Facial Recognition System for Driver Vigilance Monitoring

M.A. Spaans

H.J. Dikkers

Res. Rep. No. LSS 169/03

Facial Recognition System for Driver Vigilance Monitoring

M.A. Spaans
H.J. Dikkers

Prague, June 2003

Delft University of Technology

Czech Technical University in Prague

June, 2003.

The authors are students at the Faculty of Information Technology and Systems. Delft University of Technology, Mekelweg 4, 2628 CD, Delft, The Netherlands.
E-mail: mike@ch.tudelft.nl, harmen@ch.tudelft.nl

Research was done at the Czech Technical University in Prague
Faculty of Transportation Sciences
Department of Control Engineering and Telematics
Joint Laboratory of System Reliability
Konviktská 20, Praha 1, 11000

Res. Rep. No. LSS 169/03

Typeset by the authors with the LaTeX 2_ε Documentation System.

"Felix qui potuit rerum cognoscere causas." – Virgil

Acknowledgements

We would like to thank prof. dr. M. Novák of the Faculty of Transportation Sciences of the Czech Technical University in Prague and drs. dr. L.J.M. Rothkrantz of the Faculty of Information Technology and Systems of the Delft University of Technology for providing us with the opportunity to enrich our knowledge and experience in Prague.

Specials thanks go to ing. Pavel Hrubeš for his hospitality and good company.

We would also like to thank “De Fundatie van de Vrijvrouwe te Renswoude” for their financial support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A great many people are injured or killed each year due to accidents involving automobiles. Table 1.1 and table 1.2 list the figures of European car accidents over the years 1996 to 2000. As is apparent there is a gradual drop in the number people killed as opposed to an increase in number of injuries (followed by a drop in 2000). This trend is, among other reasons, the result of several developments in traffic safety. One of these trends is the use of technology by car manufacturers to ensure safety of vehicles. Consider the airbag as an example. Over the past years we have seen it evolve from novelty to mainstream safety enhancing equipment installed in all modern cars.

As statistics indicate that over 40% [2] off all traffic accidents are related to lack of vigilance on behalf of the driver, modern research on car safety at several institutions around the world has shifted focus to understand the phenomena of somnolence in full. At the Faculty of Transportation Sciences at the Czech Technical University in Prague (CTU) researchers use electroencephalography (EEG) to attempt to understand the activity in the brain at the onset of sleep. Brain wave patterns are considered a primary factor in the determination/ prediction of sleep as the brain wave patterns will change immediately at the onset of sleeps as compared to secondary factors such as cardiac rhythm or respiratory activity, which change more slowly once sleep set in. Table 1.3 lists the primary and secondary factors related to vigilance monitoring.

	1996	1997	1998	1999	2000
France	8,541	8,444	8,918	8,487	8,079
Germany	8,758	8,549	7,792	7,772	7,503
Italy	6,688	6,724	6,326	6,633	6,410
Spain	5,483	5,604	5,957	5,738	5,776
UK	3,740	3,743	3,581	3,564	3,580
EU (15)	43,545	43,473	42,696	41,867	40,890

Table 1.1: European road accident figures – number of deaths

	1996	1997	1998	1999	2000
France	170,117	169,578	168,535	167,572	162,117
Germany	493,158	501,094	497,319	521,127	504,074
Italy	272,115	270,962	293,842	316,698	301,559
Spain	124,157	125,247	141,377	142,894	149,781
UK	329,758	336,758	335,033	330,195	331,423
EU (15)	1,707,566	1,725,464	1,760,777	1,808,912	1,770,214

Table 1.2: European road accident figures – number of injured people

Primary	Brain wave patterns	(EEG)
Secondary	Cardiac rhythm	(ECG)
	Eye movements	(EOG)
	Muscular activity	(EMG)
	Respiratory patterns	(EOG)

Table 1.3: Primary and secondary factors related to vigilance monitoring

1.1 Research Goals

In this report we present a proof-of-concept (or feasibility study) of a system that monitors the activity of the face, in particular the eyes, and can determine/predict expressions of somnolence. Our research complements the research done at CTU's Faculty of Transportation Sciences in such a way that data obtained from our system can be combined with EEG and other data to provide a complete map of human physiology at the onset of sleep.

Our system is intended to form a framework for subsequent research done at CTU in this particular area and can be integrated in a simulation environment as illustrated in figure 1.1.

Figure 1.1: Simulation environment

1.2 Report

In chapter 2 of this report we provide an overview of the EEG research done at the Faculty of Transportation Sciences of CTU and in chapter 3 a theoretical background on the problem of facial expression classification is provided. In chapter 4 the HADES-1 system is presented and chapter 5 provides data and results obtained from the system. In chapter 6 we present our recommendations and conclusion. In two appendices we have provided rough data from our system and a brief user's guide. The final appendix contains the system specifications.

Chapter 2

Prediction of Microsleeps Based on EEG

At the Faculty of Transportation Sciences of the Czech Technical University in Prague, ongoing research focuses on understanding the relationship between brain activity and attention level (or vigilance). In a laboratory environment clinical trails are performed to discover what occurs at the onset of sleep (so called microsleeps). In [2] Faber et al. present a framework in which this research is used to predict microsleeps in order to react appropriately in an automated way, that is interrupting microsleeps and restoring full vigilance. This framework is illustrated in Table 2.1. The following sections will give an in-depth overview of EEG and its application to the prediction of microsleeps.

1. Subject performs activity
2. EEG sensors and amplifier register the brain's bio-potential
3. EEG is analysed using Fourier analysis or other spectral filtering
4. Results are processed by logic control elements
5. An alarm is sound if microsleeps are determined

Table 2.1: Framework of EEG research at CTU

2.1 Data Acquisition

As was discovered by the German psychiatrist Hans Berger in the 19th century, the brain generates a bio-potential of very low voltage.[1] When electrodes are attached to the cranial surface of the brain, the electric changes that occur in the extracellular fluid of the brain in response to changes in potential among large groups of neurons (ion flux), can be measured when amplified. The signal thus obtained is a fluctuation of the brain's bio-potential in time. Its recording is

[1] mere tens of $\mu V's$

known as electroencephalography (EEG). When a set of electrodes is distributed over the cranial surface multichannel data can be obtained.

In the work of Faber et al. [2] 35 volunteers were exposed to extended periods of sleeplessness and subjected to EEG testing. All volunteers were asked to perform a standardized series of tasks, including simple arithmetic, while brain activity was constantly monitored using EEG. Reaction time was also monitored by measuring the delay in a subject's response to oral signals.

2.2 Spectral Analysis

Collected brain wave data can be analyzed by computer. Using Fast Fourier analysis or Gabor filtration it is possible to determine what frequency components are present in a single channel of EEG data. In the field of neuroscience a distinction is made between four frequency bands: δ, ϑ, α and β. The various frequency components are depicted in Figure 2.1. As is apparent, brain waves

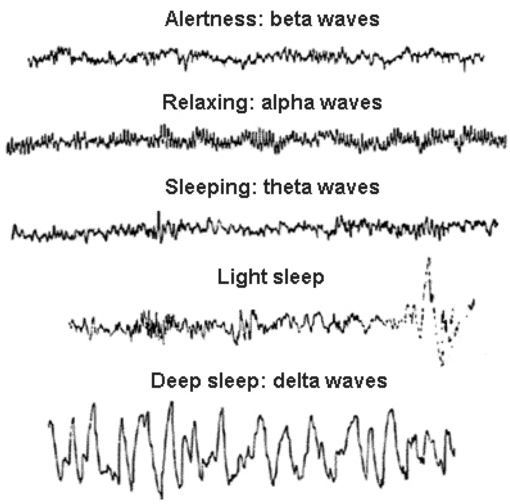


Figure 2.1: EEG in states of vigilance

change with regard to different states of somnolence. Table 2.2 illustrates this as well. α Activity is predominant in normal, awake subjects. Measured reaction times are illustrated in Table 2.3. In the course of relaxation reaction times slow and in most subjects an increase of activity in the α band is apparent. At the onset of sleep α activity disintegrates into activity in the ϑ and later the δ band. Associated reaction times drop from approximately 800 ms to 1200 ms. Reaction times longer than 1200 ms seldom appear; when reaction times drop that low the subject is usually close to sleep.

Band	*Freq.*	*Vigilant state*
δ	$1-3\ Hz$	A person is deep asleep
ϑ	$3-7\ Hz$	A person is sleepy, asleep or in sleep transition
α	$8-13\ Hz$	A person is awake and relaxed
β	$13-30\ Hz$	A person is awake and active, alertness

Table 2.2: EEG frequency components and associated states of vigilance

Vigilant state	*Reaction time*
Alert	$100\ ms - 400\ ms$
Relaxed	$400\ ms - 800\ ms$
Sleep (NREM1)	$800\ ms - 1000\ ms - 1200\ ms - \infty$

Table 2.3: Reaction times at different level of vigilance

2.3 Results

As was mentioned in the previous section, in normal, awake subjects activity in the α band is predominant. When asleep α activity disintegrates into ϑ and later δ activity. Frequency changes that occur when a subject is in a transitory state between awake and asleep do not occur simultaneously all over the brain; each part of the cortex reacts in a different way, at different times. When the subject is fully asleep though, δ activity is registered foremost. As there is a large difference between the frequencies measured at different electrodes, and also a large difference between different subjects, Faber et al. in [2] use a ratio to express the state of somnolence.

$$\frac{\alpha}{\delta} \tag{2.1}$$

If there is more α activity than δ activity this ratio will be larger than one, and thus the subject is awake. If there is more activity in the δ band than in the α band, the ratio will be smaller than one and the subject is asleep. Due to the fact that there is less energy in the ϑ band, its frequency changes are less well suited to somnolence determination. Using the ratio as described in equation 2.1, Faber et al. can accurately predict the onset of sleep (microsleeps) using constant EEG monitoring. A system based on this principle can be implemented in environments where constant attention levels are critical, such as driving or piloting aircraft.

Chapter 3

Theoretical Background

This chapter describes the model used by the HADES system, underlined by relevant research. The scientific field of research of facial expression classification is close to the problem of somnolence detection in facial images. Typically, these classification systems are capable of recognizing about six different facial expressions from visual data. Determining expressions of somnolence is actually a subproblem of these problems, since the data involved need only be classified into two distinct expressions.

All facial expression recognition models can be placed within a three-tier framework, as described by Pantic and Rothkrantz [6]. Several recent approaches will be discussed [1], focusing on systems that handle problems that are strongly correlated to the driver vigilance problem. The framework consists of the following steps:

1. Face detection
2. Facial expression data extraction
3. Facial expression classification

The following sections will describe these steps in more detail, in order to extend the reader's background knowledge in the relevant subject matter.

3.1 Face Detection

In our research we have only considered the problem of face detection in facial images, and have left the problem of face detection in arbitrary images out of scope. There exist two different models to represent the face: the holistic model, which represents the face as a whole, and the analytic model, which represents the face as a set of facial features.

A holistic approach has been proposed by Huang and Huang [3], who use something called a Canny edge detector to roughly estimate the location of the face in

[1] The research by Pantic et al. [6] is the source of the approaches described in this section

the image. This detector observes the valley in pixel intensity that lies between the lips and the two symmetrical vertical edges representing the outer vertical boundaries of the face. This system has limitations with regard to rigid head movements, facial hair and glasses, and has several illumination constraints.

Pantic and Rothkrantz [7] use dual view facial images in their holistic approach to facial expression classification. They analyze the horizontal and vertical histograms of the frontal view image in order to determine the boundaries of a rectangle around the face. In order to determine the contour of the face, they use an algorithm based on the HSV color model. No facial hair or glasses are allowed in their system and they require a camera to be mounted on the subject's head.

Kobayashi and Hara [5] use an analytic approach; they use a CCD camera in monochrome mode to obtain brightness distribution data of the face. Their system determines the position of the irises in real-time, by comparing the brightness distribution of the currently examined data to an average distribution obtained by averaging data of ten subjects. The subjects face the camera at a distance of approximately one meter; no rigid head rotations are allowed.

Kimura and Yachida [4] propose a potential net for face representation. First they normalize an image by using the centers of the eyes and the center of mouth, which are found by an integral projection method based on color and edge information. Then the potential net is fitted to the normalized image to model the face and its movement. Analyzed faces are without facial hair and glasses and face the camera directly; no rigid head rotations are allowed.

3.2 Facial Expression Data Extraction

Eye closure and narrowing eyelids are the most obvious signs of the onset of somnolence. In our research we have decided to focus on just the eyes in determining somnolence. We are assuming other facial features are less relevant to the classification problem. Other facial features might however provide extra information. The corners of the mouth are for instance likely to be a little lower than normal.

As there are two approaches in face detection systems, there are also two different approaches in facial expression data extraction systems: the holistic and the analytic approach. Additionally it is possible to combine the two approaches in the hybrid approach.

Using the analytical approach and selecting the appropriate (i.e. eye-based) features is the most obvious solution in our particular situation. In holistic and hybrid approaches many non-relevant features are taken into account which make results less reliable. A holistic approach which focuses on the eyes only might however provide good results.

In their analytic system Kobayashi and Hara [5] use a geometric face model of 30 Facial Characteristic Points (FCPs), 16 of which concern the eyes. A set of brightness distributions of 13 vertical lines crossing these FCPs is used on a normalized image. The data thus obtained is fed to a neural network as input.

No facial hair and no glasses are allowed in their system. An advantage of this system is its real-time property.

Cohn [1] uses a model of facial landmark points near the facial features. In the first frame of a sequence of recorded images the landmark points are selected manually. For the other frames an optical flow method is used. The displacement of each landmark point is calculated by subtracting its normalized position in the first frame from its current normalized position. All frames of an input sequence are normalized manually. The displacement vectors, calculated between the initial and the peak frame, represent the facial information used for recognition of the displayed facial actions. Analyzed faces are without facial hair and glasses, no rigid head motions are allowed and the face in the first frame must be neutral (or expressionless).

3.3 Facial Expression Classification

With regard to the facial expression classification problem three methods are considered:

- Template-based methods
- Neural-network-based methods
- Rule-based methods

The template-based methods compare an arbitrary image to prototypic templates in each expression category, and classify this image to the category that matches best. In general, it is difficult to achieve quantified template-based recognition of non-prototypic images, meaning images can only be exclusively categorized into one class without a level of uncertainty. The fact that each person has his own maximal intensity of displaying a certain facial action makes the situation even more difficult.

Neural networks could be considered as template-based methods due to their black-box behavior. Pantic et al. [6] however distinguishes neural networks from the template-based methods, as they do in fact perform quantified facial expression categorization. When utilizing a neural-network-based classification, a facial expression is classified according to the categorization process that the network learned during a training phase. Recognition of non-prototypic facial expressions is feasible if each neural network output is associated with a weight from the interval $[0, 1]$, instead of being associated with either 0 or 1.

Kobayashi and Hara [5] apply a 234 x 50 x 6 back-propagation neural network. The units of the input layer correspond to the number of the brightness distribution data extracted from an input facial image (see the previous section), while each unit of the output layer corresponds to one emotion category.

The rule-based systems surveyed by Pantic classify the examined facial expressions into the basic emotion categories, based on previously encoded facial actions (a means of describing the face). In order to achieve this, the prototypic expressions are first described in terms of facial actions, after which the examined expression is compared to the prototypic expressions defined for each of the emotion categories and classified in the optimal fitting category.

Pantic and Rothkrantz [7] use the localized contours of the face in order to extract model features. The difference between the currently extracted model features and the same features extracted from an expressionless face of the same person is calculated and compared to prior acquired knowledge. The production rules can thus classify the images into the appropriate classes.

Chapter 4

HADES-1 System

In this chapter the HADES-1 system is presented.[1] The HADES-1 system (Hybrid Approach to Determining Expressions of Somnolence) is a system designed to analyze a stream of images taken by a video camera of a subject's face, and detect expressions of somnolence. The HADES-1 system is essentially a prototype, or proof-of-concept, of a system which can be embedded into cars or used in any situation where constant vigilance is of paramount importance.

We will first give an overview of the system, in which its general working is described, after which we will focus on the system design in detail. We will also attempt to classify the HADES-1 system according to the taxonomy of all recent systems dealing with automated analysis of facial expressions, as given by Pantic et al. in [6]. A brief user's guide of the system has been provided in appendix B.

4.1 System Overview

As was described in the previous chapter, the problem of emotional expression classification consists of three basic steps. Although the problem of detecting an expression of somnolence is a subproblem of general emotional classification, the steps are in effect similar. To the three basic steps, *face detection*, *facial expression data extraction* and *facial expression classification* we have added two extra steps, *image acquisition* and *determination of somnolence*, relevant to our more particular problem. The steps are illustrated in table 4.1.

[1]Hades is the lord of the dead and ruler of the nether world, which is referred to as the domain of Hades. He ruled the underworld together with Persephone, whom he abducted from the upperworld. Zeus ordered him to release Persephone back into the care of her mother Demeter, but before she left he gave her a pomegranate. When she ate it, it bound her to the underworld. Hades rules the dead, assisted by various (demonic) helpers, such as Thanatos and Hypnos, the ferryman Charon, and the hound Cerberus.

