PERGAMON

Transportation Research Part C 9 (2001) 399–413

TRANSPORTATION RESEARCH PART C

www.elsevier.com/locate/trc

# Driver fatigue: a vision-based approach to automatic diagnosis

Martin Eriksson, Nikolaos P. Papanikolopoulos *

*Artificial Intelligence, Robotics and Vision Laboratory, Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, USA*

Received 15 December 1997; accepted 21 June 2000

**Abstract**

In this paper, we describe a system that locates and tracks the eyes of a driver. The purpose of such a system is to perform detection of driver fatigue. By mounting a small camera inside the car, we can monitor the face of the driver and look for eye movements which indicate that the driver is no longer in condition to drive. In such a case, a warning signal should be issued. This paper describes how to find and track the eyes. We also describe a method that can determine if the eyes are open or closed. The primary criterion for this system is that it must be highly non-intrusive. The system must also operate regardless of the texture and the color of the face. It must also be able to handle changing conditions such as changes in light, shadows, reflections, etc. Initial experimental results are very promising even when the driver moves his/her head in a way such that the camera does not have a frontal view of the driver's face. 

*Keywords:* Driver fatigue; Visual tracking; Blink rate; Microsleeps; Detection of fatigue

## 1. Introduction

Driver fatigue is an important factor in a large number of accidents. Reducing the number of fatigue-related accidents would save the society a significant amount financially, in addition to reducing personal suffering. We believe that by monitoring the eyes, the symptoms of driver fatigue can be detected early enough to avoid many of these accidents. In recent years, the problem of driver fatigue has been closely investigated. An increased knowledge about the correlation between sleep deprivation and lack of alertness has caused the professional trucking

* Corresponding author. Tel.: +1-612-625-4002; fax: +1-612-625-0572.
*E-mail addresses:* eriksson@cs.umn.edu (M. Eriksson), npapas@cs.umn.edu (N.P. Papanikolopoulos).

0968-090X/01/$ - see front matter 
PII: S0968-090X(00)00045-0

industry to adjust the scheduling of shifts for the drivers. Also, the circadian rhythms of the body have been studied in order to improve sleep management and napping strategies amongst professional drivers. Countermeasures such as drinking caffeine, playing loud music, driving with company, etc. have also been studied. Unfortunately, there is no "silver bullet" to the driver-fatigue problem. The fundamental solution would be to make people aware of the risks of driving sleep deprived. It is important to understand that a system such as the one we propose in this paper can never serve as a substitute for sleep. The intention is to avoid a portion of the accidents that would occur when the driver persists to drive when fatigued. In this work, we have focused on driver fatigue, even though similar symptoms occur when the driver is intoxicated. There are many indicators of oncoming fatigue, some of which are possible to detect only by using a camera.

Two of the most urgent symptoms that we consider feasible to detect accurately are:

(a) Microsleeps – These are short periods (2–3 s) during which the driver rapidly loses consciousness.

(b) The forward "bouncing" movement of the head.

In this paper, we primarily deal with the detection of microsleeps. A driver who is having microsleeps is a potential danger to himself/herself as well as to others. By giving this driver a warning signal, one could potentially save lives. The input to the system is a continuous sequence of images fed from the camera. From this sequence, the system can analyze the eyes in each image, as well as comparing the eyes between images. The analysis of face images is a popular research area with applications such as face recognition, virtual tools and handicap aids (Tello, 1983; White et al., 1993), human identification and database retrieval (Cox et al., 1995). There are also many real-time systems being developed in order to track face features (Tock and Craw, 1995; Xie et al., 1995; Stiefelhagen et al., 1996). In addition to the detection of driver fatigue, these systems can also be used for teleconferencing, virtual pointing tools (virtual mouse) or for medical research. These kinds of real-time systems generally consist of three components:

(a) Localization of the eyes (in the first frame).

(b) Tracking the eyes in the subsequent frames.

(c) Detection of failure in tracking.

Localization of the eyes involves looking at the entire image of the face and determining the eye envelopes (the areas around the eyes). During tracking in subsequent frames, the search space is reduced to the area corresponding to the eye-envelopes in the current frame. This tracking can be done at relatively low computational effort since the search space is significantly reduced. Also, since there are less distracting features in the reduced area, less advanced methods will suffice. In order to detect failure in the tracking, general constraints such as the distance between the eyes and the horizontal alignment of the two eyes can be used. The eyes should be relocated after a certain period (i.e., 100 frames), even though no failure has been detected, to periodically make sure that the correct feature is being tracked. This means that despite the fact that the main component of the system is tracking, a relatively efficient method has to be used for the localization phase.

This paper is organized as follows: In Section 2, we describe some of the previous works in this area. Section 3 includes experimental setup and a description of how the system operates. In Section 4, we describe how we perform the detection of fatigue. Results and future work are discussed in Section 5. Finally, Section 6 contains the conclusions.

