EPICS

Image Processing and Analysis

Sleep Team Final Report

Spring 2016, April 29 2016

David Tang, Erich Fischer, Apoorv Gaur, Laolu Peters, Ian Lioe, Yang Hou

Advisors: Professor Edward J. Delp, Professor A.J. Schwichtenberg

Department of Electrical and Computer Engineering, Purdue University

Table of Contents

I. Introduction
   A. Background
   B. Goals
   C. Legal Considerations
II. Exploration of Sleep Research Methods
   A. Summary of Sleep Research Methods
   B. Videosomnography
III. Image Processing
IV. Input Data Specifications
   A. Equipment
   B. Data
V. Motion Detection Methods
   A. Image Contouring
   B. White Pixel
   C. Optical Flow
VI. Threshold for Sleep Status
VII. Graphic User Interface
   A. Design
   B. Usage
VIII. Results
   A. Image Contour
   B. Optical Flow
   C. White Pixel
   D. Threshold Algorithm
IX. Conclusion
   A. Challenges
   B. Recommendations for Future Work
X. Appendix A
   A. Team Member Participation
XI. Acknowledgement
XII. References

I. Introduction

A. Background

Sleep is a major factor in child development. Researchers are interested in cataloging sleep patterns of young children so that later they can see if sleep disorders are indicative of developmental disorders such as autism. One technique used by researchers is video somnography, using video to record subject's sleep outside of a clinical setting, which allows researchers to catalog sleep with a high degree of accuracy, little intrusiveness and disruption of the subject's sleep, and which can be reviewed later. However, cataloging those sleep patterns by hand requires hours of labor intensive work per child per night, limiting the ability of researchers to cover a large group of children in a reasonable amount of time. To aid researchers , we endeavored to use image processing to automatically detect movement, the common way to catalog sleep from observation, so that computers can do the hours of cataloging, leaving researchers to review whole nights in minutes instead of hours, and thus allowing large numbers of subject's sleep pattern to be studied and any link between sleep disorders and developmental disorders to be found in a reasonable amount of time.

B. Goals

The overarching aim of the project is for infant sleep detection using automated video analysis. Sleep problems are common for children with autism spectrum disorder (ASD) but assessing their sleep can be challenging. This project includes improving home-based sleep studies for children with and at elevated risk for ASD. The goal of this project is determine when a child is sleeping/awake by designing an automated system that includes the development of image/video processing tools for automatic video processing. Using the video recordings of young children sleeping the team will (1) determine activity during sleep, (2) compare (1) to an ankle worn accelerometer, and (3) determine sleep/wake thresholds using standard behavioral codes for awake and asleep.

C. Legal Considerations

Handling video data of children sleeping is an important, and serious, responsibility. Parents entrusted us with sensitive data about their children, and it was our duty to handle it with care, and protect it from misuse. Each member of the project went through (CITI) training for how to handle the data properly. Data was stored only on local machines, used only for research purposes, except a few samples used in progress reports expressly approved for such use. Finally, at the end of the semester, all data was removed from the local machines and thus again resided only with the sleep researchers.

II. Exploration of Sleep Research Methods

A. Summary of Sleep Research Methods

Sleep can be monitored based on EEG activity. Sleep is characterized by its duration, its distribution and by its quality (consolidated or fragmented). The various methods to assess sleep include: polysomnography, videosomnography, actigraphy, direct observations, sleep diaries and questionnaires. Sleep is a behavioral, physiological and neural phenomena. A brief overview of other sleep research methods:

Polysomnography

This is the monitoring of EEG (brain activity), EMG (muscle activation), EOG (eye movements), oxygen saturation sensors (oximetry). This method provides very detailed information and it can also be used for the multiple sleep latency test (a test used to assess daytime sleepiness). The advantages of this method include: It produces the most detailed set of data about a person's sleep, it is done in a lab setting with standardized conditions which reduces the possibility of error, using this method also allows for testing of multiple latency to assess daytime sleepiness. There are also a couple disadvantages that come with using this method which include: there are too many sensors which have to be place on the body which makes it very hard for infants to sleep, it is very costly due to the various high-tech machines used to monitor sleep, and finally most infants sleep a lot with an unexpected schedule during the daytime which makes it hard for the researchers to predict when to place the various monitors on them.

Direct Behavioural Observation

By definition, this is the direct observation of young infants through the night to assess their sleep. The advantages of this method include: it can be done at home in a normal sleep setting. While the disadvantages are: it is very labor intensive, and very difficult to be done overnight.

Actigraphy

This method is when the researcher places a wrist-watch like device on a young infant, the device is placed on either the hand or the ankle that monitors body movements and provides information on sleep-wake patterns. The advantages to this method include: there is no intrusion to the sleep of a young infant, the method doesn't require any arduous installations like polysomnography, it provides good data about movement during sleep. The disadvantages to this method are: it only measures activity and doesn't provide any data on breathing, sleep staging and other behaviors which the polysomnography method can provide.

Sleep Diaries

This is a diary completed by child or caregiver, it can be used to control the quality of the actigraphy data.

Sleep Questionnaires

Most researchers use questionnaires and diaries but when they need information for sleep quality or sleep architecture, more sophisticated methods are applied.

B. Videosomnography

This method involves video recordings of young infants when they are asleep. It can be used to find events like night terror, rhythmic behaviors, REM behavior disorders and other parasomnias that might occur during sleep. The disadvantages of this method include: intrusion into the privacy of a family with a camera, the data inferences due to the position of the camera and the infant's movements during sleep.

This method was the one chosen by our community partners for the research project it is a combination of "Diagnosis by behavioral observation" and home-videosomnography (HVS). HVS is done using three-steps - 16x (basic overview and classification), 4-8x (detailed descriptions), normal speed in-depth description. The hardware used are an infrared security camera and a microphone connected to a PC netbook through a USB video capture device. While the software - bit-rate software: requires a higher sampling rate than frame rate of the camera and compression software using codecs.

HVS enables recording of patient behaviors, their attributes and interactions. HVS works better than actigraphy because HVS shows all necessary information and any marked Point of Interest can be visually reviewed and analyzed further. HVS also allows for clinical description of the symptoms associated with a sleep problem.

III. Image Processing

Basics

Images are made up of pixels. Each pixel has a specific value, a red, green, and blue value for RGB color images, or a luminosity value for grayscale images. These pixels are stored in order so that they can be pieced together to form an image. Since images are two dimensional, pixels are often referred to by the (x,y) coordinate position within the image, and while when stored in a file the pixels are simply a long sequence of values, in most programs and algorithms these values are stored in a two dimensional array which mimics the visual space they are reconstructed into.

Videos are made up on images, much as images are made up of pixels. Videos are played by constructing each image in order, one after another. Often, each image is only slightly different from the previous image giving the impression of smooth movement as the video is played. These individual images which are part of a video are called frames.

When doing image processing, we break the videos down into their individual frames, and then break the frames down into their sequence of pixels. We look at how these pixels vary from one frame to the next to detect movement, and at how each pixel differs from its' neighbors to detect edges and outline objects.

Motion Detection

Before we concluded on the three motion detection methods that were used, we tried different methods for motion detection which includes: background subtraction, double-difference imaging, hybrid of three frame differentiation & background frame subtraction and hybrid of background frame subtraction and frame-by-frame difference. All these methods had errors related to ghosting, which is when the presence of an object in previous frame generates false alarms, and foreground aperture, which is the similarity between pixels when objects speed is too slow or is motionless.

IV. Input Data Specifications

A. Equipment

The goal has been to record video of children sleeping and so Swann ADW-400 Digital Wireless Cameras with Receiver were used. This equipment was chosen because it stored the video in a lossless compression format, worked well in low light levels, and was small so as to reduce intrusions. It is important to note that while the equipment was the same for all videos, the ambient light levels, camera position and angle, and even in-frame inanimate movement were inconsistent.

B. Data

Video data was ofcourse the most important part for determining the sleep state of children, but the text data provided with the actigraphy data was also displayed within the user interface as researchers requested being able to view that data alongside any we output.

Video Data

The data was provided in 10 minute segments in avi format. All videos were at 16 frames per second, at a resolution of 320x240, and included audio as well.

Actigraphy Data

Actigraphy data was provided as a text file containing the output from the proprietary software that came with the motion sensor which provided its own determination of sleep, as well as whether the child was in lying down or active. A histogram was also provided in pdf format.

Sleep Researcher Notes

Pdf files of scanned notes taken by sleep researchers was also provided. While almost impossible to integrate into the user interface, it was a useful tool in determining the accuracy of our image processing methods.