1.	Image acquisition	Argos
2.	Face detection	Hypnos
3.	Facial expression data extraction	Hypnos
4.	Facial expression classification	Hypnos
5.	Determination of somnolence	Persephone

Table 4.1: Steps involved in facial expression classification

Image Acquisition

Image acquisition in the HADES-1 system is done using a digital video camera in a controlled manner, that is, under ideal circumstances with regard to lighting, etc. The results of our research, presented in the next chapter, are based on two different camera setups: a close-up of the eyes and a view of the head in its entirety. In both cases the subject is sitting in front of a white screen, to eliminate background disturbances, and does not tilt or rotate his head. The system can be considered as performing *analysis of static image sequences* according to the classification give by Pantic et al.

Face Detection

Our research has focused primarily on the eyes as a measure of somnolence detection. The detection of the eyes within the facial images is considered outside the scope of our research and is done by hand. The system operator monitors the incoming video stream and using a pointing device, marks the upper-left corner of the eyes and the bottom-right corner, creating a rectangle marking the area of interest. According to Pantic et al. in [6] this is the *analytical approach*; only the individual features of the face, in our case the eyes, are considered, instead of the face as a whole.

Facial Expression Data Extraction

Using the marked rectangle of interest mentioned previously, we extract a vector of data based on three methods that will be described further on in this chapter. These methods are based on the light/color intensity of pixels in the image. The HADES-1 system can easily be adapted to include different methods of data extraction, based on the rectangle of interest. The vector of data is continuously refreshed as the camera captures a new image. This is the *analytical* approach according to Pantic et al. [6]

Facial Expression Classification

When a vector of facial data has been obtained, we need to determine whether an expression of somnolence is present. In order to do this the HADES-1 system requires two calibration images: **calib_0** and **calib_1**. The first calibration image is essentially an image of the subject in neutral condition, without emotion and with open eyes. The second calibration image is an image of the subject in sleepy condition, that is, with eyes closed. From these calibration images two data vectors are obtained using the same rectangle of interest mentioned previously and using the same method. HADES-1 can now determine whether an expression of somnolence is present in the input images by comparing the data vector with the vectors obtained from the calibration images. This principle is illustrated in figure 4.1.

Essentially we have three vectors in an n-dimensional space. HADES-1 calculates the distance between the data vector and the **calib_0** vector and the distance between the data vector and the **calib_1** vector. Of the two distances thus obtained the shortest is chosen to classify the expression. If the distance to the second calibration image, the subject with eyes closed, is smallest, an expression of somnolence is determined; if the distance to the first calibration image is smallest, the subject's expression is neutral.

In our research we have, in principle, assumed that the difference between the two calibration images (i.e. eyes-open and eyes-shut) is large enough for us to use in determining expressions of somnolence. The next chapter will provide data to underline this assumption.

According to Pantic et al. in [6] our chosen approach is a *template-based* method.

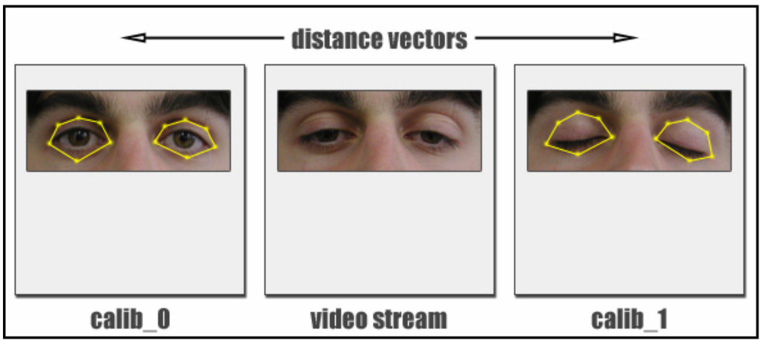


Figure 4.1: comparison of data vectors to determine somnolence

Determination of Somnolence

This final step is necessary when the HADES-1 system is implemented in environments where vigilance is of critical importance. In such situations an alarm, or other device, might need to be triggered to alert a driver or operator to his lack of attention. This alarm can not be triggered on the first determined expression of somnolence by the HADES-1 system, for it is possible when capturing many frames, i.e. data vectors per time instant, that a blink of an eye can be classified as an expression of somnolence. The way this is handled by HADES-1 is described further on in this section.

4.2 System Design

The HADES-1 system is written in C++, taking advantage of the mechanism of inheritance to make it easily extendible. The basic steps as discussed previously in this chapter, are translated into three logical modules: *Argos*, *Hypnos* and *Persephone*.[2] *Argos* is the input module (step 1 in table 4.1), reading the data as obtained from the camera; *Hypnos* does all the calculations of the system (step 2, 3 and 4 in table 4.1) and *Persephone* takes care of somnolence determination (step 5 in table 4.1) as well as miscellaneous Graphical User Interface issues.[3] Figure 4.2 illustrates the individual modules and their relationships. The rest of this chapter will describe in detail the individual modules of the system.

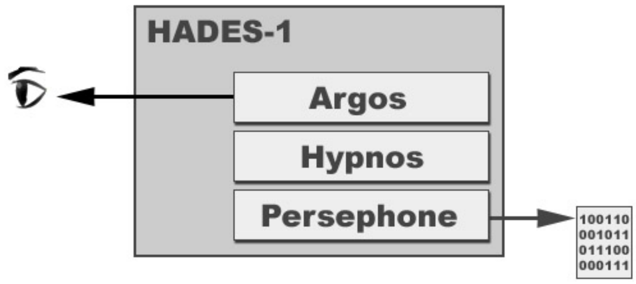


Figure 4.2: HADES-1 system diagram

4.3 Argos

Argos is responsible for processing the camera input and image acquisition. The HADES-1 system assumes that a video capturing device is connected to a USB or FireWire port of the workstation. The USB and FireWire standards are both integrated into 32-bit Microsoft Windows systems (XP, NT, 2000).

Argos sets up a form of data graph in memory, containing several *filters* as nodes.[4] A *capture filter* captures the video signal from the camera and passes it on to the *DV video decoder filter*, which converts it into an RGB-signal. The *sample grabber filter* grabs frames from the RGB-signal. Frames are not constantly grabbed by Argos, but only on request by other parts of the system. The *video renderer filter* draws the live feed to a window. The data graph is depicted in figure 4.3.

The digital video camera typically provides 25 frames per second, compliant with the PAL standard. This frame rate is an upper bound of our real-time system.

The Hypnos algorithms are pixel-based. In order to do proper classification high-resolution images are required, though low-resolution images are necessary

[2] C++classes
[3] A responsible job for HADES' wife;-)
[4] The libraries used by this graph are provided by the DirectX 9.0 DirectShow module

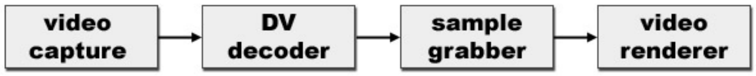


Figure 4.3: HADES-1 system diagram

for fast processing. In our system the captured video images have a resolution of 720x568 pixels, which is sufficient for the currently implemented Hypnos algorithms to do proper classification at reasonable speeds.[5]

4.4 Hypnos

As was mentioned previously, the Hypnos module does the calculation of the system. Hypnos is the parent class of several child classes, each of which implements a different method of obtaining data vectors of the calibration images and the input image and calculating the distances between them. Figure 4.4 illustrates their dependency.

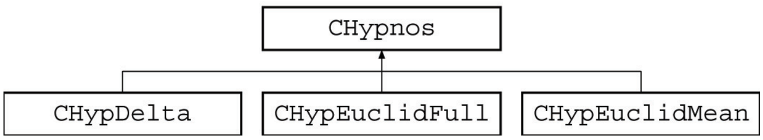


Figure 4.4: Class diagram of Hypnos and subclasses

HypnosEuclidMean

The data vectors as they are calculated by the HypnosEuclidMean class are the averages of pixel intensity along horizontal and vertical scanlines, inside the rectangle of interest (i.e. the eyes). This is illustrated by the matrix in equation 4.1. The main assumption being yet again, the fact, that the intensity vector of the sleepy calibration image will differ substantially to the intensity vector of the neutral calibration image, thus allowing accurate comparison.

$$\begin{array}{c} \begin{pmatrix} 120 & 229 & 127 & 89 & 21 & 154 & 189 & 88 & 231 & 32 \\ 243 & 174 & 173 & 84 & 135 & 98 & 227 & 215 & 56 & 70 \\ 143 & 175 & 143 & 132 & 85 & 86 & 223 & 234 & 146 & 140 \\ 202 & 239 & 123 & 47 & 75 & 236 & 173 & 114 & 29 & 30 \\ 165 & 87 & 43 & 234 & 138 & 64 & 217 & 176 & 121 & 243 \end{pmatrix} \\ \begin{matrix} 175 & 181 & 122 & 117 & 91 & 128 & 206 & 165 & 117 & 103 \end{matrix} \end{array} \quad \begin{matrix} 128 \\ 148 \\ 151 \\ 127 \\ 149 \end{matrix} \tag{4.1}$$

[5]Chapter five will focus on performance of the algorithms

To determine the distance between the data vectors HypnosEuclidMean uses the Euclidean distance metric as given by equation 4.2.[6] The Euclidean distance is calculated separately for the means in the horizontal and vertical direction. A correction factor called the λ factor is used to attach more importance to either the horizontal or vertical distances. The HypnosEuclidMean algorithm is simple and fast and proves very effective as will be demonstrated in the next chapter.

$$d_M = \left\{ \sum_{i=1}^{p} (x_i - y_i)^m \right\}^{\frac{1}{m}} \tag{4.2}$$

HypnosEuclidFull

HypnosEuclidFull is very similar to HypnosEuclidMean. It uses the same distance metric, as given by equation 4.2. The main difference is that HypnosEuclidFull uses the full rectangle of interest to obtain its data vectors and does no do any statistical pre-processing on them. In performance HypnosEuclidFull is also quite similar to HypnosEuclidMean.

HypnosDelta

The HypnosDelta subclass uses a different principle to calculate its data vectors; it determines the *difference* between subsequent pixel intensities on each scanline (vertical and horizontal). The matrix in equation 4.3 shows the differences in intensity values taken from an image of the outer left corner of an open left eye. As is apparent there is a significant rise and fall in values along the vertical scanline. The peak in value indicates the position where the eye 'begins'; at the boundary between the white of the inside eye (sclera) and outside the eye there is a sharp rise in intensity values. Figure 4.5 illustrates this as well; it is a plot of the same data point, only taken from a larger area. The second peak indicates where the eye 'ends'.

$$\begin{pmatrix} 1 & 0 & 1 & 1 & -1 \\ 3 & 0 & 2 & 2 & 0 \\ 2 & 1 & 2 & 1 & 2 \\ 2 & 4 & 3 & 4 & 4 \\ 10 & 11 & 11 & 10 & 9 \\ 24 & 23 & 23 & 22 & 24 \\ 40 & 40 & 37 & 38 & 40 \\ 18 & 17 & 19 & 18 & 19 \\ 5 & 5 & 6 & 4 & 5 \\ 3 & 4 & 6 & 6 & 3 \end{pmatrix} \tag{4.3}$$

The main reason for using intensity differences is to attempt to make the system less sensitive to head movement. A slight movement of the head causes a severe drop in classification accuracy of the HypnosEuclidMean and HypnosEuclidFull

[6] Actually the Minkowski of order m distance is given, which is a general form of the Euclidean distance.($m = 2$)

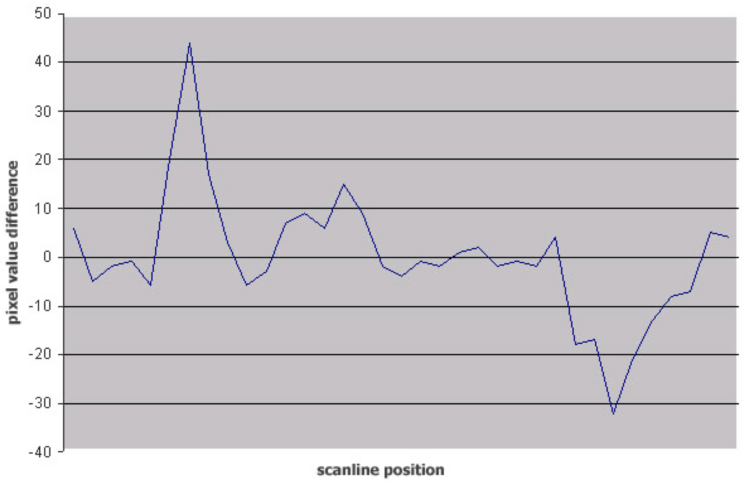


Figure 4.5: Intensity differences along vertical scanline

algorithms for their data vectors are based on actual pixel intensity values. The HypnosDelta data vector contains only *differences* as values.

We found the Euclidean distance metric unsatisfactory for comparing the HypnosDelta data vector with the HypnosDelta vectors taken from the calibration images. Instead an absolute sum is used for comparison. This is done because there are very little boundary (i.e. 'large') values in the sleepy calibration image, thus producing a low vector sum. When compared to an arbitrary data vector, a relative lack of boundary values (i.e. low sum) will immediately indicate a small distance to the sleepy data vector, meaning an expression of somnolence is detected.

This HypnosDelta subclass performs more calculations than HypnosMean and HypnosDelta per captured frame and is therefore slightly slower than its brethren classes.

4.5 Persephone

As was mentioned previously in this chapter, when a single captured image is classified as an expression of somnolence, it does not automatically mean a subject is in a 'state of somnolence'. Persephone determines whether a subject is actually asleep. The Persephone class does this determination based on the *number* of captured images that have been classified as an expression of somnolence; a system variable maintains a count which, when reaching a certain threshold[7] triggers an action.[8]

Persephone also performance miscellaneous actions with respect to the user

[7] in HADES-1 this defaults to 3

[8] in HADES-1 an alarm goes off

interface and the storing of obtained data.

Chapter 5

Results

In order to measure and analyze the performance of our system, we have subjected HADES-1 to two thorough testings. In the first set-up a close-up of the eyes was taken, whereas the second provided us with the face as a whole. In both tests, the lighting conditions were controlled and the subject kept his head reasonably still.

This chapter presents the data obtained from the first set-up only. The results of the second set-up were surprisingly similar and will not be described in detail. The first section describes the performance of HADES-1 and the three algorithms that have been explained in the previous chapter. These results will be compared in the Analysis section. Finally, the boundary conditions will be scrutinized.

5.1 Performance

One of the system requirements is its real-time or quasi real-time performance. The camera we used, a Canon MV10 Digital Video Camcorder, provides a maximum of 25 frames per second. Measurements of the speed of the different algorithms we used justify the conclusion that the HADES-1 is indeed quasi real-time. Table 5.1 shows the results when testing the HADES-1 system on a Pentium 4 2.4 GHz, with a selected rectangle of interest of approximately 20 percent the size of the image.

HypnosEuclidMean	18–19 fps
HypnosEuclidFull	16–17 fps
HypnosDelta	16–17 fps

Table 5.1: Frame rates of algorithms

In the following subsections we will discuss the three algorithms briefly.

HypnosEuclidMean

The HypnosEuclidMean algorithm shows some decent results as are depicted in figure 5.1. The x-axis represents the elapsed time in seconds, whereas the y-axis represents the distance of the examined image to the **calib_0** and **calib_1** image. The HADES-1 system classifies an image as 'sleepy' when the distance to the neutral face surpasses the distance to the sleepy face. Comparing these results to our personal observations we can conclude that the system did not misclassify any image in this particular image sequence. The blinking of the eye at around T=0.5 and T=2.3 can clearly be seen in the graph. This confirms our assumption that eye closure in only one image does not necessarily mean the test subject is in a state of somnolence. In table A.1 (which contains the data of figure 5.1) and table A.2 it becomes apparent there are at most two values between the extreme values of **calib_0** and **calib_1**, which account for the steepness of the slope in transition.

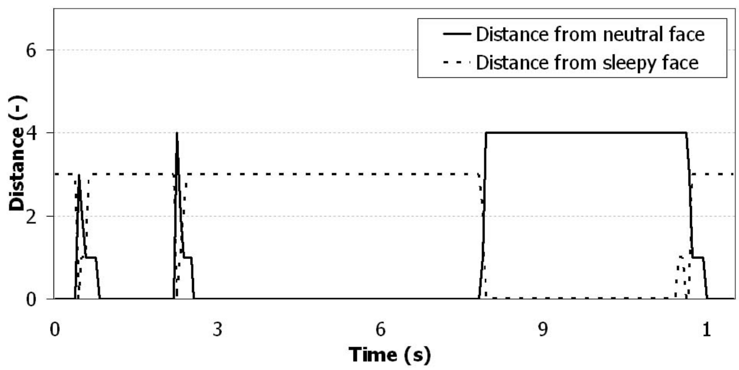


Figure 5.1: HypnosEuclidMean results

HypnosEuclidFull

The results of the HypnosEuclidFull algorithm are roughly similar to the HypnosEuclidMean results, as is clear from figure 5.2. The classification of whether the eyes are closed is similar to classification by human observation. HypnosEuclidFull however does not have the coarse scale problems, as HypnosEuclidMeans has, where lack of detail prevents proper analysis of transitions. Although the slopes of the graph remain steep, the distance to the **calib_0** or **calib_1** image increases or decreases within four to five subsequent images when closing or opening eyes.

