COBVIS-D: A Computer Vision System for Describing the Cephalo-Ocular Behavior of Drivers in a Driving Simulator

Steven Beauchemin[1], Parisa D. Z. Varcheie[2], Langis Gagnon[2], Denis Laurendeau[3], Martin Lavallière[4] Thierry Moszkowicz[3], Florent Prel[3], Normand Teasdale[4]

[1] Department of Computer Science, University of Western Ontario, London, ON, Canada, N6A 5B7
[2]Centre de recherche informatique de Montréal, 550, rue Sherbrooke Est, Bureau 100, Montréal, QC, Canada, H3A 1B9
[3]Department of Electrical and Computer Engineering, Laval University, Quebec, QC, Canada, G1V 0A6
[4]Division of kinesiology, Laval University, Quebec, QC, Canada, G1K 7P4
beau@csd.uwo.ca, {parisa.darvish, langis.gagnon}@crim.ca, {denis.laurendeau, thierry}@gel.ulaval.ca, florent.prel.1@ulaval.ca, normand.teasdale@kin.msp.ulaval.ca

Abstract. This paper describes current research combining computer vision, virtual reality and kinesiology for analyzing the cephalo-ocular behavior of drivers in realistic driving contexts. The different components of the system are described and results are provided for each one. The ultimate goal of the system is to achieve automatic analysis of drivers' behavior in order to design training programs tailored to their driving habits.

Keywords: computer vision, tracking, facial expression detection, driving simulator, virtual reality.

1 Introduction

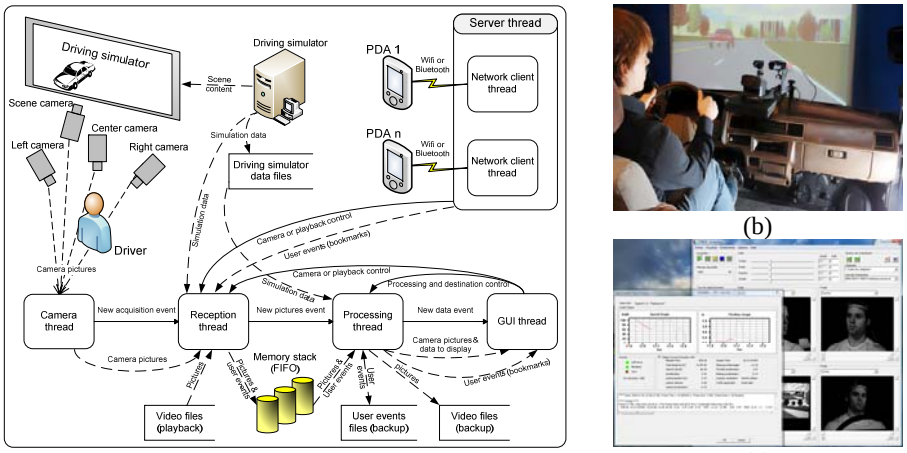
COBVIS-D aims to develop a system for describing the cephalo-ocular behavior and visual search patterns of drivers in realistic driving situations. The system uses computer vision to extract eye and head motion as well as facial expressions of the human test subjects as they drive in a simulator. The pose (position and orientation) of the head in 3D space is estimated as well. This information is used to assess driving performance for typical driving scenarios such as lane change, overtaking, merging and stopping at intersections. In the driving simulator, the driver watches the road on a virtual reality screen. The driver is observed by three synchronized and calibrated b&w cameras running at 30 images per second. The image sequences are processed in order to extract the information mentioned above.

In the following, a short description of the experimental set up is provided first. Secondly, two different approaches for head/eye detection are described. The first one is simple and runs in real-time while the second, more complex, is more robust but

still remains to be tuned to achieve real-time performance. The approach for estimating head pose is described briefly, followed by the approach that is being implemented for detecting facial expressions and, more precisely, facial action units related to the mouth of the driver. Results are provided for the different algorithms. The paper concludes by a description of the development of a platform that integrates the different components that can be used for analyzing the cephalo-ocular behavior of drivers.

