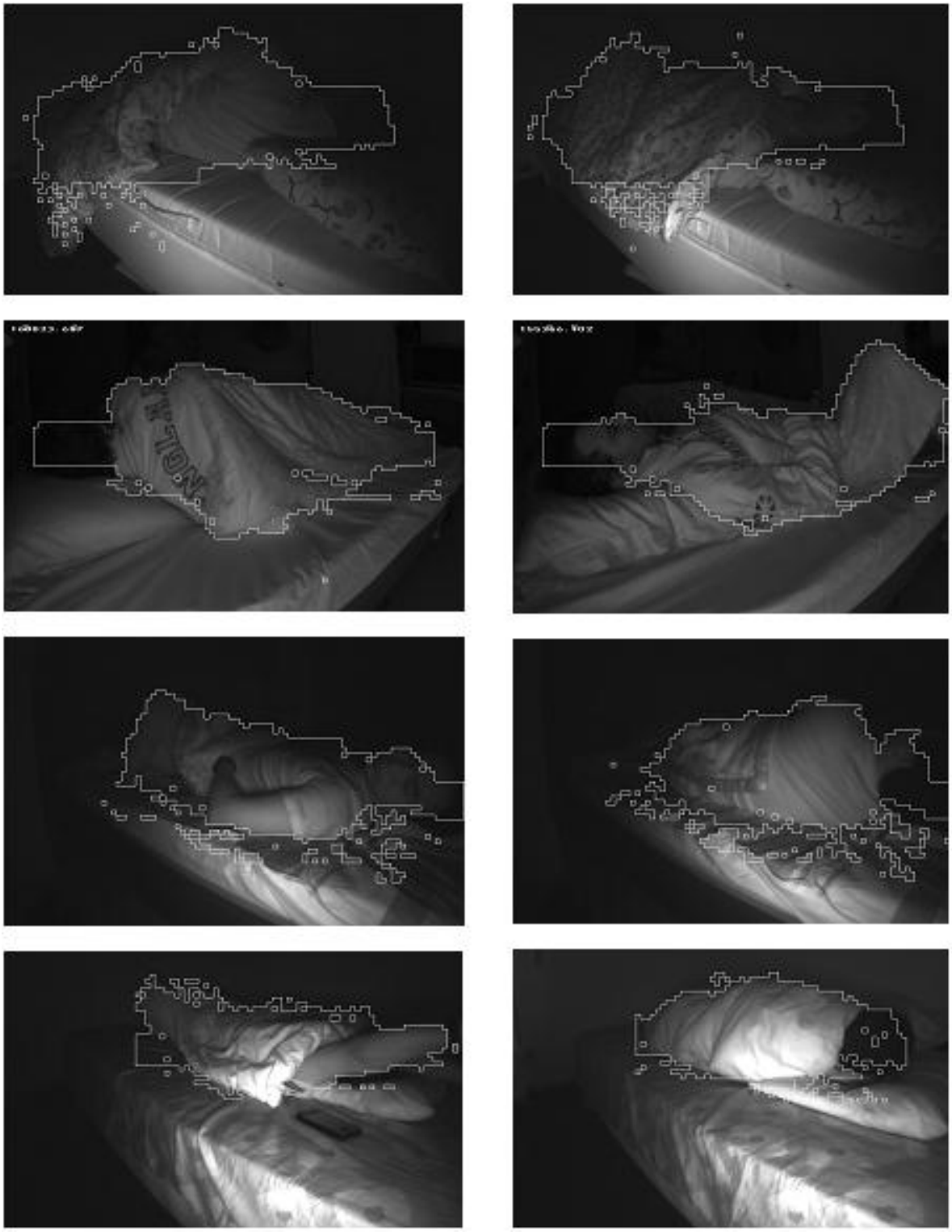


Figure 45: Segmentation Results - training and testing with same data set. The boundary of (segmented) POI is given by the white lines.

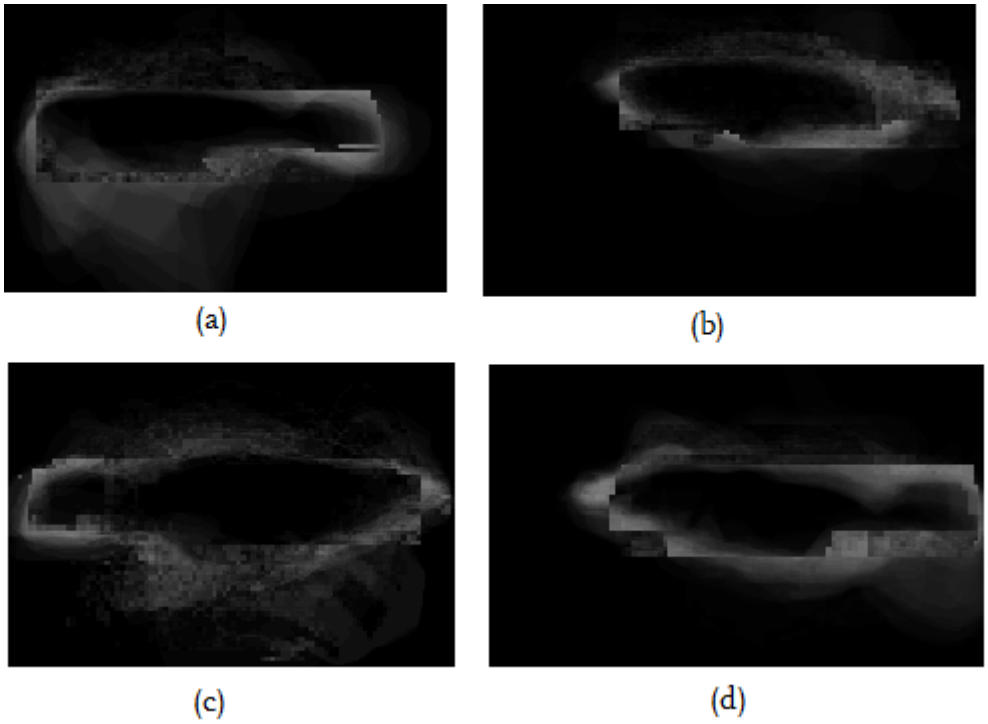


Figure 46: Heat maps indicate the accumulated classification error of all data sets (*intra data set*) (a) *DATA_Home_1* (b) *DATA_Home_2* (c) *DATA_Synt_1* (d) *DATA_Synt_2*

4.1.5 Cross-validation (*inter data set*)

The classifier evaluation for *inter data set* is presented in this section. In *inter data set,* the classifier is trained and tested with frames from different recordings. The classifier is trained with one of the four data sets (*DATA_Home_1, DATA_Home_2, DATA_Synt_1, DATA_Synt_2*) and tested with the other one. This procedure is repeated till all the data sets are tested. The classifier evaluation without pre-processing is presented in Table 8 and Table 9 and with pre-processing in Table 10 and Table 11.

Training Data Set	Testing Data Set	Accuracy
DATA_Home_2, DATA_Synt_1, DATA_Synt_2	*DATA_Home_1*	89.87
DATA_Home_1, DATA_Synt_1, DATA_Synt_2	*DATA_Home_2*	83.31
DATA_Home_1, DATA_Home_2, DATA_Synt_2	*DATA_Synt_1*	89.00
DATA_Home_1, DATA_Home_2, DATA_Synt_1	*DATA_Synt_2*	85.53
Average classification accuracy without pre-processing: cross-validation(*inter data set*)		86.95

Table 8: Classifier statistics - Accuracy without pre-processing (trained and tested with frames from different data sets)

Training Data Set	Testing Data Set	Precision
DATA_Home_2, DATA_Synt_1, DATA_Synt_2	*DATA_Home_1*	81.52
DATA_Home_1, DATA_Synt_1, DATA_Synt_2	*DATA_Home_2*	51.48
DATA_Home_1, DATA_Home_2, DATA_Synt_2	*DATA_Synt_1*	90.61
DATA_Home_1, DATA_Home_2, DATA_Synt_1	*DATA_Synt_2*	77.50
Average Precision (Positive predictive value) without pre-processing: cross- validation(*inter data set*)		75.27

Table 9: Classifier statistics - Precision without pre-processing (trained and tested with frames from different data sets)

Training Data Set	Testing Data Set	Accuracy
DATA_Home_2, DATA_Synt_1, DATA_Synt_2	*DATA_Home_1*	89.98
DATA_Home_1, DATA_Synt_1, DATA_Synt_2	*DATA_Home_2*	88.89
DATA_Home_1, DATA_Home_2, DATA_Synt_2	*DATA_Synt_1*	86.74
DATA_Home_1, DATA_Home_2, DATA_Synt_1	*DATA_Synt_2*	88.23
Average classification accuracy with pre-processing : cross-validation(*inter data set*)		88.46

Table 10: Classifier statistics - Accuracy with pre-processing (trained and tested with frames from different data sets)

Training Data Set	Testing Data Set	Precision
DATA_Home_2, DATA_Synt_1, DATA_Synt_2	*DATA_Home_1*	82.63
DATA_Home_1, DATA_Synt_1, DATA_Synt_2	*DATA_Home_2*	62.58
DATA_Home_1, DATA_Home_2, DATA_Synt_2	*DATA_Synt_1*	89.74
DATA_Home_1, DATA_Home_2, DATA_Synt_1	*DATA_Synt_2*	81.49
Average Precision (Positive predictive value) with pre-processing: cross-validation(*inter data set*)		79.11

Table 11: Classifier statistics - Precision with pre-processing (trained and tested with frames from different data sets)

The average *accuracy* and average *precision* of cross-validation (*inter data set*) compared to that of cross-validation (*intra data set*) is lower. This is expected as the training and testing frames for *inter data set* evaluation are from different recordings. This shows that that the classifier is not completely independent to changes in illumination and setup. However, these results are more important for the application and reflect how the classifier generalizes to different data sets.

The average *precision* (with and without pre-processing) when testing with *DATA_Home_2* is low (62.58 from Table 11 and 51.48 from Table 9) because the classifier was trained with data sets *DATA_Home_1, DATA_Synt_1, DATA_Synt_2* which have POI

centred (approximately) in the same part of the image (head of POI appears closer to camera Figure 37, Figure 39, Figure 40). As a result of training (with data sets in which POI is centred in the same part of the image), location based features gain prominence, thus affecting the classification accuracy when tested on a data set (*DATA_Home_2*) with different camera setup. By training the classifier over larger data sets would make it robust against any such variation in camera setup.

The segmentation results of cross-validation (*inter data set* with pre-processing) are shown in Figure 47. The heat maps of classification errors for all data sets (with pre-processing) are shown in Figure 48. From the heat maps, it can be seen that the misclassification of pixels in POI area for cross-validation (*inter data set*) is slightly higher than cross-validation (*intra data set*) (shown in Figure 46) as discussed above.