V. Motion Detection Methods

A. Image Contouring

Introduction

When researching for possible motion detection techniques, we began by looking at what was the desired output needed by our users: Prof. A.J. Schwichtenberg's lab. Given the problem statement of extracting as much relevant motion information as possible, we determined that in the long run, it would be useful to detect body parts along with a motion count for a given second/minute in time. Image contouring perfectly fit that requirement because it could be used for:

a. Object Detection and Tracking
b. Intensity of Motion

We focused on finding the intensity of motion per second, as that would give a meaningful parameter to integrate with the Graphical User Interface (GUI) as quickly as possible, but the use for object detection and tracking could be helpful in future function implementations.

Concept

Image contouring utilizes finding the outline (contour) of the object in question, and then tracking that object's contour in future frames to detect motion via checking for position changes of said contours in future frames.

To find intensity of motion per second using image contouring, four steps are taken:

1. Converting the video frames from color to grayscale
2. Frame differencing consecutive grayscale frames in a video
3. Thresholding frame differentiated frames
4. Counting number of 'contours' in a thresholded frame

Step 1: Converting the video frames from color to grayscale

Given video frames in the red, green, and blue (RGB) color scale, the formula to convert each pixel from RGB color to grayscale, equation 5.A.1 is achieved using the values expanded on by the luminosity equation 5.A.2.

$$f(x,y) = (r,g,b) \rightarrow f(x,y) = Y \qquad (5.A.1)$$

$$Y = (0.299)r + (0.587)g + (0.114)b \qquad (5.A.2)$$

(x,y) is the cartesian coordinate position of the pixel in the frame matrix
(r,g,b) is the red, green and blue values for a single pixel

Figure 5.A.1 Example of a color frame converted to grayscale

Step 2: Frame differencing consecutive grayscale frames in a video

A frame delta (or frame difference) can be calculated by taking the difference in pixel values of grayscale equivalents of 2 consecutive frames in a 16 fps video. This procedure is represented by the formula:

$$f(x,y,T-t) = f(x,y,T) - f(x,y,t) = \sum_{x=0}^{height} \sum_{y=0}^{width} Y_T(x,y) - Y_t(x,y) \qquad (5.A.3)$$

Where, Y_T = Grayscale equivalent at time T

Y_t = Grayscale equivalent at time t

For a 16 fps video, $T - t = \frac{1}{16} sec$

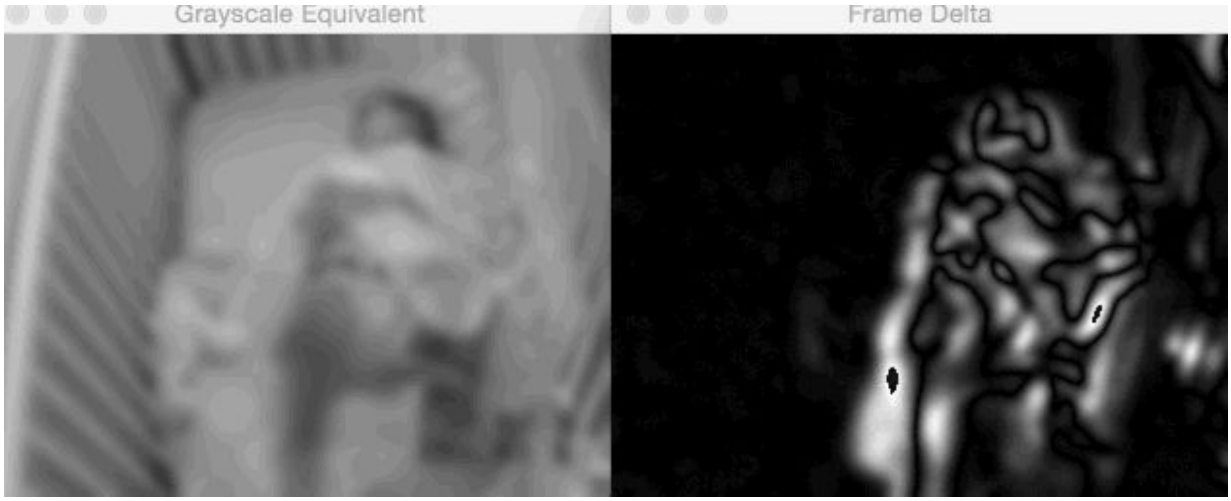


Figure 5.A.2: Example of frame difference on 2 consecutive grayscales in a video

Step 3: Applying binary threshold on a frame delta

To isolate areas of interest in a frame delta, binary thresholding is applied

$$f(x,y,T-t) = 255,\ Y_{T-t} > 25 \qquad (5.A.4)$$
$$f(x,y,T-t) = 0,\ Y_{T-t} \leq 25 \qquad (5.A.5)$$

The value of 25 was chosen to be the best as a measure of observation

Figure 5.A.3: Example of a frame delta vs. its thresholded frame

Step 4: Counting number of 'contours' in a thresholded frame

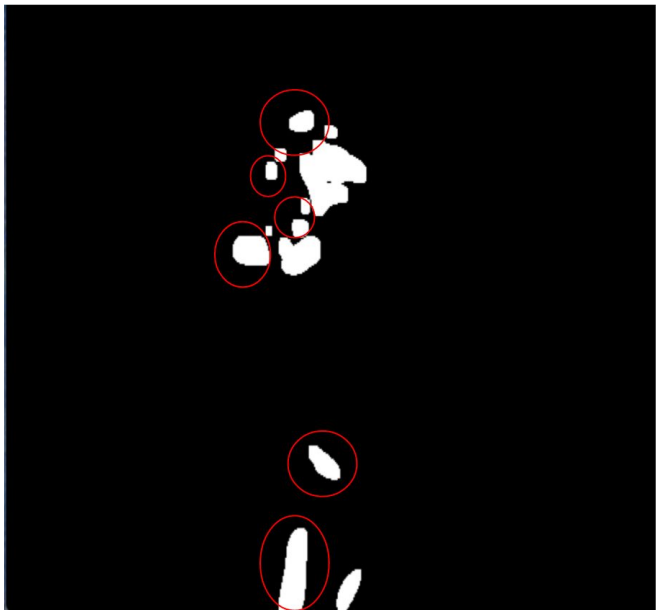


Figure 5.A.4: Counting the number of 'contours' on a thresholded image

Contour tracing is a method by which the boundaries of unique objects on a given background are traced. In our case, it's the number of unique white portions separated by black background, as depicted in Figure 5.A.4. This is done by an algorithm called Border Tracing.

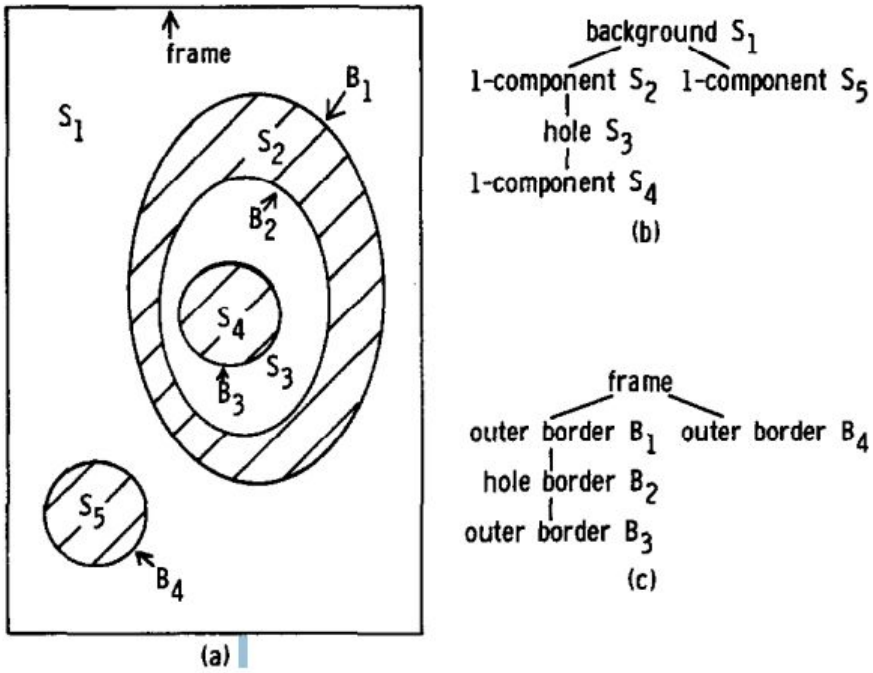


Figure 5.A.5: Example of Border Tracing Algorithm

Depicted in Figure 5.A.5 is the procedure followed by a border tracing algorithm. A pixel on the outer border of a white portion is found and followed until a cluster of pixels complete an outer border. This cluster of outer border pixels forms a contour. If a smaller white portion is enclosed in a larger white portion, as depicted by component S4 enclosed by component S2 in Figure 5.A.5, the inner border of the larger white portion, called a hole border, is also traced so that a distinct differentiation can be made between the a hole border and an outer border of smaller white portion.