2 Driving Simulator Setup and Computer Vision System

The quality of immersion is a key element in using a driving simulator. It can be achieved when the subject is in the dark and observes the display screen with sufficient intensity. However, if computer vision is to be used for estimating the cephalo-ocular behavior of the driver, a sufficient intensity level is required. In the driving simulator, the lighting environment can be controlled in order to meet both requirements. The driver's head and torso are illuminated by three LED spots in the near infrared and observed by 3 off-the-shelf b&w cameras with specially designed lenses that match the spots' wavelength (see Fig. 1 (a). Note that a fourth camera, the "scene" camera, is oriented towards the VR screen). Being in the near infrared part of the spectrum, the light sources do not spoil the sense of immersion and provide enough intensity for the vision algorithms to work properly. The cameras are positioned so as to minimize specular reflections for subjects wearing glasses.



(a)

(b)

(c)

Fig. 1 Simulator cockpit with the three cameras positioned in front of the driver (box in top-left) (a). Simulator screen as seen from the cockpit (b). User interface for the instructor (c).

3 Head and Eye Tracking

Two different approaches are being investigated for tracking the head and eyes of the driver. The first one is simple and works in real-time and has been fully integrated into the system. The second should be more robust but still needs to achieve real-time performance.

3.1 Simple Head and Eye Tracking Algorithm

The techniques developed by Leinhart and Maydt [1] extend upon a machine-learning approach that has originally been proposed by Viola and Jones [2]. The rapid object detector they propose consists of a cascade of boosted classifiers. Boosting is a machine learning meta-algorithm used for performing supervised learning. These boosted classifiers are trained on simple Haar-like, rectangular features chosen by a learning algorithm based on AdaBoost [3]. Viola and Jones have successfully applied their object detection method to faces [4], while Cristinacce and Cootes [5] have used the same method to detect facial features. Leinhart and Maydt extend the work of Viola and Jones by establishing a new set of rotated Haar-like features which can also be calculated very rapidly while reducing the false alarm rate of the detector. In the techniques proposed by Zhu and Ji [6], a trained AdaBoost face detector is employed to locate a face in a given scene. A trained AdaBoost eye detector is applied onto the resulting face region to find the eyes; a face mesh, representing the landmark points model, is resized and imposed onto the face region as a rough estimate. Refinement of the model by Zhu and Ji is accomplished by fast phase-based displacement estimation on the Gabor coefficient vectors associated with each facial feature. To cope with varying pose scenarios, Wang et al. [7] use asymmetric rectangular features, extended from the original symmetric rectangular features described by Viola and Jones to represent asymmetric gray-level features in profile facial images.

In our system, the detection of the head with the cameras is performed using a Cascade of boosted classifiers. The “Haar like features” provide accuracy and efficiency for head detection and, depending on the training of the Cascade classifiers, can limit the detection to the face and for small angles of the head relative to the cameras. This allows for the use of only the two cameras which are best positioned for capturing the image sequences and circumvents the problem of processing images in which one eye is not visible from a given point of view. The robustness of the head detection provided by the classifiers allows for the detection of a region of interest (ROI) in the area around and including the eyes.

Once this ROI is found, a blob-finding approach is used for detecting the exact location of the eyes. The blobs are retrieved after image saturation and application of a connected components algorithm. It is assumed that a pupil is circular or elliptical in shape and that the distance between two eyes is limited to a given range of values depending on the distance between the cameras and the subject. Once candidate blobs for the eyes are detected in the ROI, false blobs are filtered out (small blobs, etc.), and pairs of remaining blobs are then constructed and the distance and the angle between the blobs in each pair are computed. A set of rules (ratio between expected distance and the measured distance, angle, radius value etc.) is used to discard pairs of blobs

that do not correspond to the eyes, usually leaving only one pair of the best matched blobs. The stereo computation described in Section 5 is used to confirm this hypothesis on the validity of this 'winning pair'. This method for detecting the eyes can be repeated several times to find the best threshold value for the blob detection algorithm when no valid pair can be found at previous iterations. Because of the very good and stable image quality, the position of the eyes can be retrieved even for people wearing glasses.

3.2 Improved Head and Eye Tracking Algorithm

In order to make the system more robust, a more complex head/eye detection and tracking algorithm that extends the boosting approach is being investigated.

We designed a number of strategies to create a framework that extends boosting for learning in real-time. As a hypothesis, we first extended Kearn's assumptions on boosting with learners towards image feature trackers [8]. While boosting meta-algorithms have been successfully used for learners, we demonstrated, in a practical manner, that this concept may be extended to image-trackers, provided that they are used as weak learners.