## 2. Previous work

A large amount or research has been conducted in order to find the correlation between sleep deprivation and accidents in traffic caused by low alertness (Dinges, 1995). Proper scheduling of shifts for professional drivers as well as education in sleep management and napping strategies are approaches taken in order to reduce the number of accidents due to fatigue. In addition, the impact of the circadian pacemaker on the sleeping activity has been closely studied (Dijk and Czeisler, 1994). The understanding of sleeping patterns and the importance of being alert while performing monotone yet critical tasks (such as driving a car) is very high. Yet, there is no simple solution to the driver-fatigue problem. However, advances in technology have produced new methods for real-time diagnosis of driver fatigue (e.g., vision-based methods, monitoring of the drivers' steering patterns, etc.). The systems we have studied are vision-based systems that use one camera in order to monitor the face of the driver. The major contribution of our approach is that we have created a combination of methods for the detection of the face, the detection, tracking and monitoring of the eyes and finally, the detection of fatigue. All these methods were designed in order to address the real-time constraints of the problem along with the environmental and task-related challenges that the fatigue detection presents.

Localizing the eyes in an image is obviously a trivial task for a human observer, while a computer requires well-defined features and rules to perform the same task. Many methods have been proposed for localizing facial features in images (Craw et al., 1987; Huang and Chen, 1992; Chow and Li, 1993; Stringa, 1993; Li and Roeder, 1995; Roeder and Li, 1996; Segawa et al., 1996; Wu et al., 1996). These methods can roughly be divided into two categories: *Template-based matching* and *Feature-based matching*. These techniques are compared by Brunelli and Poggio (1993). Template-based matching uses knowledge about the shape of an object and searches the image for that particular object. One popular template-matching technique for extraction of face features is to use *deformable templates* (Xie et al., 1994; De Silva et al., 1995). Deformable templates are similar to the *active snakes* introduced by Kass et al. (1988) in the sense that they both use energy minimization to compute image forces. A deformable template is constructed using some knowledge about the shape to be extracted. The template can then deform and move across the image in order to minimize some cost function. In feature-based matching, the system uses knowledge about some geometrical constraints. For example, a face has two eyes, one mouth and one nose in specific relative locations. In this paper, we describe a combination of these techniques in a top-down manner. We use features in order to find the approximate location, and then we apply a template inside this region in order to determine the exact location.

One interesting method for face recognition was developed by Stringa (1993). He used the observation that the eyes are regions of rapidly changing intensity. By projecting edges onto the vertical axis, he determined the vertical location of the eyes. We use a similar approach on a reduced version of the image. Another approach described by Stiefelhagen et al. (1996) uses connected regions in order to extract the dark disks corresponding to the pupils. Rather than looking for the pupils, we used the fact that the entire eye regions are darker than their surroundings, again allowing us to use the reduced image in order to extract these rough regions at a reduced computational cost. This approach requires a threshold in order to perform the region

extraction. In order to improve performance, adaptive thresholding is used. Another real-time system intended for detection of driver fatigue (Tock and Craw, 1995) uses a set of stored eye templates and thresholds in order to localize the region of interest. Since we are concerned about robustness regarding changing conditions, we use adaptive thresholding. The two systems mentioned above use color information in order to extract the head from the background. Stiefelhagen et al. used a statistical color model in order to determine which pixels are likely to correspond to face colors. As described in Tock and Craw (1995), the observation that the head differs significantly from the background (head lining and seats, etc.) in HSV-space (Hue-Saturation-Value) will allow the system to extract the skin color. In order to avoid dependence on a fairly colorless background, we decided to use again the reduced image and localize the symmetry axis (Yoo and Oh, 1996). Since the driver will be looking almost straight ahead, there will be a well-defined vertical-symmetry line between the eyes.

Once the rough eye regions are localized, we use a template in order to find the exact location of the iris. Many different templates have been described for finding the shape of an eye. Xie et al. (1994) developed a deformable template consisting of 10 cost equations based on image intensity, image gradient and internal forces of the template. This template worked well on still images of fairly high resolution. Since we are greatly concerned about computational speed, we decided to use only the two cost equations dealing with image intensity. For our purposes, this will work well since we narrow down the search space significantly before applying the template. Also, we do not use any cost concerning internal forces. Once the eyes are found, the search space in the subsequent frames is limited to the area surrounding the eye regions found. In the system by Stiefelhagen et al., the darkest pixel (which is likely to be a pixel inside the pupil) is used for tracking, allowing high computational speed. Another approach (De Silva et al., 1995) is to perform an edge detection on the region of interest and then track the region with a high concentration of edges. This method will run significantly slower.

## 3. The system

We would like to emphasize that in order to make a system such as this useful in practice, it is of major importance to make it highly non-intrusive. The system must operate on its own, without the driver having to interact with it. This means that the system must be able to automatically find the eyes, find the parameters required to perform the tracking and be able to tell when the tracking has failed. When the system starts, frames are continuously fed from the camera to the computer. As mentioned in the introduction, we use the initial frame in order to localize the eye positions. Once the eyes are localized, we start the tracking process by using information in previous frames in order to achieve localization in subsequent frames. During tracking, error detection is performed in order to recover from possible tracking failure. When a tracking failure is detected, the eyes are relocalized. During tracking, we also perform the detection of fatigue. At this point, we limit this to detection of microsleeps. At each frame, when the eyes are localized, the system determines whether the eyes are open or not. Thus, we are able to tell when the eyes have been closed for too long. In practice, we count the number of consecutive frames during which the eyes are closed. If this number gets too large, we issue a warning signal. A flowchart diagram of the system is shown in Fig. 1.