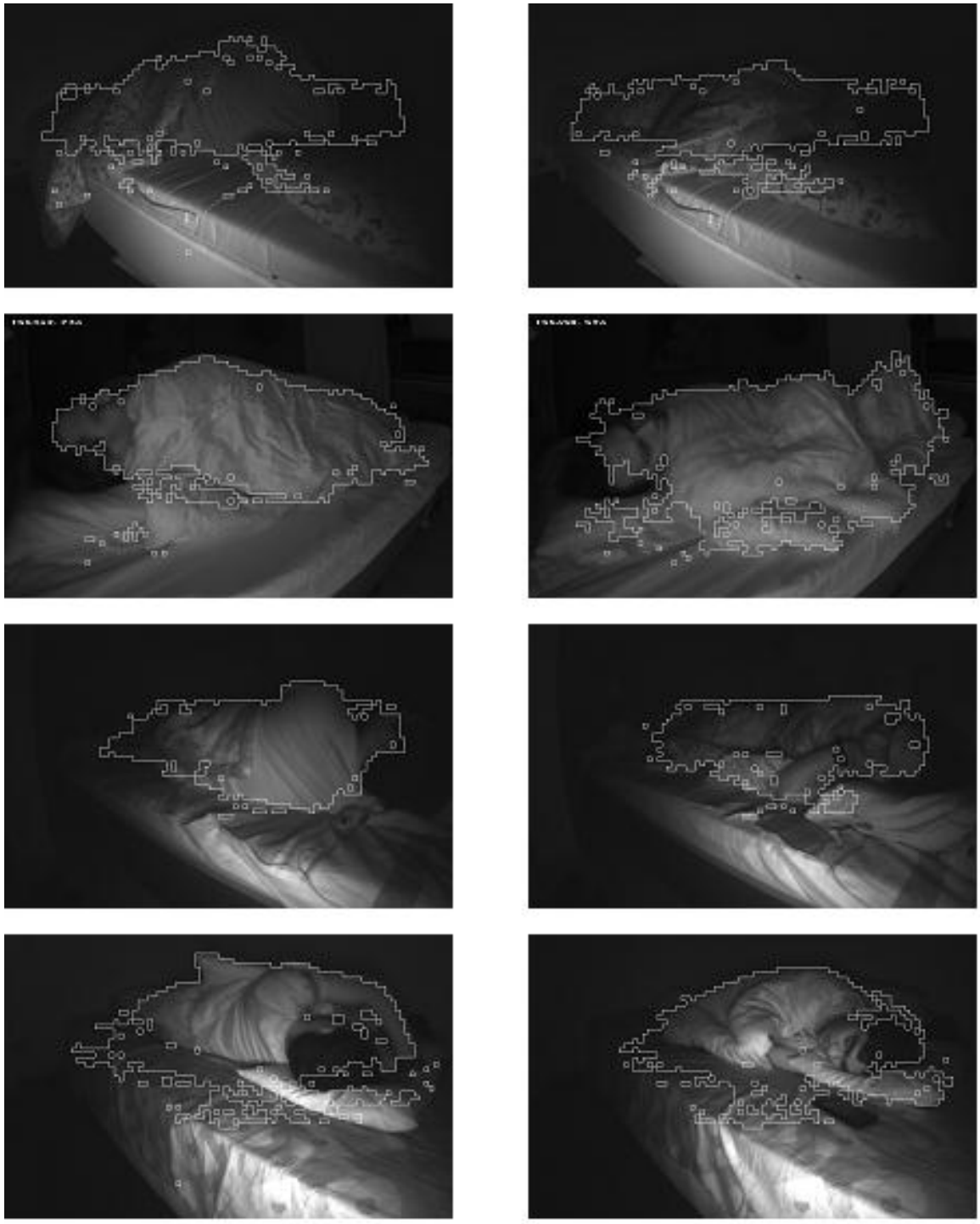


Figure 47: Segmentation result- training and testing with different data sets. The boundary of (segmented) POI is given by the white lines.

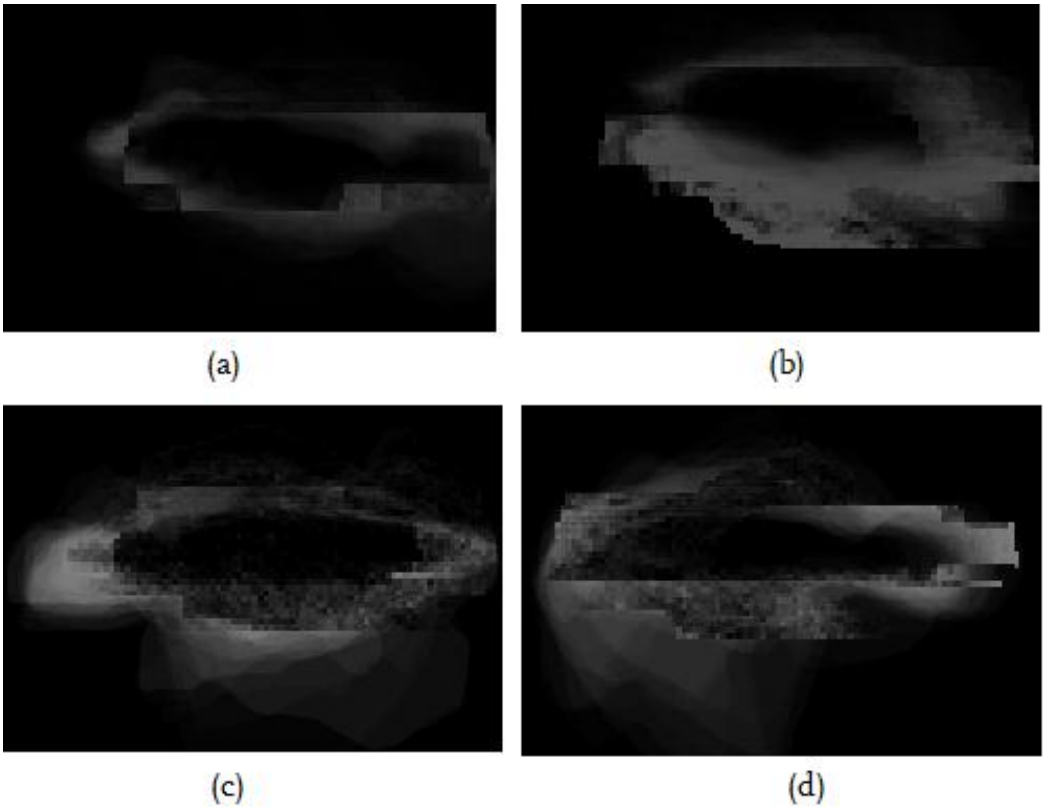


Figure 48: Heat maps indicate the accumulated classification error of all data sets (*inter data set*) (a) *DATA_Home_1* (b) *DATA_Home_2* (c) *DATA_Synt_1* (d) *DATA_Synt_2*

4.1.6 Discussion pre-processing step

The comparison of cross-validation (*inter data set*) results before and after pre-processing is shown in Table 12. It can be seen that average *accuracy* and *precision* improve by adding contrast enhancement as a pre-processing step. By pre-processing the signal strength in image areas near the head and feet is boosted. The features calculated in these areas after pre-processing become more discriminative, thus increasing overall accuracy and precision. This shows that the classifier is not completely independent to variation in illumination (pixel intensity).

Parameter (*intra data set*)	Without pre-processing	With pre-processing
Average Accuracy	86.95	88.46
Average Precision	75.27	79.11

Table 12: Comparison of classifier statistics (with and without) contrast enhancement

4.2 Actigraphy results

The *video actigraph* is generated by calculating the Sum of Absolute Differences (SAD) of all pixels in two consequent frames. The *video actigraph* generation is explained in detail in Section 1.3. The application generates *video actigraph* by performing frame differencing only in the segmented POI area. Hence, the video actigraph is expected to have components only from POI. In Section 4.2.1 and Section 4.2.2, the evaluation of application generated *video actigraph* is presented. Section 4.2.3, presents the analysis of the proposed algorithm (Section 2.9) to compensate for false detection of POI movement caused due to blanket movement by BP.

4.2.1 Comparison of *video actigraph* with *wrist actigraph*

In this section, the generated *video actigraph* is compared with *wrist actigraph*. The evaluation is done separately for video samples in which either POI or BP move at a time. All the video sequences used for evaluation are taken from *DATA_Synt_2*. For POI segmentation, the classifier was trained with all annotated frames from datasets, *DATA_Home_1, DATA_Home_2* and *DATA_Synt_1*.

The comparisons of *video actigraphs* (generated with and without segmentation) with *wrist actigraph* of videos in which only POI is moving are shown in Figure 49, Figure 50 and Figure 51. From these figures, it can be seen that for all peaks in the *wrist actigraph* signal (shown in blue) there are corresponding peaks in the *video actigraph* signal (generated with and without segmentation - marked in green and red respectively). In addition there are extra peaks in *video actigraph* signal which are due to the movement of POI originating from parts of the body other than the wrist. At times, the *wrist actigraph* records a larger peak (movement) than the *video actigraph* peak (movement). These are instances where POI's wrist moves sharply than other POI body parts. The Actiwatch placed on POI's wrist records large *wrist actigraph* signal due to the large acceleration recorded by it. However, when only the wrist of POI moves, the numbers of pixels belonging to POI's wrist are very small in number and the SAD value of these pixels in consecutive frames would result in weak signal in *video actigraph*. In Figure 51, it can be seen that in some instances there is only a *wrist actigraph* signal and no corresponding *video actigraph* signal. These are caused by movements from POI's wrist which are not recorded in the video (POI turned towards BP and POI's hands not visible).

There is also a high correlation between *video actigraph*s generated with (shown in green) and without segmentation (shown in red). This shows that even with segmentation, no movements of POI have been missed out in the *video actigraph* signal (with segmentation).

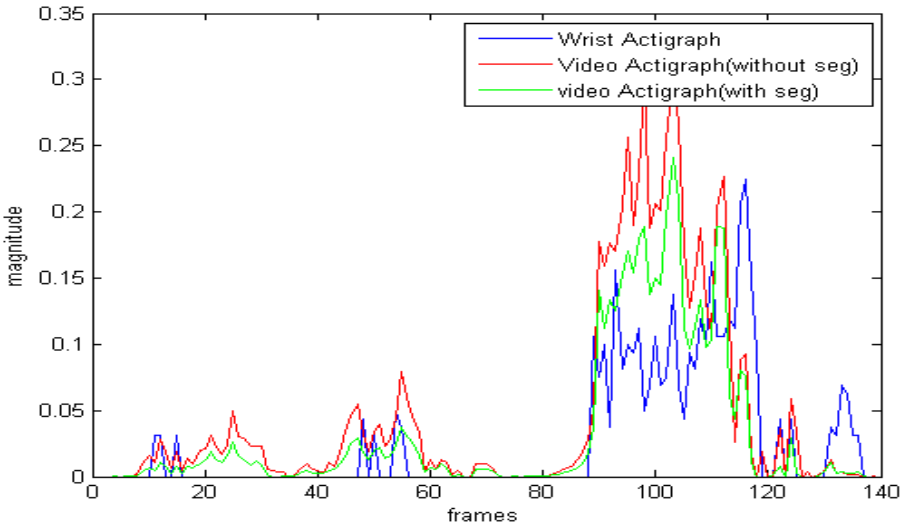


Figure 49: Comparison of *wrist actigraph* with *video actigraph* generated with and without segmentation (only POI moving in the video sequence). There is a high correlation between *video actigraph* generated without segmentation to that of *video actigraph* generated with segmentation.