Finding and counting the existence of contours in a difference frame is an indication that there was motion in the 1/16th of a second that the difference frame represents. Counting the number of contours in 15 difference frames in a second represents the motion seen in the video during that given second.

Let $C(t_s)$ be the number the contours found in time t_s, measured in seconds,

$$C(t_s) = \sum_{dframe=1}^{15} C(dframe) \qquad (5.A.6)$$

To find the motion status in a minute,

$$C(t_m) = \sum_{t_s = 1}^{60} C(t_s) \qquad (5.A.7)$$

Where t_m is a given time in minutes.

B. <u>White Pixel</u>

Introduction

One of the primary goals given to us by our community partners is to try to determine and estimate whether or not the subject (the baby) is asleep or awake at the time in the video given to us with only resources from the video itself. We realized the most intuitive way of realizing this is to detect the amount of motion in the video that is exerted by the baby. After researching about various motion detection techniques, we found that the simplest and most effective method that could realize our goal is by detecting differences in consecutive frames to find motion. We call it the white pixels method because in the process we transform these differences into pixels with the value of a pure white pixel and non-differences into pixels with the value of a pure black pixel to better differentiate the two, and count the number of white pixels to estimate the amount of motion to estimate the sleep status (awake or asleep).

Concept

White pixel utilizes finding the differences in consecutive frames (only works when camera is stationary), and then counting the amount of differences (white pixels) in order to determine whether or not anything moved in the video.

To find the amount of motion in the video (very similar to image contouring), four steps are taken:

1. Converting the video frames from color to grayscale
2. Frame differencing consecutive grayscale frames in a video
3. Thresholding frame differentiated frames
4. Counting number of 'white pixels' in a thresholded frame

(Note: The first steps are exactly the same as image contouring, but will be also written here in case the reader skipped directly to this section)

<u>Step 1: Converting the video frames from color to grayscale</u>

Given video frames in the red, green, and blue (RGB) color scale, the formula to convert each pixel from RGB color to grayscale, equation 5.B.1 is achieved using the values expanded on by the luminosity equation 5.B.2.

$$f(x,y) = (r,g,b) \rightarrow f(x,y) = Y \qquad (5.B.1)$$

$$Y = (0.299)r + (0.587)g + (0.114)b \qquad (5.B.2)$$

(x,y) is the cartesian coordinate position of the pixel in the frame matrix
(r,g,b) is the red, green and blue values for a single pixel

Figure 5.B.1 Example of a color frame converted to grayscale

Step 2: Frame differencing consecutive grayscale frames in a video

A frame delta (or frame difference) can be calculated by taking the difference in pixel values of grayscale equivalents of 2 consecutive frames in a 16 fps video. This procedure is represented by the formula:

$$f(x,y,T-t) = f(x,y,T) - f(x,y,t) = \sum_{x=0}^{height} \sum_{y=0}^{width} Y_T(x,y) - Y_t(x,y) \qquad (5.B.3)$$

Where, Y_T = Grayscale equivalent at time T
Y_t = Grayscale equivalent at time t

For a 16 fps video, $T - t = \frac{1}{16} sec$

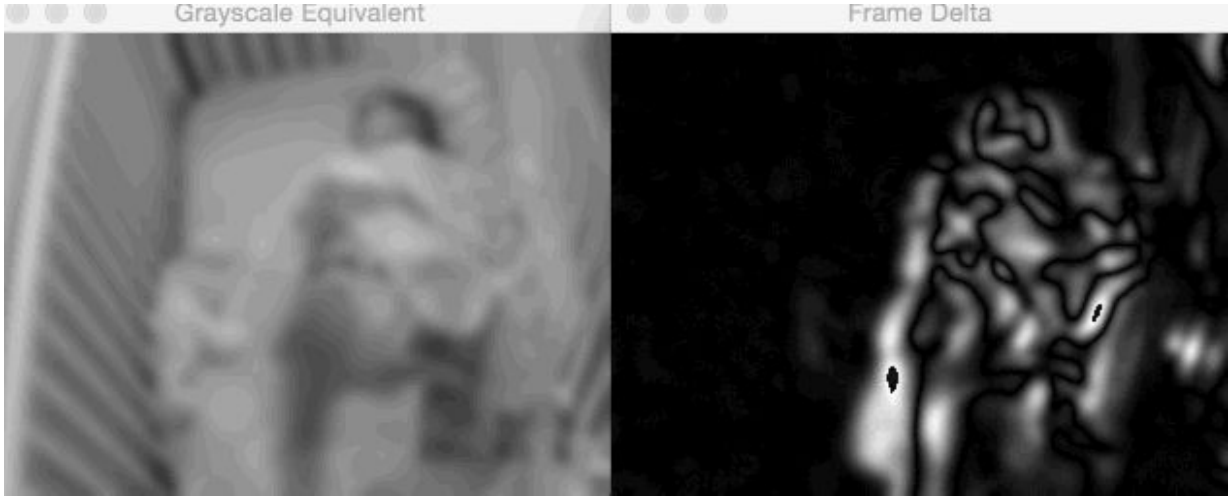


Figure 5.B.2: Example of frame difference on 2 consecutive grayscales in a video

Step 3: Applying binary threshold on a frame delta

To isolate areas of interest in a frame delta, a variable threshold is applied

Let Th be the calculated threshold value

$$f(x,y,T-t) = 255,\ Y_{T-t} > Th \qquad (5.B.4)$$

$$f(x,y,T-t) = 0,\ Y_{T-t} \leq Th \qquad (5.B.5)$$

The value of Th is calculated by taking all values in the difference frame matrix, and dividing it by the number of pixels in a frame, and adding 25 to the result. This way of calculating the threshold ignores negative effects of lighting changes in the frame.

Figure 5.B.3: Example of a frame delta vs. its thresholded frame

Step 4: Counting number of 'white pixels' in a thresholded frame

This step simply uses the thresholded frame produced from the previous step and counts the numbers of pixels that have the value of 255. The number of these white pixels then is stored and added up to compute total number of white pixels in a minute (60 (seconds) × 16 (frames per second)). That is the value we used to estimate sleep status, a high amount of white pixels would indicate lots of motion in the video, which then would indicate that the baby is awake.

C. Optical Flow

Introduction

The need to accurately calculate motion was a prerequisite for the project because of the needs of the project partner. That said, after perusing various Computer Vision Scholarly articles, we found a method that would solve the previously said problem. This method had a lot of promise because it monitored intensity directly which was an important reason for why the prior methods were obtaining less accurate results; the videos tend to have various intensities which made it hard for other methods to find intensities implicitly and obtain accurate results.

Concept

Optical flow is a general concept that is the basis for several sub methods of calculating motion in the motion detection field. The term optical flow itself was coined from the theory of how animals perceive the motion of objects, surfaces, and edges as they move through the world. In motion estimation theory, the basic concept of optical flow is calculating motion between two frames as a sequence of images, which are taken at time t and t + Δt. The base estimation for every optical flow method is that there is a brightness constancy in between two sequences of image frames. For a 2-dimensional image with an object inside with a particular intensity the consequent frame at time t+deltat must have the same object with same intensity but in a different location, where;

$I(x,y,t) = I(x+\Delta x, y+\Delta y, t+\Delta t)$ (5.C.1)

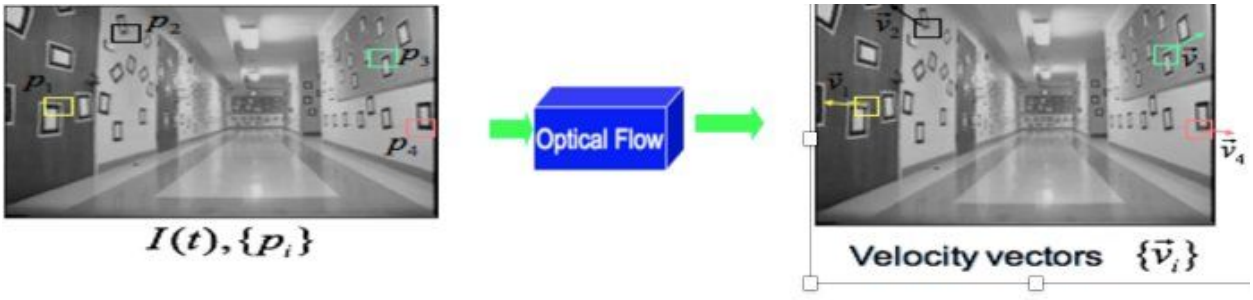


Figure 5.C.1

To expand upon this constraint, using taylor series we obtain;

$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,t) + \frac{\delta I}{\delta x}\Delta x + \frac{\delta I}{\delta y}\Delta y + \frac{\delta I}{\delta t}\Delta t + \beta$ (5.C.2)
Where β is a series of higher order terms which can be negligible.