HypnosDelta

HypnosDelta, which focuses on differences between subsequent pixel intensities, shows us different results. As is apparent by comparing figure 5.3 to figures

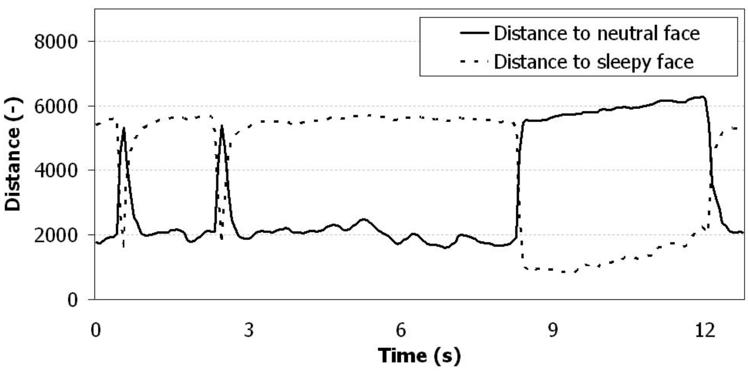


Figure 5.2: HypnosEuclidFull results

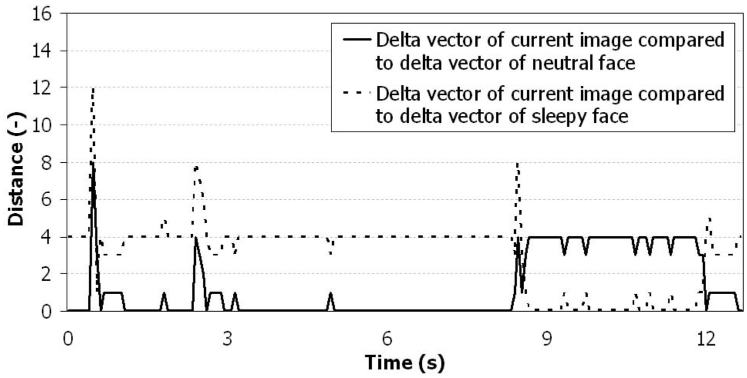


Figure 5.3: HypnosDelta results

5.1 and 5.2, the HypnosDelta algorithm classifies closed eyes correctly in non-transitionary, stable situations.

The *delta vector of the current image compared to the delta vector of the neutral face* as well as the *delta vector of the current image compared to the delta vector of the sleepy face* however both show peak values in transitions. In the HypnosEuclidMean and -Full algorithm the distance between the vectors in transitions show both a peak and a drop in value. The double-peak values in the HypnosDelta algorithm transitions lead to unreliable classification, although the transition data could be valuable for purposeful study of transitionary behavior.

5.2 Analysis

In non-transitionary states where the eyes are either open or closed, the three algorithms perform equally well. When comparing the results from the different algorithms, the transitions between states provide the most interesting data.

Table 5.2 lists the data of a single transition between closed and open eyes and table 5.3 lists the data of the first eye blink in the first test, which is actually a double transition. In these tables the 'N' signifies the distance from the acquired image to the **calib_0** image, and 'S' signifies the distance to the **calib_1** image.

As was mentioned in the previous section, HypnosDelta does not classify this blink as 'closed eyes'. Both the Euclidean based algorithms do, in fact, detect an eye closure, where HypnosEuclidFull classifies eye closure earlier than HypnosEuclidMeans. This last observation is even more apparent in table 5.2.

Although the algorithms use the exact same images as a source, they might classify these images totally different. This is apparent from row 3 in the same table, where the HypnosDelta classifies a 'neutral' with full confidence, and HypnosEuclidFull does this same classification two images later (which means 250 ms!). An examination of table A.1 and table A.2 in the appendix confirms the claim that this is not a coincidence.

Time		*HypnosEuclidMean*			*HypnosEuclidFull*			*HypnosDelta*		
time	*ms*	*N*	*S*	*sleepy*	*N*	*S*	*sleepy*	*N*	*S*	*sleepy*
14:38:18	225	4	0	TRUE	4205	1518	TRUE	4	0	TRUE
14:38:18	365	4	1	TRUE	4195	1539	TRUE	4	0	TRUE
14:38:18	506	2	0	TRUE	3734	1943	TRUE	0	4	FALSE
14:38:18	631	1	1	FALSE	3035	2774	TRUE	0	4	FALSE
14:38:18	772	0	3	FALSE	1927	3865	FALSE	1	3	FALSE
14:38:18	912	0	3	FALSE	1916	3915	FALSE	1	3	FALSE

Table 5.2: Excerpt of Hypnos data at a transition

Time		*HypnosEuclidMean*			*HypnosEuclidFull*			*HypnosDelta*		
time	*ms*	*N*	*S*	*sleepy*	*N*	*S*	*sleepy*	*N*	*S*	*sleepy*
14:38:06	694	0	3	FALSE	1707	3926	FALSE	0	4	FALSE
14:38:06	819	0	3	FALSE	1788	4135	FALSE	0	4	FALSE
14:38:06	960	2	1	TRUE	3074	1674	TRUE	8	12	FALSE
14:38:07	085	1	1	FALSE	3000	2481	TRUE	1	5	FALSE
14:38:07	225	1	2	FALSE	2304	3345	FALSE	1	3	FALSE
14:38:07	350	1	3	FALSE	2054	3592	FALSE	1	3	FALSE

Table 5.3: Excerpt of Hypnos data at eye blink

Our observations of the three implemented algorithms lead us to conclude that neither of the algorithms performs significantly better than the others, which is not surprising considering their similarity.

5.3 System Drawbacks

The test results in table A.1 have been obtained under strict controlled conditions. This raises questions about the robustness of the HADES-1 system. Exhaustive testing under different circumstances has led to several observations.

The subject's head should not be rotated or tilted for the system to be able to do proper classification; the system is relatively insensitive to minute movements of the head, but not when it is rotated or tilted.

HADES-1 is capable of dealing with subjects with glasses, although the eyes stand out less clear in the image and the distinguishability of the **calib_0** and **calib_1** images decreases. This leads to a more noisy classification of the facial images compared to a subject without glasses.

As we have considered the problem of face detection within images out of the scope of our research, the system is not capable of tracking any substantial head movements. All rigid head movements result in a misclassification by the system.

The HADES-1 system has a considerable amount of drawbacks, many of which could be solved easily. The next chapter will discuss possible enhancements in detail.

Chapter 6

Conclusion

6.1 Recommendations

Upon examining the HADES-1 system, we discover several areas for improvement. The face or eye detection is of great interest, since the current system misclassifies expressions when there is substantial movement of the head. Other measurement tools than the current algorithms might enhance the system. Finally, the data comparison method in our algorithms could be improved. These areas of improvement mirror the steps in the three-tier framework as introduced in chapter 3: 'face detection', 'facial expression feature extraction' and 'facial expression classification'.

Replacement of the user-selected rectangle of interest by an automated eye detection module would greatly enhance the system. This module can utilize the fact that drivers tend to keep their heads practically in the same position. The spatio-temporal relation between images can be used for tracking movements of the head, instead of processing each image independently.

When it comes to facial expression feature extraction, many different approaches have been discussed by Pantic et al. [6]. Kobayashi and Hara [5] use brightness distributions at certain fixed vertical lines in the face, which is similar to the HADES-1 system. Changing the current color model from RGB to HSV, a model based on brightness values, might enable us to retrieve more information from the images. Other, holistic, approaches such as the fitting of elastic graphs to facial images, or utilization of eigenfaces based on PCA, are possible and described in Pantic et al..

The classification of the images could also be improved. The Mahalanobis distance is a commonly used metric in the field of image classification. Compared to the Euclidean distance it corrects for correlation between the different features, since it is very sensitive to inter-variable changes in the template images. Besides changing the distance calculations it is possible to use alternative classification methods, such as neural networks or rule-based systems. Especially the former is commonly used in the world of facial expression recognition.

HADES-1 utilizes images acquired by a digital video camera. This digital video camera might be replaced with an infra-red camera. The result would be a typical red illumination of the eyes, comparable to the red eyes when taking a picture with flash, making them easy to detect.

6.2 Final Remarks

In this report we have presented a system for recognizing expressions of somnolence in the human face. When assessing its performance, based on data and results as presented in chapter five, we may conclude the HADES-1 system works well under ideal circumstances. In ideal situations the subject under scrutiny keeps his head straight and facing forward continuously. Even slight movements of the head however severely affect correct classification of somnolence. Yet more demanding circumstances such as person independency have not been taken into account at all in the current HADES system.

Considering the current level of technology we consider the application of a HADES-like system inside a moving vehicle, or anywhere else for that matter, where constant vigilance is of critical importance, well within the scope of possibilities.

Bibliography

[1] J.F. Cohn, A.J. Zlochower, J.J. Lien, and T. Kanade, "Feature-Point Tracking by Optical Flow Discriminates Subtle Differences in Facial Expression", *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pp. 396–401, 1998.

[2] J. Faber, M. Novák, P. Svoboda, and V. Tatarinov, "Electrical Brain Wave Analysis During Hypnagogium", *Neural Network World*, vol. 1, pp. 41–54, 2003.

[3] C.L. Huang and Y.M. Huang, "Facial Expression Recognition Using Model-Based Feature Extraction and Action Parameters Classification", *Journal of Visual Communication and Image Representation*, vol. 8, no. 3, pp. 278–290, 1997.

[4] S. Kimura and M. Yachida, "Facial Expression Recognition and Its Degree Estimation", *Proceedings of Computer Vision and Pattern Recognition*, pp. 295–300, 1997.

[5] H. Kobayashi and F. Hara, "Recognition of Six Basic Facial Expressions and Their Strength by Neural Network", *Proceedings of the International Workshop on Robot and Human Communication*, pp. 387–391, 1992.

[6] M. Pantic and L.J.M. Rothkrantz, "Automatic analysis of facial expressions: the state of the art", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1424–1445, 2000.

[7] M. Pantic and L.J.M. Rothkrantz, "An Expert System for Multiple Emotional Classification of Facial Expressions", *Proceedings of the International Conference on Tools with Artificial Intelligence*, pp. 113–120, 1999.

Appendix A

Hades Data

This Hades log file was created on Tuesday June 10th, 2003, when running all three algorithms simultaneously. The resulting frame rate was 6–7 frames per second for table A.1 and 9–10 frames per second for table A.2. Running any algorithm exclusively will result in a higher frame rate, see table 5.1.

The first two columns represent the time and the amount of milliseconds. The other columns list the data of each of the three algorithms: first the distance to the neutral face, then the distance to the sleepy face and finally the boolean whether the eyes are closed (TRUE) or not (FALSE).

Table A.1: Hypnos data – close-up of eyes

Time		*HypnosEuclidMean*			*HypnosEuclidFull*			*HypnosDelta*		
14:38:01	257	4	0	TRUE	4163	1281	TRUE	4	0	TRUE
14:38:01	476	4	0	TRUE	4207	1321	TRUE	4	0	TRUE
14:38:01	601	4	0	TRUE	4303	1471	TRUE	4	0	TRUE
14:38:01	741	4	0	TRUE	4392	1538	TRUE	4	0	TRUE
14:38:02	288	2	1	TRUE	3505	1726	TRUE	1	5	FALSE
14:38:02	444	1	2	FALSE	2707	2887	FALSE	2	6	FALSE
14:38:02	569	0	3	FALSE	1803	3790	FALSE	1	5	FALSE
14:38:02	710	0	3	FALSE	1578	3908	FALSE	0	4	FALSE
14:38:02	851	0	3	FALSE	1394	4018	FALSE	0	4	FALSE
14:38:02	976	0	3	FALSE	1477	4030	FALSE	0	4	FALSE
14:38:03	116	0	3	FALSE	1609	4073	FALSE	0	4	FALSE
14:38:03	241	0	3	FALSE	1708	4036	FALSE	0	4	FALSE
14:38:03	382	0	3	FALSE	1741	3999	FALSE	0	4	FALSE
14:38:03	522	0	3	FALSE	1752	4008	FALSE	0	4	FALSE
14:38:03	647	0	3	FALSE	1808	4026	FALSE	0	4	FALSE
14:38:03	788	0	3	FALSE	1876	4029	FALSE	0	4	FALSE
14:38:03	913	0	3	FALSE	1862	3984	FALSE	0	4	FALSE
14:38:04	038	0	3	FALSE	1852	3979	FALSE	0	4	FALSE
14:38:04	179	0	3	FALSE	1856	3974	FALSE	0	4	FALSE
14:38:04	304	0	3	FALSE	1874	3989	FALSE	0	4	FALSE
14:38:04	444	0	3	FALSE	1878	4010	FALSE	0	4	FALSE
14:38:04	569	0	3	FALSE	1763	3954	FALSE	0	4	FALSE
14:38:04	710	0	3	FALSE	1710	3955	FALSE	0	4	FALSE
14:38:04	835	0	3	FALSE	1780	4016	FALSE	0	4	FALSE
14:38:04	976	0	3	FALSE	1769	4018	FALSE	0	4	FALSE
14:38:05	101	0	3	FALSE	1730	3982	FALSE	0	4	FALSE
14:38:05	241	0	3	FALSE	1695	3999	FALSE	0	4	FALSE
14:38:05	366	0	3	FALSE	1715	4025	FALSE	0	4	FALSE
14:38:05	491	0	3	FALSE	1737	3978	FALSE	0	4	FALSE
14:38:05	632	0	3	FALSE	1818	4071	FALSE	0	4	FALSE
14:38:05	757	0	3	FALSE	1868	4072	FALSE	0	4	FALSE
14:38:05	897	0	3	FALSE	1934	4133	FALSE	0	4	FALSE
14:38:06	022	0	3	FALSE	1880	3985	FALSE	0	4	FALSE
14:38:06	163	0	3	FALSE	1806	3901	FALSE	0	4	FALSE
14:38:06	288	0	3	FALSE	1802	3807	FALSE	0	4	FALSE
14:38:06	429	0	3	FALSE	1743	3712	FALSE	0	4	FALSE
14:38:06	554	0	3	FALSE	1680	3782	FALSE	0	4	FALSE
14:38:06	694	0	3	FALSE	1707	3926	FALSE	0	4	FALSE