Then computational strategies have been devised to satisfy the real-time constraints imposed by the requirement of tracking the eyes of drivers in a car simulator. Toward this end, sets of weak trackers are applied to the images in an interlaced manner. For instance, computationally expensive trackers would be used less frequently in an image sequence than those with a lesser cost. While this approach has the potential to reduce the accuracy of these trackers, we may still use the outputs of those that are applied more frequently in order to perform auto-correction operations to the entire set of trackers. The combined outputs of our weak trackers executing at different time intervals produce a robust real-time eye tracking system.

Additionally, the eye tracking system so derived represents only one instance of our framework, as identical principles and strategies may be used to track various types of image features, depending on the training stage that is used, if such a stage is required for the weak trackers that are chosen.

Our approach makes use of several techniques for processing input sequences of drivers following given scenarios in the simulator. Such techniques have been used successfully on their own [1], [9] and as part of a more extended framework [10]. Acceptable face and facial feature detections were produced at good success rates. Each technique used in our framework is treated as a module and these modules are classified into two major groups: detectors, and trackers. Detectors localize the facial regions automatically and lay a **base image** to be used for tracking by other modules. A base image is a visual capture of a particular facial region and can be used to perform a match against several other regions throughout the input sequence. Trackers use the base image set out by the detectors and employ matching algorithms to retrieve the correct position of the same facial region across following frames. Our framework uses a tracker based on Scale-Invariant Feature Transform (SIFT) [11] and a second tracker that uses a normalized correlation coefficient (NCC) method.

The rapid object detector (ROD) proposed by Viola and Jones [2] is a hybrid in that it can be classified as both a detector and tracker; it is employed to detect the face

and localize the eyes, while the SIFT and NCC trackers only deal with the eye regions. Often, a tracker in our framework may lose a target due to fast movement of the driver's head; a false positive base image may be registered at that time and the **off-target** tracker may eventually be tracking the wrong region as a consequence. As a detector, the ROD localizes the face and facial features automatically. As a tracker, it is used to track the eyes in between frames and to correct off-target trackers allowing for a second registration of a base image.

The framework uses a look-up table composed of blurred, scaled down Gaussian images. The Gaussian pyramid method [12] creates a stack of images that are successively smaller; the base image of the pyramid is defined as the original image and each image at any given level is composed of local averages of pixel neighborhoods of the preceding level of the pyramid. Detectors employed in our framework process the highest level of the pyramid first. In the case where an object of interest is not detected, the next level down is processed in a similar manner. The bottom-up approach used to detect faces and eyes in our framework reduces the processing time required by the detectors.

The various methods used in our framework produce good results, whether for detecting or tracking objects of interest in a given scene. The quality of the Haar-based classifier used by the rapid object detector is determined by its ability to correctly predict classifications for unseen (negative) examples. The classifier must be evaluated with a set of examples that is separate from the training set of samples used to build it. In many cases, some of the negative examples that should be used during the training phase of the classifier are missed and, in such a case, errors are introduced when the same negative example is seen by the classifier.

Detectors and trackers can be corrective in that errors introduced by one module in our framework are likely to be corrected by one or more of the modules throughout the input sequence. An off-target tracker can be corrected by a hybrid detector/tracker in order to allow for a second registration of a base image of the eye and, vice versa, where a false positive detection of the eye region by the hybrid detector/tracker can be rectified by one or more trackers with a true positive base image. Trackers, in terms of their corrective nature, in our framework are classified as follows:

1. Long-term correcting: Trackers that are long-term correcting display excellent tracking results and are usually associated with a high computational expense. They are not applied to every frame of an image sequence.
2. Short-term correcting: Short-term correcting trackers are less computationally expensive when compared to long-term correcting trackers, and are involved in correcting most of the errors introduced by the other modules. They are not applied to every image frame, yet more frequently than long-term correcting ones.
3. Real-time correcting: Real-time correcting trackers are associated with the lowest computational cost in processing the frames and are usually the least robust modules when compared to their peers. They are applied to most image frames from a sequence.

Tracking modules in our framework are classified into the above three categories based on their effectiveness as trackers, and according to their computational cost. In

our framework, the hybrid detector/tracker serves as a good short-term correcting tracker, while the SIFT and NCC trackers are used as long-term and real-time correcting trackers, respectively. As a general constraint toward the application of weak trackers, the framework ensures that only one such tracker is applied per frame.