It is worth noting at this point that the values $\frac{\delta I}{\delta x}$, $\frac{\delta I}{\delta y}$, and $\frac{\delta I}{\delta t}$ can be identifiable as gradient images of the particular frame, where;

$\frac{\delta I}{\delta x} = I_x = I(x,y,t) * \begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix}$ (5.C.3.1)

$\frac{\delta I}{\delta x} = I_y = I(x,y,t) * \begin{pmatrix} -1 & -1 \\ 1 & 1 \end{pmatrix}$ (5.C.3.2)

$\frac{\delta I}{\delta x} = I_t = I(x,y,t) * \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + I(x,y,t+\Delta t) * \begin{pmatrix} -1 & -1 \\ -1 & -1 \end{pmatrix}$ (5.C.3.3)

(Note that the $*$ sign signifies convolution)

From the brightness constancy constraint seen in equation 5.C.1, we can further simplify equation 5.C.2 into;

$\frac{\delta I}{\delta x}\Delta x + \frac{\delta I}{\delta y}\Delta y + \frac{\delta I}{\delta t}\Delta t = 0$ (5.C.4.1)
Or
$I_x\Delta x + I_y\Delta y + I_t\Delta t = 0$ (5.C.4.2)

Where we can further simplify it into;

$I_x \frac{\Delta x}{\Delta t} + I_y \frac{\Delta y}{\Delta t} + I_t \frac{\Delta t}{\Delta t} = 0$ (5.C.4.3)

$I_x v_x + I_y v_y + I_t = 0$ (5.C.4.4)

$I_x v_x + I_y v_y = -I_t$ (5.C.4.5)

Where in this case v_x and v_y are velocities of the image in between those two frames. At this particular point it is plain to see that it is not possible to solve for these two velocity values even if we can obtain all 3 gradient images, as there are two unknowns(v_x and v_y). To solve this problem our team looked into two different methods to solve for these values which are: Lucas-Kanade and Gunnar-Farneback method of Two-Frame Motion Estimation Based on Polynomial Expansion.

Lucas Kanade

After initially researching about optical flow, we decided on testing a method of calculating the velocities, which is called Lucas Kanade. The first step that we took was to test using an example code given from the OpenCV website to see how it would work in an actual video, as you can see from the image below;

Figure 5.C.2

The method calculates the velocity of the movement of the car from particular points of the image. Notice that even though it is working most of the time, there are some points left off where it is calculating the flow of points that are not corresponding to moving objects at all, which correlates to an error. To further test this code, we then tried it on a sample of one of the baby videos that was given to our team by Prof. A.J. Schwichtenberg. The result can be seen as below;

Figure 5.C.3

As can be seen from the image above, the method does not detect the baby at all, instead focusing on the timestamp of the recorded video on the bottom right corner of the image. In order to solve this problem we further analyzed into the theory of how Lucas Kanade works.

Concept

Lucas Kanade uses the method of calculating patches of the image instead of the entire image in order to calculate its velocity, therefore simplifying equation 5.C.4.5 enough such that it makes it possible to calculate v_x and v_y. Following equation 5.C.4.5, we can express it in terms of vectors such that;

$$I_x v_x + I_y v_y = -I_t \quad \Leftrightarrow \quad \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = -I_t \quad \text{(5.C.5.1)}$$

$$\begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = -I_t \begin{bmatrix} I_x \\ I_y \end{bmatrix} \quad \text{(5.C.5.2)}$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \quad \text{(5.C.5.3)}$$

Taking a look at the resultant equation shown in 5.C.5.3 it is plain to see that the matrix has values that correspond to the changes of the frames with respect to time. The summation inside the matrices refer to a windowing operation, in which instead of calculating the gradient points of the entirety of the image at once, we calculate only in increments. As a result, the summation would be of the form; $\sum_{i-w}^{i+w} \sum_{j-w}^{j+w}$, where i and j indicate the x and y coordinates of the image and w is the length of the windowing operation. Paying attention to the right hand side of the equation, the multiplication of $I_x I_t$ can be interpreted as the change in intensities of the image in the x component, multiplied with the change in time, which, is very similar to the physical world's equivalent of distance multiplied by time. Intuitively, we can also assume that if for example we equate $w = 0$ we will have absolutely no windowing operation, which means that we will calculate the entirety of the image at once, resulting in a matrix out of the entire movement velocity of the image. The reason why we would require a windowing operation as opposed to leaving it without, is because in the case of where there are two objects with similar colour intensities, there could be an error caused by the algorithm assuming the two objects are the same, when in truth they are two different objects.

If for experimental reasons, we decide to calculate *u* and *v* without the use of the correlation matrix(the left hand side matrix next to vector u,v of which we will discuss a little bit later.) we will obtain an output similar to the one seen below;

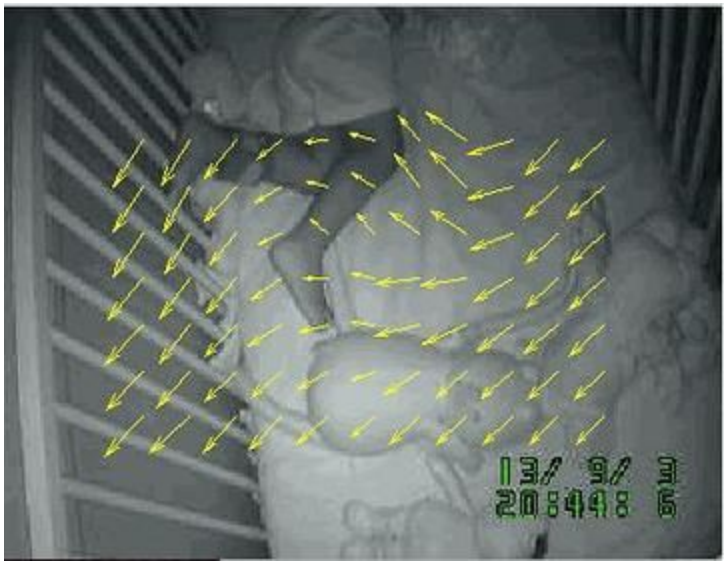


Figure 5.C.4

Which in itself has it's own uses, but most of the time, we will need to calculate only certain objects that are required in the image, and not just any particular one. So for that matter of fact the correlation matrix exists.
The correlation matrix, as seen below;

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \quad (5.C.5.4)$$

Is a relation of all the corners and edges in a particular patch of image. Where $\sum I_x I_x$ represents to a straight line in an image in the *x*-component, $\sum I_y I_y$ represents to a straight line in the y-component, while $\sum I_x I_y$ represents a corner. Assuming we have been using a summation over a particular window, the resultant of that matrix will represent the locations of of all the corners and edges of a particular image. In which case, if multiplied with the flow matrix as seen below;

$$\begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}^{-1} \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} (5.5.5)$$

Will result in a calculation of movement of the particular image over all edges and corners over every patch of window. Intuitively, we can also say that if we do not use any window

calculations, it may cause the image to have much more errors due to the fact that there more points to choose from with similar intensities. There are also multitudes of variations of Lucas Kanade methods, of which one of them is where there are different ways of representing equation 5.C.5.4. These different methods are normally corner detection methods such as harris corner detection, shi-tomasi corner detection, and much more. During our first test using the OpenCV version of Lucas Kanade, the algorithm utilized the Shi-Tomasi Corner detection method, in which replaces the autocorrelation matrix with an equivalent equation. Since we weren't really using it into that much depth the formulas for calculating it will not be discussed that much further.

