Monitoring Head/Eye Motion for Driver Alertness with One Camera

Paul Smith, Mubarak Shah, and N. da Vitoria Lobo
Computer Science, University of Central Florida, Orlando, FL 32816
{rps43158,shah,niels}@cs.ucf.edu

Abstract

We describe a system for analyzing human driver alertness. It relies on optical flow and color predicates to robustly track a person's head and facial features. Our system classifies rotation in all viewing directions, detects eye/mouth occlusion, detects eye blinking, and recovers the 3D gaze of the eyes. We show results and discuss how this system can be used for monitoring driver alertness.

1. Introduction

A system for classifying head movements would be useful in warning drivers when they fell asleep. Also, it could be used to gather statistics about a driver's gaze.

We describe a framework for analyzing movies of driving and determining when the driver is not paying adequate attention to the road. We use a single camera placed on the car dashboard. We focus on rotation of the head and blinking, two important cues for determining driver alertness.

Our head tracker consists of tracking the lip corners, eye centers, and sides of face. Automatic initialization of all features is achieved using color predicates[5] and connected component algorithms.

Occlusion of the eyes and mouth often occurs when the head rotates or the eyes close, so our system tracks through such occlusion and can automatically reinitialize when it mis-tracks. We implement blink detection and demonstrate that we can obtain 3-D direction of gaze from a single camera. These components allow us to classify rotation in all viewing directions and detect blinking, which, in turn, are necessary components for monitoring driver alertness.

First, we describe previous work and then describe our system in detail. We then present results, discuss driver alertness, and conclude.

1.1. Previous Work

Work on driver alertness [3] [4] [7] [8] [9] [10], to our knowledge, has not yet led to a system that works in a moving vehicle. The most recent of these [3], did not present any methods to acquire the driver's state. Further their method relies on LEDs, and uses multiple cameras to estimate facial orientation. A moving vehicle presents new challenges like variable lighting and changing backgrounds. The first step in analyzing driver alertness is to track the head. Several researchers have worked on head tracking [6] [2], and the various methods each have their pros and cons.

1.2. Input Data

The movies were acquired using a video camera placed on the car dashboard. The system runs on an UltraSparc using 320x240 size images with 30 fps video. Two drivers were tested under different daylight conditions ranging from broad daylight to parking garages. Some movies were taken in moving vehicles and others in stationary vehicles.

1.3. Parts Used from Other Research

A color predicate was originally developed by Kjeldsen *et al.* [5]. The idea, there, is to manually mark subsets of the RGB color space that the algorithm should recognize in future test images.

Anandan's optical flow algorithm [1] produces affine optical flow. It computes the global motion of a scene.

2. The Algorithm

Here is an overview of our algorithm.

1. Automatically initialize lips with color predicate and connected components
2. Automatically initialize eyes using color predicate and connected components
3. Track lip corners with dark line and color predicates
4. Track eyes with affine optical flow and color predicates
5. Construct a bounding box of head using color predicate
6. Determine rotation using distance between eye and lip feature points and sides of face
7. Determine blinking and eye disappearance using the number and intensity of pixels in eye region
8. Reconstruct 3D gaze using constant projection assumptions
9. Make inferences regarding driver's state using rotation and eye occlusion information
10. Decide, using rotation and distance constraints, if eye or lip tracking needs reinitialization
11. Repeat from step 3 for next frame

2.1. Initializing Lip and Eye Feature Points

2.1.1. Automatic Lip Initialization

A color predicate was generated using 7 images of people's lips. A few of the training images together with their manually drawn lip regions and automatically selected lip colored pixels by the color predicate are shown in Fig 1.

Figure 1. lip color predicate training.

The reason for the salt and pepper noise throughout the image is that backgrounds have lip-like colors in them. Also, parts of the faces were lip colored due to lighting conditions. Fig 2 shows the results of running the lip color predicate on non-training images. After obtaining this lip

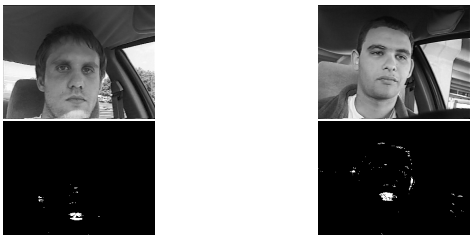

Figure 2. color predicate non-trained images.

image, we apply a connected component algorithm to it, and the biggest lip colored region is identified as the mouth. We compute edges of this mouth region and declare these as the lip corners. The initialization does not need pinpoint accuracy as the lip tracker itself will overcome inaccuracies. Fig 3 shows the results of automatic lip initialization on the previously shown input images.