The following describes the algorithm employed by our framework:

1. Acquire new frame: Given the selected view thought to contain a face, a new frame is acquired.
2. Build Gaussian pyramid: Once a new frame is acquired, a Gaussian pyramid is built.
3. Detect a face: The face detector is applied to the entire scene, in search of a profile face.
4. Track the eyes: If the eyes have not been detected by the system yet, the eye detector is run on the regions of interest. In the case where the eyes have already been registered by the trackers, the system employs the appropriate tracker.
5. Update detection windows: The detection window for the left and right eyes are updated according to the displacement vector produced by the tracker employed, and adjusted using a confidence parameter associated with the tracker.
6. View switching assessment: Once the frame has been fully processed, results from the detectors and some of the trackers are used to assess the current, primary view, according to preset thresholds. A view switch is performed if necessary.

The full integration of our trackers into a single, corrective framework shows an excellent performance. In terms of hit rate, our framework produces a correct-hit rate of 99.4%. However, with a high level of accuracy come severe computational costs. All the experiments were performed on a laptop running an Intel ® Pentium ® M processor at 2 GHz. The mean frame rate we obtained under such conditions was found to be *12.31* frames per second. The low frame rate comes back to the implementation of SIFT, as it is computationally costly. A slight change in configuration of the parameters for the framework could, potentially, produce higher frame rates while preserving excellent performance levels. We are currently working on this issue.

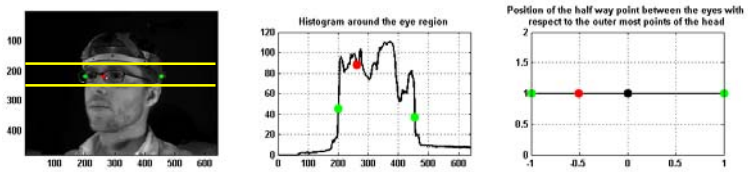
4 Head Pose Estimation

The detection of the location of the eyes with the algorithm of Section 2.1 allows the position (and orientation) of the driver's head to be estimated. This is achieved by computing the location of the half-way point between eyes (close to the nose) with respect to the outer most points of the head. In this case, this ratio is computed easily because the background is dark and uniform. The edges of the head in the image are found by thresholding the histogram of the eye area (Fig. 2 (a)). When the midpoint between the eyes is in the middle of the head the driver is looking straight at the camera. When this camera is the center camera (Fig. 1 (a)), then the driver is looking straight towards the screen. When the midpoint point is off-centre, the value of the

distance to the center allows the direction head to be estimated with respect to the camera. The knowledge of the position of the left and right cameras with respect to the center camera allows the horizontal direction of the head with respect to the center of the screen to be estimated roughly. Figure 3 shows results of this head orientation estimation approach for an elderly driver for different frames, cameras, and angles.

5 Nose Detection and 3D reconstruction

According to Gorodnichy the nose is the best facial feature to be tracked because it is clearly visible when the head is moving [13]. More precisely, the tip of the nose can often be approximated by a sphere. Once the position of the eyes and the rough position of the head are obtained, an accurate estimate of the position of the nose can be computed. After the application of an image threshold and a connected component algorithm, the brightest point in the connected regions is found and is considered as being the tip of the nose (see images in Fig. 3).



(a) (b) (c)

Fig. 2 On (a), (b), and (c), the green points are the outer most points of the head and the red point represents the half way point between the eyes. The mean histogram is computed between the two yellow lines. The helmet on the driver is a head tracker that can be used as a reference for our computation. It is not used by the algorithm (a). b) Histogram between the yellow lines, and position of the half way point between the eyes and the outer most points of the head (b). Normalized pixel value of the position of the points (c).

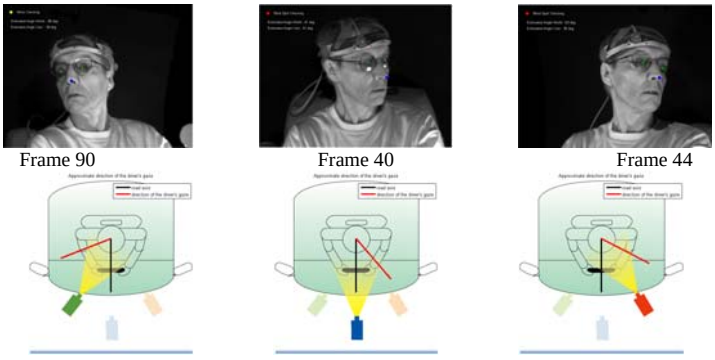


Fig. 3 Head orientation estimation for an elderly driver wearing glasses, for different frames, different cameras (the camera used is highlighted in yellow), and for different angles. The upper images are extracted from a video sequence, and the bottom images represent the head

pose estimation of the driver (red line) with respect to the driving simulator**.** The right-hand side shows the geometry for the computation of stereo.