Summary of Steps for Lucas Kanade

- Calculate gradient images as per equation 5.C.3.1, 5.C.3.2, and 5.C.3.3
- Decide on a window frame *w* of pixels(in our case it was 40)
- Solve for u, v as per equation 5.C.5.5 for a summation of increments $\sum_{i-w}^{i+w}\sum_{j-w}^{j+w}$ for calculation of velocity of every edge corner and line.
- Alternate methods: remove autocorrelation matrix (eq. 5.C.5.4) and solve for u,v as per equation 5.C.5.5 to find u,v of patches of points for the entire image.
- Alternate method: replace autocorrelation matrix (eq. 5.C.5.4) for any of the corner detection methods, of which will not be discussed here but can be found elsewhere.

Problems

There were a couple of possible uses for this method that we had initially planned which came into consideration after several tests using the default Lucas Kanade Optical Flow method. The first was to combine this method with contouring, where it became possible to track body parts of the baby, which would then make it an efficient way of calculating motion. The second would be to calculate just the edges of the baby to calculate its motion to make it more accurate.

However we encountered a problem where due to the calculation taking an incredibly long time it was very difficult to get a frame by frame calculation of the velocity, which instead we would calculate velocity over a couple of frames. This made it almost impossible to get an accurate representation of motion over an entire video.

Future Prospects

For future references the code used to calculate Lucas Kanade is saved in the SLEEP Git, and if for any reason at all would need to be used, would need to be optimized to calculate faster, or must be used by calculating videos frame by frame.

VI. Threshold for Sleep Status

Introduction

After we have the sleep motion information from the video, we need a mechanism to link motion information, either the frequency of motion or the intensity of motion, to actual sleep status, i.e. sleep or awake. So we have this algorithm which could identify the sleep status of a child based on the motion information provided by the different motion detection methods introduced above. The general idea of the algorithm is to generate several critical variable parameters for different sleep patterns from a large scale of training set. Then whenever a new sleep video needs to be evaluated, we pick the set of parameters which match the sleep patterns in the video best to generate corresponding sleep status.

Assumption

1. The algorithm is based on the assumption that the state of sleep S can be estimated by a function of frequency of movement m and sleep time t:
$$S = f(m,t)$$
2. For each patient, the function f(m,t) is different.
3. The videos always start at the time when the child is awake.
4. For the actigraphy data used as training set, the very last minute of the video is guaranteed to be out of the bed.

Model

Based on the analyzation of sleep patterns from the relating sleep research paper provided, three parameters were set up to assist generating sleep status. They are:

1. Window length: the minimum duration of time a child need to stay motionless in order to be identified as asleep
2. Threshold 1: under what level of frequency of movement can we identify the child as motionless
3. Threshold 2: once the child is asleep, what level of frequency of movement should we use as threshold to evaluate the status of awake

The three parameters could be shown on a standard histogram, as the figure show below, in which the green arrow indicates the variable window length, the orange arrow indicates the threshold 1 and the red arrow illustrates the setup of threshold 2.

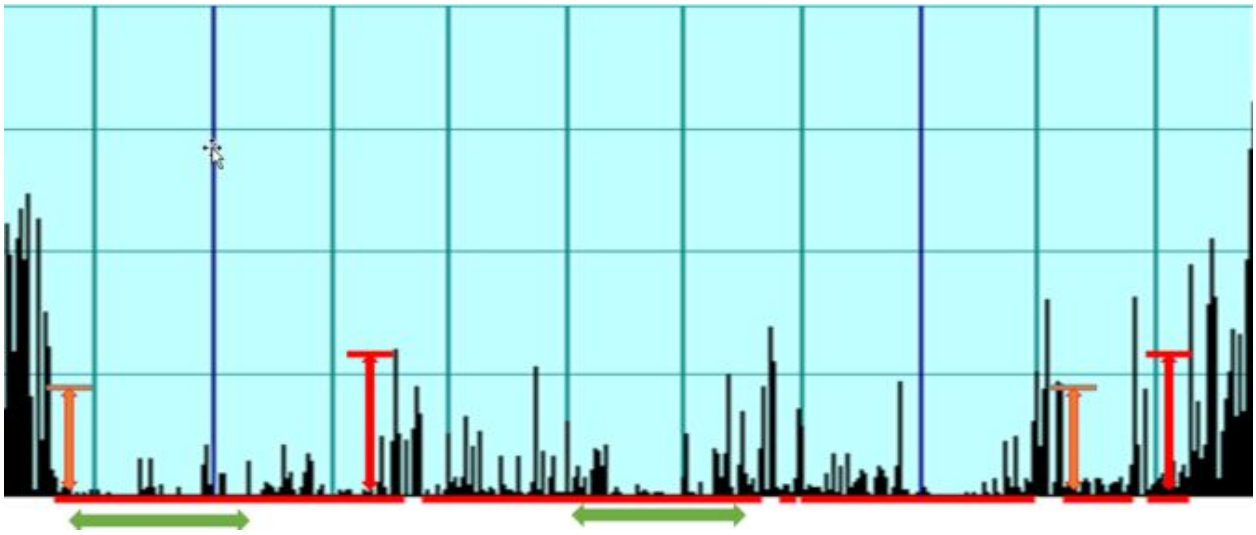


Figure 6.1

The three parameters could be shown on a standard histogram, as the figure show below, in which the green arrow indicates the variable window length, the orange arrow indicates the threshold 1 and the red arrow illustrates the setup of threshold 2.

Algorithm implementation

The algorithm follows the idea of iterating key parameters from reasonable ranges in the model to generate different sets of sleep status corresponding to each night in the training set. Then the algorithm will compare the sleep status generated by different set of parameters to the training set's sleep status, find the best matching one and save the corresponding parameters. Different nights in the training set are also categorized into different sleep patterns, and the set of parameters selected from the process mentioned above will be saved under the particular sleep pattern index. For example, after analyzing one night in the training set, the algorithm will identify this night's sleep pattern, window length, threshold 1 and threshold 2, and save it in the file.

Training methodology discussion and decision

Along the development of the threshold algorithm, two training methods are at the table. A per-person method, and a database method.

1. The per-person method: when having a new patient in. Record several nights of videos. And select one night with typical behavior according to parent's notes or questionnaires. Manually analyze the night as a sample data. Input it to the algorithm, then the algorithm will automatically analyze all future incoming video.
    a. Pros: more accurate, since the thresholds are set based on different individuals, and are used by the same individual.

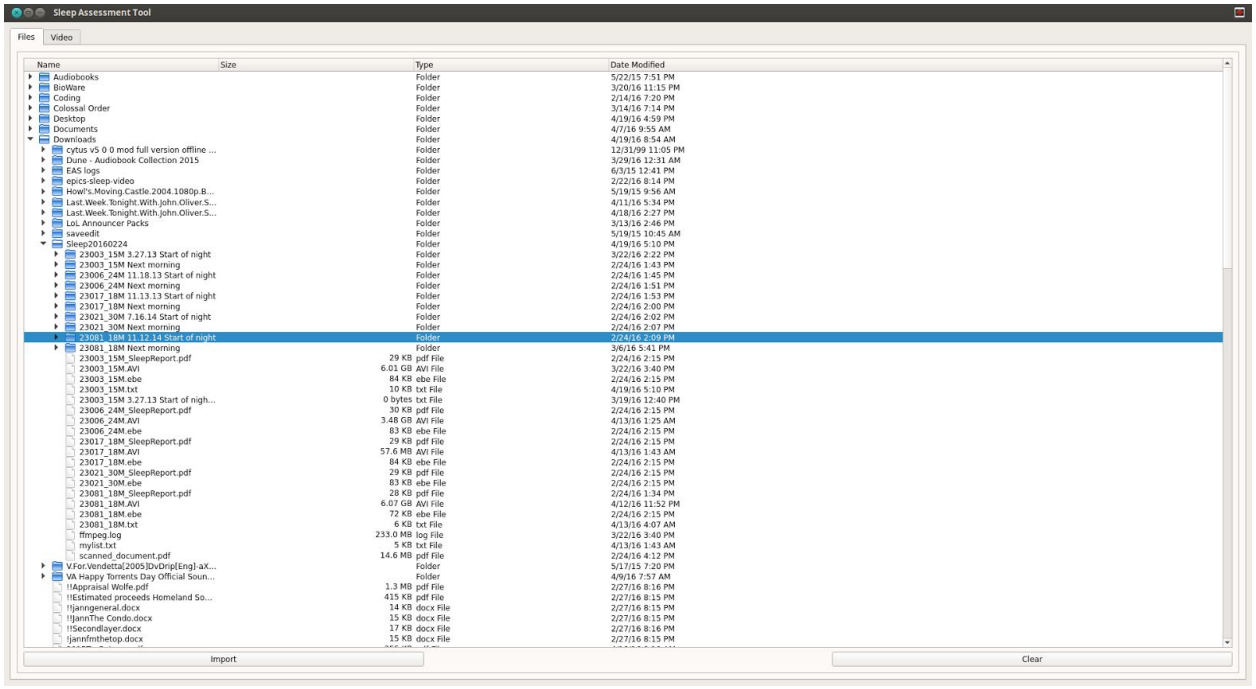
    b. Cons: not 100% auto, since for each new patient, a manually generated file is needed.
2. The database method: The idea of this method is to have a set of input samples as the training set to generate a list of variable sets to match different sleep patterns. After the training is done, all future incoming sleep data will be first identified into different categories, then use the corresponding threshold.
    a. Pros: after the initial training is done, the algorithm is 100% auto
    b. Cons: might need a large scale of dataset for training in order to establish a big enough sleep pattern database for higher analyze accuracy

After proper communication with our project partner, as well as thoroughly consideration, we decided to use the database method.