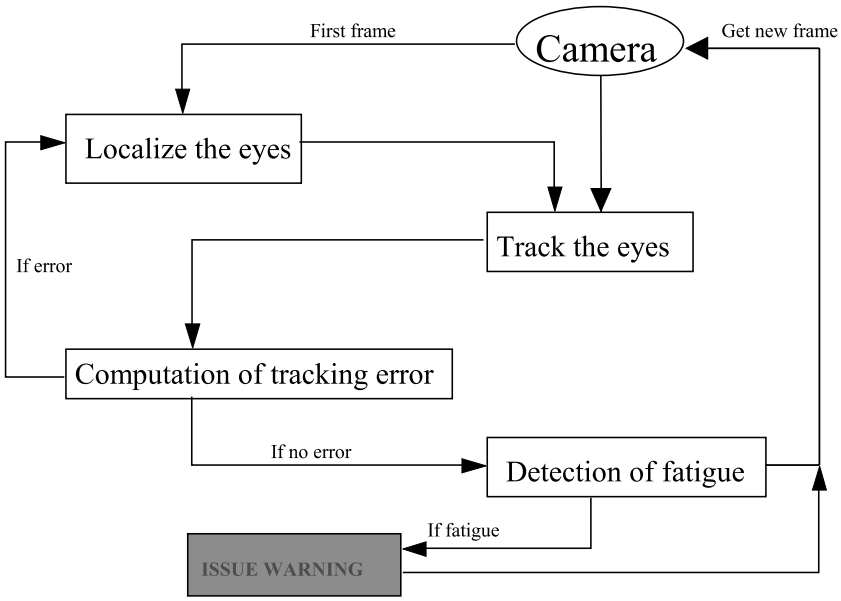


Fig. 1. Flowchart diagram of the processing of each frame.

### 3.1. Experimental setup

The final system will consist of a camera pointing at the driver. The camera is to be mounted on the dashboard inside the vehicle. For the system we are developing, the camera is stationary and will not adjust its position or zoom during operation. This means that in order to allow for normal variations in head positioning, the camera cannot be zoomed in too closely. The reason is that the eyes are then likely to end up outside the field of view of the camera. In the future, we are planning to use a camera that will be able to dynamically adjust zoom and direction in order to focus closely on the eye regions, and thus give a better resolution inside the region of interest. However, for this system, there is a trade-off between field of view and resolution. This means we have to optimize, so that the field of view is "big enough" to contain the eyes during normal driving posture and "small enough" to allow for detection of fine eye movements. For experimentation, we are using a JVC color video camera, sending the frames to a Silicon Graphics Indigo. The grabbed frames are represented in RGB space with 8-bit pixels (256 colors). We do not use any specialized hardware for image processing.

### 3.2. Localization of the eyes

We localize the eyes in a top-down manner, reducing the search space at each step. The steps involved are:

1. Localization of the face.
2. Computation of the vertical location of the eyes.
3. Computation of the exact location of the eyes.

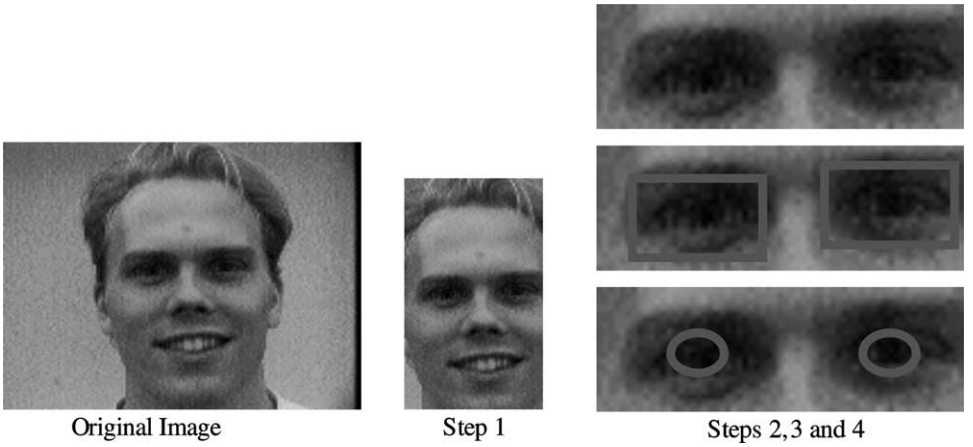


Fig. 2. The steps involved in the localization of the eyes.

4. Estimation of the position of the iris.

Each one of these steps is shown in Fig. 2.