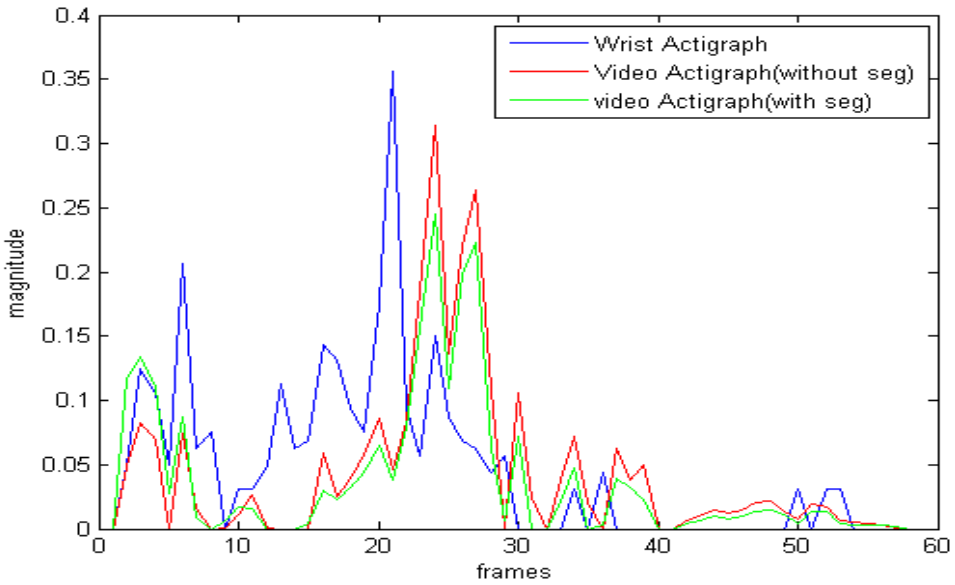


Figure 50: Comparison of *wrist actigraph* with *video actigraph* generated with and without segmentation (only POI moving in the video sequence). Large peaks in *wrist actigraph* than *video actigraph* due to sharp movement of POI's wrist.

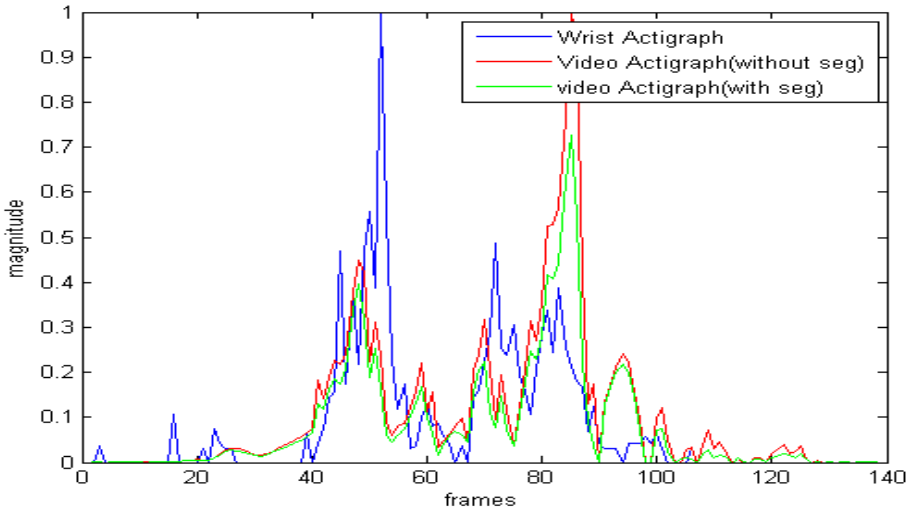


Figure 51: Comparison of *wrist actigraph* with *video actigraph* generated with and without segmentation (only POI moving in the video sequence). Extra peaks in *wrist actigraph* signal are due to POI's wrist movement which are not recorded in the video (POI turned towards BP and POI's hands not visible).

In Figure 52, Figure 53 and Figure 54, the comparison of *video actigraphs* (generated with and without segmentation) with *wrist actigraph* are shown for video sequences in which only the BP is moving. It can be seen from these figures that there is no *wrist actigraph* signal as there is no POI movement in these video sequences. The *video actigraph* (without segmentation - marked in red) signal is due to the BP movement.

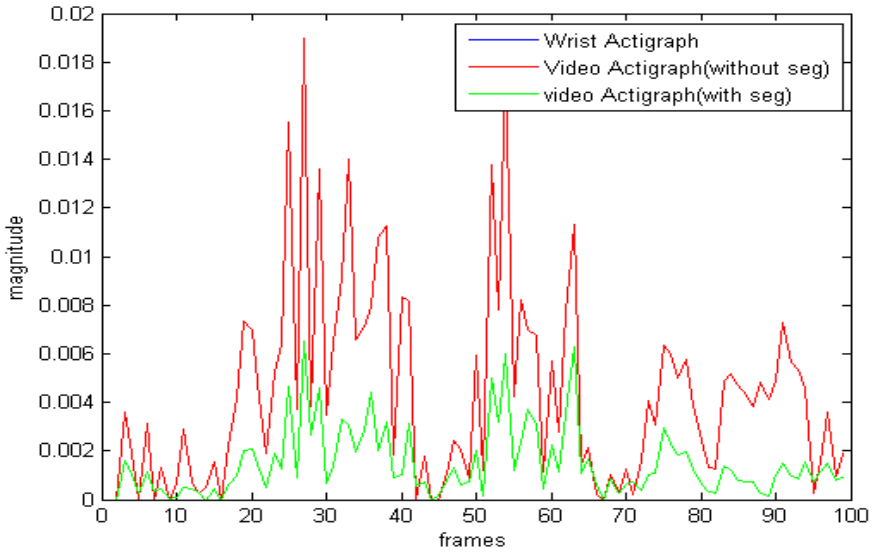


Figure 52: Comparison of *wrist actigraph* with *video actigraph* generated with and without segmentation (only BP moving in the video sequence). In the video sequence, the POI lies in supine position and BP moves without any restrictions. There is no *wrist actigraph* signal as there is no POI movement.

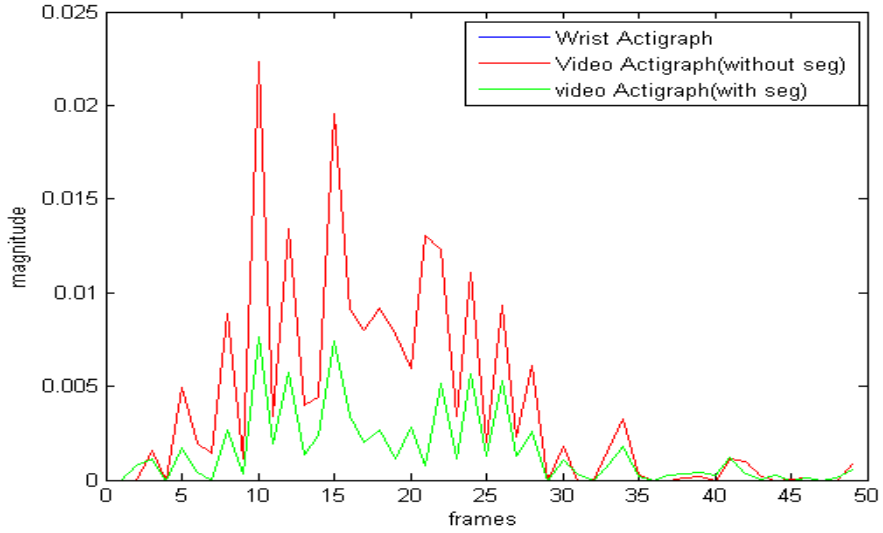


Figure 53: Comparison of *wrist actigraph* with *video actigraph* generated with and without segmentation (only BP moving in the video sequence). In the video sequence, the POI lies in right lateral recumbent position (sideways towards camera) BP moves without any restrictions. The signal strength of *video actigraph* (without segmentation – when only BP moving) is lower than that of *video actigraph* (without segmentation – when only POI moving – as shown in figures 49-51).

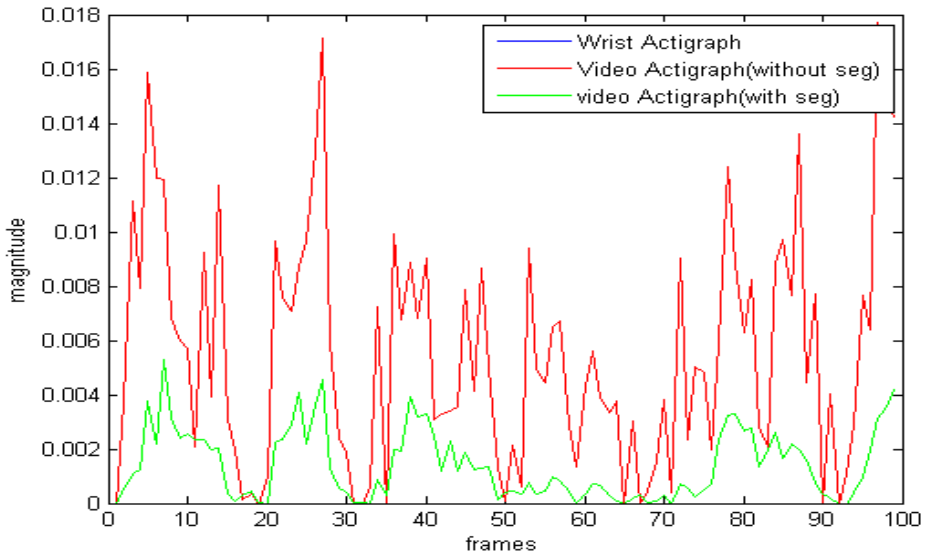


Figure 54 : Comparison of *wrist actigraph* with *video actigraph* generated with and without segmentation (only BP moving in the video sequence). The magnitude of the *video actigraph* signal (without segmentation) is attenuated with segmentation (shown in green).

The *video actigraph* signal (without segmentation) when BP is moving is not as strong as in case of *video actigraph* generated when there is POI movement (as seen in figures 49-51). This is because of the fact that the BP area in the image is weakly illuminated

resulting in low signal strength of image pixels in these regions which in turn evaluates to low SAD value of pixels (*video actigraph signal*) when there is motion. In a well illuminated scene, the *video actigraph* signal (without segmentation) could be strong and comparable to *video actigraph* generated when there is POI movement.

From figures 52-54, it can be seen that the magnitude of the *video actigraph* signal (generated without segmentation - shown in red) is attenuated with segmentation (*video actigraph* generated with segmentation – signal shown in green). By thresholding, the *video actigraph* signal (with segmentation) would correlate to POI movement. The threshold value has been chosen as 0.010 which is lower than the average magnitude (0.025) of the small POI movements. The threshold has been determined from observations made on 40 video sequences.

4.2.2 Accuracy of *video actigraph and wrist actigraph* in comparison to ground truth

In this section, the accuracy of *video actigraph* and *wrist actigraph* in comparison with the ground truth is presented. The ground truth is annotations of POI movements in video sequences. The annotations are binary label (value of '1 (True)' assigned if there is POI movement and value of '0 (False)' if there is either no movement or BP movement in the video sequence). The annotations have been done by two independent experts on 40 video sequences sampled from all data sets each having 5-7 frames.

These annotations (ground truth) are compared to *wrist actigraph* and *video actigraphs* (generated with and without segmentation) and the accuracy is determined. If there is at least a single peak in the *video actigraph* or *wrist actigraph* signal of the video sequences (5-7 frames length), it is assumed to be a POI movement and is labelled 'positive (1)'. In case there are no peaks, it is labelled 'negative (0)'. The accuracy of *wrist actigraph* and *video actigraphs* (generated with and without segmentation) is shown in Table 13.