VII. Graphic User Interface

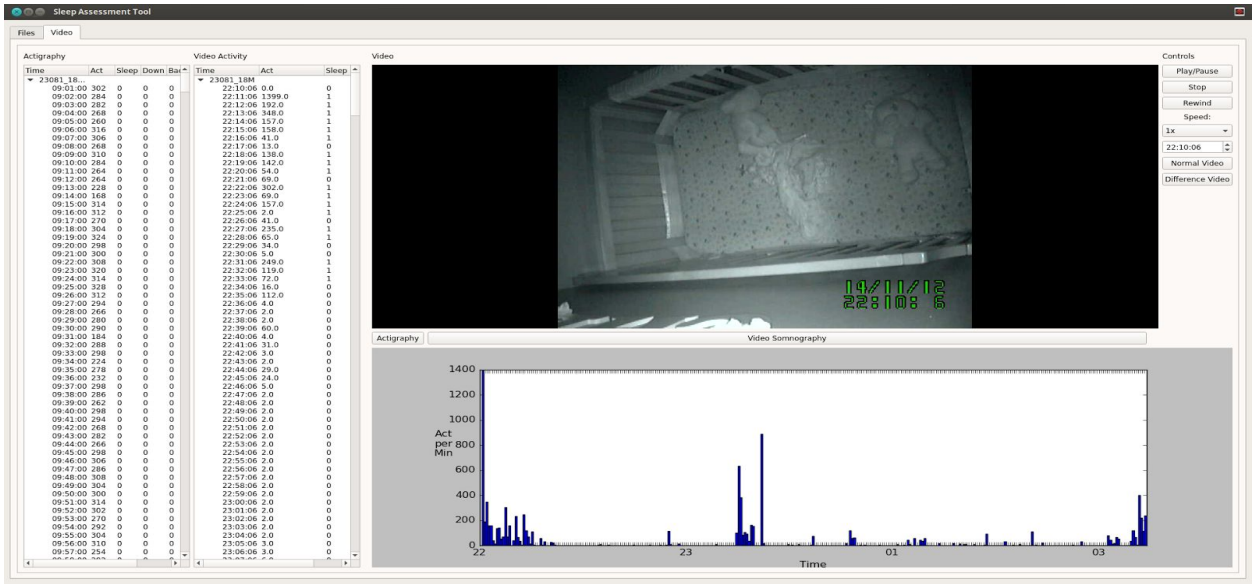
A. Design

The graphical user interface design is specifically geared to be as intuitive as possible. Obviously selecting the files or folder with the appropriate data is necessary in processing those files and thus a file browser is included.



File selection tab

Once the selected files have been processed it is important to provide as much data as possible without overwhelming the user, or cluttering the display. Researchers requested the raw data be visible (the two columns), we felt histograms were important to impart a visual view of the overall sleep pattern, and allowing the researchers to view the video allows them to spot check and determine causes of interruptions quickly.

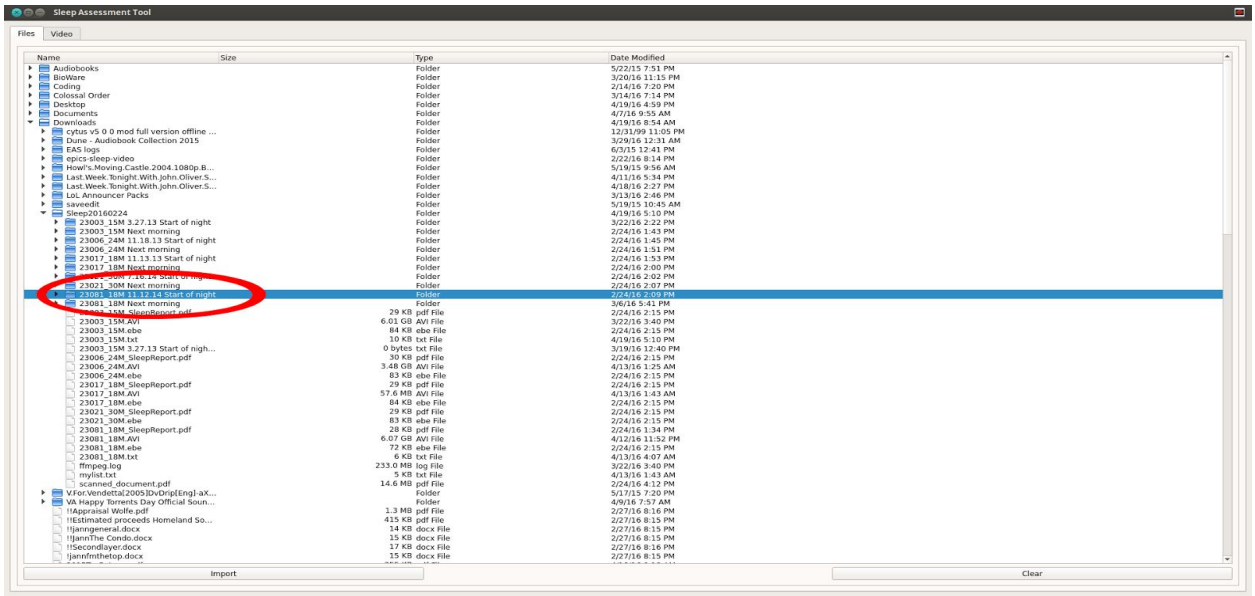


Video data display tab

B. Usage

Using the program begins by opening the sleep assessment tool, sleep_assessment_tool.py. Double clicking will run the tool.

Step 1: Choose folder containing videos (Please select Start of Night, not Next Morning)



Note: Program works best when folders are set up in the following structure:

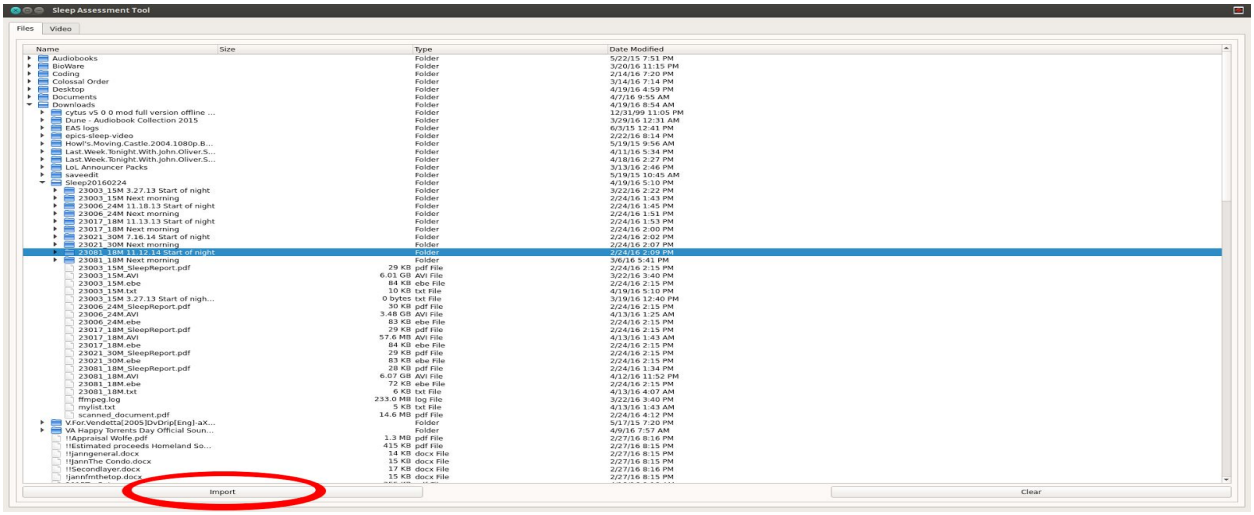
Folder: Date of videos

--->Folder: Date of videos- Start of night

--->Folder: Date of videos Next Morning

--->file: Date of videos.ebe (optional) (recommended)