2.1.2. Automatic Eye Initialization

Automatic eye initialization uses skin color predicates as well, though in a different way. Fig 4 shows input images, manually selected skin regions, and the output of the color predicate program on training images.

Figure 3. Output of automatic lip initialization

Figure 4. skin color predicate training

Figure 5. skin found in non-trained images

Fig 5 shows output of the skin color predicate on non-training images. Since eyes are not skin, they always show up as holes. Hence, we find connected components of non-skin pixels to find the eye holes. We find the two holes that are above the previously found lip region, and that satisfy the following size criteria for eyes. Since our dashboard camera is at a fixed distance from the face, we estimate the relative size of eyes to be between 15 and 800 pixels. For all images we tested(several hundred), we found these criteria to be reliable. Fig 6 shows results of automatic eye and lip initialization from various data sets.

2.2. Lip Tracking

We have a multi-stage lip tracker. The first stage is the most accurate but unstable. The second stage is not as accurate but more stable. The third stage is coarse but very stable. We use the first stage estimate, if it is correct. If not

Figure 6. automatic eye and lip initialization

we take the second stage estimate. If both stages fail, we take the third stage estimate as the lip corners.

For the first stage, we automatically find the dark line between the lips, shown in Fig 7 as a white line. We compute this dark line as follows. We find the center of the lips from $\frac{PreviousCornerLeft+PreviousCornerRight}{2}$ and for each side of the mouth we start examining each pixel outward from the lip center. For each pixel, we consider a vertical line and find the darkest pixel, $\min \frac{R+G+B}{3}$, on this vertical line. The darkest pixel will generally be a pixel on the dark line between the lips. We do this for 35 subsequent pixels, which is why the mouth line extends beyond the sides of the mouth. To determine where the lip corners are we obtain $f(x) = \frac{1}{DistanceFromClosestCorner} + \frac{1}{Brightness}$ for each pixel; this is because we want a pixel that is close to the previous lip corner, but if it is too bright, then it cannot be the lip corner. The function maximum is the lip corner. If this estimate is too far from the previous lip corner, we run the second stage of our algorithm, described next.

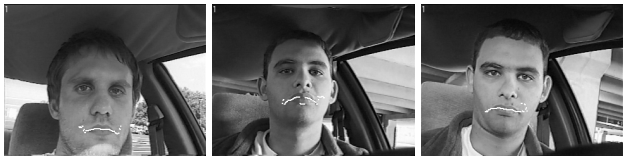

Figure 7. Examples of dark line between lips

Here we use a stricter color constraint. With the darkest line found, we select the pixel closest to the previous lip corner that has lip colored pixels above and below it.

If the second stage fails then we employ the third stage, which is simply reinitialization of the system, as described above in section 2.1.1, within the most recent lip region. In this way we have a method automatically able to correct itself when the tracking is lost due to occlusion. In subsequent frames the previous lip tracking steps will resume control and regain the exact position of the lip corners.

The reason for our hierarchical lip tracker is that large rotation, occlusion, or rapidly changing lighting breaks down the accurate(first) stage. The two other stages are more coarse, but they are more robust. Fig 8 shows the output of the lip tracker for a variety of images.

Figure 8. lip tracker for a variety of sequences

2.3. Eye Tracking

We have a multi-stage eye tracker with similar constraints to the multi-stage lip tracker. For the first stage, we go to the eye center in the previous frame and find the center of mass of the eye region pixels. Then we search a 5×5 window around the center of mass and look for the darkest pixel, which corresponds to the pupil. If this estimate produces a new eye center close to the previous eye center then we take this measurement.