The 3D reconstruction aims at computing the position and orientation of the head relative to a base reference frame (which happens to be the reference frame of one of the cameras) and is performed using the linear triangulation method [14]. The 3D coordinates of both eyes and the tip of the nose are computed with the approach described before and the orientation of the triangle they form in 3D space provides head orientation.

6 Detection of Facial Expressions and Mouth Actions Units

6.1 Background Material on Action Units

Facial expressions (FE) are important cognitive load markers in the context of car driving. FE can be defined globally or locally using the Facial Action Coding System (FACS) [15]. Literature review conducted during the project has identified the FACS-based approach as the more appropriate for in-car driving situations. FACS is based on muscular activity underlying momentary changes in facial aspects where each change is coded as a facial Action Unit (AU). A certified FACS coder has manually analyzed 74 video sequences of 12 subjects, acquired in our driving simulator, in order to identify the set of most important AUs depicted by car drivers. A total of 140 instances of AUs were identified, for which the most common were eye blinks, lowering brow, jaw drops, lips apart, lip corner puller and lips suck [16].

A real-time eye blink detector has been implemented and has been integrated in the car simulator and for which performances were reported elsewhere [17]. We are now focusing on mouth-related AUs. Recent progress with respect to real-time lip points tracking of simulator sequences are presented in the next sections. Lip points tracking is the basic step for the detection of all mouth-related AUs.

Among the facial features, mouth and lips are difficult to track since they are highly deformable. Our approach is based on geometric features (lip corners) and appearance features (texture around lips). Geometric- and appearance-based features are two general methods used in face recognition. It has been reported that geometric features are often outperformed by appearance ones [18][19]. However, recent studies show that in some cases geometric features can outperform appearance-based ones [19]. Yet it seems that using both features might be the best choice for certain FE.

6.2 Lip Tracking

Our lip tracker involves two steps: target modeling and tracking. Modeling characterizes lip points for the tracker while tracking is carried out by extracting and scoring lip-corner features based on a fuzzy rule. The fuzzy modeling is used because of the uncertainty and nonlinearity characteristics of the lip changes.

Target modeling. Left and right lip corners (targets) are manually localized in the first frame. A range filter is applied around them. Histogram equalization is used to enhance contrast and image is converted to binary with a high threshold to retain only

the strong edges. Five geometric and appearance features are used to characterize the target over a 30x30 window: gray level images, gray level histogram, edge images, normalized x and y projection histogram pattern of the obtained edge map.

Tracking. In each frame, candidate corners around the previous lip corners positions are extracted. For all of them, only those aligned along the same row, with a maximum of 10 pixel difference along the vertical direction of the left and right corners, are selected. For the candidate corners, the above features are calculated. Then, the longest segment of the lip edge is extracted from the edge map image. To localize the true corners, features of the candidate corners are compared with the initial target model features and a score is given based on a fuzzy rule. The true corners are the ones with the highest score value on each mouth sides. Seven fuzzy inputs (e.g. distances) with normalized membership functions (value between 0 and 1) are used:

1. Euclidean distance of the candidate corners from the previous position of the lip corners
2. Euclidean distance of the gray-level histogram around each candidate corner with the gray-level histogram around lip corners in the previous frame
3. and 4. Euclidean distances of the normalized x and y projection histogram around each candidate corners with the normalized x and y projection histograms of the lip corners in the previous frame
5. Similarity of the cropped gray level images around the candidate corners with the same size cropped gray level images around the lip corners from the previous frame
6. Similarity of the cropped edge images around the candidate corners with the same size cropped edge images around the lip corners from the previous frame
7. Euclidean distance of the candidate corners with the detected left and right corners, which are obtained from the longest segment of the lip image.

The best corners are those having the maximum score which is calculated by multiplying the seven fuzzy membership outputs. After detecting the true corners, the target model is updated according to the last detection results. Fig. 5 shows an example of the best lip corner candidates among all of the lip corners.