### 3.2.1. Localization of the face

Since the face of a driver is symmetric, we use a symmetry-based approach, similar to Yoo and Oh (1996). We found that in order for this method to work, it is enough to use a smaller, gray-scale version of the image. In order to construct the smaller version, we used subsampling by averaging the pixel values in the subsampling window. A symmetry value is then computed for every pixel-column in the reduced image. If the image is represented as $I(x,y)$, then the symmetry value for a pixel-column is given by

$$S(x) = \sum_{w=1}^{k} \sum_{y=1}^{ysize} [abs(I(x,y-w) - I(x,y+w))].$$

$S(x)$ is computed for $x \in [k, xsize - k]$ where $k$ is the maximum distance from the pixel-column that symmetry is measured, and $xsize$ is the width of the image. The $x$ corresponding to the lowest value of $S(x)$ is the center of the face. The result from this process is shown in Fig. 3. The search space is now limited to the area around this line, which reduces the probability of having distracting features in the background.

### 3.2.2. Computation of the vertical location of the eyes

As suggested in Stringa (1993), we use the observation that eye regions correspond to regions of high spatial frequency. Again, we are working with the reduced image. We create the gradient map, $G(x,y)$, by applying an edge-detection algorithm on the reduced image. Any edge-detection method could be used. We choose to use a very simple and fast method called pixel differentiation, that assigns $G(x,y) = I(x,y) - I(x-1,y)$. We selected this method since it does not involve any convolution. $G(x,y)$ will now reveal areas of high spatial frequency. By projecting $G(x,y)$ onto its vertical axis, we get a histogram $H(y)$

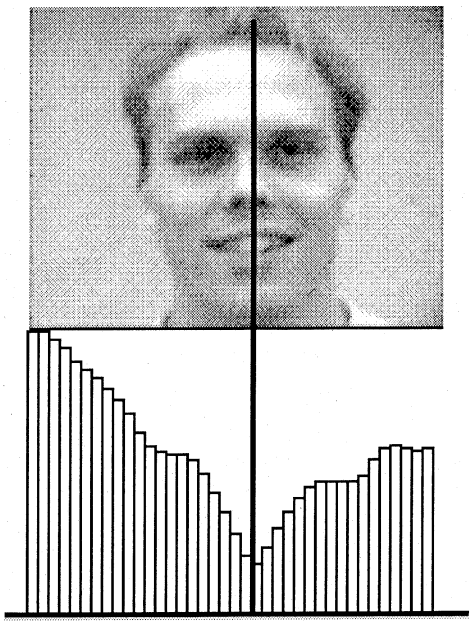

Fig. 3. The symmetry histogram.

$$H(y) = \sum_{i=1}^{xsize} G(i,y).$$

Since both eyes are likely to be positioned in the same row, $H(y)$ will have a strong peak on that row. However, in order to reduce the risk of error, we consider the best three peaks in $H(y)$ for further search rather than just the maximum. This process is illustrated in Fig. 4.

*3.2.3. Find the exact location of the eyes*

In order to find the eye regions given the previously mentioned processing steps, we rely on the fact that the eyes correspond to intensity valleys in the image. Given that, we can threshold the image by keeping only pixels with a value lower than a certain threshold value, and then extract the connected regions in the resulting image. We used a raster-scan algorithm on the reduced

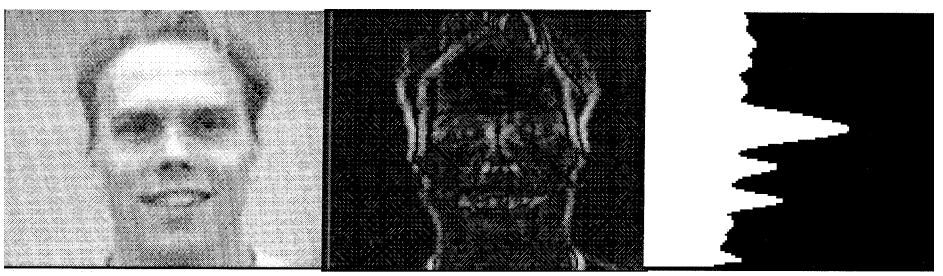

Fig. 4. The original image, the edges and the histogram of projected edges.

image in order to extract these regions. In general, our raster-scan algorithm found 4–5 regions. In order to resolve which of these regions correspond to the eyes, we use the information in $H(y)$. We try to find a peak corresponding to a row in the image with two connected regions on. The three best peaks in $H(y)$ are considered. We also use general constraints, such that both eyes must be located "fairly close" to the center of the face.

One difficulty with this method is to find a threshold that will generate the correct eye regions. As shown in Fig. 5, we can see that the same threshold value generates significantly different results for different images.

We used a method called adaptive thresholding (Stiefelhagen et al., 1996) that starts out with a low threshold. If two good eye regions are found, that threshold is stored and used the next time the eyes have to be localized.

If no good eye regions are found, the system automatically attempts with a higher threshold until the regions are found. When the threshold gets too high, the system will start over with a low threshold again. If no good regions are found on any of the three biggest peaks in $H(y)$ with any of the attempted thresholds, we conclude that there are no eyes in the image (e.g., the driver is looking over his shoulder). In this case, the system will continuously attempt the same process with subsequent frames, until the eye regions are back in the image. An example of adaptive thresholding is shown in Fig. 6.