Continued on next page

Table A.1 – continued from previous page

Time		*HypnosEuclidMean*			*HypnosEuclidFull*			*HypnosDelta*		
14:38:06	819	0	3	FALSE	1788	4135	FALSE	0	4	FALSE
14:38:06	960	2	1	TRUE	3074	1674	TRUE	8	12	FALSE
14:38:07	085	1	1	FALSE	3000	2481	TRUE	1	5	FALSE
14:38:07	225	1	2	FALSE	2304	3345	FALSE	1	3	FALSE
14:38:07	350	1	3	FALSE	2054	3592	FALSE	1	3	FALSE
14:38:07	491	0	3	FALSE	1847	3824	FALSE	1	3	FALSE
14:38:07	616	0	3	FALSE	1862	3923	FALSE	1	3	FALSE
14:38:07	757	0	3	FALSE	1919	3987	FALSE	0	4	FALSE
14:38:07	882	0	3	FALSE	1913	4102	FALSE	0	4	FALSE
14:38:08	022	0	3	FALSE	1932	4161	FALSE	0	4	FALSE
14:38:08	147	0	3	FALSE	1950	4176	FALSE	0	4	FALSE
14:38:08	288	0	3	FALSE	1623	4006	FALSE	0	4	FALSE
14:38:08	413	0	3	FALSE	1625	4049	FALSE	0	4	FALSE
14:38:08	553	0	3	FALSE	1707	4101	FALSE	0	4	FALSE
14:38:08	678	0	3	FALSE	1853	4166	FALSE	0	4	FALSE
14:38:08	819	0	3	FALSE	1824	3620	FALSE	4	8	FALSE
14:38:08	944	3	0	TRUE	3693	972	TRUE	2	2	FALSE
14:38:09	085	1	1	FALSE	2821	2558	TRUE	0	4	FALSE
14:38:09	210	1	2	FALSE	2135	3242	FALSE	1	3	FALSE
14:38:09	350	1	3	FALSE	1763	3568	FALSE	1	3	FALSE
14:38:09	475	0	3	FALSE	1672	3769	FALSE	1	3	FALSE
14:38:09	616	0	3	FALSE	1732	3892	FALSE	1	3	FALSE
14:38:09	741	0	3	FALSE	1977	4054	FALSE	1	3	FALSE
14:38:09	882	0	3	FALSE	1963	4050	FALSE	0	4	FALSE
14:38:10	007	0	3	FALSE	1958	4035	FALSE	0	4	FALSE
14:38:10	147	0	3	FALSE	2035	4069	FALSE	0	4	FALSE
14:38:10	272	0	3	FALSE	2052	4040	FALSE	0	4	FALSE
14:38:10	413	0	3	FALSE	1976	4020	FALSE	0	4	FALSE
14:38:10	553	0	3	FALSE	2031	4087	FALSE	0	4	FALSE
14:38:10	678	0	3	FALSE	2062	4135	FALSE	0	4	FALSE
14:38:10	819	0	3	FALSE	2047	4135	FALSE	0	4	FALSE
14:38:10	944	0	3	FALSE	2178	4225	FALSE	0	4	FALSE
14:38:11	085	0	3	FALSE	2258	4285	FALSE	0	4	FALSE
14:38:11	210	0	3	FALSE	2126	4202	FALSE	1	3	FALSE
14:38:11	350	0	3	FALSE	2341	4384	FALSE	0	4	FALSE
14:38:11	475	0	3	FALSE	2417	4447	FALSE	0	4	FALSE
14:38:11	616	0	3	FALSE	2215	4318	FALSE	0	4	FALSE
14:38:11	756	0	3	FALSE	2083	4255	FALSE	0	4	FALSE
14:38:11	881	0	3	FALSE	1893	4158	FALSE	1	3	FALSE
14:38:12	022	0	3	FALSE	1774	4037	FALSE	0	4	FALSE
14:38:12	147	0	3	FALSE	1686	4025	FALSE	1	3	FALSE
14:38:12	288	0	3	FALSE	1747	4112	FALSE	1	3	FALSE
14:38:12	413	0	3	FALSE	1844	4143	FALSE	0	4	FALSE
14:38:12	553	0	3	FALSE	1877	4191	FALSE	0	4	FALSE
14:38:12	678	0	3	FALSE	1781	4143	FALSE	0	4	FALSE
14:38:12	819	0	3	FALSE	1644	4056	FALSE	0	4	FALSE
14:38:12	944	0	3	FALSE	1555	3941	FALSE	0	4	FALSE
14:38:13	084	0	3	FALSE	1576	3952	FALSE	1	3	FALSE
14:38:13	209	0	3	FALSE	1808	4115	FALSE	0	4	FALSE
14:38:13	350	0	3	FALSE	1916	4172	FALSE	0	4	FALSE
14:38:13	475	0	3	FALSE	1928	4176	FALSE	0	4	FALSE
14:38:13	616	0	3	FALSE	1778	4059	FALSE	0	4	FALSE
14:38:13	741	0	3	FALSE	1746	4022	FALSE	1	3	FALSE
14:38:13	881	0	3	FALSE	1675	4015	FALSE	0	4	FALSE
14:38:14	006	0	3	FALSE	1653	3987	FALSE	0	4	FALSE
14:38:14	147	0	3	FALSE	1603	3903	FALSE	0	4	FALSE
14:38:14	272	0	3	FALSE	1648	3955	FALSE	0	4	FALSE
14:38:14	413	0	3	FALSE	1749	3968	FALSE	1	3	FALSE
14:38:14	553	2	1	TRUE	3023	1745	TRUE	3	7	FALSE
14:38:14	678	4	0	TRUE	3709	826	TRUE	4	0	TRUE
14:38:14	819	3	0	TRUE	3739	770	TRUE	4	0	TRUE
14:38:14	944	3	0	TRUE	3717	771	TRUE	4	0	TRUE
14:38:15	084	4	0	TRUE	3785	819	TRUE	4	0	TRUE
14:38:15	225	4	0	TRUE	3849	881	TRUE	4	0	TRUE
14:38:15	350	4	0	TRUE	3841	885	TRUE	4	0	TRUE
14:38:15	491	4	0	TRUE	3831	914	TRUE	4	0	TRUE
14:38:15	631	4	0	TRUE	3846	910	TRUE	4	0	TRUE
14:38:15	756	4	0	TRUE	3863	988	TRUE	4	0	TRUE
14:38:15	897	4	0	TRUE	3855	955	TRUE	4	0	TRUE
14:38:16	037	4	0	TRUE	3888	946	TRUE	4	0	TRUE
14:38:16	178	4	0	TRUE	3969	1039	TRUE	4	0	TRUE
14:38:16	319	4	0	TRUE	4003	1085	TRUE	4	0	TRUE
14:38:16	444	4	0	TRUE	3989	1055	TRUE	4	0	TRUE
14:38:16	584	4	0	TRUE	3964	1055	TRUE	4	0	TRUE
14:38:16	725	4	0	TRUE	4024	1154	TRUE	4	0	TRUE
14:38:16	850	4	0	TRUE	3984	1150	TRUE	4	0	TRUE
14:38:16	991	4	0	TRUE	4000	1147	TRUE	4	0	TRUE
14:38:17	131	4	0	TRUE	4113	1271	TRUE	4	0	TRUE
14:38:17	272	4	0	TRUE	4133	1301	TRUE	4	0	TRUE
14:38:17	412	4	0	TRUE	4178	1382	TRUE	4	0	TRUE
14:38:17	537	4	0	TRUE	4208	1418	TRUE	4	0	TRUE
14:38:17	678	4	0	TRUE	4227	1460	TRUE	4	0	TRUE
14:38:17	819	4	0	TRUE	4120	1335	TRUE	4	0	TRUE
14:38:17	944	4	0	TRUE	4109	1318	TRUE	4	0	TRUE
14:38:18	084	4	0	TRUE	4187	1443	TRUE	4	0	TRUE
14:38:18	225	4	0	TRUE	4205	1518	TRUE	4	0	TRUE
14:38:18	365	4	1	TRUE	4195	1539	TRUE	4	0	TRUE
14:38:18	506	2	0	TRUE	3734	1943	TRUE	0	4	FALSE
14:38:18	631	1	1	FALSE	3035	2774	TRUE	0	4	FALSE
14:38:18	772	0	3	FALSE	1927	3865	FALSE	1	3	FALSE
14:38:18	912	0	3	FALSE	1916	3915	FALSE	1	3	FALSE
14:38:19	053	0	3	FALSE	1940	3884	FALSE	1	3	FALSE

Continued on next page

Table A.1 – continued from previous page

Time		*HypnosEuclidMean*			*HypnosEuclidFull*			*HypnosDelta*		
14:38:19	178	0	3	FALSE	1910	3842	FALSE	1	3	FALSE
14:38:19	319	0	3	FALSE	1943	3809	FALSE	1	3	FALSE
14:38:19	459	0	3	FALSE	1974	3745	FALSE	0	4	FALSE
14:38:19	600	0	3	FALSE	2013	3675	FALSE	1	3	FALSE
14:38:19	725	0	3	FALSE	1971	3669	FALSE	1	3	FALSE
14:38:19	865	0	3	FALSE	1943	3839	FALSE	1	3	FALSE
14:38:19	990	0	3	FALSE	1962	3882	FALSE	1	3	FALSE
14:38:20	131	0	3	FALSE	1974	3830	FALSE	1	3	FALSE
14:38:20	256	0	3	FALSE	1962	3807	FALSE	1	3	FALSE
14:38:20	397	1	3	FALSE	2024	3750	FALSE	1	3	FALSE
14:38:20	537	1	3	FALSE	2018	3754	FALSE	1	3	FALSE
14:38:20	662	1	3	FALSE	2037	3819	FALSE	0	4	FALSE
14:38:20	803	0	3	FALSE	2028	3871	FALSE	1	3	FALSE
14:38:20	928	0	3	FALSE	2022	3783	FALSE	1	3	FALSE
14:38:21	068	0	3	FALSE	1983	3741	FALSE	1	3	FALSE
14:38:21	193	0	3	FALSE	1949	3794	FALSE	0	4	FALSE
14:38:21	334	0	3	FALSE	1944	3826	FALSE	0	4	FALSE
14:38:21	475	0	3	FALSE	2001	3797	FALSE	0	4	FALSE
14:38:21	600	0	3	FALSE	1926	3875	FALSE	1	3	FALSE
14:38:21	740	0	3	FALSE	1958	3904	FALSE	0	4	FALSE
14:38:21	865	1	3	FALSE	2008	3942	FALSE	1	3	FALSE
14:38:22	006	0	3	FALSE	1976	3949	FALSE	1	3	FALSE
14:38:22	131	0	3	FALSE	1940	3874	FALSE	0	4	FALSE
14:38:22	272	1	3	FALSE	1980	3852	FALSE	0	4	FALSE
14:38:22	412	2	0	TRUE	3270	1403	TRUE	3	7	FALSE
14:38:22	537	4	0	TRUE	3879	1077	TRUE	4	0	TRUE
14:38:22	678	4	0	TRUE	3889	1060	TRUE	4	0	TRUE
14:38:22	803	4	0	TRUE	3870	1038	TRUE	4	0	TRUE
14:38:22	943	4	0	TRUE	3887	1058	TRUE	4	0	TRUE
14:38:23	084	4	0	TRUE	3887	1038	TRUE	4	0	TRUE
14:38:23	209	4	0	TRUE	3898	1046	TRUE	4	0	TRUE
14:38:23	350	4	0	TRUE	3865	1078	TRUE	4	0	TRUE
14:38:23	490	4	0	TRUE	3882	1080	TRUE	4	0	TRUE
14:38:23	631	4	0	TRUE	3888	1095	TRUE	4	0	TRUE
14:38:23	756	4	0	TRUE	4028	1220	TRUE	4	0	TRUE
14:38:23	896	4	0	TRUE	4020	1217	TRUE	4	0	TRUE
14:38:24	037	4	0	TRUE	3973	1254	TRUE	4	0	TRUE
14:38:24	178	4	0	TRUE	3944	1311	TRUE	4	0	TRUE
14:38:24	318	4	0	TRUE	3892	1332	TRUE	4	0	TRUE
14:38:24	443	4	0	TRUE	3891	1270	TRUE	4	0	TRUE
14:38:24	584	4	0	TRUE	4029	1303	TRUE	4	0	TRUE
14:38:24	725	4	0	TRUE	4037	1308	TRUE	4	0	TRUE
14:38:24	865	4	0	TRUE	4043	1332	TRUE	4	0	TRUE
14:38:24	990	4	0	TRUE	4020	1312	TRUE	4	0	TRUE
14:38:25	131	4	0	TRUE	3991	1326	TRUE	4	0	TRUE
14:38:25	271	4	0	TRUE	3972	1317	TRUE	4	0	TRUE
14:38:25	412	4	0	TRUE	4102	1358	TRUE	4	0	TRUE
14:38:25	553	4	0	TRUE	4172	1421	TRUE	4	0	TRUE
14:38:25	678	4	0	TRUE	4146	1432	TRUE	4	0	TRUE
14:38:25	818	4	0	TRUE	4148	1440	TRUE	4	0	TRUE
14:38:25	959	4	0	TRUE	4091	1441	TRUE	4	0	TRUE
14:38:26	099	4	0	TRUE	4074	1424	TRUE	4	0	TRUE
14:38:26	240	3	0	TRUE	4085	1680	TRUE	1	3	FALSE
14:38:26	365	2	1	TRUE	3617	2120	TRUE	2	6	FALSE
14:38:26	506	1	2	FALSE	2998	2897	TRUE	1	3	FALSE
14:38:26	646	1	3	FALSE	2345	3544	FALSE	1	3	FALSE
14:38:26	771	1	3	FALSE	2139	3731	FALSE	1	3	FALSE
14:38:26	912	1	3	FALSE	2113	3732	FALSE	1	3	FALSE
14:38:27	037	1	3	FALSE	2133	3660	FALSE	1	3	FALSE
14:38:27	178	1	3	FALSE	2120	3631	FALSE	1	3	FALSE
14:38:27	318	1	3	FALSE	2143	3763	FALSE	1	3	FALSE
14:38:27	443	1	3	FALSE	2140	3665	FALSE	1	3	FALSE
14:38:27	584	1	3	FALSE	2222	3583	FALSE	1	3	FALSE
14:38:27	709	1	3	FALSE	2276	3561	FALSE	1	3	FALSE
14:38:27	849	1	3	FALSE	2358	3482	FALSE	1	3	FALSE
14:38:27	990	1	3	FALSE	2360	3428	FALSE	1	3	FALSE
14:38:28	115	1	3	FALSE	2255	3537	FALSE	1	3	FALSE
14:38:28	256	1	3	FALSE	2215	3615	FALSE	1	3	FALSE
14:38:28	381	1	3	FALSE	2186	3662	FALSE	1	3	FALSE
14:38:28	521	1	3	FALSE	2174	3754	FALSE	1	3	FALSE
14:38:28	662	1	3	FALSE	2175	3766	FALSE	1	3	FALSE
14:38:28	787	1	3	FALSE	2211	3658	FALSE	1	3	FALSE
14:38:28	927	1	3	FALSE	2155	3733	FALSE	1	3	FALSE
14:38:29	068	1	3	FALSE	2108	3837	FALSE	1	3	FALSE
14:38:29	193	1	3	FALSE	2077	3913	FALSE	1	3	FALSE
14:38:29	334	1	3	FALSE	2068	3897	FALSE	0	4	FALSE
14:38:29	459	1	3	FALSE	2085	3864	FALSE	1	3	FALSE
14:38:29	599	1	3	FALSE	2342	3787	FALSE	0	4	FALSE
14:38:29	740	1	3	FALSE	2310	3709	FALSE	1	3	FALSE
14:38:29	865	1	3	FALSE	2166	3803	FALSE	1	3	FALSE
14:38:30	006	1	3	FALSE	2141	3937	FALSE	0	4	FALSE
14:38:30	130	1	3	FALSE	2154	4036	FALSE	1	3	FALSE
14:38:30	271	1	3	FALSE	2120	3927	FALSE	1	3	FALSE
14:38:30	412	1	3	FALSE	2145	3902	FALSE	0	4	FALSE
14:38:30	537	1	3	FALSE	2314	3625	FALSE	1	3	FALSE
14:38:30	677	1	3	FALSE	2284	3629	FALSE	1	3	FALSE
14:38:30	818	4	0	TRUE	3928	1282	TRUE	4	0	TRUE
14:38:30	943	4	0	TRUE	3940	1221	TRUE	4	0	TRUE
14:38:31	084	4	0	TRUE	3963	1198	TRUE	4	0	TRUE
14:38:31	209	4	0	TRUE	4002	1207	TRUE	4	0	TRUE
14:38:31	349	4	0	TRUE	4020	1242	TRUE	4	0	TRUE
14:38:31	490	4	0	TRUE	4019	1272	TRUE	4	0	TRUE

Continued on next page

Table A.1 – continued from previous page

Time		*HypnosEuclidMean*			*HypnosEuclidFull*			*HypnosDelta*		
14:38:31	630	4	0	TRUE	4010	1248	TRUE	4	0	TRUE
14:38:31	771	4	0	TRUE	4032	1282	TRUE	4	0	TRUE
14:38:31	896	4	0	TRUE	4139	1426	TRUE	4	0	TRUE
14:38:32	037	4	0	TRUE	4136	1412	TRUE	4	0	TRUE
14:38:32	177	4	0	TRUE	4186	1488	TRUE	4	0	TRUE
14:38:32	318	4	0	TRUE	4261	1625	TRUE	4	0	TRUE
14:38:32	459	4	0	TRUE	4266	1598	TRUE	4	0	TRUE
14:38:32	599	4	0	TRUE	4354	1669	TRUE	4	0	TRUE
14:38:32	724	4	0	TRUE	4433	1759	TRUE	4	0	TRUE
14:38:32	865	4	0	TRUE	4452	1775	TRUE	4	0	TRUE
14:38:33	005	4	0	TRUE	4454	1812	TRUE	4	0	TRUE
14:38:33	146	4	0	TRUE	4437	1791	TRUE	4	0	TRUE
14:38:33	287	4	0	TRUE	4425	1765	TRUE	4	0	TRUE
14:38:33	427	4	0	TRUE	4285	1586	TRUE	4	0	TRUE
14:38:33	568	4	0	TRUE	4378	1717	TRUE	4	0	TRUE
14:38:33	693	4	0	TRUE	4426	1759	TRUE	4	0	TRUE
14:38:33	833	4	0	TRUE	4439	1827	TRUE	4	0	TRUE
14:38:33	974	4	1	TRUE	4480	1902	TRUE	4	0	TRUE
14:38:34	115	4	1	TRUE	4491	1946	TRUE	4	0	TRUE
14:38:34	255	4	1	TRUE	4452	1904	TRUE	4	0	TRUE
14:38:34	396	4	1	TRUE	4479	1923	TRUE	4	0	TRUE
14:38:34	521	4	1	TRUE	4545	2006	TRUE	4	0	TRUE
14:38:34	677	4	1	TRUE	4555	2026	TRUE	4	0	TRUE
14:38:34	802	4	1	TRUE	4521	1990	TRUE	4	0	TRUE
14:38:34	943	0	3	FALSE	2260	4122	FALSE	1	5	FALSE
14:38:35	083	0	4	FALSE	2203	4647	FALSE	0	4	FALSE
14:38:35	224	0	4	FALSE	2433	4937	FALSE	0	4	FALSE
14:38:35	365	0	4	FALSE	2496	5010	FALSE	0	4	FALSE
14:38:35	490	0	4	FALSE	2322	4904	FALSE	0	4	FALSE
14:38:35	630	0	4	FALSE	2252	4858	FALSE	0	4	FALSE
14:38:35	771	0	4	FALSE	2254	4854	FALSE	0	4	FALSE
14:38:35	896	1	3	FALSE	2513	3708	FALSE	11	15	FALSE
14:38:36	036	1	3	FALSE	2947	3406	FALSE	1	3	FALSE
14:38:36	177	1	3	FALSE	2771	3670	FALSE	1	3	FALSE
14:38:36	302	1	3	FALSE	2610	3905	FALSE	1	3	FALSE
14:38:36	443	1	3	FALSE	2489	3834	FALSE	1	3	FALSE
14:38:36	583	1	3	FALSE	2289	3740	FALSE	0	4	FALSE