	Wrist Actigraph	*Video actigraph* (generated without segmentation)	*Video actigraph* (generated with segmentation)
Accuracy (%)	75.00	70.00	85.00

Table 13: accuracy of *wrist actigraph* and *video actigraphs* (generated with and without segmentation)

4.2.3 Video actigraph with suggested blanket compensation

In a shared bed it is common for POI and BP to share a common blanket and there can be instances where the blanket on top of POI moves when it is pulled by BP. This would cause a false signal in *video actigraph* even after segmentation. This type of false detections of POI movements caused due to blanket pulling by BP might be compensated as described in Section 2.9. Figure 55 and Figure 56 show the comparison of *video actigraph* generated with segmentation (shown in red) with *video actigraph* generated with segmentation and blanket compensation (shown in green) in two different video sequences.

Figure 55 shows the comparison where only POI moves in the video sequence. For evaluation, the *video actigraph* generated with segmentation (shown in red) is considered as the ground truth (signal assumed to exactly correspond to POI movements). All the peaks in the *video actigraph* (with segmentation- shown in red) signal correspond to POI movements. It can be seen that even with blanket compensation, most of the *video actigraph* peaks (with segmentation and blanket compensation – shown in green) match with *video actigraph* peaks (with segmentation). This shows that the proposed blanket compensation method does not falsely classify the actual POI movements as false movements caused by blanket pulling by BP.

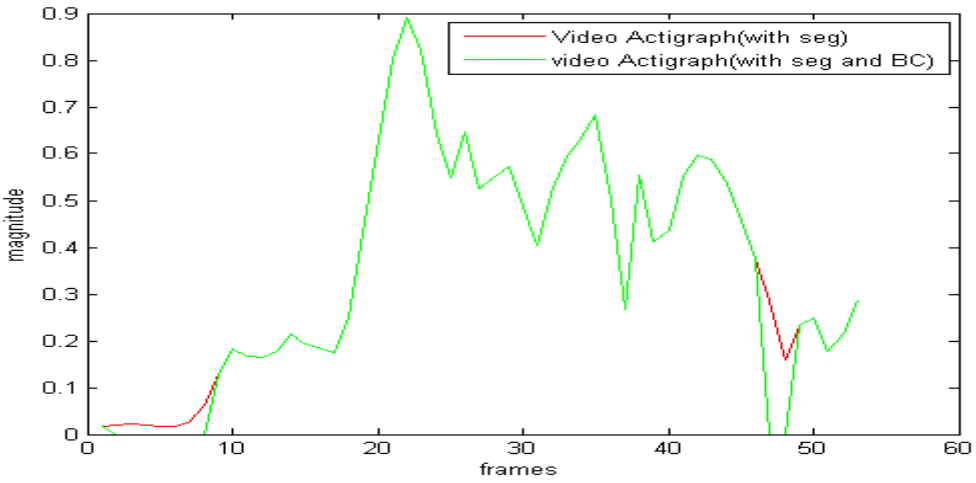


Figure 55: Comparison of *video actigraph* (generated with segmentation – shown in red) with *video actigraph* (generated with segmentation and blanket compensation – shown in green). Only POI moves in the video sequence. Most of the POI movements are correctly classified as POI movements in *video actigraph* (generated with segmentation and blanket compensation).

Figure 56 shows the comparison of *video actigraph* generated with segmentation to that of *video actigraph* generated with segmentation and blanket compensation in video sequence in which only BP moves. The peaks in *video actigraph* (generated with segmentation- shown in red) are due to false POI movements recorded due to blanket pulling by BP. It

can be seen that with blanket compensation (*video actigraph* with segmentation and blanket compensation- shown in green), most of the false POI movements are compensated. The effect of blanket compensation on accuracy of *video actigraph* is shown in Table 14. It can be seen that the accuracy of the *video actigraph* generated with segmentation and blanket compensation is 92.5% (tested on the same 40 sequences as described in Section 4.2.2) which is higher than the *video actigraph* generated (only) with segmentation.

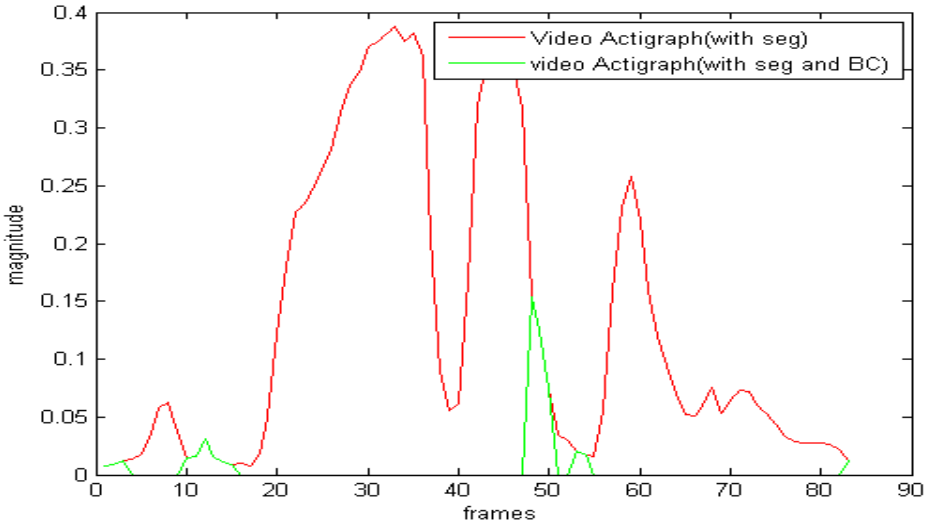


Figure 56: Comparison of *video actigraph* (generated with segmentation – shown in red) with *video actigraph* (generated with segmentation and blanket compensation – shown in green). Most of the false POI movements caused due to blanket pulling by BP are compensated (*video actigraph generated* with segmentation and blanket compensation).

	Video actigraph (generated with segmentation)	*Video actigraph* (generated with segmentation and blanket compensation)
Accuracy (%)	85.00	92.50

Table 14: The accuracy of *video actigraph* increases significantly with blanket compensation

4.3 Discussion

As described in Section 2.8, the classifier was trained by several discriminative features that have been carefully selected. The feature set includes several existing (e.g. HOG, focus measure) and newly developed or adapted features (e.g. brightness, region based motion feature). As expected the key features that have been selected by the classifier are the brightness feature (Section 2.1), focus feature (Section 2.2), location based features

(Section 2.5) and HOG feature (Section 2.4). The illumination of the scene from a single IR light source mounted on top of the camera near the POI head causes the brightness decay from POI towards BP making brightness feature one of the most discriminative features. Location based features gain prominence because of the standard setup which results in POI being centred in the image. The Focus feature (measure of defocus) figures out among the top discriminative features as the camera is made to focus on POI as part of the installation. HOG scores over other gradient features as it describes the boundaries more robustly against variation in illumination due to the intrinsic local contrast normalization.

The classifier has been tested over various data sets and the average precision and accuracy for cross validation (*intra data set*) with pre-processing has been found to be 79.11 and 88.46 (Table 12) respectively. The misclassification of POI segmentation has been found to be less near the POI centre and more around the POI boundary as expected due to the increased difficulty in classification at the boundaries. We believe that the segmentation result can be further improved by using an adaptive contrast enhancement technique that enhances contrast based on the spatial position in the image (more near the image corners close to POI head & legs and less in other parts). The segmentation result can also be improved by building a confidence measure from the temporal component of POI segmentation results (basing the current POI segmentation on past segmentation results – segmentation results from previous frames). The segmentation result seems unaffected by different blanket types (*DATA_Synt_2* having thin blanket and others having thick blanket) and sleeping positions (many different postures of POI and BP have been recorded in data sets *DATA_Synt_1* and *DATA_Synt_2*).

The application generates a *video actigraph* by calculating the sum of absolute differences of pixels belonging to POI in two consequent frames. The accuracy (correctly detecting POI movements) of the application generated *video actigraph* has been found to be 85% (Table 13) compared to 70% accuracy of *video actigraph* generated with full frame differencing and 75% accuracy of *wrist actigraph*. The accuracy of application generated *video actigraph* is found to go up to 92.5% (Table 14) with the suggested blanket compensation method (Section 2.9).

5 Hardware realization

The application should be able to generate the *video actigraph* in real time (at the desired frame rate), preferably while running on low cost commodity hardware. To make the development fasters and easy, we have decided to use an existing prefabricated system on chip (*BeagleBoard*[42]) with on-board power management and ready to use peripherals. In Section 5.1, a brief description of the *BeagleBoard* is given. In Section 5.2, the comparison of execution times of the application on *BeagleBoard* and PC are presented. Section 5.3 discusses the possible optimizations that can be made to the application.

5.1 BeagleBoard

BeagleBoard is a low power open source single board computer produced by Texas Instruments (TI). The *BeagleBoard* has a DM3730 digital media processor[43] which is one of TI's DaVinci series of processors optimized for media applications[44]. The DM3730 has an ARM Cortex A8 CPU [45], a TMS320DM64x+ DSP [46] and a PowerVR SGX graphics accelerator [47]. Apart from these, the DM3730 supports multimedia codecs, accelerators and peripherals that aid the development of optimized digital video applications.

The advantage of having a general purpose processor (ARM) and a digital signal processor (TMS320c64x+) together is that, the control and media portions can both be executed by processors that excel at their respective tasks making it an ideal platform for multimedia applications. The technical specifications of the *BeagleBoard* can be found in *Appendix - C*. The *BeagleBoard* ships with preinstalled '*Angstrom*' Operating System (OS). Angstrom is an open source Linux distribution for embedded devices built using OpenEmbedded build system[48]. The advantages of *Angstrom* operating system (on *BeagleBoard*) are free license, availability of cross-compilation tools, availability of drivers for using peripherals and presence of large developer community for support. The µEye camera module can be connected to *BeagleBoard* easily via the USB port. The proposed final sleep monitoring application is shown in Figure 57.