### *3.2.4. Estimation of the position of the iris*

Once the eye regions are localized, we can apply a very simple template in order to localize the iris. As mentioned in the introduction, many templates have been developed for this purpose. We constructed a template consisting of two circles, one inside the other. A good match would result

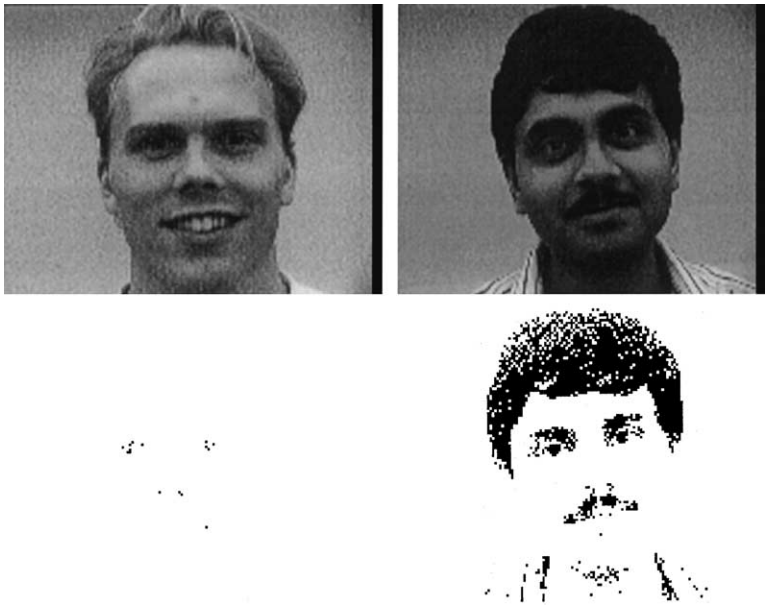

Fig. 5. The same threshold value applied to two different images.

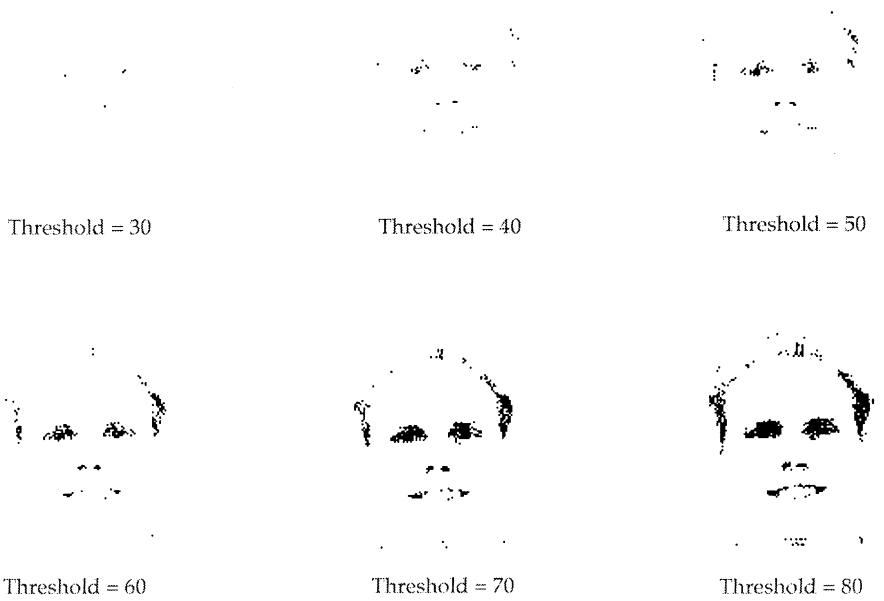


Fig. 6. Adaptive thresholding.

in many dark pixels in the area inside the inner circle, and many bright pixels in the area between the two circles. The template is shown in Fig. 7. This match occurs when the inner circle is centered on the iris and the outside circle covers the sclera. We decided to use a fixed size template that does not deform. In our case this works, since we have narrowed down the search space to a very small area, which minimizes the risk of finding a better match than the iris.

The match $M(a1, a2)$ is computed as

$$M(a1, a2) = \sum_{(p,q)\in a1} I(p,q) - \sum_{(p,q)\in a2} I(p,q).$$

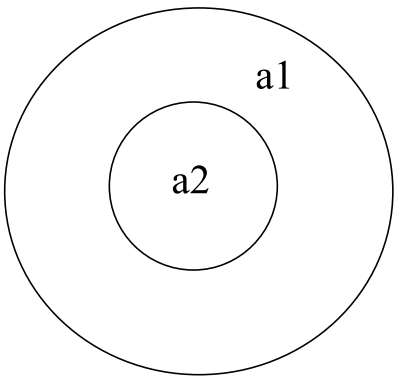


Fig. 7. The eye template.

A low value for $M(a1, a2)$ corresponds to a good match. The template is matched across the predicted eye region and the best match is reported.