Table A.2: Hypnos data – distant from eyes

Time		*HypnosEuclidMean*			*HypnosEuclidFull*			*HypnosDelta*		
18:06:46	362	5	0	TRUE	2090	494	TRUE	8	0	TRUE
18:06:46	471	5	0	TRUE	2083	483	TRUE	8	0	TRUE
18:06:46	565	5	0	TRUE	2022	457	TRUE	7	1	TRUE
18:06:46	674	2	2	FALSE	1574	1137	TRUE	2	6	FALSE
18:06:46	799	1	4	FALSE	1030	1746	FALSE	1	7	FALSE
18:06:46	877	0	4	FALSE	842	1845	FALSE	1	7	FALSE
18:06:47	002	0	4	FALSE	909	1806	FALSE	0	8	FALSE
18:06:47	112	0	4	FALSE	965	1759	FALSE	0	8	FALSE
18:06:47	237	0	4	FALSE	987	1750	FALSE	1	7	FALSE
18:06:47	315	0	4	FALSE	1004	1747	FALSE	0	8	FALSE
18:06:47	440	0	4	FALSE	1006	1721	FALSE	0	8	FALSE
18:06:47	565	0	4	FALSE	799	1783	FALSE	1	7	FALSE
18:06:47	674	0	4	FALSE	663	1822	FALSE	1	7	FALSE
18:06:47	752	0	4	FALSE	636	1843	FALSE	0	8	FALSE
18:06:47	877	0	4	FALSE	658	1829	FALSE	1	7	FALSE
18:06:48	002	0	4	FALSE	771	1788	FALSE	1	7	FALSE
18:06:48	112	0	4	FALSE	864	1751	FALSE	1	7	FALSE
18:06:48	206	0	4	FALSE	920	1730	FALSE	0	8	FALSE
18:06:48	315	0	4	FALSE	1023	1689	FALSE	1	7	FALSE
18:06:48	440	0	4	FALSE	970	1706	FALSE	1	7	FALSE
18:06:48	518	0	4	FALSE	891	1732	FALSE	1	7	FALSE
18:06:48	643	0	4	FALSE	736	1797	FALSE	1	7	FALSE
18:06:48	752	0	4	FALSE	750	1799	FALSE	0	8	FALSE
18:06:48	877	0	4	FALSE	808	1767	FALSE	0	8	FALSE
18:06:48	956	0	4	FALSE	869	1746	FALSE	1	7	FALSE
18:06:49	081	0	4	FALSE	861	1750	FALSE	1	7	FALSE
18:06:49	206	0	4	FALSE	936	1732	FALSE	1	7	FALSE
18:06:49	284	0	4	FALSE	876	1732	FALSE	1	7	FALSE
18:06:49	393	0	4	FALSE	763	1777	FALSE	1	7	FALSE
18:06:49	518	0	5	FALSE	634	1825	FALSE	1	7	FALSE
18:06:49	643	0	5	FALSE	643	1837	FALSE	1	7	FALSE
18:06:49	721	0	5	FALSE	645	1822	FALSE	1	7	FALSE
18:06:49	846	0	5	FALSE	651	1816	FALSE	1	7	FALSE
18:06:49	956	0	5	FALSE	688	1836	FALSE	1	7	FALSE
18:06:50	081	0	5	FALSE	679	1821	FALSE	0	8	FALSE
18:06:50	159	0	5	FALSE	704	1800	FALSE	1	7	FALSE
18:06:50	284	0	5	FALSE	715	1823	FALSE	1	7	FALSE
18:06:50	393	0	5	FALSE	696	1833	FALSE	1	7	FALSE
18:06:50	518	0	5	FALSE	718	1864	FALSE	1	7	FALSE
18:06:50	596	0	5	FALSE	698	1844	FALSE	1	7	FALSE

Continued on next page

Table A.2 – continued from previous page

Time		*HypnosEuclidMean*			*HypnosEuclidFull*			*HypnosDelta*		
18:06:50	721	0	5	FALSE	712	1839	FALSE	1	7	FALSE
18:06:50	846	0	5	FALSE	710	1819	FALSE	1	7	FALSE
18:06:50	956	0	5	FALSE	696	1789	FALSE	0	8	FALSE
18:06:51	034	0	4	FALSE	783	1751	FALSE	1	7	FALSE
18:06:51	159	0	5	FALSE	782	1777	FALSE	1	7	FALSE
18:06:51	284	0	5	FALSE	755	1778	FALSE	0	8	FALSE
18:06:51	362	1	5	FALSE	764	1805	FALSE	1	7	FALSE
18:06:51	471	1	5	FALSE	774	1788	FALSE	1	7	FALSE
18:06:51	596	1	5	FALSE	821	1753	FALSE	0	8	FALSE
18:06:51	721	1	5	FALSE	832	1757	FALSE	1	7	FALSE
18:06:51	799	1	5	FALSE	852	1744	FALSE	1	7	FALSE
18:06:51	924	1	5	FALSE	893	1719	FALSE	0	8	FALSE
18:06:52	034	1	5	FALSE	930	1699	FALSE	1	7	FALSE
18:06:52	159	1	5	FALSE	884	1728	FALSE	1	7	FALSE
18:06:52	237	1	5	FALSE	866	1762	FALSE	0	8	FALSE
18:06:52	362	1	5	FALSE	874	1804	FALSE	1	7	FALSE
18:06:52	471	1	5	FALSE	883	1783	FALSE	1	7	FALSE
18:06:52	596	1	5	FALSE	926	1767	FALSE	1	7	FALSE
18:06:52	674	1	5	FALSE	948	1769	FALSE	1	7	FALSE
18:06:52	799	1	5	FALSE	996	1792	FALSE	1	7	FALSE
18:06:52	924	1	5	FALSE	1036	1779	FALSE	1	7	FALSE
18:06:53	034	1	5	FALSE	1026	1768	FALSE	2	6	FALSE
18:06:53	112	1	5	FALSE	1013	1771	FALSE	2	6	FALSE
18:06:53	237	1	5	FALSE	1004	1800	FALSE	1	7	FALSE
18:06:53	362	1	5	FALSE	1056	1816	FALSE	1	7	FALSE
18:06:53	440	1	5	FALSE	1028	1795	FALSE	1	7	FALSE
18:06:53	565	1	5	FALSE	998	1761	FALSE	1	7	FALSE
18:06:53	674	1	5	FALSE	966	1730	FALSE	1	7	FALSE
18:06:53	799	1	5	FALSE	990	1719	FALSE	1	7	FALSE
18:06:53	878	1	5	FALSE	993	1708	FALSE	1	7	FALSE
18:06:54	003	1	5	FALSE	1022	1687	FALSE	1	7	FALSE
18:06:54	112	1	5	FALSE	1029	1715	FALSE	1	7	FALSE
18:06:54	206	1	5	FALSE	1017	1738	FALSE	1	7	FALSE
18:06:54	315	1	5	FALSE	1032	1766	FALSE	1	7	FALSE
18:06:54	440	1	5	FALSE	1044	1762	FALSE	1	7	FALSE
18:06:54	565	1	5	FALSE	1069	1762	FALSE	1	7	FALSE
18:06:54	674	1	5	FALSE	1071	1769	FALSE	1	7	FALSE
18:06:54	753	1	5	FALSE	1064	1778	FALSE	1	7	FALSE
18:06:54	878	1	5	FALSE	1031	1743	FALSE	1	7	FALSE
18:06:55	003	1	5	FALSE	1028	1741	FALSE	1	7	FALSE
18:06:55	112	1	5	FALSE	1022	1723	FALSE	0	8	FALSE
18:06:55	206	1	3	FALSE	1164	1469	FALSE	1	9	FALSE
18:06:55	315	4	2	TRUE	1850	892	TRUE	7	1	TRUE
18:06:55	440	4	2	TRUE	1896	877	TRUE	7	1	TRUE
18:06:55	518	4	2	TRUE	1888	876	TRUE	8	0	TRUE
18:06:55	643	4	2	TRUE	1897	867	TRUE	7	1	TRUE
18:06:56	034	4	1	TRUE	1920	841	TRUE	7	1	TRUE
18:06:56	096	4	1	TRUE	1916	841	TRUE	7	1	TRUE
18:06:56	206	4	1	TRUE	1928	844	TRUE	7	1	TRUE
18:06:56	315	4	1	TRUE	1927	874	TRUE	7	1	TRUE
18:06:56	393	4	1	TRUE	1927	873	TRUE	7	1	TRUE
18:06:56	518	4	1	TRUE	1927	866	TRUE	7	1	TRUE
18:06:56	643	4	1	TRUE	1936	840	TRUE	8	0	TRUE
18:06:56	721	4	1	TRUE	1957	818	TRUE	7	1	TRUE
18:06:56	846	4	1	TRUE	1955	826	TRUE	7	1	TRUE
18:06:56	956	4	1	TRUE	1940	858	TRUE	7	1	TRUE
18:06:57	081	4	1	TRUE	1966	916	TRUE	7	1	TRUE
18:06:57	159	4	1	TRUE	1977	922	TRUE	7	1	TRUE
18:06:57	284	4	1	TRUE	1960	920	TRUE	7	1	TRUE
18:06:57	393	4	1	TRUE	1979	889	TRUE	7	1	TRUE
18:06:57	518	4	1	TRUE	1981	884	TRUE	7	1	TRUE
18:06:57	596	4	1	TRUE	1975	865	TRUE	7	1	TRUE
18:06:57	721	4	1	TRUE	1966	836	TRUE	7	1	TRUE
18:06:57	846	4	1	TRUE	1969	869	TRUE	7	1	TRUE
18:06:57	956	4	1	TRUE	1961	901	TRUE	7	1	TRUE
18:06:58	034	4	1	TRUE	1971	923	TRUE	7	1	TRUE
18:06:58	159	4	1	TRUE	1968	894	TRUE	7	1	TRUE
18:06:58	284	4	1	TRUE	1962	873	TRUE	7	1	TRUE
18:06:58	362	4	1	TRUE	1958	865	TRUE	7	1	TRUE
18:06:58	471	4	1	TRUE	1964	901	TRUE	7	1	TRUE
18:06:58	596	4	1	TRUE	1970	914	TRUE	7	1	TRUE
18:06:58	721	4	1	TRUE	1976	895	TRUE	7	1	TRUE
18:06:58	800	4	1	TRUE	1985	935	TRUE	7	1	TRUE
18:06:58	925	4	1	TRUE	1991	978	TRUE	7	1	TRUE
18:06:59	034	4	1	TRUE	2014	974	TRUE	7	1	TRUE
18:06:59	159	4	1	TRUE	1992	931	TRUE	7	1	TRUE
18:06:59	237	4	1	TRUE	1982	892	TRUE	7	1	TRUE
18:06:59	362	4	1	TRUE	1896	940	TRUE	7	1	TRUE
18:06:59	471	3	2	TRUE	1496	1349	TRUE	4	4	FALSE
18:06:59	596	1	4	FALSE	1121	1740	FALSE	2	6	FALSE
18:06:59	675	1	5	FALSE	1102	1840	FALSE	2	6	FALSE
18:06:59	800	1	5	FALSE	1161	1927	FALSE	2	6	FALSE
18:06:59	925	1	5	FALSE	1144	1917	FALSE	2	6	FALSE
18:07:00	003	1	5	FALSE	1151	1920	FALSE	2	6	FALSE
18:07:00	112	1	5	FALSE	1120	1907	FALSE	2	6	FALSE
18:07:00	237	1	5	FALSE	1112	1890	FALSE	2	6	FALSE
18:07:00	362	1	5	FALSE	1116	1897	FALSE	2	6	FALSE
18:07:00	471	1	5	FALSE	1050	1871	FALSE	2	6	FALSE
18:07:00	565	1	5	FALSE	1084	1881	FALSE	2	6	FALSE
18:07:00	675	1	5	FALSE	1144	1924	FALSE	2	6	FALSE
18:07:00	800	1	5	FALSE	1196	1978	FALSE	2	6	FALSE
18:07:00	878	1	5	FALSE	1216	1985	FALSE	2	6	FALSE

Continued on next page

Table A.2 – continued from previous page

Time		*HypnosEuclidMean*			*HypnosEuclidFull*			*HypnosDelta*		
18:07:01	003	1	5	FALSE	1216	1959	FALSE	2	6	FALSE
18:07:01	112	1	5	FALSE	1207	1955	FALSE	1	7	FALSE
18:07:01	237	1	5	FALSE	1208	1949	FALSE	1	7	FALSE
18:07:01	315	1	5	FALSE	1212	1948	FALSE	1	7	FALSE
18:07:01	440	1	5	FALSE	1123	1919	FALSE	1	7	FALSE
18:07:01	565	1	5	FALSE	1159	1930	FALSE	1	7	FALSE
18:07:01	675	1	5	FALSE	1153	1928	FALSE	1	7	FALSE
18:07:01	753	1	5	FALSE	1182	1951	FALSE	1	7	FALSE
18:07:01	878	1	5	FALSE	1208	1960	FALSE	1	7	FALSE
18:07:02	003	1	5	FALSE	1228	1960	FALSE	1	7	FALSE
18:07:02	081	1	5	FALSE	1250	1973	FALSE	1	7	FALSE
18:07:02	206	1	5	FALSE	1286	2006	FALSE	1	7	FALSE
18:07:02	315	1	5	FALSE	1259	1999	FALSE	1	7	FALSE
18:07:02	440	1	5	FALSE	1253	1996	FALSE	1	7	FALSE
18:07:02	518	1	5	FALSE	1313	2022	FALSE	1	7	FALSE
18:07:02	643	1	5	FALSE	1352	2038	FALSE	1	7	FALSE
18:07:02	753	1	5	FALSE	1391	2059	FALSE	2	6	FALSE
18:07:02	847	1	5	FALSE	1368	2037	FALSE	1	7	FALSE
18:07:02	956	1	5	FALSE	1375	2048	FALSE	1	7	FALSE
18:07:03	081	1	5	FALSE	1422	2075	FALSE	1	7	FALSE
18:07:03	206	1	5	FALSE	1386	2067	FALSE	2	6	FALSE
18:07:03	315	1	5	FALSE	1347	2019	FALSE	2	6	FALSE
18:07:03	393	1	5	FALSE	1307	1910	FALSE	2	6	FALSE
18:07:03	518	4	2	TRUE	1737	1297	TRUE	7	1	TRUE
18:07:03	643	4	2	TRUE	1882	1229	TRUE	7	1	TRUE
18:07:03	722	4	2	TRUE	1906	1199	TRUE	7	1	TRUE
18:07:03	847	4	2	TRUE	1926	1194	TRUE	7	1	TRUE
18:07:03	956	4	2	TRUE	1945	1155	TRUE	7	1	TRUE
18:07:04	081	4	2	TRUE	1955	1100	TRUE	7	1	TRUE
18:07:04	159	4	2	TRUE	1953	1048	TRUE	7	1	TRUE
18:07:04	284	4	2	TRUE	1960	1071	TRUE	7	1	TRUE
18:07:04	393	4	2	TRUE	1993	1101	TRUE	7	1	TRUE
18:07:04	518	4	2	TRUE	2015	1116	TRUE	7	1	TRUE
18:07:04	597	4	2	TRUE	2008	1101	TRUE	7	1	TRUE
18:07:04	722	4	2	TRUE	2013	1082	TRUE	7	1	TRUE
18:07:04	847	4	1	TRUE	2022	1042	TRUE	7	1	TRUE
18:07:04	956	4	1	TRUE	2046	1050	TRUE	7	1	TRUE
18:07:05	034	4	1	TRUE	2028	1023	TRUE	7	1	TRUE
18:07:05	159	4	1	TRUE	2057	1057	TRUE	7	1	TRUE
18:07:05	284	4	1	TRUE	2068	1073	TRUE	7	1	TRUE
18:07:05	362	4	1	TRUE	2081	1098	TRUE	7	1	TRUE
18:07:05	472	4	1	TRUE	2084	1101	TRUE	7	1	TRUE
18:07:05	597	4	1	TRUE	2075	1084	TRUE	7	1	TRUE
18:07:05	722	4	1	TRUE	2084	1101	TRUE	7	1	TRUE
18:07:05	800	4	1	TRUE	2098	1098	TRUE	7	1	TRUE
18:07:05	925	4	1	TRUE	2106	1072	TRUE	7	1	TRUE
18:07:06	034	4	1	TRUE	2089	1105	TRUE	7	1	TRUE
18:07:06	159	4	2	TRUE	2110	1186	TRUE	7	1	TRUE
18:07:06	237	4	2	TRUE	2117	1216	TRUE	7	1	TRUE
18:07:06	362	4	2	TRUE	2147	1282	TRUE	7	1	TRUE
18:07:06	472	4	3	TRUE	2120	1313	TRUE	7	1	TRUE
18:07:06	565	4	3	TRUE	2124	1316	TRUE	7	1	TRUE
18:07:06	675	4	3	TRUE	2176	1359	TRUE	7	1	TRUE
18:07:06	800	4	2	TRUE	2208	1306	TRUE	7	1	TRUE
18:07:06	925	4	2	TRUE	2196	1268	TRUE	7	1	TRUE
18:07:07	034	5	2	TRUE	2229	1272	TRUE	7	1	TRUE
18:07:07	112	5	2	TRUE	2203	1309	TRUE	7	1	TRUE
18:07:07	237	3	4	FALSE	1679	1766	FALSE	4	4	FALSE
18:07:07	362	2	4	FALSE	1451	2068	FALSE	3	5	FALSE
18:07:07	440	2	5	FALSE	1462	2103	FALSE	3	5	FALSE
18:07:07	565	2	5	FALSE	1498	2144	FALSE	2	6	FALSE
18:07:07	675	2	5	FALSE	1434	2121	FALSE	2	6	FALSE
18:07:07	800	1	5	FALSE	1410	2093	FALSE	2	6	FALSE
18:07:07	878	1	5	FALSE	1383	2048	FALSE	2	6	FALSE
18:07:08	003	1	5	FALSE	1370	2001	FALSE	2	6	FALSE
18:07:08	112	1	5	FALSE	1374	1997	FALSE	2	6	FALSE
18:07:08	237	1	5	FALSE	1367	2001	FALSE	2	6	FALSE
18:07:08	315	1	5	FALSE	1366	2008	FALSE	2	6	FALSE
18:07:08	440	1	5	FALSE	1421	2062	FALSE	2	6	FALSE
18:07:08	565	1	5	FALSE	1453	2106	FALSE	2	6	FALSE
18:07:08	644	1	5	FALSE	1434	2079	FALSE	2	6	FALSE
18:07:08	753	1	5	FALSE	1439	2083	FALSE	2	6	FALSE
18:07:08	878	1	5	FALSE	1476	2113	FALSE	2	6	FALSE
18:07:09	003	2	5	FALSE	1513	2134	FALSE	2	6	FALSE
18:07:09	081	1	5	FALSE	1495	2115	FALSE	2	6	FALSE
18:07:09	206	1	5	FALSE	1500	2121	FALSE	2	6	FALSE
18:07:09	315	2	5	FALSE	1495	2122	FALSE	2	6	FALSE
18:07:09	440	2	5	FALSE	1499	2122	FALSE	2	6	FALSE
18:07:09	519	1	5	FALSE	1443	2098	FALSE	2	6	FALSE
18:07:09	644	2	5	FALSE	1499	2124	FALSE	2	6	FALSE
18:07:09	753	2	5	FALSE	1580	2173	FALSE	2	6	FALSE
18:07:09	878	2	5	FALSE	1694	2219	FALSE	1	7	FALSE
18:07:09	956	2	5	FALSE	1686	2214	FALSE	2	6	FALSE
18:07:10	081	2	5	FALSE	1662	2189	FALSE	2	6	FALSE
18:07:10	206	2	5	FALSE	1650	2195	FALSE	2	6	FALSE
18:07:10	284	2	5	FALSE	1655	2197	FALSE	1	7	FALSE
18:07:10	394	2	5	FALSE	1622	2176	FALSE	2	6	FALSE
18:07:10	519	2	5	FALSE	1629	2164	FALSE	2	6	FALSE
18:07:10	644	3	4	FALSE	1498	1893	FALSE	2	6	FALSE
18:07:10	722	4	4	FALSE	1769	1552	TRUE	6	2	TRUE
18:07:10	847	4	3	TRUE	2017	1415	TRUE	7	1	TRUE
18:07:10	956	4	3	TRUE	2030	1417	TRUE	7	1	TRUE