If this stage fails, we run the second stage, where we search a window around the eyes and analyze the likelihood of each non-skin connected region being an eye. We limit the search space to a 7×20 window around the eye. We find the slant of the line between the lip corners. The eye centers we select are the centroids that have the closest slant to that of the lip corners. Still, this method by itself can get lost after occlusion. For simplicity in our description, we refer to these two stages together as the eye black hole tracker.

The third stage, which we call the affine tracker, runs in parallel with the first two stages. Since automatic initialization yields the eye centers, we construct windows around them, and then in subsequent frames, consider a second window centered around the same point. We compute the affine transformation between the windowed subimages and then, since we know the eye center in the previous frame, we warp the subimage of the current frame to find the new eye center. Thus, we have two estimates for the eye centers, one from the eye black hole tracker and one from the affine tracker. When there is rotation or occlusion or when the eye black hole tracker produces an estimate that is too far away from the previous frame, we use the affine tracker solely. In all other cases we take an average of the two trackers to be the eye center. Later, we discuss how we detect rotation.

We use Anandan's algorithm to compute the affine transformation. It is less likely to break down during heavy occlusion. The affine tracker is not as accurate as the eye black hole tracker, because of the interpolation in warping, which is why we don't use it exclusively unless as a last resort. Figs 9 and 10 show some results of the eye and mouth tracker in various images from the data sets.

Figure 9. whole head tracker

Figure 10. head tracker with eye occlusion

Sometimes, after occlusion, the eye tracker mis-tracks. To compensate, whenever the distance between the eyes gets to more than $\frac{1}{4}$M (where M is horizontal image size), we reinitialize the eyes. This criteria was adopted because we know both the location of the camera in the car and the approximate size of the head. We also reinitialize the eyes when the the lips reappear after complete occlusion, which we determine when the number of lip pixels in the lip region drops below five pixels and comes back. The reasoning being that if the lips are fully occluded, then the eyes will not be visible, so when they reappear we should reinitialize.

This eye tracker is very robust; it tracks successfully through occlusion and blinking in our experiments. Further, it is not affected by a moving background, and it has been verified to track continuously on sequences of 400 frames.

2.4. Bounding Box of Face

We can determine face rotation if we have the face's bounding box. To find this box, we start at the center of the head region, which is computed using the average of the eye centers and lip corners. We can do this because the center of the head is approximately located in between these four feature points. We could have found a more accurate centroid of the head, but only a rough estimate is needed here. Then for each side of the face, we start our search at a constant distance from the center of face and look inward finding the first consecutive five pixels that are all skin. Using five pixels, protects us from selecting the first spurious skin pixel. This approach gives an acceptable face contour. We show each side of the face as a straight line, produced from an average of all the positions for that (curved) side of the face in Fig 11, along with the tracked eyes and mouth.

Figure 11. face trace with head tracker

2.5. Occlusion, Rotation, and Blinking

Often the driver blinks or rotates the head, so occlusion of the eyes or lips occurs, which we need to detect. To clarify: our tracker is able to track through most occlusion, but it does not recognize that occlusion (from rotation or blinking) occurred. For driver alertness, we need to develop algorithms to model occlusion so that we can identify these activities. Our occlusion model deals with rotation and blinking, important factors for driver alertness.

Because of foreshortening, when rotation occurs, depending on which direction rotation is occuring in, the distance between the feature points and sides of face will increase or decrease. So, in each frame we compute the distance from the sides and top of the face to the eye centers. We also compute the distance from the side of face from the mouth corners. We take the derivative of these measurements over time and when there is consistent decrease or increase in the distance, this indicates rotation. Formally, when more than half of the distances of a particular feature point indicate rotation in the same direction, then this feature point is assumed to be involved in head rotation. Fig 11 shows how the distance between the face sides and eye and mouth feature points increase and decrease during rotation.

Next, a voting system is constructed where each feature point predicts the direction of rotation. When half or more of the feature points detect rotation, then we declare rotation in this particular direction. Each feature point can be involved in rotation along combinations of directions, but some cases are mutually exclusive(e.g. simultaneous left and right rotation). We have verified that the system can detect rotation along combined directions (e.g. up and left). By considering the distance from the sides and top of head we can discriminate rotation from translation of the head.