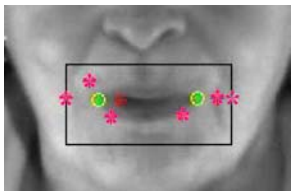


Fig. 4 True lip corners (circles) among all the lip corner candidates (stars) inside the search window (rectangle).

The performance of the algorithm for different drivers with various mouth changes has been evaluated. The performance is calculated on a dataset of 18 video sequences, one sequence per subject, with an average of 20,000 frames/sequences. Only the lip and mouth movement sub-sequences are kept (e.g. smile, talking, lip stretch, lip press, etc.). The number of frames with correct corner detection and the total number of

frames for each video are given in Table 1. Columns indicate if the detection has been done within circles of radius 5%, 10%, 20% and 50% of the current lip length around the ground truth corners.

Table 1. Left and right lip corners detection versus frame number for 18 subjects

Video	*5% Left*	*10% Left*	*20% Left*	*50% Left*	*5% Right*	*10% Right*	*20% Right*	*50% Right*	*Total frames*
P1 (talk)	85	85	85	85	85	85	85	85	85
P2 (smile)	51	51	51	51	51	51	51	51	51
P3 (lip press)	19	34	38	58	40	42	44	62	62
P4 (lip suck)	3	6	6	16	12	13	16	16	16
P5 (talk)	40	48	57	57	30	40	55	57	57
P6 (lip pucker)	40	52	63	68	66	66	68	68	68
P7 (talk)	58	58	58	62	62	62	62	62	62
P8 (lip funneler)	245	250	250	250	250	250	250	250	250
P9 (smile)	44	50	50	50	33	44	50	50	50
P10 (talk)	33	38	39	44	37	41	46	46	46
P11 (lip tight.)	48	48	50	50	47	47	50	50	50
P12 (smile)	236	236	236	236	236	236	236	236	236
P13 (smile)	26	32	36	36	34	35	36	36	36
P14 (talk)	61	66	69	69	69	69	69	69	69
P15 (talk)	23	26	26	26	26	26	26	26	26
P16 (lip suck)	136	142	143	143	138	140	143	143	143
P17 (smile)	7	9	9	11	8	9	9	11	11
P18 (talk/smile)	89	91	91	91	91	91	91	91	91
Total	1244	1322	1357	1403	1315	1347	1387	1409	1409

7 Integrated Environment

The processing modules described in previous sections are integrated in a common software platform in order to allow instructors to analyze the cephalo-ocular behavior of drivers. The overall structure of the software platform is shown in Fig. 1 (a). The system receives data from synchronized cameras, from video files (on playback mode), user events (bookmarks on typical head movements or driving errors) and driving simulator data. In return, it can provide real time plots and data for the analysts, instructor or driver**.**

The building blocks of the software platform operate as follows:

- A driving simulator (STISIM Drive from Systems Technology Inc.) which runs scenarios and sends simulation data to files. Scenarios and driving environments are fully.
- The camera thread downloads the current frames from the synchronized video cameras. Once all the pictures are fully downloaded, it sends an acquisition event to the reception thread.
- The reception thread can receive camera or playback control commands and user events from the GUI or a network client (PDA). User events are pushed in a temporary buffer. They are sent from the user to mark a zone in the video stream for future playback. This could, for instance, allow a driving instructor to highlight

a zone where the driver made a mistake. Once the thread receives an acquisition event from the camera thread or once new pictures are available from the video files (playback mode) it pushes new pictures, data and user events in the FIFO memory stack for the processing thread. When the CPU usage is close to 100% for the processing thread, the number of buffered pictures increases when computing process is to slow (when the head is moving for instance) and decreases when it's fast enough because the reception thread priority is higher than the processing thread priority. When the processing thread receives the new pictures event it pulls out the last pictures, data and user events from its current frame; reads the corresponding data from the simulator data file and user event files (containing the previously stored data in playback mode); generate user readable data and graph for the GUI and generate backup video files and new user events files if requested.

Each plug-in can be activated or disabled in the processing thread, but some plug-ins can depend from others and can't be activated alone. If a plug-in is optional, it is processed only when another requests it. The first mandatory plug-in to be called is "3D Head Pose" described in Section 4 and 5. When necessary, it calls the optional "Head Detection" plug-in which role is to detect and extract the driver's head position in the video camera pictures. The "Head Detection" plug-in is not frequently called because the driver's head usually moves only during blind spot checking. When activated, the "Face Analysis" plug-in described in Section 6 is mandatory and generates detailed information on the eye blinking and facial expressions from the center camera pictures.