Continued on next page

Table A.2 – continued from previous page

Time		*HypnosEuclidMean*			*HypnosEuclidFull*			*HypnosDelta*		
18:07:11	081	4	3	TRUE	2052	1410	TRUE	7	1	TRUE
18:07:11	159	4	3	TRUE	2050	1405	TRUE	7	1	TRUE
18:07:11	284	4	3	TRUE	2076	1385	TRUE	7	1	TRUE
18:07:11	394	4	3	TRUE	2091	1297	TRUE	7	1	TRUE
18:07:11	519	4	3	TRUE	2111	1326	TRUE	7	1	TRUE
18:07:11	597	4	3	TRUE	2115	1338	TRUE	7	1	TRUE
18:07:11	722	4	3	TRUE	2146	1344	TRUE	7	1	TRUE
18:07:11	847	4	3	TRUE	2118	1319	TRUE	7	1	TRUE
18:07:11	956	5	3	TRUE	2138	1307	TRUE	7	1	TRUE
18:07:12	034	5	3	TRUE	2150	1262	TRUE	7	1	TRUE
18:07:12	159	5	3	TRUE	2190	1202	TRUE	7	1	TRUE
18:07:12	284	5	2	TRUE	2214	1136	TRUE	7	1	TRUE
18:07:12	394	5	2	TRUE	2215	1167	TRUE	7	1	TRUE
18:07:12	472	5	3	TRUE	2131	1299	TRUE	7	1	TRUE
18:07:12	597	4	4	FALSE	1739	1695	TRUE	4	4	FALSE
18:07:12	722	2	5	FALSE	1464	2034	FALSE	3	5	FALSE
18:07:12	800	2	5	FALSE	1522	2094	FALSE	3	5	FALSE
18:07:12	925	3	5	FALSE	1529	2121	FALSE	2	6	FALSE
18:07:13	034	3	5	FALSE	1557	2130	FALSE	2	6	FALSE
18:07:13	159	3	5	FALSE	1551	2135	FALSE	2	6	FALSE
18:07:13	237	3	5	FALSE	1507	2108	FALSE	2	6	FALSE
18:07:13	362	3	5	FALSE	1559	2117	FALSE	2	6	FALSE
18:07:13	472	3	5	FALSE	1618	2160	FALSE	2	6	FALSE
18:07:13	597	3	5	FALSE	1675	2193	FALSE	2	6	FALSE
18:07:13	675	3	5	FALSE	1659	2193	FALSE	2	6	FALSE
18:07:13	800	3	5	FALSE	1635	2169	FALSE	2	6	FALSE
18:07:13	925	3	5	FALSE	1632	2162	FALSE	2	6	FALSE
18:07:14	034	3	5	FALSE	1633	2162	FALSE	1	7	FALSE
18:07:14	112	3	5	FALSE	1620	2159	FALSE	1	7	FALSE
18:07:14	237	3	5	FALSE	1609	2147	FALSE	2	6	FALSE
18:07:14	362	3	5	FALSE	1658	2174	FALSE	1	7	FALSE
18:07:14	441	3	5	FALSE	1661	2172	FALSE	1	7	FALSE
18:07:14	566	3	5	FALSE	1662	2167	FALSE	1	7	FALSE
18:07:14	675	3	5	FALSE	1643	2150	FALSE	1	7	FALSE
18:07:14	800	3	5	FALSE	1614	2136	FALSE	1	7	FALSE
18:07:14	878	3	5	FALSE	1591	2129	FALSE	1	7	FALSE
18:07:15	003	3	5	FALSE	1555	2101	FALSE	1	7	FALSE
18:07:15	112	2	5	FALSE	1490	2053	FALSE	1	7	FALSE
18:07:15	237	2	5	FALSE	1499	2038	FALSE	1	7	FALSE
18:07:15	316	2	5	FALSE	1522	2063	FALSE	1	7	FALSE
18:07:15	441	3	5	FALSE	1524	2078	FALSE	1	7	FALSE
18:07:15	566	3	5	FALSE	1524	2066	FALSE	1	7	FALSE
18:07:15	675	3	5	FALSE	1506	2043	FALSE	1	7	FALSE
18:07:15	753	3	5	FALSE	1498	2028	FALSE	1	7	FALSE
18:07:15	878	3	5	FALSE	1485	2021	FALSE	1	7	FALSE
18:07:16	003	3	5	FALSE	1417	1967	FALSE	0	8	FALSE
18:07:16	081	3	5	FALSE	1419	1961	FALSE	1	7	FALSE

Appendix B

Hades User's Guide

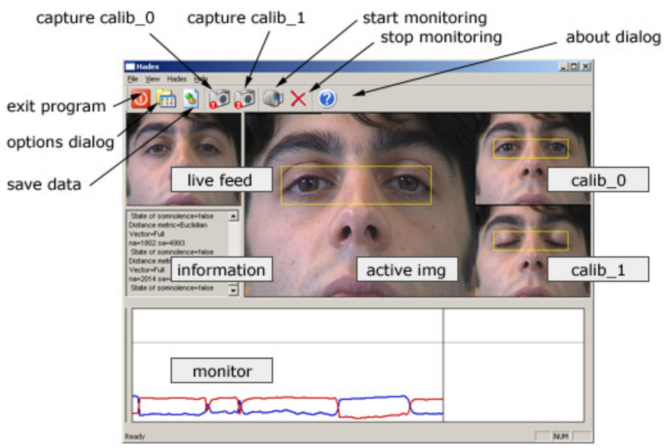


Figure B.1: HADES-1 User Interface

When the program executes, the user is presented with an interface as is illustrated in figure B.1. Upon commencing the program will immediately show the camera input in the *live feed* window.[1] By pressing the buttons labeled *calib_0* and *calib_1* the user can obtain the two calibration images. The *start recording* button will bring the *monitor* window and the *active image* window online and the *stop recording* button will stop the monitoring. Information about the algorithm and its parameters will be show in the *information* window.

The *options dialog* can be opened by pressing the appropriate button. The system parameters that can be altered in the dialog are shown in tabel B.1.

[1]assuming a camera is connected, functioning properly, etc.

General	
Alarm on/off	*on*
Euclidean distance parameter	2.0
Lambda factor	1.0
Hypnos	
HypnosEuclidMean algorithm	*default*
HypnosEuclidFull algorithm	
HypnosDelta algorithm	
Persephone	
Alarm delay	3
Frame rate	10 fps
Error threshold	0.7

Table B.1: System options

Appendix C

HADES-1 Documentation

C.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

C.2 CAboutDlg Class Reference

C.2.1 Detailed Description

The Class CAboutDlg.

Author:

Harmen Dikkers and Mike Spaans

Version:

1.0, 20-05-2003

Public Types

- enum { **IDD** = IDD_ABOUTBOX }

Public Member Functions

- **CAboutDlg** ()
  CAboutDlg constructor.

Protected Member Functions

- virtual void **DoDataExchange** (CDataExchange *pDX)
  Void.

C.2.2 Constructor & Destructor Documentation

C.2.2.1 CAboutDlg::CAboutDlg ()

CAboutDlg constructor.

Author:
Harmen Dikkers and Mike Spaans

Version:
1.0, 20-05-2003

C.2.3 Member Function Documentation

C.2.3.1 void CAboutDlg::DoDataExchange (CDataExchange * *pDX*) `[protected, virtual]`

Void.

Author:
Harmen Dikkers and Mike Spaans

Version:
1.0, 20-05-2003

C.3 CArgos Class Reference

C.3.1 Detailed Description

The Class CArgos.

Class CArgos is responsible for the camera input and creating CMyBitmaps from this input. Based on the DirectShow StillCap example, provided by Microsoft DirectX SDK 9.0

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Public Member Functions

- **CArgos** ()

  CArgos constructor.

- virtual ~**CArgos** ()

  CArgos destructor.

- void **CreateBitmap** (**CMyBitmap** *bmp)

  Creates a bitmap.

Protected Member Functions

- void **GetDefaultCapDevice** (IBaseFilter **ppCap)

  Gets the first video capture device.

- HRESULT **InitGraph** ()

  Initializes the graph and its filters.

- void **ClearGraph** ()

  Clears the graph and its filters.

- void **Error** (TCHAR *pText)

  Shows an error box.

Protected Attributes

- CComPtr< IGraphBuilder > **m_pGraph**
  Pointers for the capture graph and filter.
- CComPtr< ISampleGrabber > **m_pGrabber**

Friends

- class **CSampleGrabberCB**

C.3.2 Constructor & Destructor Documentation

C.3.2.1 CArgos::CArgos ()

CArgos constructor.

Calls **InitGraph( )**, that initializes the graph and its filters.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.3.2.2 CArgos::~CArgos () `[virtual]`

CArgos destructor.

Calls **ClearGraph( )**, that clears the graph and its filters.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.3.3 Member Function Documentation

C.3.3.1 void CArgos::CreateBitmap (CMyBitmap * *bmp*)

Creates a bitmap.

The public function to grab a bitmap and store it in the given parameter.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Parameters:
bmp pointer to output bitmap

C.3.3.2 void CArgos::GetDefaultCapDevice (IBaseFilter ** *ppCap*) `[protected]`

Gets the first video capture device.

Enumerates video capture devices and takes the first one.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Parameters:
ppCap pointer to pointer to capture device

C.3.3.3 HRESULT CArgos::InitGraph () `[protected]`

Initializes the graph and its filters.

Creates a graph with capture filter, samplegrabber filter and live rendering; Sets the specifications for the live rendering 'frame'; Sets **CSampleGrabberCB** m-CB as the callback.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Returns:
hr result of the initializing

C.3.3.4 void CArgos::ClearGraph () `[protected]`

Clears the graph and its filters.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.3.3.5 void CArgos::Error (TCHAR * *pText*) [protected]

Shows an error box.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Parameters:
pText pointer to error text

C.4 CHadesApp Class Reference

C.4.1 Detailed Description

The Class CHadesApp.

Class CHadesApp is the main application.

Author:
Harmen Dikkers and Mike Spaans

Version:
1.0, 20-05-2003

Public Member Functions

- **CHadesApp** ()
  CHadesApp constructor.
- virtual BOOL **InitInstance** ()
  Initializing.
- afx_msg void **OnAppAbout** ()
  App command to run the dialog.

C.4.2 Constructor & Destructor Documentation

C.4.2.1 CHadesApp::CHadesApp ()

CHadesApp constructor.

Author:
Harmen Dikkers and Mike Spaans

Version:
1.0, 20-05-2003

C.4.3 Member Function Documentation

C.4.3.1 BOOL CHadesApp::InitInstance () `[virtual]`

Initializing.

Sets frame coordinates.

Author:
Harmen Dikkers and Mike Spaans

Version:
1.0, 20-05-2003

C.4.3.2 void CHadesApp::OnAppAbout ()

App command to run the dialog.

Author:
Harmen Dikkers and Mike Spaans

Version:
1.0, 20-05-2003

C.5 CHadesOptionsDlg Class Reference

C.5.1 Detailed Description

The Class HadesOptionsDlg.

The MFC generated class files associated with the options dialog.

Author:
Mike Spaans

Version:
1.0, 01-06-2003

[NOHEADER]

- enum { **IDD** = IDD_DIALOG_OPTIONS }
  Misc attributes.

- int **m_algorithm**
- int **m_alarm**
- double **m_euclid**
- int **m_framerate**
- double **m_lambda**
- BOOL **m_wav**
- double **m_errorThreshold**

Public Types

Public Member Functions

- **CHadesOptionsDlg** (CWnd *pParent=NULL)
  CHadesOptionsDlg constructor.

Protected Member Functions

- virtual void **DoDataExchange** (CDataExchange *pDX)
  MFC GUI function.

C.5.2 Constructor & Destructor Documentation

C.5.2.1 CHadesOptionsDlg::CHadesOptionsDlg (CWnd * *pParent* = NULL)

CHadesOptionsDlg constructor.

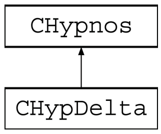
Attributes are assigned default values.

Author:
Mike Spaans

Version:
1.0, 23-05-2003

C.6 CHypDelta Class Reference

Inheritance diagram for CHypDelta::



C.6.1 Detailed Description

The Class HypDelta.

The class HypDetal determines somnolence in a bitmap based on difference in pixel intensity values.

Author:
Mike Spaans

Version:
1.0, 23-05-2003

Public Member Functions

- **CSomnValue Somnolence** (CRect, **CMyBitmap** ∗, **CMyBitmap** ∗, **CMyBitmap** ∗)

  Determines whether a bitmap is found to be in state of somnolence with regard to 'neutral' and 'sleepy' bitmaps.

- int **GetMax** ()

  Returns maximum range of values.

Private Member Functions

- **CSomnValue EuclidDelta** (CRect, **CMyBitmap** ∗, **CMyBitmap** ∗, **CMyBitmap** ∗)

  Determines somnolence based on vectors that are all bytes in selection Rect.

- void **CalcDeltaVectors** (CRect, **CMyBitmap** ∗, intVector &, intVector &)

  Calculates delta vectors.

- void **WriteVerData** (CRect, intVector &)

Misc. data dump functions. Mainly for debugging.

- void **WriteHorData** (CRect, intVector &)
  Save data to file.
- void **Dump** (intVector &)
- void **Dump2** (intVector &, intVector &, intVector &)

Private Attributes

- intVector **m_vectorHorNeutral**
  Data vectors.
- intVector **m_vectorVerNeutral**
- intVector **m_vectorHorSleepy**
- intVector **m_vectorVerSleepy**
- intVector **m_vectorHorBuffer**
- intVector **m_vectorVerBuffer**

C.6.2 Member Function Documentation

C.6.2.1 CSomnValue CHypDelta::Somnolence (CRect *rect*, CMyBitmap ∗ *neutral*, CMyBitmap ∗ *sleepy*, CMyBitmap ∗ *buffer*) `[virtual]`

Determines whether a bitmap is found to be in state of somnolence with regard to 'neutral' and 'sleepy' bitmaps.