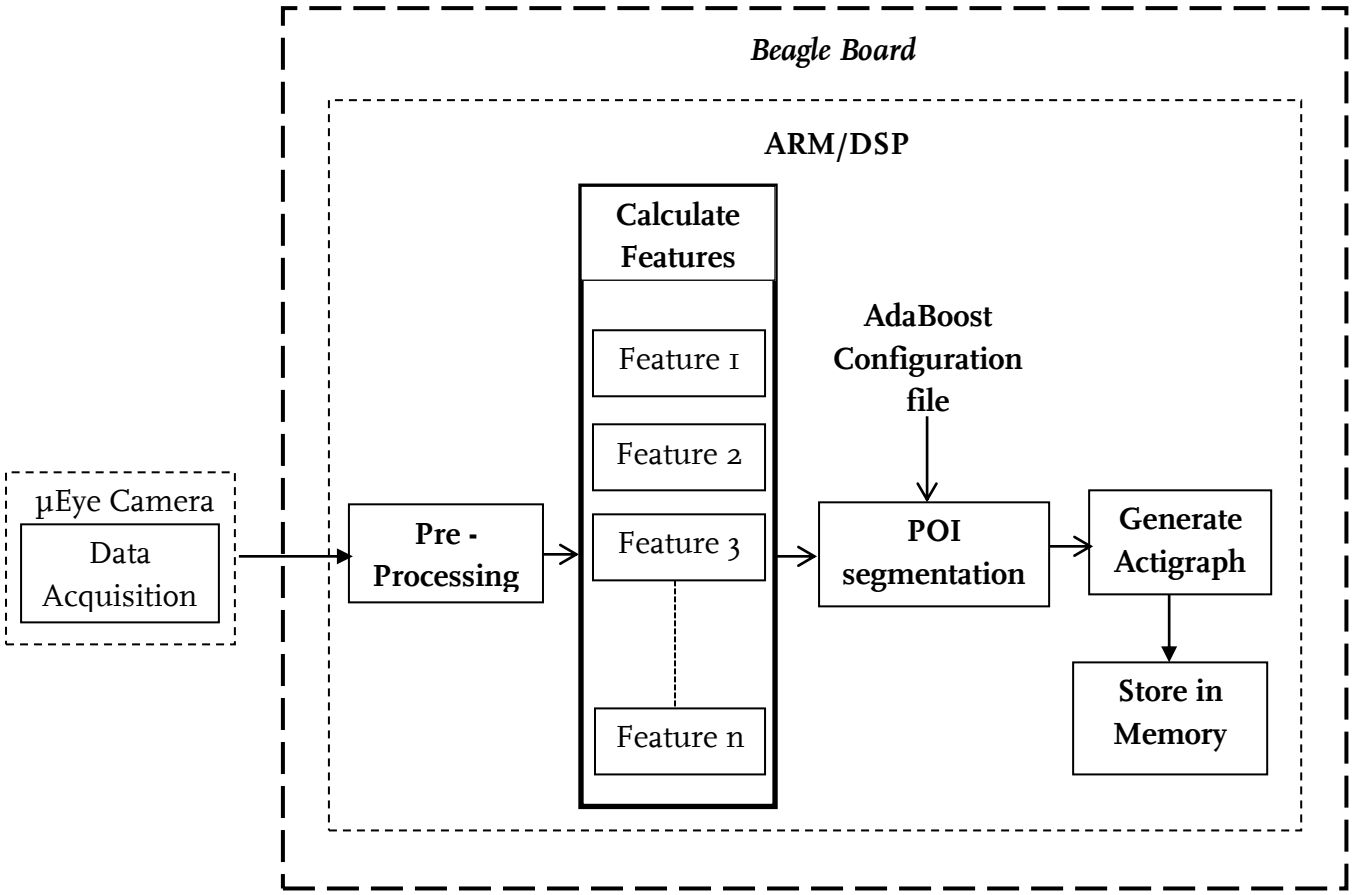


Figure 57: Sleep Monitoring Application Overview

5.2 Comparison of execution times

The sleep monitoring application has been ported to the *BeagleBoard* and the comparison of execution time with a PC is shown in Figure 58. The current implementation on the *BeagleBoard* processes pre-recorded video sequences and the algorithm only runs on the ARM processor of the board.

Among all the features used for segmentation, HOG (Section 2.4), Focus (Section 2.2), RBMF – region based motion feature (Section 2.3) and Brightness (Section 2.1) contribute the major part of the total execution time. To save computation time the features are only calculated for some pixels (one pixel per 8X8 non overlapping blocks) in the full scale image. The result is a sub-sampled feature image for every feature that is used for classification. The classification result is a sub-sampled image with positive (POI) and negative (background) labels. The sub-sampled classified image is up scaled to full scale image by pixel repetition (all pixels in the block are assigned the same pixel value) to get POI area estimation in the full scale image. In Figure 58, the comparison of execution times (per frame) of the (computationally intensive) features on *BeagleBoard*

with a PC is presented. Currently the application has an epoch length of 11.27 seconds (able to process one frame for every 11.27 seconds).

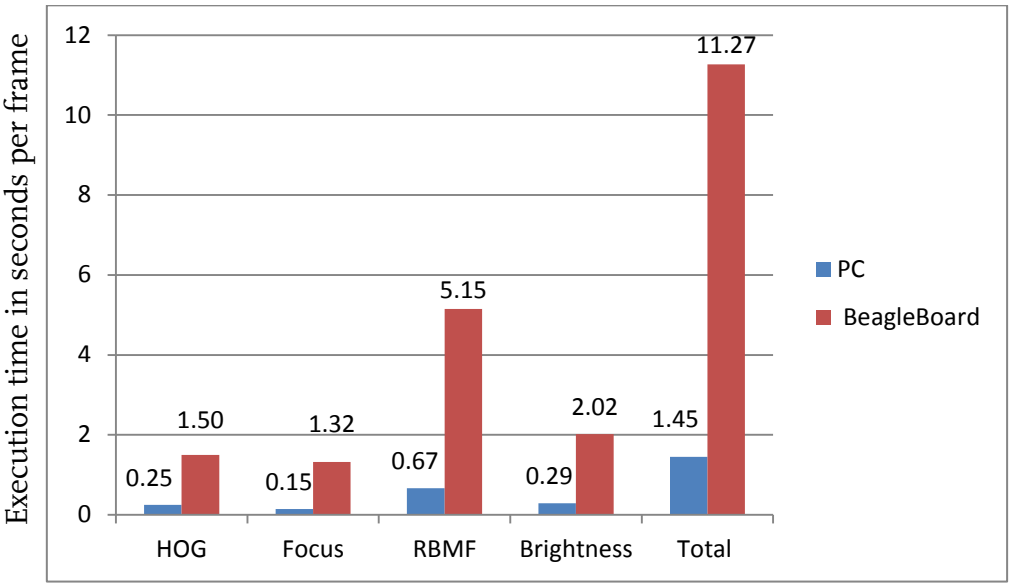


Figure 58: Comparison of execution times (PC vs BeagleBoard)

5.3 Possible optimizations

The current classifier implementation requires all the features to be calculated for all the pixels prior to the classification. An optimized implementation which only calculates the necessary features required at a particular cascade level of the classifier and only for pixels that have been passed onto the current cascade layer (True positives and False positives) from previous cascade layer would improve the execution time significantly.

Most of the methods (Focus, Edge Detectors-Gradient Features) discussed in Chapter 2, use MAC instructions. The parallel computation power of the DSP would reduce the computational time required by them and would bring the application a step closer in meeting the real time requirement (ability to process frames at the required frame rate). However, in some cases it may turn out that some parts of the algorithm would execute faster on the CPU (ARM) than on DSP. This would be due to the communication overhead between ARM and DSP as both the cores are connected via L3 and L4 interconnects on DM3730. During task mapping, the communication overhead has to be considered and care should be taken that the performance gained by mapping a task on DSP should not be dominated by the loss of performance due to the communication overhead. *Angstrom* supports the setup of *DSPLink* that enables the usage of DSP and the procedure can be found in [49].

Apart from using DSP, the NEON extension of ARM can also be used to improve execution time. *NEON* or *MPE* (Media Processing Engine) is an advanced SIMD (Single Instruction Multiple Data) technology [50]. The communication overhead between ARM and NEON may be negligible as they communicate via the L1 interconnect. Features which require frequent data fetching and which have small MAC computations would

perform better on NEON than on DSP due to low communication overhead. The gradient computations with edge operators can be suitable candidates for running on NEON. Motion estimation which requires frequent random fetches can also be a suitable candidate. The use of *NEON* coprocessor on the *BeagleBoard* is demonstrated for *Angstrom* in [51].

6 Conclusion and future work

This thesis details the design and implementation of a sleep monitoring application that is able to distinguish movements between Person of Interest and Bed Partner. The application gets the video data from a camera placed on a bed side table and generates a *video actigraph* that measures the gross motor activity (movements) of person of interest. The *video actigraph* generated can be used to assess the overall sleep quality of a person and to diagnose various sleep related disorders. The *video actigraph* is generated by computing the sum of absolute differences of all pixels in two consequent low-pass filtered images (frames). The extensions are, segmenting POI area from the image and performing frame differencing only in the POI area of the image.

The segmentation algorithm developed uses AdaBoost, a supervised machine learning technique. The classifier uses several weak features together to segment POI area from the image. The feature set includes several existing (e.g. HOG, focus measure) and newly developed or adapted (e.g. brightness, region based motion feature) features. The segmentation algorithm can be broadly divided into two phases, training and testing. In the training phase, the classifier is trained with samples from various data sets that are recorded in various illumination conditions. The training set consists of feature values and labels (ground truth) for every training sample. Each sample represents a pixel from a frame and hence the annotations (ground truth) have to be pixel accurate. The annotations were performed by using ITK-SNAP tool. 240 frames from four data sets have been annotated and are used for training the classifier.

In the testing phase, the classifier configuration file generated from the training phase is used to classify positive and negative samples. Positive samples represent POI area in the image and negative samples the background. The resultant POI segmentation is evaluated using *K*-fold cross-validation method. The average segmentation accuracy and precision are 86.95 and 75.27 respectively for tested samples from a different data set than that used for training.

The *video actigraph* generated after segmentation has been compared to ground truth. The ground truths are labels for short video sequences (5-7 frames length. The accuracy of detecting POI movements after segmentation has been found to be 85% which is greater than the 75% accuracy achieved by actiwatch. A blanket compensation methodology to detect false POI motions that have been recorded due to blanket pulling by BP has been presented. With segmentation and blanket compensation, the application can detect POI movements up to accuracy of 92.5%.

The sleep monitoring application developed is aimed at a stand-alone consumer product and has to have components that are cheap and embedded in nature. The entire application consists of a visual system to acquire video data and an embedded computing unit to process the data and generate *video actigraph*. The visual system consists of a µEye CMOS camera with a fish eye lens and IR light. The compact size of the camera, the USB interface, its wide resolution and easy control of internal parameters makes it suitable for its use with the embedded platform. The use of fish eye lens, greatly increases the view

angle of the camera thus making it possible to observe POI completely from head to foot. The embedded computing unit selected for the current project is the BeagleBoard from TI which has interfaces to connect to the camera and other peripherals like memory card, display device when required. The BeagleBoard has 1 GHz ARM processor and a DSP to satisfy the performance needs for a real-time application. An implementation which processes pre-recorded video sequences has been ported onto the BeagleBoard and the comparison of execution time with a PC has been documented.

Future Work

The segmentation accuracy has been observed to be poor near the foot and head parts of POI. This is due to the weak illumination in these areas. To overcome this, a light source capable of illuminating these areas within the safety limits has to be investigated. The segmentation accuracy in these areas can also be improved by using a suitable contrast enhancement technique that boosts the contrast in these areas without affecting the pixels in the background. An adaptive technique that enhances contrast based on spatial position in the image can be an interesting option to investigate. We believe that the segmentation result can be improved by building a confidence measure from the temporal component of POI segmentation results (basing the current POI segmentation result on the past segmentation results from previous frames).

It has been observed that location based features gain prominence when trained with data sets having similar placement of the camera. This is because in data sets with similar setup, POI appears to be localized to certain part in the image. The segmentation accuracy degrades when tested with data sets having different camera placement. The classifier has to be trained over a larger data set to make it more robust against such variations in camera placement or necessary camera alignment support has to be made available for the product installation.

An efficient implementation of the application can be achieved by exploiting the multi-core architecture of the digital media processor on BeagleBoard. Most of the features being used for segmentation are independent and can be computed in parallel. Focus feature calculation has complex MAC calculations and implementing it on DSP may gain performance. The SIMD extension on ARM can be used for parallel computation of other features that have small window operations using the NEON instruction set available for *BeagleBoard*.

The classifier implementation used for the current project requires all features to be calculated for all samples. A modified implementation in which features are calculated only for the samples that are being passed onto the next stage of the classifier can gain significant performance.