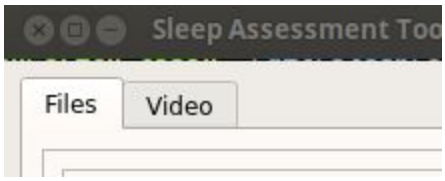
Step 2: Press import button at the left bottom of the file browser



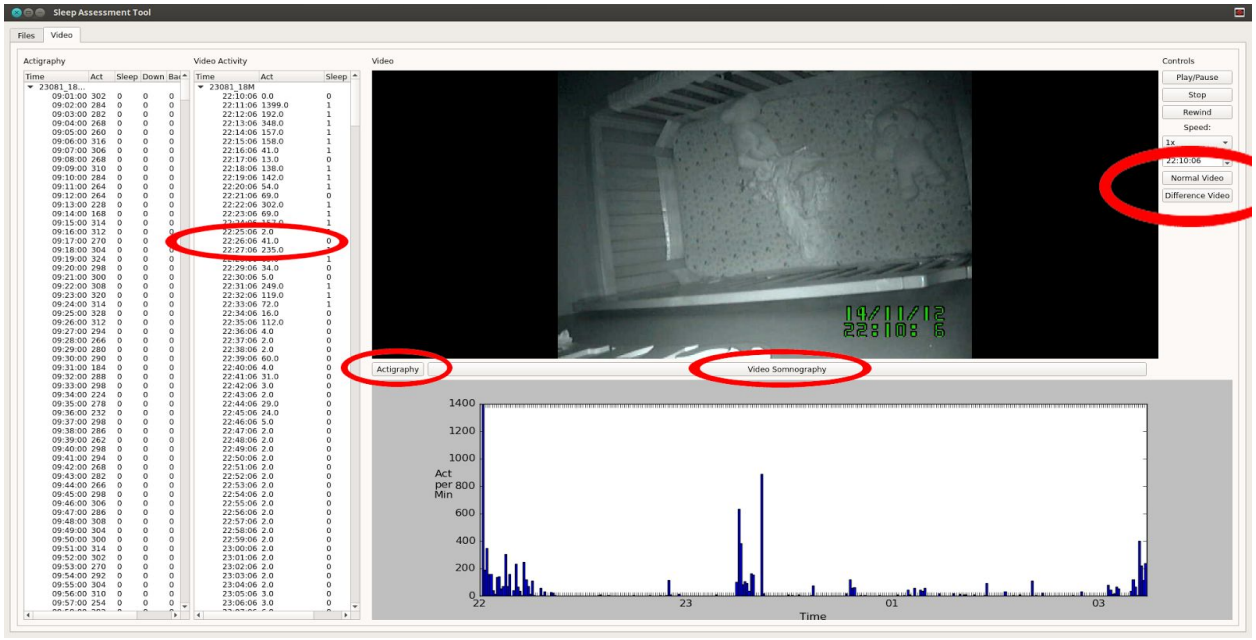
Step 3: Wait for files to be processed

Note: Command prompt terminals will appear showing the processing. If no windows appear then the video has already been processed and you can safely proceed.

Step 4: Press Video tab and view data



Step 5: Customize view with available buttons



Clicking in the data tables will move the video to the corresponding time.

Clicking *Actigraphy* or *Video Somnography* will switch the histogram to the one corresponding to its' respective data.

Clicking *Normal Video* or *Difference Video* will switch between the recorded video or the masked video highlighting movement (currently disabled, in progress)

VII. Results

A. Image Contouring

Sample Output

Time	Activity
418.0m	159.0
419.0m	502.0
420.0m	352.0
421.0m	387.0
422.0m	172.0
423.0m	487.0
424.0m	319.0
425.0m	254.0
426.0m	248.0
427.0m	406.0
428.0m	582.0
429.0m	179.0
430.0m	360.0
431.0m	266.0
432.0m	581.0
433.0m	749.0

Figure 7.A.1: Sample output from image contouring

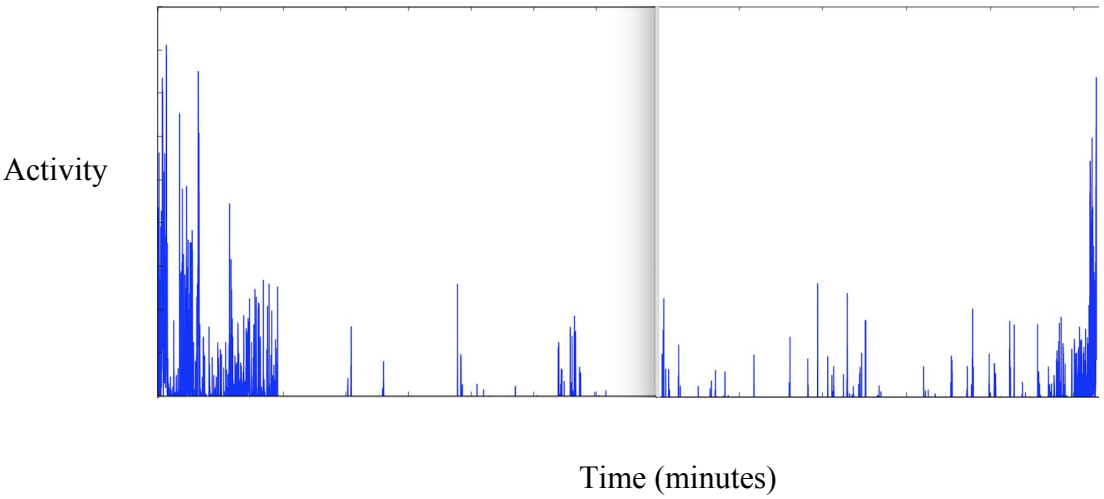


Figure 7.A.2: Histogram generated from a night's set of video

B. Optical Flow

Sample Output

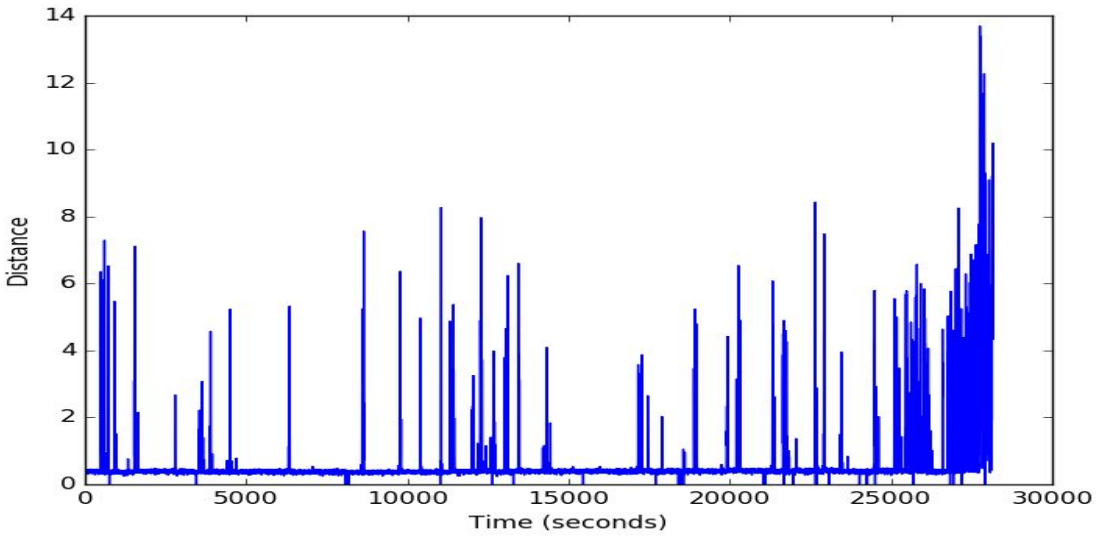


Figure 7.B.1: Histogram generated from a night's set of video

Distance	Time
0.381549	00:00:01
0.395963	00:00:02
0.348781	00:00:03
0.412668	00:00:04
0.360942	00:00:05
0.394801	00:00:06
0.389712	00:00:07
0.385899	00:00:08
0.283347	00:00:09
0.326674	00:00:10
0.405628	00:00:11
0.418951	00:00:12
0.363155	00:00:13
0.348867	00:00:14
0.405323	00:00:15
0.402541	00:00:16
0.370522	00:00:17
0.388304	00:00:18
0.345402	00:00:19
0.348625	00:00:20
0.391427	00:00:21
0.283281	00:00:22
0.345087	00:00:23
0.407187	00:00:24
0.372379	00:00:25
0.451130	00:00:26
0.455240	00:00:27
0.393266	00:00:28