Calls EuclidDelta.

Author:
Mike Spaans

Version:
1.0, 27-05-2003

Returns:
measure of somnolence. A value of '1' indicates full somnolence '0' full vigilance.

Parameters:
rect selection region
neutral pointer to a **CMyBitmap** containing neutral 'image'
sleepy pointer to a **CMyBitmap** containing sleepy 'image'
buffer pointer to a **CMyBitmap** containing buffer 'image'

Warning:
assert: bitmaps are NOT empty!

Implements **CHypnos** (p. 62).

C.6.2.2 CSomnValue CHypDelta::EuclidDelta (CRect *rect*, CMyBitmap ∗ *neutral*, CMyBitmap ∗ *sleepy*, CMyBitmap ∗ *buffer*) [private]

Determines somnolence based on vectors that are all bytes in selection Rect.

- Determine means
- Correct for means not having same dimension
- Correct with lambda factor
- Return smallest

Author:
Mike Spaans

Version:
1.0, 01-06-2003

Returns:
measure of somnolence. A value of '1' indicates full somnolence '0' full vigilance.

Parameters:
rect selection region
neutral pointer to a **CMyBitmap** containing neutral 'image'
sleepy pointer to a **CMyBitmap** containing sleepy 'image'
buffer pointer to a **CMyBitmap** containing buffer 'image'

Warning:
uses _horizontal_ vectors only.

C.6.2.3 void CHypDelta::CalcDeltaVectors (CRect *selectionRect*, CMyBitmap ∗ *bitmap*, intVector & *horVector*, intVector & *verVector*) [private]

Calculates delta vectors.

Author:
Mike Spaans

Version:
1.0, 01-06-2003

Parameters:
selectionRect gives bounding box
bitmap a pointer to an image buffer
horVector array of int's that will store horizontal means
verVector array of int's that will store vertical means

C.6.2.4 void CHypDelta::WriteVerData (CRect *rect*, intVector & *__vector*) `[private]`

Misc. data dump functions. Mainly for debugging.

Data is saved to file.

Author:
Mike Spaans

Version:
1.0, 29-05-2003

Warning:
File is overwritten. Unfinished.

Todo
Finish

C.6.2.5 void CHypDelta::WriteHorData (CRect *rect*, intVector & *__vector*) `[private]`

Save data to file.

Data is saved to file.

Author:
Mike Spaans

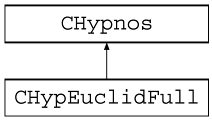
Version:
1.0, 29-05-2003

Warning:
File is overwritten. Unfinished.

Todo
Finish

C.7 CHypEuclidFull Class Reference

Inheritance diagram for CHypEuclidFull::



C.7.1 Detailed Description

The Class HypEuclidFull.

The class HypEuclidFull determines somnolence in a bitmap using all value whithin rectangle of interest.

Author:
Mike Spaans

Version:
1.0, 23-05-2003

Public Member Functions

- **CSomnValue Somnolence** (CRect, **CMyBitmap** ∗, **CMyBitmap** ∗, **CMyBitmap** ∗)

  Determines whether a bitmap is found to be in state of somnolence with regard to 'neutral' and 'sleepy' bitmaps.

- int **GetMax** ()

  Returns maximum range of values.

Private Member Functions

- **CSomnValue EuclidFull** (CRect, **CMyBitmap** ∗, **CMyBitmap** ∗, **CMyBitmap** ∗)

  Determines somnolence based on vectors that are all bytes in selectionRect.

- void **CalcVectors** (CRect, **CMyBitmap** ∗, intVector &)

  Calculate a data vector based on all values whitch selectionRect.

Private Attributes

- intVector **m_vectorNeutral**

  Data vectors.

- intVector **m_vectorSleepy**
- intVector **m_vectorBuffer**

C.7.2 Member Function Documentation

C.7.2.1 CSomnValue CHypEuclidFull::Somnolence (CRect *rect*, CMyBitmap ∗ *neutral*, CMyBitmap ∗ *sleepy*, CMyBitmap ∗ *buffer*) `[virtual]`

Determines whether a bitmap is found to be in state of somnolence with regard to 'neutral' and 'sleepy' bitmaps.

Calls EuclidFull to do calculation.

Author:
Mike Spaans

Version:
1.0, 27-05-2003

Returns:
measure of somnolence.

Parameters:
rect selection region

neutral pointer to a **CMyBitmap** containing neutral 'image'

sleepy pointer to a **CMyBitmap** containing sleepy 'image'

buffer pointer to a **CMyBitmap** containing buffer 'image'

Warning:
assert: bitmaps are NOT empty!

Implements **CHypnos** (p. 62).

C.7.2.2 CSomnValue CHypEuclidFull::EuclidFull (CRect *rect*, CMyBitmap ∗ *neutral*, CMyBitmap ∗ *sleepy*, CMyBitmap ∗ *buffer*) `[private]`

Determines somnolence based on vectors that are all bytes in selectionRect.

Author:
Mike Spaans

Version:
1.0, 27-05-2003

Returns:
measure of somnolence. A value of '1' indicates full somnolence '0' full vigilance.

Parameters:
rect selection region

neutral pointer to a **CMyBitmap** containing neutral 'image'

sleepy pointer to a **CMyBitmap** containing sleepy 'image'

buffer pointer to a **CMyBitmap** containing buffer 'image'

C.7.2.3 void CHypEuclidFull::CalcVectors (CRect *selectionRect*, CMyBitmap * *bitmap*, intVector & *__vector*) [private]

Calculate a data vector based on all values whitch selectionRect.

Author:
Mike Spaans

Version:
1.0, 16-05-2003

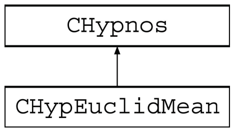
Parameters:
selectionRect bounding box

bitmap a pointer to an image buffer

__vector array of int's that will store values

C.8 CHypEuclidMean Class Reference

Inheritance diagram for CHypEuclidMean::



C.8.1 Detailed Description

The Class HypEuclidMean.

The class HypEuclidMean determines somnolence in a bitmap based on scanline means.

Author:
Mike Spaans

Version:
1.0, 23-05-2003

Public Member Functions

- **CSomnValue Somnolence** (CRect, **CMyBitmap** ∗, **CMyBitmap** ∗, **CMyBitmap** ∗)

  Determines whether a bitmap is found to be in state of somnolence with regard to 'neutral' and 'sleepy' bitmaps.

- int **GetMax** ()

  Returns maximum range of values.

- void **SetLamda** (double d)

  Function to set lambda parameter from options dialog.

- double **GetLambda** ()

  Function to get lambda parameter from options dialog.

Public Attributes

- double **m_lambda**

  parameter for somnolence determination

Private Member Functions

- **CSomnValue EuclidMeans** (CRect, **CMyBitmap** ∗, **CMyBitmap** ∗, **CMyBitmap** ∗)

  Determines somnolence based on calculation of scanline means.

- void **CalcMeans** (CRect, **CMyBitmap** ∗, intVector &, intVector &)

  Determine means of R,G and B values of all rows and cols within selection rectangle if it exists.

Private Attributes

- intVector **m_vectorVerticalNeutral**

  Data vectors.

- intVector **m_vectorHorizontalNeutral**
- intVector **m_vectorVerticalSleepy**
- intVector **m_vectorHorizontalSleepy**
- intVector **m_vectorVerticalBuffer**
- intVector **m_vectorHorizontalBuffer**

C.8.2 Member Function Documentation

C.8.2.1 CSomnValue CHypEuclidMean::Somnolence (CRect *rect*, CMyBitmap ∗ *neutral*, CMyBitmap ∗ *sleepy*, CMyBitmap ∗ *buffer*) `[virtual]`

Determines whether a bitmap is found to be in state of somnolence with regard to 'neutral' and 'sleepy' bitmaps.

Calls EuclidMeans to perform calculation.

Author:
Mike Spaans

Version:
1.0, 27-05-2003

Returns:
measure of somnolence. A value of '1' indicates full somnolence '0' full vigilance.

Parameters:

rect selection region

neutral pointer to a **CMyBitmap** containing neutral 'image'

sleepy pointer to a **CMyBitmap** containing sleepy 'image'

buffer pointer to a **CMyBitmap** containing buffer 'image'

Warning:
assert: bitmaps are NOT empty!

Implements **CHypnos** (p. 62).

C.8.2.2 CSomnValue CHypEuclidMean::EuclidMeans (CRect *rect*, CMyBitmap ∗ *neutral*, CMyBitmap ∗ *sleepy*, CMyBitmap ∗ *buffer*) `[private]`

Determines somnolence based on calculation of scanline means.

- Determine means
- Correct for means not having same dimension
- Correct with lambda factor
- Return smallest

The Lambda factor is a parameter that favors horizontal means over vertical means for being relatively less sensitive to movement, which usually occurs left<->right.

Author:
Mike Spaans

Version:
1.0, 27-05-2003

Returns:
measure of somnolence.

Parameters:
rect selection region

neutral pointer to a **CMyBitmap** containing neutral 'image'

sleepy pointer to a **CMyBitmap** containing sleepy 'image'

buffer pointer to a **CMyBitmap** containing buffer 'image'

Warning:
Asset: bitmaps are NOT empty!

C.8.2.3 void CHypEuclidMean::CalcMeans (CRect *selectionRect*, CMyBitmap ∗ *bitmap*, intVector & *horizontalMean*, intVector & *verticalMean*) [private]

Determine means of R,G and B values of all rows and cols within selection rectangle if it exists.

Value are stored in intVectors.

```
| R-mean | G-mean | B-mean | R-mean | G-mean | B-mean | ..
|-------- -------- --------+-------- -------- --------+---
|      First Column        |      Second Column       | ..
```

Author:

Mike Spaans

Version:

1.0, 12-05-2003

Parameters:

selectionRect gives bounding box

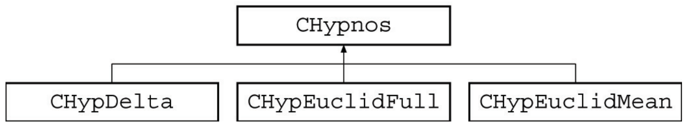
bitmap a pointer to an image buffer

horizontalMean array of int's that will store horizontal means

verticalMean array of int's that will store vertical means

C.9 CHypnos Class Reference

Inheritance diagram for CHypnos::



C.9.1 Detailed Description

The Class Hypnos.

Hypnos does all calculations with regard to somnolence detection, through its three child classes: **CHypEuclidMean**, **CHypEuclidFull** and **CHypDelta**. The function Somnolence is a pure virtual function, meaning it is only implemented in its child classes.

In this class we define the type intVector which is used for most calculations. It is based on the STL vector template.

Several statistical functions are implemented but not used by the HADES-1 system.

Author:
Mike Spaans

Version:
1.0, 23-05-2003

Public Member Functions

- **CHypnos** ()

  CHypnos constructor.

- virtual ∼**CHypnos** ()

  CHypnos destructor.

- virtual **CSomnValue Somnolence** (CRect, **CMyBitmap** ∗, **CMyBitmap** ∗, **CMyBitmap** ∗)=0

  Pure virtual member function.

- virtual int **GetMax** ()=0

  Pure virtual member function.

Public Attributes

- double **m_euclidDistanceParam**

  parameter for calculation of Euclidean distance

Protected Member Functions

- int **EuclidDistance** (intVector &, intVector &, double)

  Determines Euclidian distance between intVector u and intVector v.

- int **AbsSum** (intVector &)

  Sum of all absolute values in vector.

- void **Rotate** (**CMyBitmap** ∗, double)

  Rotate.

- double **RoundDouble** (double, int)

  Rounds a double to nearest whole.

- double **Mean** (intVector &)

  Determines mean of a given input vector.

- double **Variance** (intVector &)

  Determines the variance of a given input vector.

- double **Std** (intVector &)

  Determines the standard deviation of a given input vector.

C.9.2 Constructor & Destructor Documentation

C.9.2.1 CHypnos::CHypnos ()

CHypnos constructor.

The Euclidean distance parameter is assigned a default value.

Author:
Mike Spaans

Version:
1.0, 23-05-2003

C.9.2.2 CHypnos::~CHypnos () [virtual]

CHypnos destructor.

Author:
Mike Spaans

Version:
1.0, 23-05-2003

C.9.3 Member Function Documentation

C.9.3.1 int CHypnos::EuclidDistance (intVector & *u*, intVector & *v*, double *d*) [protected]

Determines Euclidian distance between intVector u and intVector v.

I.e. (sum [(u_i - v_i)^d])^1/d

Author:
Mike Spaans

Version:
1.0, 14-05-2003

Parameters:
u input vector 1

v input vector 2

d parameter that determines power

C.9.3.2 int CHypnos::AbsSum (intVector & *__intVector*) [protected]

Sum of all absolute values in vector.

Detailed description

Author:
Mike Spaans

Version:
1.0, 06-06-2003

Parameters:
__intVector input vector

C.9.3.3 void CHypnos::Rotate (CMyBitmap * *bitmap*, double *r*) `[protected]`

Rotate.

Do rotation

```
( x y ) (                 )
        (  cos(r)    sin(r) )
        (               )
        ( -sin(r)    cos(r) )
        (               )
```

Author:
Mike Spaans

Version:
1.0, 23-05-2003

Parameters:
bitmap input bitmap

r angle of rotation

Warning:
volatile!

C.9.3.4 double CHypnos::RoundDouble (double *doValue*, int *nPrecision*) `[protected]`

Rounds a double to nearest whole.

This function rounds a double to nearest whole at given precision.

Author:
Mike Spaans

Version:
1.0, 09-06-2003

Returns:
rounded double

Parameters:
doValue value to be rounded

nPrecision int specifiying at what precision to round

C.9.3.5 double CHypnos::Mean (intVector & *vec*) [protected]

Determines mean of a given input vector.

This function determines the mean of a vector of integers.

Author:
Mike Spaans

Version:
1.0, 09-06-2003

Returns:
double mean

Parameters:
vec input vector

C.9.3.6 double CHypnos::Variance (intVector & *vec*) [protected]

Determines the variance of a given input vector.

This function determines variance of a vector of integers.

Author:
Mike Spaans

Version:
1.0, 09-06-2003

Returns:
double variance

Parameters:
vec input vector

C.9.3.7 double CHypnos::Std (intVector & *vec*) [protected]

Determines the standard deviation of a given input vector.

This function determines the standard deviation of a vector of integers.

Author:
Mike Spaans

Version:
1.0, 09-06-2003

Returns:
double standard deviation

Parameters:
vec input vector

C.10 CLogView Class Reference

C.10.1 Detailed Description

The Class CLogView.

Class CLogView is responsible for the graph shown in the lower part of the interface

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Protected Member Functions

- **CLogView** ()
  CLogView constructor.
- virtual void **OnDraw** (CDC ∗pDC)
  Draws the graph.
- virtual BOOL **PreCreateWindow** (CREATESTRUCT &cs)
  Make sure the background color is grey.
- virtual void **PostNcDestroy** ()
  The pseudo destructor.
- virtual ∼**CLogView** ()
  CArgos *destructor.*
- afx_msg void **OnHadesLogView** (WPARAM wParam, LPARAM l-Param)
  Handle 'new data' message.
- afx_msg void **OnChangeParams** (WPARAM wParam, LPARAM l-Param)
  Changes internal parameters.

Protected Attributes

- int **m_buffersize**
  Storing the points for drawing.
- CPoint ∗ **points1**

- CPoint ∗ **points2**
- CPoint **lastPoint1**
- CPoint **lastPoint2**

- int **m_max**

  Attributes needed for drawing.

- int **m_speed**

  Attributes needed for drawing.

- int **m_counter**

  Attributes needed for drawing.

- double **m_errorThreshold**
- bool **fullbuffer**

C.10.2 Constructor & Destructor Documentation

C.10.2.1 CLogView::CLogView () `[protected]`

CLogView constructor.

Initializes attributes.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.10.2.2 CLogView::∼CLogView () `[protected, virtual]`

CArgos destructor.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.10.3 Member Function Documentation

C.10.3.1 void CLogView::OnDraw (CDC ∗ *pDC*) `[protected, virtual]`

Draws the graph.

Draws the threshold line, the points in the buffer and the 'refresh' line

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.10.3.2 BOOL CLogView::PreCreateWindow (CREATESTRUCT & *cs*) `[protected, virtual]`

Make sure the background color is grey.

Author:
Mike Spaans

Version:
1.0, 20-05-2003

C.10.3.3 void CLogView::PostNcDestroy () `[protected, virtual]`

The pseudo destructor.

Due to dynamic creation (forced by CWindowSplitter) the destructor is not called, causing memory leaks. Processing this final message provides the solution.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.10.3.4 void CLogView::OnHadesLogView (WPARAM *wParam*, LPARAM *lParam*) `[protected]`

Handle 'new data' message.

Updates the array of points with a new point calculated from the Hades data that has been passed by the message.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Parameters:
wParam pointer to **CSomnValue**

lParam NULL

Warning:
The **CSomnValue** object is deleted in this message handler!

C.10.3.5 void CLogView::OnChangeParams (WPARAM *wParam*, LPARAM *lParam*) `[protected]`

Changes internal parameters.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Parameters:
wParam pointer to double, the new m_errorThreshold value

lParam the new m_max value

Warning:
The double object is deleted in this message handler!