Fig 12 shows the output of the whole head tracker including rotation analysis messages, automatically displayed by the system. Next, we present a method to determine when blinking occurs.

We have two methods to eye occlusion detection. The first method computes the likelihood of rotational occlusion of the eyes. To determine occlusion of the eyes we look for the number of skin pixels in the eye region, and when this increases to more than $\frac{4}{5}S$, where S is the size of the eye region, then we assume rotational occlusion is occuring. We know the size of the eye region because we have the non-skin region from the skin color predicate. The reason why we do not just announce rotational occlusion when rotation occurs is that rotation is not a sufficient condition to infer eye occlusion. During small rotation, both eyes will still be visible. Our method works well in rotational occlusion of the eyes.

The next method computes the likelihood of the eyes being closed (blinking). This method relies on the simple fact

Figure 12. head tracker rotation messages

that the eyes contain eye whites. In each frame as long as there are eye-white pixels in the eye region then we assume that the eyes are open. If not, then we assume blinking in the particular eye is occuring. To determine what is considered eye-white color, in the first frame of each sequence we find the brightest pixel in the eye region. This allows the blink method to adapt to various lighting conditions.

For the above eye occlusion detection, each eye is independent of the other. Now we are able to distinguish between rotational occlusion of eyes and the eyes closing(blinking). These methods give very good results. Fig 14 shows some of the results from blink detection for both short blinks and long eye closures.

2.6. Reconstructing 3D Gaze Direction

The problem of 3D reconstruction is a difficult one. Already, we have determined rotation information and now we provide a solution to 3D gaze tracking problem with a single camera. The reason we can do this is that we only need the direction of the gaze. For all practical purposes, the gaze could go on through the windshield. By making this assumption we eliminate the need to know the distance from the head to the camera. Also, if we assume that head size is relatively constant between people then we have all the information we need. Since we only want the direction of gaze, we need the x,y locations of the eyes and back of head. With this information, we can construct the line which passes through all the z coordinates. We know the eye locations, so if we can find the back of the head, then we can reconstruct the gaze. We have found through our experimentation that when rotation occurs, the back of the head can be approximated well by the average of the two eyes subtracted from its distance from the center of the head in the first frame. This assumption is valid because when rotation occurs, the average position of the two eyes moves in the opposite direction to the back of the head. When head translation occurs, we add the translation of the average position of the eyes to the original back of the head. This gives us the relative location of the back of the head(relative to the first frame). Since we have the x,y location of the eyes and the back of the head, we can draw lines in xyz space showing the direction of the gaze. This method allows us to derive the gaze direction.

Fig 13 shows some results from acquiring 3D gaze information. The first two pictures are the input pictures. The next two figures are the graphical 3D representation of the scene from the y,z plane (with the x axis coming out of the page). The lowest figure was just one of the pictures from the sequence which shows that the head moves up in this picture with no rotation. This is an important component of the system because now it is possible to generate statistics of where the driver's gaze is.

The system is not foolproof. Given a low-lighting picture the method of head tracking would break down. However, we have tested our program on twelve sequences ranging from 30-400 frames, and the system appears to be very robust and stable.

3. Driver Alertness

In this section we propose ideas about how to use our system to acquire the driver's state.

When the driver is looking away for too long, then we warn that the driver's alertness is too low. Similarly, when the driver's eyes are occluded (either from blinking or rotation occlusion), for too long we warn that the driver's alertness is too low. More of a rigorous analysis on the physiology of sleep behaviors is necessary before accurately determining when a driver has fallen asleep. For our system however, we are able to determine a basic set of criteria to determine driver vigilance. We can do the following. Since we know when the driver's eyes are closed we assume the driver has a low vigilance level if the eyes are closed for more than 40/60 frames. In each frame we always know the number of lip pixels in the image. So we can threshold this number, and whenever there are too few lip pixels we will assume that the head is heavily rotated. We can then print that the driver's vigilance level is too low. However, since it is natural for a driver to look left and right we will only print a driver inalertness message if the heavy lip occlusion occurs for more than 10/20 frames. Finally for general rotation that does not completely occlude the eyes, we will not give driver inalertness warnings unless the rotation is prolonged.