7 References

[1] K. Bloch, "*Polysomnography: a systematic review*," Technol Health Care, pp. 285-305, 1997.

[2] W. Liao and C. Yang, "*Video-based activity and movement pattern analysis in overnight sleep studies*," 19th International conference on pattern recognition, vols 1-6, pp. 1774-1777, 2008.

[3] A. Muzet, "*Dynamics of body movements in normal sleep*," in Sleep '86: proceedings of the eighth European congress on sleep research, Szeged (Hungary), pp. 232-234, 1988.

[4] B. Jansen and K. Shankar, "*Sleep staging with movement-related signals*," International Journal of bio-medical computing, vol. 32, pp. 289-297, 1993.

[5] C. Wang, A. Hunter, N. Gravill, and S. Matusiewicz, "*Real time pose recognition of covered human for diagnosis of sleep apnoea*," Computerized medical imaging and graphics, vol. 34, pp. 523-533, 2010.

[6] B. C. L. Richard Reily, T, "*Electrograms (ECG, EEG, EMG, EOG)*," Technology and health care, vol. 18, pp. 443-458, 2010.

[7] L. Pei, A. Heinrich, and C. Shan, "*Background segmentation from depth estimation for sleep analysis*," Philips research Europe, Technical note PR-TN 2009/00924, 2009.

[8] "*FACTS - Sleep disorders,*" Available: http://www.cmu.edu/qolt/Foundry/documents/sleepmedia_factsheet.pdf

[9] J. Hedner, G. Pillar, S. D. Pittman, D. Zou, L. Grote, and D. P. White, "*A novel adaptive wrist actigraphy algorithm for sleep-wake assessment in sleep apnea patients*," Sleep, vol. 27, pp. 1560-1566, 2004.

[10] Y. Sivan, A. Kornecki, and T. Schonfeld, "*Screening obstructive sleep apnoea syndrome by home videotape recording in children*," European respiratory journal, vol. 9, pp. 2127-2131, 1996.

[11] K. Nakajima, Y. Matsumoto, and T. Tamura, "*A monitor for posture changes and respiration in bed using real time image sequence analysis*," in 22nd Annual International conference of the IEEE-Engineering-in-Medicine-and-Biology-Society, Chicago, pp. 51-54, 2000.

[12] F. C. Yang, C. H. Kuo, M. Y. Tsai and S. C. Huang , "*Image-based sleep motion recognition using artificial neural networks*," Proceedings of the 2003 International conference on machine learning and cybernetics, vols 1-5, pp. 2775-2780, 2003.

[13] C.W. Wang, A. Ahmed and A. Hunter, "*Locating the upper body of covered humans in application to diagnosis of obstructive sleep apnea*," World congress on engineering 2007, vols 1 and 2, pp. 662-667, 2007.

[14] D. Ramanan and D. A. Forsyth, "*Finding and tracking people from the bottom up*," in Conference on computer vision and pattern recognition, Madison, Wi, pp. 467-474, 2003.

[15] B. Wu and R. Nevatia, "*Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors*," International journal of computer vision, vol. 75, pp. 247-266, 2007.

[16] J. Deutscher and I. Reid, "*Articulated body motion capture by stochastic search*," International journal of computer vision, vol. 61, pp. 185-205, 2005.

[17] C. Sminchisescu, A. Kanaujia, L. Zhiguo, and D. Metaxas, "*Discriminative density propagation for 3D human motion estimation*," in IEEE computer society conference computer vision and pattern recognition, vol. 1, pp. 390-397, 2005.

[18] W. H. Liao and D. Y. Li, "*Homomorphic processing techniques for near-infrared images*," IEEE international conference on acoustics, speech, and signal processing, vol. III, pp. 461-464, 2003.

[19] J. Chase, F. Agogue, C. Starfinger, Z. Lam, G. Shaw, A. Rudge, and H. Sirisena, "*Quantifying agitation in sedated ICU patients using digital imaging*," Computer methods and programs in biomedicine, vol. 76, pp. 131-141, 2004.

[20] A.Heinrich and H.C.v.Vugt, "*Non-contact analysis of body movement during sleep with video actigraphy*," Philips research, Technical note PR-TN 2010/00395.

[21] "*TI Chronos : eZ430,"* Available : http://www.ti.com/lit/ug/slau292c/slau292c.pdf

[22] M. Bichsel and A. P. Pentland, "*A simple algorithm for shape from shading*," in Proceedings of the IEEE computer society conference on computer vision and pattern recognition, pp. 459-465, 1992.

[23] J. Oliensis, "*Uniqueness in shape from shading*," International journal of computer vision, vol. 6, pp. 75-104, 1991.

[24] J. Oliensis, "*Shape from shading as a partially well-constrained problem*," in Proceedings of the IEEE computer society conference on computer vision and pattern recognition, vol- I, pp. 559-564, 1991.

[25] J. Oliensis, P. Dupuis, and B. C. Vemuri, "*Direct method for reconstructing shape from shading*," Geometric methods in computer vision, vol. 1570, pp. 116-128, 1991.

[26] D. Tsai and H. Wang, "*Segmenting focused objects in complex visual images*," Pattern recognition letters, vol. 19, pp. 929-940, 1998.

[27] A. Pentland, "*A new sense for depth of field*," IEEE transactions on pattern analysis and machine intelligence, vol. 9, pp. 523-531, 1987.

[28] S. Lai, C. Fu and S. Chang, "*A generalized depth estimation algorithm with a single image*," IEEE transactions on pattern analysis and machine intelligence, vol. 14, pp. 405-411, 1992.

[29] W. Tsai, "*Moment-preserving thresholding- a new approach*," Computer vision graphics and image processing, vol. 29, pp. 377-393, 1985.

[30] R. C. W. Gonzalez and R.E. Woods, Chapter 10: Image segmentation ,"*Digital Image Processing,"* MA: Addison-Wesley, 1992.

[31] J. Badenas, J. Sanchiz, and F. Pla, "*Motion-based segmentation and region tracking in image sequences*," Pattern recognition, vol. 34, pp. 661-670, 2001

[32] G. de Haan, P. W. A. C. Biezen, H. Huijgen, and O. A. Ojo, "*True-motion estimation with 3-D recursive search block matching*," IEEE transactions on circuits and systems for video technology, vol. 3, pp.368-379, 1993.

[33] G. d. Haan, "*Digital video post processing,"* 2010.

[34] X.Jin, A.Heinrich, C.Shan, H. C. v. Vugt, and G. d. Haan, "*Two-person segmentation in a shared bed by motion estimation*," Philips research Europe, Technical note PR-TN 2010/00305, 2010.

[35] N. Dalal, B. Triggs, C. Schmid, S. Soatto, and C. Tomasi, "*Histograms of oriented gradients for human detection*," Proceedings of the IEEE computer society conference on computer vision and pattern recognition, vol 1, pp. 886-893, 2005.

[36] P. Viola and M. Jones, "*Robust real-time object detection*," International journal of computer vision, 2001.

[37] Y. Freund and R. E. Schapire, "*A decision-theoretic generalization of on-line learning and an application to boosting*," Journal of computer and system sciences, vol. 55, pp. 119-139, 1997.

[38] Y. S. Freund and Robert E. , "*Experiments with a new boosting algorithm*," ICML, pp. 148-156, 1996.

[39] "*µEye camera*," Available: http://www.ids-imaging.com/frontend/products.php?cam_id=10

[40] "*Fish eye lens : Fujinon FE185C086HA-1,"* Available: http://www.fujinonbroadcast.com/Security/Product.aspx?cat=1019&id=1072

[41] "*ITK- SNAP,"* Available: http://www.itksnap.org/pmwiki/pmwiki.php

[42] "*BeagleBoard-xM system reference manual*," Available: http://beagleboard.org/static/BBxMSRM_latest.pdf

[43] "*DM3730, DM3725 Digital media processors*," Available: http://www.ti.com/lit/ds/symlink/dm3730.pdf

[44] "*TI DaVinci digital processors*," Available: http://www.ti.com/lsds/ti/dsp/platform/davinci/device.page

[45] "*ARM Cortex A8*," Available: http://www.arm.com/products/processors/cortex-a/cortex-a8.php

[46] "*TMS32064x+ DSP*," Available: http://www.ti.com/product/tms320c6416

[47] "*PowerVR SGX530*," Available: http://www.imgtec.com/powervr/sgx_series5.asp

[48] "*OpenEmbedded buils system*," Available: http://www.openembedded.org/wiki/Main_Page

[49] "*Installing DSPLink on BeagleBoard*," Available: http://ossie.wireless.vt.edu/trac/wiki/BeagleBoard_DSPLink

[50] "I*ntroducing NEON*," ARM, Development article, 2009.

[51] "*BeagleBoard demo image*," Available: http://www.angstrom-distribution.org/beagleboard-demo-image-available

8 Appendices

A. Document - Recording Protocol

Sleep Monitoring

Experiment – Video Recording Guideline – Protocol for Home Sequence

This document describes the experimental setup for recording video sequences at home for the sleep monitoring project. Please read the document before you actually start with the setup. Contact, Bhargava Puvvula on **xxxxxxxxx** if you have any queries.

Equipment checklist:

- ☐ µEye camera with fish eye lens
- ☐ Tripod (with extension handle)
- ☐ Actiwatch(2 No's),Chronos watch (1 No.)
- ☐ Ruler
- ☐ Silent platform with adapter
- ☐ USB cable (2 No's)
- ☐ IR light source with power cord

Procedure:

1) **Assembling the Silent platform:**
   - The silent platform has two units, the CPU (Figure 59) and the monitor (Figure 60) which is to be connected together with the USB cable.

Figure 59: Silent Platform – CPU

Figure 60: Silent platform - Monitor

Note: 1. The Micro-USB port of the monitor is under the cover as marked in Figure 60.
2. The monitor should be connected to one of the USB 2.0 ports (marked in Figure 61) of the CPU.

- Connect the AC adapter to the power port (Figure 61) of the CPU.

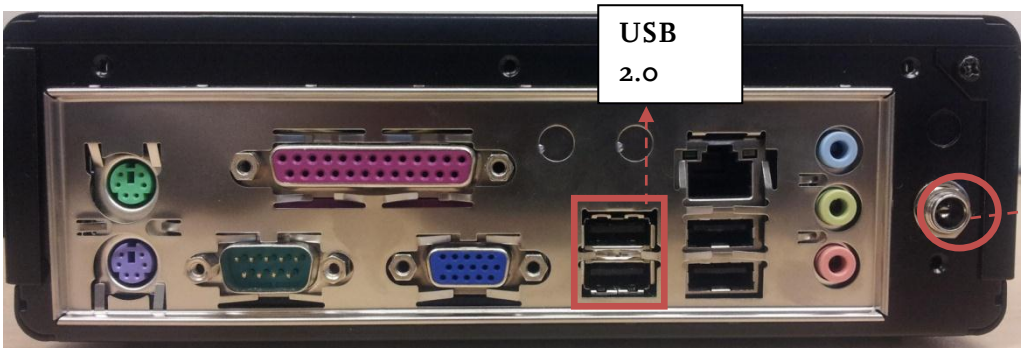


Figure 61: Rear view of the CPU

- Don't place the monitor on top of the CPU. Place it next to the CPU.
- Don't turn on the CPU yet.

2) The µEye camera with the fish eye lens connected looks as in Figure 62.