C.11 CMainFrame Class Reference

C.11.1 Detailed Description

The Class CMainFrame.

CMainFrame creates the toolbar and a splitterwindow

Author:
Harmen Dikkers and Mike Spaans

Version:
1.0, 20-05-2003

Public Member Functions

- **CMainFrame** ()
  CMainFrame constructor.
- virtual BOOL **PreCreateWindow** (CREATESTRUCT &cs)
  Override for some lay-out stuff.
- virtual BOOL **OnCmdMsg** (UINT nID, int nCode, void *pExtra, AFX_-CMDHANDLERINFO *pHandlerInfo)
  Override to pass command messages to children.
- virtual ~**CMainFrame** ()
  CMainFrame destructor.

Protected Member Functions

- virtual BOOL **OnCreateClient** (LPCREATESTRUCT lpcs, CCreate-Context *pContext)
  Override to enable splitted windows.
- virtual LRESULT **WindowProc** (UINT message, WPARAM wParam, LPARAM lParam)
  Override to pass user-defined messages to children.
- afx_msg int **OnCreate** (LPCREATESTRUCT lpCreateStruct)
  Override to enable custom toolbar.

- void **ReplaceBackgroundColor** (CBitmap &ioBM)
- void **MakeToolbarImageList** (UINT inBitmapID, CImageList &out-ImageList)

- void **AttachToolbarImages** (UINT inNormalImageID, UINT inDisabledImageID, UINT inHotImageID)

  24-Bit toolbar stuff.

Protected Attributes

- CStatusBar **m_wndStatusBar**

  Some window objects.

- CToolBar **m_wndToolBar**
- **CSplitterWndEx m_rowSplitter**
- **CSplitterWndEx m_colSplitter**

- CImageList **m_ToolbarImages**

  24-bit toolbar stuff.

- CImageList **m_ToolbarImagesDisabled**
- CImageList **m_ToolbarImagesHot**

C.11.2 Constructor & Destructor Documentation

C.11.2.1 CMainFrame::CMainFrame ()

CMainFrame constructor.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.11.2.2 CMainFrame::~CMainFrame () `[virtual]`

CMainFrame destructor.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.11.3 Member Function Documentation

C.11.3.1 BOOL CMainFrame::PreCreateWindow (CREATESTRUCT & *cs*) `[virtual]`

Override for some lay-out stuff.

Author:
Mike Spaans

Version:
1.0, 20-05-2003

Parameters:
&cs information on the current object

C.11.3.2 BOOL CMainFrame::OnCmdMsg (UINT *nID*, int *nCode*, void * *pExtra*, AFX_CMDHANDLERINFO * *pHandlerInfo*) `[virtual]`

Override to pass command messages to children.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Parameters:
nID command ID
nCode command notification code
****pExtra*** dependent on nCode
****pHandlerInfo*** should be NULL

C.11.3.3 BOOL CMainFrame::OnCreateClient (LPCREATESTRUCT *lpcs*, CCreateContext * *pContext*) `[protected, virtual]`

Override to enable splitted windows.

Author:
Mike Spaans

Version:
1.0, 20-05-2003

Parameters:
lpcs information on the current object
****pContext*** pointer to CCreateContext

C.11.3.4 LRESULT CMainFrame::WindowProc (UINT *message*, WPARAM *wParam*, LPARAM *lParam*) [protected, virtual]

Override to pass user-defined messages to children.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Parameters:
message message
wParam message parameter
lParam message parameter

C.11.3.5 void CMainFrame::AttachToolbarImages (UINT *inNormalImageID*, UINT *inDisabledImageID*, UINT *inHotImageID*) [protected]

24-Bit toolbar stuff.

Nothing very interesting

C.11.3.6 int CMainFrame::OnCreate (LPCREATESTRUCT *lpCreateStruct*) [protected]

Override to enable custom toolbar.

Author:
Mike Spaans

Version:
1.0, 20-05-2003

Parameters:
∗***lpCreateStruct*** information on the current object

C.11.4 Member Data Documentation

C.11.4.1 CStatusBar CMainFrame::m_wndStatusBar [protected]

Some window objects.

C.11.4.2 CImageList CMainFrame::m_ToolbarImages [protected]

24-bit toolbar stuff.

C.12 CMessWind Class Reference

C.12.1 Detailed Description

The Class CMessWind.

MFC GUI class. Sub class of CFormView. CEdit control is passed information to display.

Author:
Mike Spaans

Version:
1.0, 01-06-2003

[NOHEADER]

- enum { **IDD** = IDD_MESS_WND }
  CEdit controls.
- CEdit **m_TextArea**

Public Types

Protected Member Functions

- virtual void **DoDataExchange** (CDataExchange *pDX)
  MFC GUI function.
- afx_msg void **OnHadesStatus** (WPARAM wParam, LPARAM lParam)
  Generated message map function,.

C.13 CMyBitmap Class Reference

C.13.1 Detailed Description

The Class CMyBitmap.

A bitmap with timestamp and some useful extra attributes.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Public Member Functions

- **CMyBitmap** ()
  CMyBitmap constructor.
- void **Paint** (CDC *pDC, CRect rc)
  Paints the bitmap in the given rectangle.
- virtual ~**CMyBitmap** ()
  CMyBitmap destructor.

Public Attributes

- SYSTEMTIME **lTimeStamp**
  Timestamp in milliseconds.
- BYTE * **pBuffer**
  The bitmap and some attributes.
- BITMAPINFOHEADER **bih**
- long **lBufferSize**
- int **m_width**
- int **m_height**

C.13.2 Constructor & Destructor Documentation

C.13.2.1 CMyBitmap::CMyBitmap ()

CMyBitmap constructor.

Initializes attributes.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.13.2.2 CMyBitmap::∼CMyBitmap () `[virtual]`

CMyBitmap destructor.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.13.3 Member Function Documentation

C.13.3.1 void CMyBitmap::Paint (CDC ∗ *pDC*, CRect *rc*)

Paints the bitmap in the given rectangle.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Parameters:
∗***pDC*** pointer to Device Context
rc rectangle to draw

C.14 CPersephone Class Reference

C.14.1 Detailed Description

The Class CPersephone.

Class **CArgos** does all the interfacing

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Public Member Functions

- **CPersephone** ()
  CLogView *constructor.*
- virtual ~**CPersephone** ()
  CPersephone destructor.

Protected Member Functions

- void **CalculateRect** (CRect rc)
  Calculate helper rectangles to prevent excessive calculations in OnDraw() and **DetermineSomnolence()***.*
- void **DetermineSomnolence** ()
  Determines somnolence and sends appropriate messages.
- void **WriteBuffer** (stringVector vec)
  Writes the data vector to a file.
- virtual BOOL **PreCreateWindow** (CREATESTRUCT &cs)
  Some lay-out things.
- virtual void **PostNcDestroy** ()
  The pseudo destructor.
- afx_msg void **OnPaint** ()
  Draws the captured neutral and sleepy face, the monitored face and the rectangles.
- afx_msg void **OnLButtonUp** (UINT nFlags, CPoint point)
  Select rectangle of interest.

- afx_msg void **OnTimer** (UINT nIDEvent)
  Capture bitmap.
- afx_msg void **OnCapturingDone** (WPARAM wParam, LPARAM l-Param)
  Determine somnolence.
- afx_msg void **OnHadesCapneutral** ()
  Capture bitmap.
- afx_msg void **OnHadesCapsleepy** ()
  Capture bitmap.
- afx_msg void **OnHadesStart** ()
  Start monitoring.
- afx_msg void **OnHadesStop** ()
  Stop monitoring and write data buffer to file.
- afx_msg void **OnToolsOptions** ()
  Creates a dialog with options.

Protected Attributes

- **CArgos** * **m_argos**
  The **CArgos** *object for capturing new CMyBitmaps.*
- **CHypnos** * **m_hypnos**
  The **CHypnos** *object for determining the somnolence.*

- int **m_alarmDelay**
  Attributes to be set by the user and helpers.
- int **m_alarmDelayCounter**
  Attributes to be set by the user and helpers.
- int **m_framesPerSecond**
  Attributes to be set by the user and helpers.
- double **m_errorThreshold**
- BOOL **m_soundAlarm**
- stringVector **output**
- bool **m_firstClick**

- UINT **m_timer**
- CRect **m_selectionRect**

  Helpers to prevent the OnDraw() function for excessive unnecessary calculations.
- CRect **m_calculatedRect**
- CRect **m_smallRect**
- **CMyBitmap** ∗ **m_neutralFace**

  Pointers to the bitmaps.
- **CMyBitmap** ∗ **m_sleepyFace**
- **CMyBitmap** ∗ **m_monitoredFace**
- **CHypnos** ∗ **m_hypnos_1**

  Temporary, just for our report.
- **CHypnos** ∗ **m_hypnos_2**
- **CHypnos** ∗ **m_hypnos_3**
- CRect **rc_neutralFace**

  The frame coordinates.
- CRect **rc_sleepyFace**
- CRect **rc_monitoredFace**

C.14.2 Constructor & Destructor Documentation

C.14.2.1 CPersephone::CPersephone ()

CLogView constructor.

Initializes attributes.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.14.2.2 CPersephone::∼CPersephone () `[virtual]`

CPersephone destructor.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.14.3 Member Function Documentation

C.14.3.1 void CPersephone::CalculateRect (CRect *rc*) [protected]

Calculate helper rectangles to prevent excessive calculations in OnDraw() and **DetermineSomnolence()**.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Parameters:
rc source rectangle

C.14.3.2 void CPersephone::DetermineSomnolence () [protected]

Determines somnolence and sends appropriate messages.

Retrieves a **CSomnValue** from the **CHypnos** class, and sends a (global) message to the message window **CMessWind** and graph **CLogView**. Also sounds alarm when threshold has been exceeded.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.14.3.3 void CPersephone::WriteBuffer (stringVector *vec*) [protected]

Writes the data vector to a file.

Due to dynamic creation (forced by CWindowSplitter) the destructor is not called, causing memory leaks. Processing this final message provides the solution.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Parameters:
vec vector of CStrings

C.14.3.4 BOOL CPersephone::PreCreateWindow (CREATESTRUCT & *cs*) [protected, virtual]

Some lay-out things.

Author:
Mike Spaans

Version:
1.0, 20-05-2003

C.14.3.5 void CPersephone::PostNcDestroy () [protected, virtual]

The pseudo destructor.

Due to dynamic creation (forced by CWindowSplitter) the destructor is not called, causing memory leaks. Processing this final message provides the solution.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.14.3.6 void CPersephone::OnPaint () [protected]

Draws the captured neutral and sleepy face, the monitored face and the rectangles.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.14.3.7 void CPersephone::OnLButtonUp (UINT *nFlags*, CPoint *point*) [protected]

Select rectangle of interest.

Select rectangle and send message to message window.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Warning:
Creates pointer to CString, needs to be deleted in ONE message handler (done by **CMessWind::OnHadesStatus**)

C.14.3.8 void CPersephone::OnTimer (UINT *nIDEvent*) `[protected]`

Capture bitmap.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.14.3.9 void CPersephone::OnCapturingDone (WPARAM *wParam*, LPARAM *lParam*) `[protected]`

Determine somnolence.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.14.3.10 void CPersephone::OnHadesCapneutral () `[protected]`

Capture bitmap.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.14.3.11 void CPersephone::OnHadesCapsleepy () `[protected]`

Capture bitmap.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.14.3.12 void CPersephone::OnHadesStart () [protected]

Start monitoring.

Send parameters needed for drawing to **CLogView** and start timer.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Warning:
Creates pointer to double, needs to be deleted in ONE message handler (done by **CLogView::OnChangeParams**)

C.14.3.13 void CPersephone::OnHadesStop () [protected]

Stop monitoring and write data buffer to file.

Kills the timer and calls WriteBuffer.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.14.3.14 void CPersephone::OnToolsOptions () [protected]

Creates a dialog with options.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.15 CSampleGrabberCB Class Reference

C.15.1 Detailed Description

The Class CSampleGrabberCB.

Class CSampleGrabberCB is responsible for grabbing the samples. This object is a SEMI-COM object, and can only be created statically. We use this little semi-com object to handle the sample-grab-callback, since the callback must provide a COM interface. We could have had an interface where you provided a function-call callback, but that would take a lot of extra work and intransparancy. Based on the DirectShow StillCap example, provided by Microsoft DirectX SDK 9.0

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Public Member Functions

- **CSampleGrabberCB** ()

  CSampleGrabberCB constructor.

- **STDMETHODIMP_** (ULONG) AddRef()

  Fake out any COM ref counting.

- **STDMETHODIMP_** (ULONG) Release()

  Fake out any COM ref counting.

- STDMETHODIMP **QueryInterface** (REFIID riid, void **ppv)

  Fake out any COM Query Interfacing.

- STDMETHODIMP **SampleCB** (double SampleTime, IMediaSample *pSample)

  Interface not implemented.

- STDMETHODIMP **BufferCB** (double dblSampleTime, BYTE *pBuffer, long lBufferSize)

  The sample grabber is calling us back on its deliver thread.

Public Attributes

- long **lWidth**

These have to be set by **CArgos::InitGraph()** *in order to write out the bitmaps correctly.*

- long **lHeight**
- **CMyBitmap** ∗ **pDestination**

C.15.2 Constructor & Destructor Documentation

C.15.2.1 CSampleGrabberCB::CSampleGrabberCB () `[inline]`

CSampleGrabberCB constructor.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.15.3 Member Function Documentation

C.15.3.1 CSampleGrabberCB::STDMETHODIMP_ (ULONG) `[inline]`

Fake out any COM ref counting.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.15.3.2 CSampleGrabberCB::STDMETHODIMP_ (ULONG) `[inline]`

Fake out any COM ref counting.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.15.3.3 STDMETHODIMP CSampleGrabberCB::QueryInterface (REFIID *riid*, void ** *ppv*) [inline]

Fake out any COM Query Interfacing.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.15.3.4 STDMETHODIMP CSampleGrabberCB::SampleCB (double *SampleTime*, IMediaSample * *pSample*) [inline]

Interface not implemented.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

C.15.3.5 STDMETHODIMP CSampleGrabberCB::BufferCB (double *dblSampleTime*, BYTE * *pBuffer*, long *lBufferSize*) [inline]

The sample grabber is calling us back on its deliver thread.

When g_bOneShot is set to true, the current bitmap is written to pDestination and a finished-message is sent when finished.

Author:
Harmen Dikkers

Version:
1.0, 20-05-2003

Parameters:

dblSampleTime sampletime in seconds, useless since we need milliseconds

****pBuffer*** the bitmap

lBufferSize size of the bitmap

C.16 CSomnValue Class Reference

C.16.1 Detailed Description

The Class CSomnValue.

A simple class that contains data passed around the HADES system about the currenlty determined state of somnolence.

Author:
Mike Spaans

Version:
1.0, 01-06-2003

Public Member Functions

- **CSomnValue** ()

  CSomnValue constructor.

- **CSomnValue** (int, int)

  CSomnValue constructor.

- virtual ~**CSomnValue** ()

  CSomnValue destructor.

Public Attributes

- int **na**

  a parameter used for drawing. Distance between Neutral and Arbitrary bitmaps.

- int **sa**

  a parameter used for drawing. Distance between Sleepy and Arbitrary bitmaps.

- bool **somn**

  a parameter used that determines the state of somnolence.

- CString **msg**

  a formatted message containing information associated with the chosen algorithm.

C.16.2 Constructor & Destructor Documentation

C.16.2.1 CSomnValue::CSomnValue ()

CSomnValue constructor.

Attributes are assigned default values.

Author:
Mike Spaans

Version:
1.0, 01-06-2003

C.16.2.2 CSomnValue::CSomnValue (int *__na*, int *__sa*)

CSomnValue constructor.

Attributes are assigned default values.

Author:
Mike Spaans

Version:
1.0, 01-06-2003

Parameters:
- ***__na*** parameter used for drawing. Distance between Neutral and Arbitrary bitmaps.
- ***__sa*** parameter used for drawing. Distance between Sleepy and Arbitrary bitmaps.

C.16.2.3 CSomnValue::~CSomnValue () `[virtual]`

CSomnValue destructor.

Author:
Mike Spaans

Version:
1.0, 01-06-2003

C.17 CSplitterWndEx Class Reference

C.17.1 Detailed Description

The Class CSplitterWndEx.

A more advanced version of CSplitterWnd. Downloaded from codegure.com

Public Member Functions

- BOOL **PreCreateWindow** (CREATESTRUCT &cs)

  Misc functions.

- void **OnMouseMove** (UINT, CPoint pt)
- void **StartTracking** (int ht)
- void **OnDrawSplitter** (CDC *pDC, ESplitType nType, const CRect &rectArg)
- BOOL **OnSetCursor** (CWnd *pWnd, UINT nHitTest, UINT message)
- BOOL **OnMouseWheel** (UINT fFlags, short zDelta, CPoint point)
- **CSplitterWndEx** (BOOL Visible=TRUE)
- **DECLARE_DYNCREATE** (CSplitterWndEx)

Protected Attributes

- BOOL **m_Visible**

  seperators visible attribute.

Index