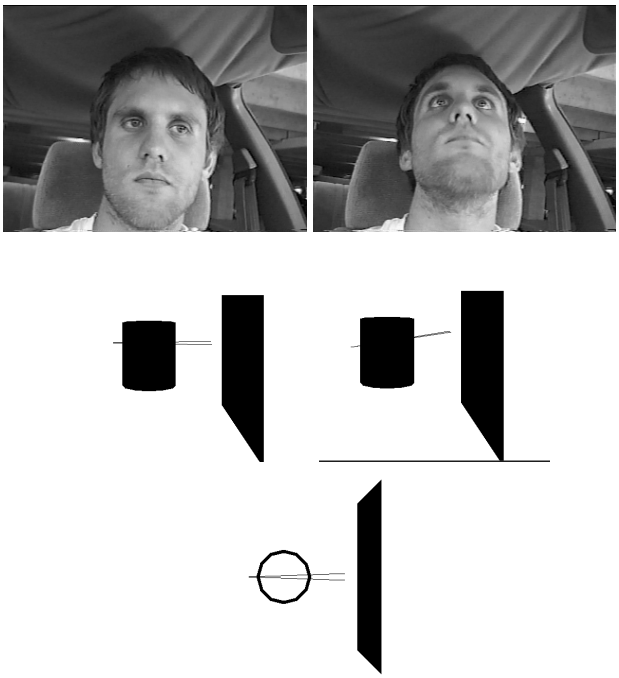

Figure 13. Acquiring 3D info

Again, it is not our intention to delve into the physiology of driver alertness at this time. We are merely demonstrating that with our framework, it is possible to collect driver information and begin to make inferences as to whether the driver is alert or not.

4. Summary and Future Directions

We presented a method to track the head, using color predicates to find the lips, eyes, and sides of the face. It was

tested under varying daylight conditions with good success. We compute eye blinking, occlusion information, and rotation information to determine the driver's alertness level.

There are many future directions for driver alertness. For aircrafts and trains, the system could monitor head motions in general and track vehicle operator alertness.

As we can recognize all gaze directions, we could develop a larger vocabulary and classify checking left/right blind spots, looking at rear view mirror, checking side mirrors, looking at the radio/speedometer controls, and looking ahead. Also we could recognize yawning. Other improvements could be coping with hands occluding the face, drinking coffee, conversation, or eye wear.

References

[1] J.R. Bergen, P. Anandan, K. Hanna, R. Hingorani. "Hierarchical Model-Based Motion Estimation." Procs. ECCV, pp. 237-252, 1992.

[2] A.Gee and R. Cipolla. "Determining the Gaze of Faces in Images." Image and Vision Computing, 30:639-647, 1994.

[3] Qiang Ji and George Bebis "Visual Cues Extraction for Monitoring Driver's Vigilance." Procs. Honda Symposium, pp. 48-55, 1999.

[4] M.K. *et al*. "Development of a Drowsiness Warning System." 11th International Conference on Enhanced Safety of Vehicle, Munich, 1994.

[5] Rick Kjedlsen and John Kender "Finding Skin in Color Images." Face and Gesture Recognition, pp. 312-317, 1996.

[6] C. Morimoto, D. Koons, A. Amir, M. Flickner. "Realtime detection of eyes and faces." Workshop on Perceptual User Interfaces, pp. 117-120, 1998.

[7] R. Onken. "Daisy, an Adaptive Knowledge-Based Driver Monitoring and Warning System." Procs. Vehicle Navigation and Information Systems Conference, pp. 3-10, 1994.

[8] H. Ueno, M. Kaneda, and M. Tsukino. "Development of Drowsiness Detection System." Procs. Vehicle Navigation and Information Systems Conference, pp. 15-20, 1994.

[9] Wierville. "Overview of Research on Driver Drowsiness Definition and Driver Drowsiness Detection." 11th International Conference on Enhanced Safety of Vehicles, Munich 1994.

[10] K. Yammamoto and S. Higuchi. "Development of a Drowsiness Warning System." Journal of SAE Japan, 46(9), 1992.

Figure 14. Blink detection with head tracker