### 3.3. Tracking the eyes

Tracking a feature in a sequence of images involves looking for that feature in a small neighborhood centered at the location of that feature, in the previous frame. The main assumption is that the feature being tracked generally moves fairly slowly between frames. In our case, we do not need to go through the entire process of localizing the eye all over again in the following frames. We track the eye by looking for the darkest pixel in the predicted region (Stiefelhagen et al., 1996). Since the darkest pixel will almost certainly be inside the pupil, this will work well if we assume that the eyes will not move very far between two consecutive frames. In order to recover from tracking errors, we make sure that the distance between the eyes remains constant and reasonable. We also control that the eyes are reasonably close horizontally to each other. If any of the geometrical constraints is violated, we relocalize the eyes in the next frame. To find the best match for the eye template, we initially center it at the darkest pixel, and then perform a gradient descent in order to find a local minimum. In Fig. 8, we show a few snapshots during tracking.

## 4. Detection of fatigue

At this point, we are trying to detect microsleeps. This is done by measuring the time period during which the eyes are closed. As the driver becomes more fatigued, we expect the eye blinks to last longer, eventually developing into microsleeps. We count the number of consecutive frames that the eyes are closed in, in order to decide the condition of the driver. For this, we need a robust way to determine if the eyes are open or closed. Our first approach was to use the match reported for the eye template described in Section 3.2.4. However, this sometimes failed in the case when the head was tilted or turned. Because of this, we developed a second method that looks at the horizontal histogram across the pupil.

During initialization (the first frames after the driver has settled down), an average match over a number of frames is calculated. When the match in a frame is "significantly" lower than the average, we call that frame a *closed* frame. If the match is close to the average, we call that an *open* frame. After $C$ consecutive closed frames, we issue a warning signal, where $C$ is the number of frames corresponding to approximately 2 to 2.5 seconds (the time when the eyes have been closed for too long). Fig. 9 shows a few snapshots of the system while performing detection of fatigue.

### 4.1. Horizontal histogram across the pupil

We use the characteristic curve generated by plotting the image intensities along the line going through the pupil from left to right, as shown in Fig. 10. The pupil is always the darkest point. Surrounding the pupil, we have the iris which is also very dark. To the right and left of the iris is the white sclera. In Fig. 10 we show two curves, one corresponding to an open eye, and one corresponding to a closed eye.

Fig. 8. Snapshots from the system during tracking. Note that in the second image, the system missed tracking of one eye.

Note that the curve corresponding to the closed eye is very flat.

We compute the matching function $M(x,y)$ as

$$M(x,y) = I(x,y)/\min\{I(x-r,y), I(x+r,y)\}$$

where $(x,y)$ is the computed center of the pupil and $r$ is the radius of the iris. $I(x,y)$ is the image intensity at $(x,y)$. When the eye is open, the valley in the intensity curve, corresponding to the pupil, will be surrounded by two large peaks corresponding to the sclera. When the eye is closed, this curve is usually very flat in the center. However, in the latter case, there is no pupil to center the curve on which can lead to a very unpredictable shape. In order to minimize the risk of having