An independent "3D camera calibration" plug-in has to be used only once or when the zoom, focal or position of an external camera are modified. This plug-in generate a XML data file used by the "3D Head Pose" plug-in.

Finally the last plug-in is the "Data Compiler" which assembles information from the previous plug-ins, the simulator and user events to display high-level user friendly data summary and plots to the GUI for scientists, driving instructor and driver (see GUI in Fig. 1(c).

8 Conclusion and Future Work

This paper has presented our on-going work on the development of an integrated system for assessing the cephalo-ocular behavior of drivers. The system uses a driving simulator for generating driving scenarios while computer vision is used for estimating head pose, eye movement and facial expressions which are believed to contain important information on the driver's condition while executing common maneuvers. Future work will concentrate on optimizing the vision algorithms and on integrating the system so it can be used for conducting training experiments on drivers with unsafe behavior.

Acknowledgments. The authors wish to acknowledge the financial support of the Auto21 Network of Centers of Excellence and of the Société d'assurance automobile du Québec (SAAQ).

References

1. Leinhart, R., Maydt, J.: An extended set of Haar-like features for rapid object detection. Proc. Int. Conf. Image Processing, Vol. 1, pp. 900-903, Rochester, NY, (2002)
2. Viola, P., Jones, M.J.: Rapid Object Detection Using A Boosted Cascade Of Simple Features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 1, pages I-511-I-518 (2001)
3. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Proc. European Conference on Computational Learning Theory, pp. 23-37, (1995)
4. Viola, P., Jones, M.: Robust Real-Time Face Detection. Int. J. Comput. Vision, Vol. 57, pp. 37-154, (2004)
5. Cristinacce, D., Cootes, T.: Facial Feature Detection using AdaBoost with Shape Constraints. Proc. British Machine Vision Conference, pp. 231-240, (2003)
6. Zhu, Z. Ji, Q.: Robust Pose Invariant Facial Feature Detection and Tracking in Real-Time. Proc. Int. Conf. Pattern Recognition, pp. 1092-1095, (2006)
7. Wang, Y., Liu, Y., Tao, L., Xu, G.: Real-Time Multi-View Face Detection and Pose Estimation in Video Stream. Conf. Pattern Recognition, Vol. 4, pp. 354-357, (2006)
8. Kearns, M.: Thoughts on Hypothesis Boosting. Unpublished manuscript, (1988)
9. Lowe, D.G.: Object Recognition from Local Scale-Invariant Features. Proc. Int. Conf. Computer Vision, Vol. 2, p. 1150, (1999)
10. Kanaujia, A., Huang, Y., Metaxas, D.: Emblem detections by tracking facial features. Proc. IEEE Computer Vision and Pattern Recognition, p. 108, (2006)
11. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. Int. J. Comput. Vision, Vol. 60, No. 2, pp. 91-110, (2004)
12. Burt, P.J., Adelson, E.H.: The Laplacian Pyramid as a compact image code. IEEE Transactions on Communications, Vol. 4, pp. 532-540, (1983)
13. Gorodnichy, D.: On Importance of Nose For Face Tracking. Proc. IEEE Intern. Conf. on Automatic Face and Gesture Recognition (FG'2002), Washington, D.C., pp. 181-186, (2002)
14. Ntawiniga, F.: Head motion tracking in 3D space for drivers. Master's Degree Thesis, Electrical Engineering, Laval University, Quebec, Canada, 98 p., (2008)
15. Ekman, P., Friesen, W.V.: Facial Action Coding System. Palo Alto, CA: Consulting Psychologists Press, Inc., (1978)
16. Lalonde, M., Byrns, D., Chapdelaine, C., Gagnon, D. : Progress Report of CRIM's Activities for the COBVIS-D Project for the Period April 2007 to March 2008, Montréal, [CRIM-08/04-02], (2008)
17. Lalonde, M., Byrns, D., Gagnon, L., Teasdale, N., Laurendeau, D.: Real-Time Eye Blink Detection with GPU-Based SIFT Tracking. Proc. 4th Canadian Conference on Computer and Robot Vision (CRV07), (2007)
18. http://www.face-rec.org/
19. Delac, K., Grgic, M.: Face Recognition. I-Tech Education and Publishing, Austria, (2007)