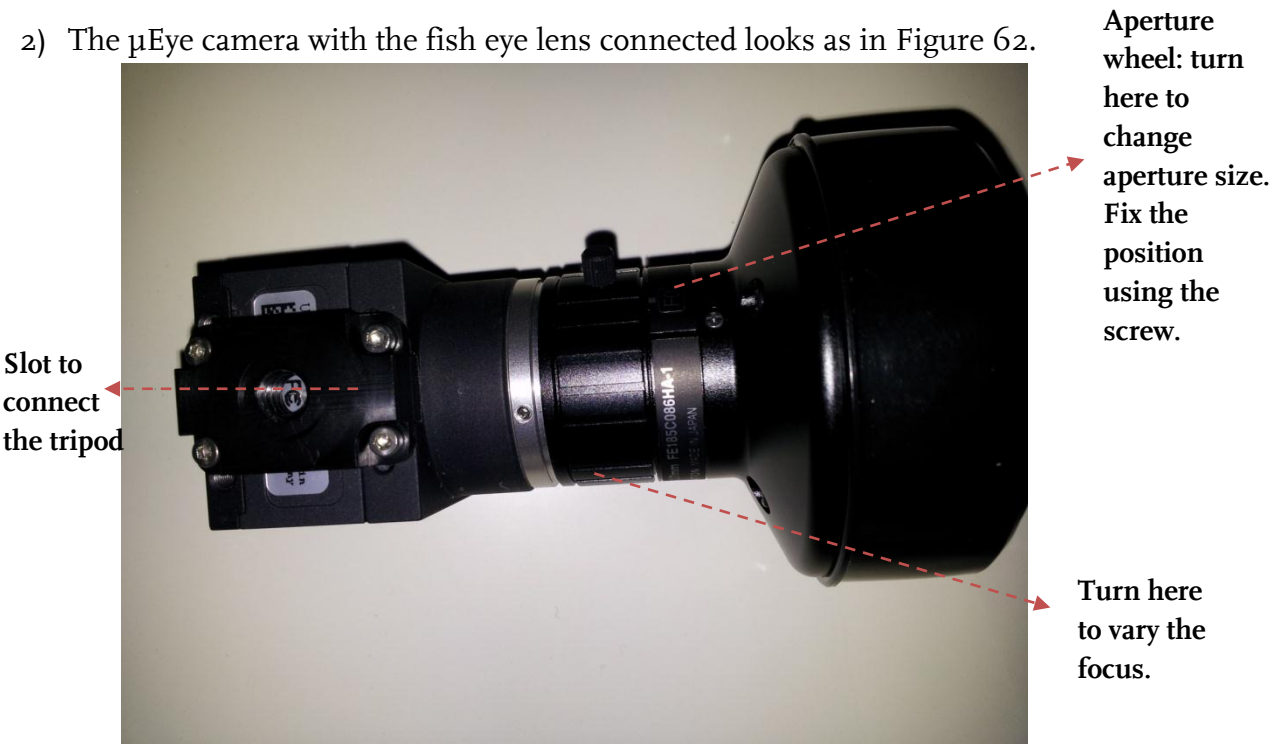


Figure 62: Camera with Fish eye lens

3) Fix the camera on the tripod (along with the extension handle) and mount it in place of bed side table (refer diagram 1 and 2) as shown in Figure 63. The height of the camera (h_c) should be 25 cms more than the height of the bed (h_b) and the distance between the camera and the bed (h_{bc}) should be 30-35 cms as shown in diagram 1.

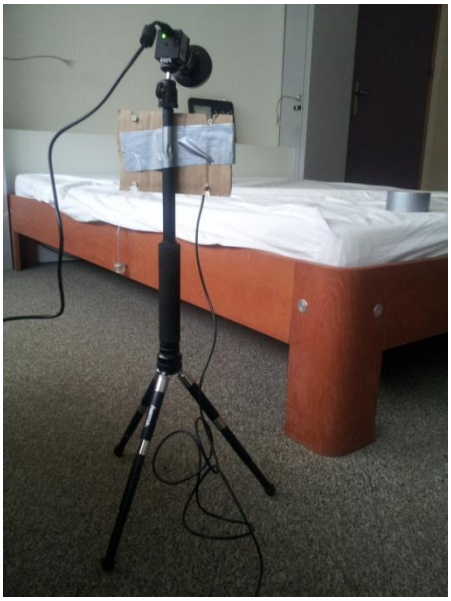


Figure 63: Camera fixed to the tripod and mounted in place of bed side table.

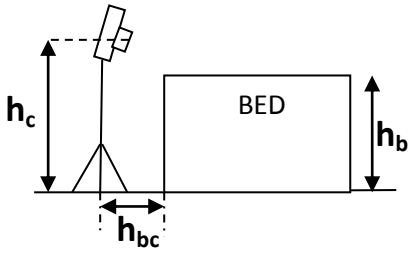
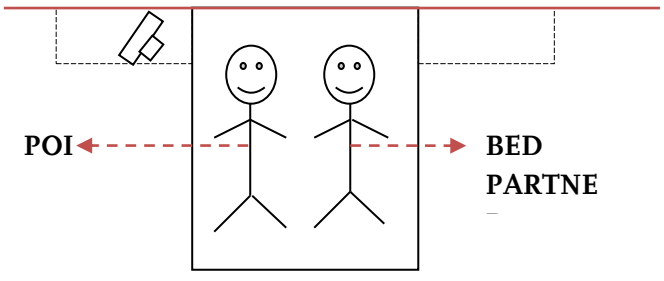


Diagram 1

Measurements

$h_c - h_b = 25$ cms
$h_{bc} = 30 - 35$ cms

4) Orient the camera in such a way that the Person of Interest (POI) is focused as shown in diagram 2.



The camera should not be placed behind the red line.

Diagram 2

NOTE: The focus can be varied by rotating the lens (as shown in Figure 62).

5) Connect the camera to the USB 2.0 port (marked in Figure 61) of the silent platform using the USB cable, if required use the extension cables.

6) Once the camera is connected, press the power button (marked in Figure 64) on the CPU. Wait for the monitor to show the application window as shown in Figure 65.

Figure 64 : Front view of CPU

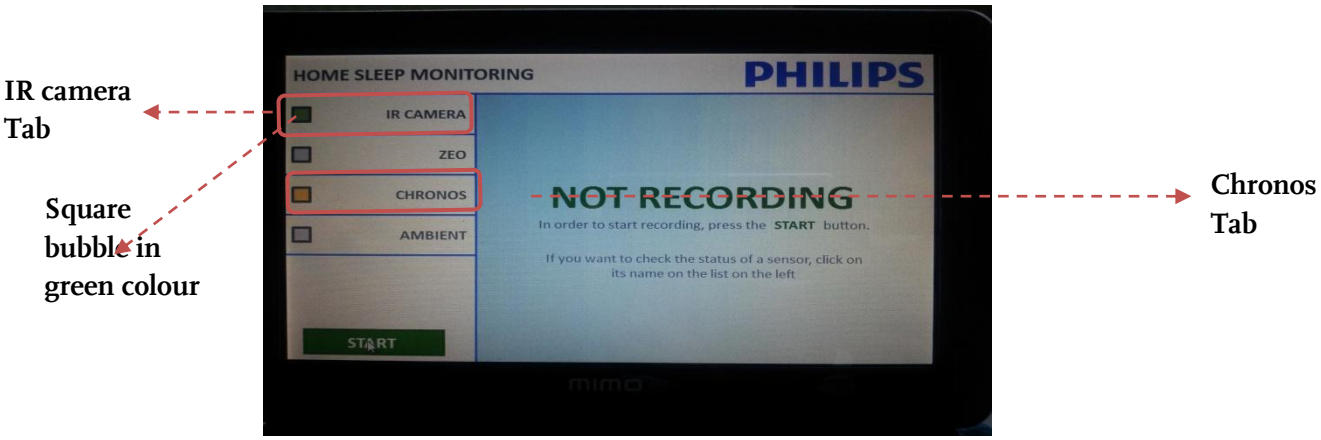


Figure 65 : Application Window - Not Recording

Note: Green colour on the square icon of the IR camera tab shows that the camera has been connected properly. If it shows other colour, please shutdown the silent platform and restart it using the power button (marked in Figure 64) on the CPU.

7) Press the IR camera tab of the application as marked in Figure 65. The video feed from the camera should be visible as shown in Figure 66.

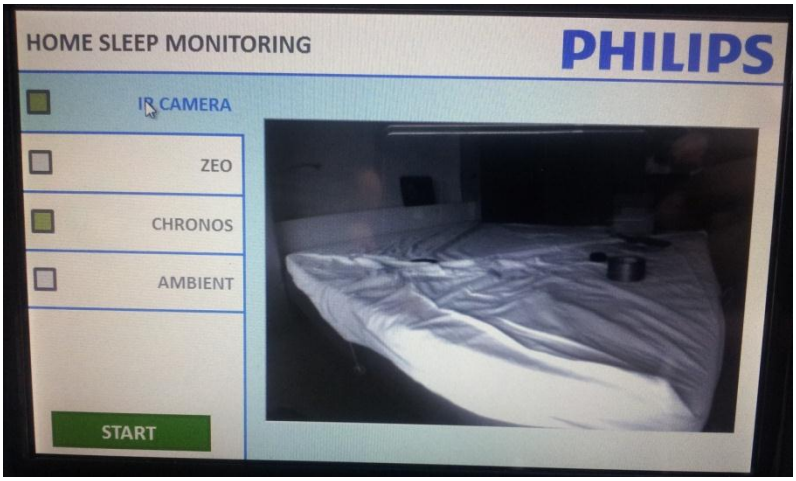


Figure 66 : Video feed from the camera

8) Adjust the orientation of the camera such that the complete bed and a significant area above the bed (Lower end and Upper end) are visible as shown in figure 67.

Figure 67 : Desired view of the recorded image.

9) Plug in the IR light source (marked in Figure 68) and mount it on the tripod below the camera. The direction of illumination should be adjusted such that both the upper and lower ends of the bed can be illuminated.

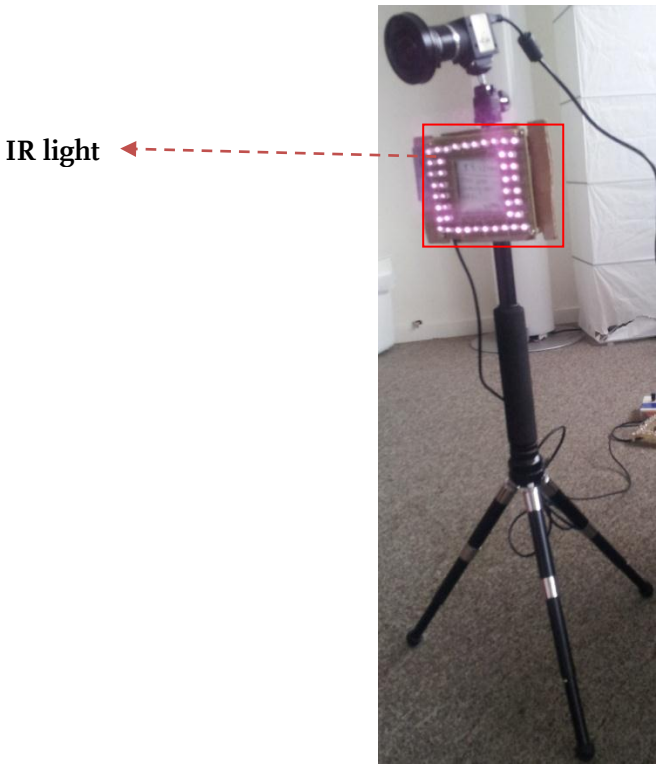


Figure 68 : Tripod with IR light source

Caution: Don't look directly into the IR light source.