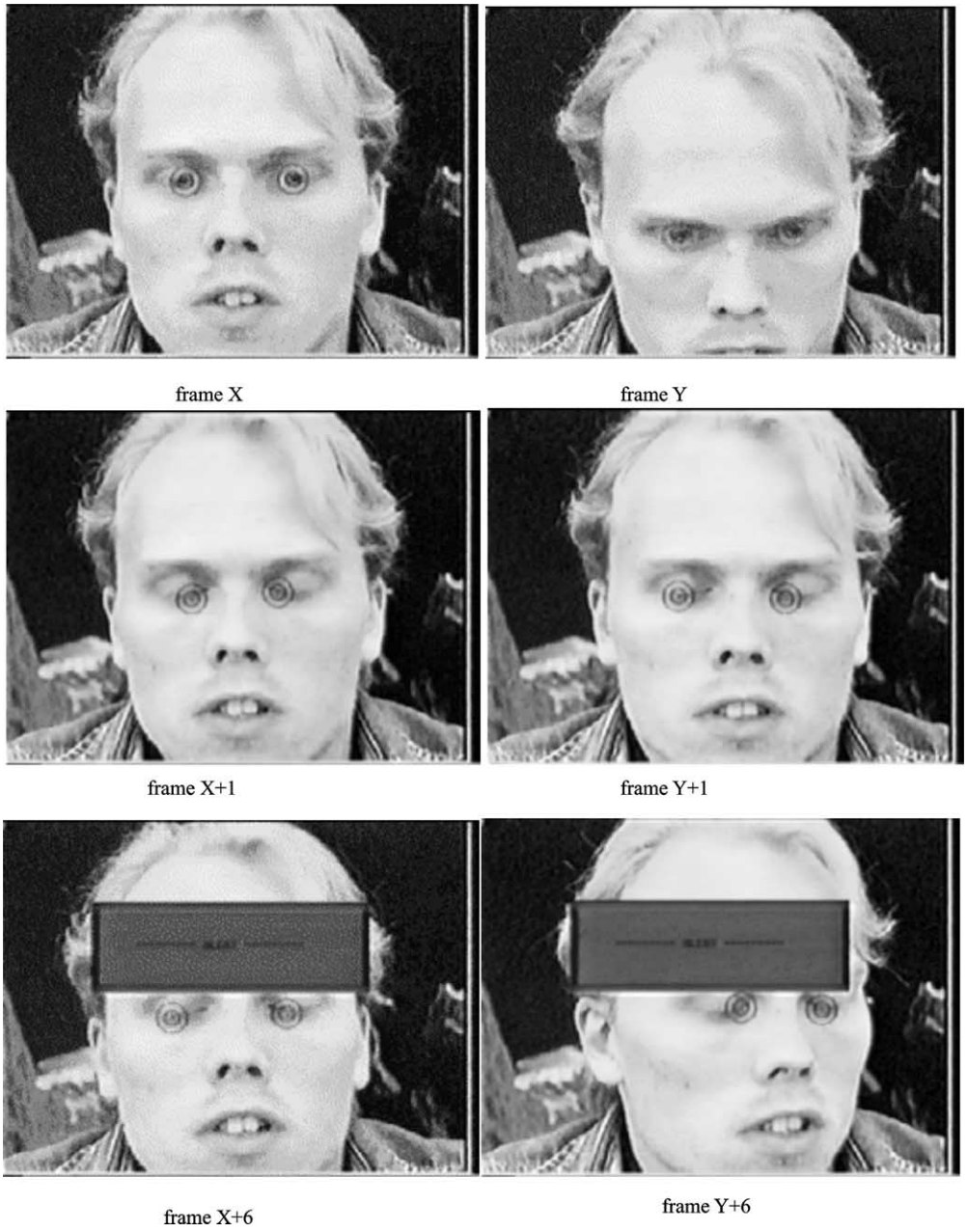


Fig. 9. Snapshots of the system while detecting fatigue.

one big peak nearby (due to noise), we always use the minimum peak at the distance $r$ from the pupil. This will lead to a good match when the eye is open, and very likely to a bad match when the eye is closed. The reason that we choose this method instead of using the match generated more accurately than a circular template, is that the horizontal line captures the characteristic eye

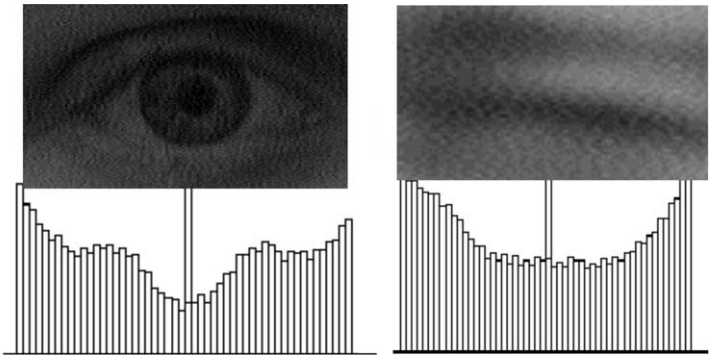

Fig. 10. Histograms corresponding to an open and a closed eye.

colours more accurately than a circular template does. A very good example of this is when the head is tilted forward. A large fraction of the circular template would end up outside the eye, since the perspective makes the eye "thinner." However, the histogram of the line going through the pupil will retain its characteristic shape. This is a very important observation since when the driver is getting tired, his or her head is very likely to fall slightly forward.

## 5. Results and future work

This section presents four "test drives" where we measured the accuracy of the detection of opened/closed eyes (for evaluation purposes, we conducted a larger number of test drives than the ones reported here). In each test drive, we simulated 10 long eye blinks, and recorded how many were computed by the system. In each test drive, the driver had the head turned in a different angle. The results are shown in Table 1. The environment where the test drives took place is a lab environment with moderate light (we tried to emulate real driving conditions). The duration of each test drive was several minutes.

For this test, we did not allow for any rapid head movements, since we wanted to simulate the situation when the driver is tired. For small head movements, the system rarely loses track of the eyes, as we can see from the results. We can also see that when the head is turned too much sideways, we had some false alarms. However, in the case where the head is tilted forward (which is the most likely posture when the driver is tired), the system operated perfectly. It should be

Table 1
Results from the tests

| | 0° left | 30° left | 45° left | 30° downwards |
|---|---|---|---|---|
| Detected alerts (10) | 10 | 10 | 10 | 10 |
| False alerts | 0 | 1 | 2 | 0 |
| Lost track | 0 | 0 | 0 | 0 |

mentioned that we are also in the process of testing the method with human subjects in a driving simulator (Human Factors Laboratory, University of Minnesota).

When we perform the detection of driver fatigue, we operate on frames of size 640 by 320. This frame size allows us to operate at approximately 5 frames/s. In order to track the eyes without detecting fatigue, it is enough to use frames of size 320 by 160, which allows a frame rate of approximately 15 frames/s.