Figure 7.B.2: Data generated from a night's set of video

C. White Pixel

Sample Output

Time(minutes)	Number of White Pixels	Status
1	273990	Awake
2	414532	Awake
3	1044523	Awake
4	605079	Awake
5	602107	Awake
6	9797	Asleep
7	4289	Asleep
8	14697	Asleep
9	9481	Asleep
10	0	Asleep

Figure 7.C.2: Data generated from a single video (10 minutes)

D. Threshold Algorithm

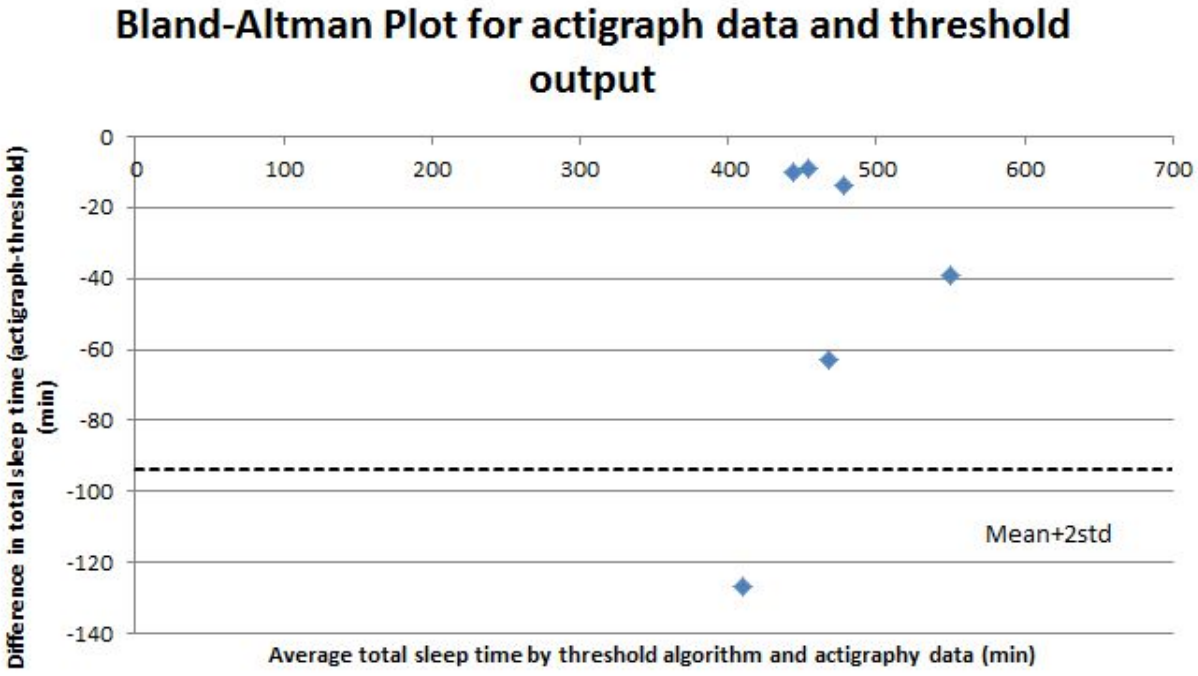


Figure 7.D.1: Bland-Altman comparison of actigraph data and threshold output

IX. Conclusion

A. Challenges

- Hard to focus on understanding the theory behind the backend models while being focused on results
- OpenCV's compatibility has major issues with windows, this provided lots of problems to members using windows OS to code. This also gave problems to the final product, where the installer had difficulties being used in the community partner's computer.
- Applying Methods without using OpenCV has it's own demerits, in that the calculation for the algorithms take much longer and less streamlined, which makes it difficult to make it as efficient.
- Experienced lack of statistical and mathematical knowledge when modeling the variable thresholding algorithm. Particularly, the accurate assessment method and a way to describe and compare different discrete functions.

B. Recommendation For Future Work

- Constantly keep testing your work on a Windows laptop because the laptops and computers in the Developmental Studies Lab have the Windows OS
- To access sleep status, find a method to learn with processing every additional set of videos.
- Properly integrate the threshold algorithm with the user interface
- Find a better method to describe the sleep pattern

X. Appendix (Team Contribution)

A. Shared Team Contributions

This includes tasks in which every member of the team completed or contributed in.

- Researched sleep research methods
- Understood the responsibilities and followed regulations for handling sensitive material
- Designed overall flow of project
- Learned the basics of image processing
- Researched motion detection techniques
- Design Review Presentation
    - Worked to effectively convey all aspects of the design process to reviewers

B. David Tang

- Optical Flow
    - Researched optical flow methods
    - Worked on starting implementation in Python
- Image Contouring
    - Understood implementation in Python, added comments
    - Worked on thresholding for frame differencing
        - Compared results from different thresholds
- White Pixel
    - Researched processes involved in the method
    - Implemented the method in Python
    - Produced predicted results which was accurate with ground truth data (manual inspection of video)
    - Compared output results with other methods as well as actigraphy data

C. Erich Fischer

- Graphical User Interface
  - Designed and tested layout
  - Input and output of algorithms
  - Display of data as lists and histograms
- Compatibility
  - Modifying code to run on Windows systems
- Documentation
  - Made user manual
  - Code commenting

D. Apoorv Gaur

- Image Contouring
  - Researched and implemented Image Contouring
  - Worked with Erich to integrate Image Contouring with the Graphical User Interface
  - Made histograms to display data
  - Commented code
- Usability
  - Worked with Erich and Laolu to install the dependencies and the GUI at the Developmental Sleep Lab

E. Laolu Peters

- Motion Detection
  - Researched various motion detection methods, and tested them on samples of our videos to see which methods worked best
- Optical Flow
  - Researched methods for optical flow
  - Tried and tested Lucas Kanade and Gunnar Farneback's method for using optical flow
  - Implemented Gunnar Farneback's method of Optical Flow
- Documentation
  - Created an installation guide for all software dependencies that our project needs to work
  - Commented in detail all code written

F. Ian Lioe

- Motion Detection
  - Researched various motion detection methods
  - Researched Multiple corner detection methods such as Harris and Shi-tomasi corner detection
- Optical Flow Lucas Kanade
  - Tested and modified OpenCV Lucas Kanade
  - Researched and tested multitudes of instances for applying Lucas Kanade without using OpenCV(with the exception of video capture)
  - Made code for non-OpenCV method of Lucas Kanade
  - Commented with instructions for further use.
- Documentation
  - Wrote down elaborate explanation of Lucas Kanade for future use in sharepoint.

G. <u>Yang Hou</u>

- Graphical User Interface
    - Implemented the conceptual design of GUI
    - Researched on Kivy library for possible better solution
- Variable Thresholding Algorithm
    - Researched sleep pattern related papers to set up the proper parameters
    - Researched statistical papers for better accuracy assessment method
    - Conceptual design of the algorithm as well as the software working process
    - Code implementation
- Documentation
    - Properly comment the code

XI. Acknowledgement

The Sleep team would like to acknowledge the assistance of Jeehyun Choe and Daniel Mas in understanding the work studied and developing and implementing the methods presented in this report.

XII. References

Suzuki, S. and Abe, K., Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)

Fleet, David, and Yair Weiss. "Optical Flow Estimation." University of Toronto Department of Computer Science. Web.

Sleep Research Society. "Basic of Sleep Behavior"

Jan Erik Solem, Programming Computer Vision with Python

Gunnar Farneback, Computer Vision Laboratory, Linkoping University, Linkoping, Sweden

Bland, Martin and Altman Douglas, Statistical Methods for Assessing Agreement Between Two Methods of Clinical Measurement

Popescu, Cezar, A Contour Based Descriptor for Object Recognition

David J. Fleet and Yair Weiss (2006). "Optical Flow Estimation". In Paragios; et al. *Handbook of Mathematical Models in Computer Vision*

Christopher M. Brown (1987). *Advances in Computer Vision*. Lawrence Erlbaum Associates. ISBN 0-89859-648-3.

Rafael C. Gonzalez; Richard E. Woods (2008). *Digital Image Processing*. Prentice Hall. pp. 1–3. ISBN 978-0-13-168728-8.