10) Turn off the lights and check the video feed displayed on the application window (Figure 66) for proper illumination of both upper and lower ends of the bed. Change the direction of the IR light source accordingly. Figure 69 shows the desired Image.

Figure 69: Desired illumination of the recorded image.

NOTE: It can be noticed that the upper end of the bed generally looks brighter than the lower end of the bed- (the user need not worry about this).

NOTE: The aperture size can be varied by rotating the aperture wheel (as shown in Figure 62).By increasing the aperture size, the image appears brighter. (Make sure that the aperture size is optimal, such that there are no brightness patches which blur the details in the image).

NOTE: The brightness of the IR light source can be changed by rotating the knob on the power cord. There is a marking that shows the optimal position of the knob in our test bedroom. Please adjust it such that the image is clear in your bedroom.

11) Chronos watch setup: The chronos watch is shown in Figue 70.

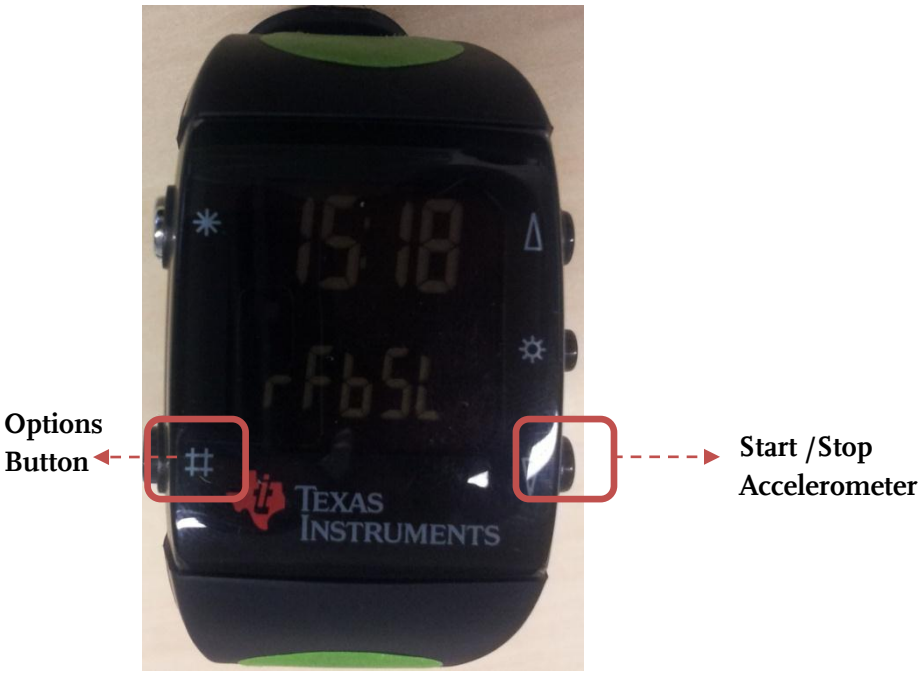


Figure 70 : Chronos Watch

- Wear the chronos watch (POI only) on the non-dominant arm (on left wrist if you are right handed, on right wrist if you are left handed)
- Press the # (Options Button) untill the display shows "**ACC**" below the time.
- Press the ∇ (Start/Stop Accelerometer) button. The square Icon on the chronos tab (marked in Figure 13) should turn to green color.
- To verify the working of the accelerometer, press the chronos tab on the application window. The application window should show the accelerometer graph as shown in Figure 71.

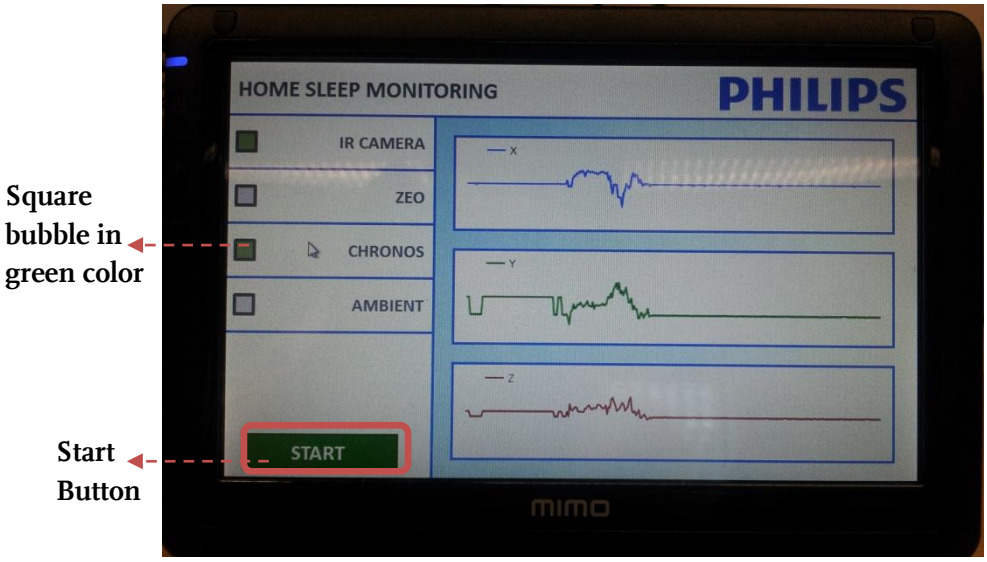


Figure 71 : Accelerometer graph shown after pressing the chronos tab

12) POI should wear the Actiwatch on the dominant arm (on left wrist if you are left handed, on right wrist if you are right handed) and the bed partner should wear the Actiwatch on the non-dominant arm (on left wrist if you are right handed, on right wrist if you are left handed)
13) After the setup, press the start button on the application window to start the recording (marked in Figure 71). The application window should look like as in Figure 72.

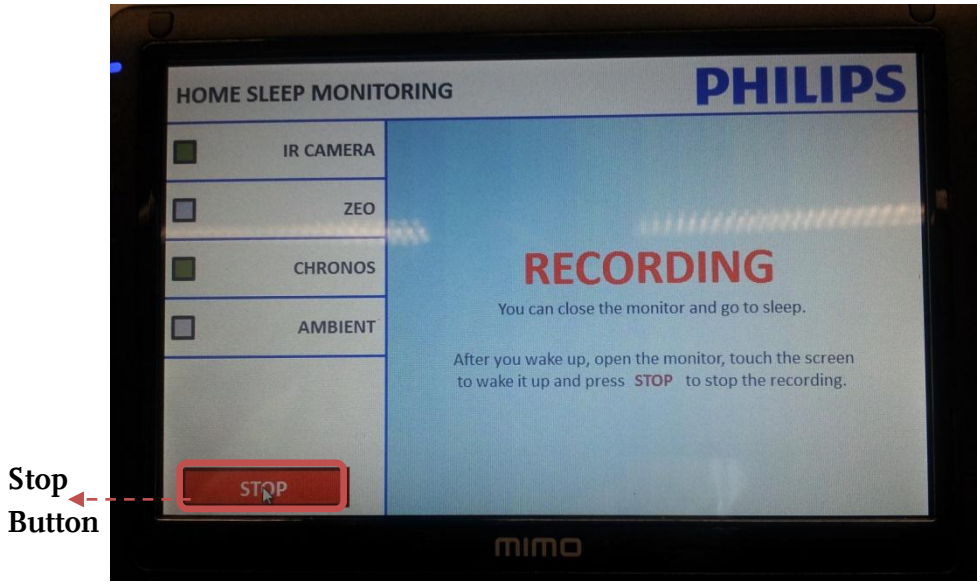


Figure 72 : Application window when recording

14) Synchronizing the Actiwatch and the video: after you are in bed, place your hands above the blanket (with the lights turned on) such that the light is registered by the light sensor on the Actiwatch. Turn off the lights after 30 seconds and press the marker button on the Actiwatch.

Note: Please remember that both the persons (POI and the bed partner) should share the same blanket.

15) Please do not watch TV in bed.
16) Please be careful with the cables lying on the ground.
17) In the morning,
    - Turn on the lights and place your hands above the blanket for 30 seconds such that the light is registered by the light sensor on the Actiwatch. Press the marker on the Actiwatch.
    - Stop the chronos watch (by pressing the ∇ button).
    - Stop the recording by pressing the stop button (marked in Figure 72) on the application window.
    - Shut down the silent platform by pressing the power button (marked in Figure 64) on the CPU

Thank you for volunteering for the experiment. We sincerely appreciate your effort.

Enjoy your sleep.

B. µEye specification

Interface	USB
Lens Mount	S-Mount
Sensor Technology	CMOS (Aptina)
Model Description (color)	UI-1226LE-C
Model Description (mono)	UI-1226LE-M
Resolution (h x v)	752 x 480
Resolution Depth	8bit (10bit ADC)
Resolution Category / Pixel Class	Wide VGA
Sensor Size	1/3"
Shutter	Global
max. fps in Freerun Mode	87 fps
Exposure Time in Freerun Mode	80µs-5.5s
Exposure Time in Trigger Mode	80µs-5,5s
AOI Modes	horizontal + vertical
Binning Modes	horizontal + vertical
Subsampling Modes	---
I/O - Strobe	1
I/O - Trigger	1
I/O - RS-232	-
I/O - GPIO	2
I/O - I2C	
HDR (High Dynamic Range)	-
Sensor Model (m/c)	MT9V032STM / MT9V032STC
Pixelpitch in µm	6.00
Optical Size	4.512 x 2.880 mm
Dimensions H/W/L	36.00 mm, 36.00 mm, 20.00 mm
Mass	12.00 g
Power Supply	USB

C. *BeagleBoard Specification*

	Feature	
Processor	Texas Instruments Cortex A8 1GHz processor	
POP Memory	Micron 4Gb MDDR SDRAM (512MB) 200MHz	
PMIC TPS65950	Power Regulators	
	Audio CODEC	
	Reset	
	USB OTG PHY	
Debug Support	14-pin JTAG	GPIO Pins
	UART	3 LEDs
PCB	3.1" x 3.0" (78.74 x 76.2mm)	6 layers
Indicators	Power, Power Error	2-User Controllable
	PMU	USB Power
HS USB 2.0 OTG Port	Mini AB USB connector	
	TPS65950 I/F	
USB Host Ports	SMSC LAN9514 Ethernet HUB	
	4 FS/LS/HS	Up to 500ma per Port if adequate power is supplied
Ethernet	10/100	From USB HUB
Audio Connectors	3.5mm	3.5mm
	L+R out	L+R Stereo In
SD/MMC Connector	MicroSD	
User Interface	1-User defined button	Reset Button
Video	DVI-D	S-Video
Camera	Connector	Supports Leopard Imaging Module
Power Connector	USB Power	DC Power
Overvoltage Protection	Shutdown @ Over voltage	
Main Expansion Connector	Power (5V & 1.8V)	UART
	McBSP	McSPI
	I2C	GPIO
	MMC2	PWM
2 LCD Connectors	Access to all of the LCD control signals plus I2C	3.3V, 5V, 1.8V
Auxiliary Audio	4 pin connector	McBSP2
Auxiliary Expansion	MMC3	GPIO,ADC,HDQ