At this point, the system has problems localizing eyes when the person is wearing glasses or has a large amount of facial hair. We believe that by using a small set of face templates similar to Tock and Craw (1995), we will be able to avoid this problem without losing anything in peformance. Also, we are not using any color information in the image. By using techniques described in Stiefelhagen et al. (1996), we can further enhance robustness.

Currently, we do not adjust zoom or direction of the camera during operation. Future work may be to automatically zoom in on the eyes, once they are localized. This would avoid the trade-off between having a wide field of view in order to locate the eyes, and a narrow field of view in order to detect fatigue.

In the current implementation we are only looking for microsleeps. At that point when the driver is having microsleeps, an accident may already have occurred. By studying the eye-movement patterns, we are hoping to find a method to generate the alert signal at an earlier stage.

## 6. Conclusions

We have developed a system that localizes and tracks the eyes of a driver in order to detect fatigue. The system uses a combination of template-based matching and feature-based matching in order to localize the eyes. During tracking, the system is able to automatically detect any error that might have occurred. In case of tracking error, the system is able to recover and resume the proper tracking. The system is also able to decide if the eyes are open or closed, which facilitates the detection of fatigue. During testing, we were able to accurately detect microsleeps when the head was directly facing the camera. As the head was increasingly turned sideways, away from the camera, the false alarm rate increased. However, performance was still very good when the head was pointing downwards, which is the most likely posture to occur when the driver is getting fatigued.

## Acknowledgements

We would like to thank the anonymous reviewer for his detailed comments. This work has been supported by the ITS Institute at the University of Minnesota, the Minnesota Department of Transportation through Contracts #71789-72983-169 and #71789-72447-159, the National Science Foundation through Contracts #IRI-9410003 and #IRI-9502245, the Center for Transportation Studies, and the McKnight Land-Grant Professorship Program at the University of Minnesota.

## References

Brunelli, R., Poggio, T., 1993. Face recognition: Features versus templates. IEEE Trans. on Pattern Analysis and Machine Intelligence 15 (10), 1042–1052.

Chow, G., Li, X., 1993. Towards a system for automatic feature detection. Pattern Recognition 26 (12), 1739–1755.

Cox, I.J., Ghosn, J., Yianilos, P.N., 1995. Feature-based recognition using mixture-distance. NEC Research Institute, Technical Report 95–09.

Craw, I., Ellis, H., Lishman, J.R., 1987. Automatic extraction of face features. Pattern Recognition Lett. 5, 183–187.

Silva, L.C., Aizawa, K., Hatori, M., 1995. Detection and tracking of facial features by using edge pixel counting and deformable circular template matching. IEICE Trans. on Information and Systems E78-D (9), 1195–1207.

Dijk, D.J., Czeisler, C.A., 1994. Paradoxal timing of the circadian rhythm of sleep propensity serves to consolidate sleep and wakefulness in humans. Neurosci. Lett. 166, 63–68.

Dinges, D.F., 1995. An overview of sleepiness and accidents. J. Sleep Res., 1–11 (suppl. 4).

Huang, C.L., Chen, C.W., 1992. Human facial feature extraction for face interpretation and recognition. Pattern Recognition 25 (12), 1435–1444.

Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: active contour models. Int. J. of Comput. Vision, 321–331.

Li, X., Roeder, N., 1995. Face contour extraction from front-view images. Pattern Recognition 28 (8), 1167–1179.

Roeder, N., Li, X., 1996. Accuracy analysis for facial feature detection. Pattern Recognition 29 (1), 143–157.

Segawa, Y., Sakai, H., Endoh, T., Murakami, K., Toriu, T., Koshimizu, H., 1996. Face recognition through hough transform for irises extraction and projection procedures for parts localization. Pacific Rim International Conference on Artificial Intelligence, 625–636.

Stringa, L., 1993. Eyes detection for face recognition. App. Artificial Intelligence 7, 365–382.

Stiefelhagen, R., Yang, J., Waibel, A., 1996. A model-based gaze-tracking system. International IEEE Joint Symposia on Intelligence and Systems, 304–310.

Tello, E.R., 1983. Between man and Machine. BYTE, 288–293.

Tock, D., Craw, I., 1995. Tracking and measuring drivers eyes. Real-Time Comput. Vision, 71–89.

White, K.P., Hutchinson, T.E., Carley, J.M., 1993. Spatially dynamic calibration of an eye-tracking system. IEEE Trans. on Systems, Man, and Cybernetics 23 (4), 1162–1168.

Wu, H., Chen, Q., Yachida, M., 1996. Facial feature extraction and face verification. IEEE Proceedings of the International Conference of Pattern Recognition, 484–488.

Xie, X., Sudhakar, R., Zhuang, H., 1994. On improving eye features extraction using deformable templates. Pattern Recognition 27 (6), 791–799.

Xie, X., Sudhakar, R., Zhuang, H., 1995. Real-time eye feature tracking from a video image sequence using Kalman filter. IEEE Trans. on Systems, Man, and Cybernetics 25 (12), 1568–1577.

Yoo, T.W., Oh, I.S., 1996. Extraction of face region and features based on chromatic properties of human faces. Pacific Rim International Conference on Artificial Intelligence, 637–645.