ML-Based Eye Tracking for Augmented Reality Heads-Up Displays (AR HUDs)

Ada Defne Tur, Deniz Yaralioglu, Cemalettin Yilmaz
CY Vision, Inc., San Jose, CA, USA

Abstract

3D Augmented Reality (AR) Heads-up Displays (HUDs) have the potential of overlaying virtual objects at the correct locations with accurate motion parallax. Accurate overlays require tracking the pupils of the driver's eyes. We developed an ML-based pupil tracking system based on a convolutional neural network (CNN) to find the precise location of the pupils.

Author Keywords

Augmented Reality; Heads-Up Display; Eye tracking; Convolutional Neural Networks.

1. Introduction

The new generation 3D HUDs will have much more compelling features than their 2D counterparts. These features will improve safety of the drivers as well as the driving experience. 3D holographic AR HUDs [1] can provide correct focus cues and can render objects at the correct depths and locations with accurate motion parallax in the real world. Practical holography applications in HUds have been enabled by the advances in the hologram calculation algorithms and the technology development in the field of spatial light modulators [2]. With the holographic techniques, instrument cluster and sensory data can be superimposed onto the windshield, virtual lane information, traffic signaling can be added in the driver's line of sight at different depths. This can provide very fast visual signals of upcoming road hazards. However, for AR HUD applications, overlaying the virtual objects accurately onto the real-world requires precise knowledge of where the driver's eyes are in the 3D space to be able to render correct parallax as well as the correct viewing angle of the objects. For this purpose, we developed the HUD system with integrated eye tracking that uses machine learning based algorithms.

2. AR HUD architecture

CY Vision's holographic HUD system is shown schematically in Figure 1. The main components of the system are the picture generation units (PGUs) and the binocular optics. The system has two picture generation units (PGUs), one per eye, to create stereoscopic 3D images. Additional benefit of having separate PGUs is the fact that the dynamic distortion correction can be custom made for each eye at each pupil positions across the eye-box volume. The binocular optics relay the image to the driver's eyes. This part has also a steerable mirror. The system includes a precision eye-tracker which generates the real-time 3D coordinates of the two pupils. The system provides separate images to each eye. To be able to steer the light correctly at the output of the optics, a mechanically scanned mirror is tilted in two dimensions based on the eye tracking data.

The pupil location is also used for rendering images at the correct locations. For example, Figure 2 shows visual objects rendered using the correct and wrong pupil locations. The green lane markers and the solid green box (indicating the closest car) show rendered images when the driver is at nominal position. However, if the head moves right and down, none of the visual objects align as shown in the Figure. The error in the location of the closer visual objects is higher (such as the box with green solid line vs the box with red dashed line indicating the closest car location) for 3D stereoscopic rendering. The pupil location is also used to render 3D virtual images with the correct viewing angle.

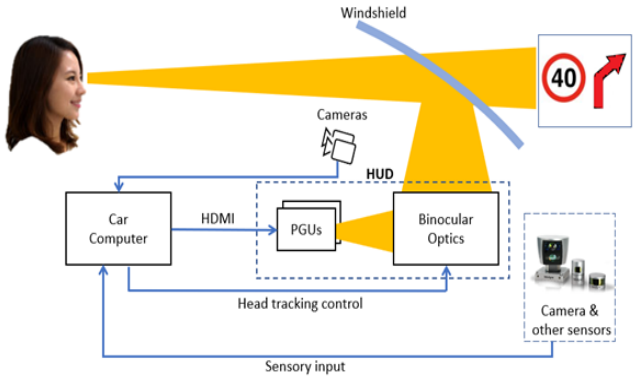

Figure 1. System schematic for a 3D holographic AR HUD system. There are two driver-looking cameras for eye tracking in the system.

The eye tracking system uses two cameras. The 3D location of each eye is tracked through stereoscopic vision obtained from these cameras. The camera parameters, such as resolution, separation, direction, are optimized to achieve the highest pupil detection accuracy. Both cameras are connected to a car computer which is ruggedized for automotive grade applications. The car computer is also used for rendering the virtual images and hologram calculation.

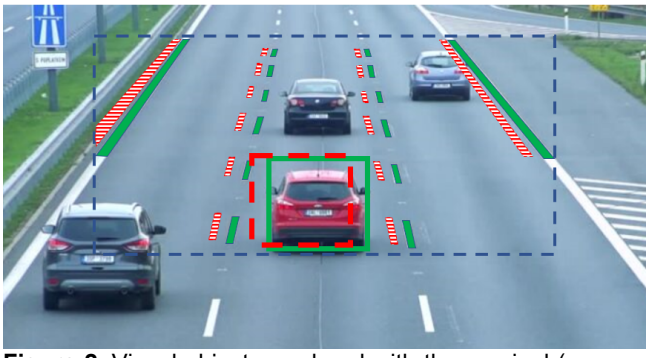

Figure 2. Visual objects rendered with the nominal (green lanes markers and a box indicating the closest car) head position. If the head position is shifted the whole scene shifts (red lane markers and the box). The box with the blue dashed line shows the FOV of the HUD for the nominal head position.

We developed Machine Learning (ML) based eye tracking to find the precise location of the pupils. The ML model is based on a convolutional neural network (CNN), aiming to detect the

 ISSN 0097-996X/21/5203-1162-$1.00 © 2021 SID

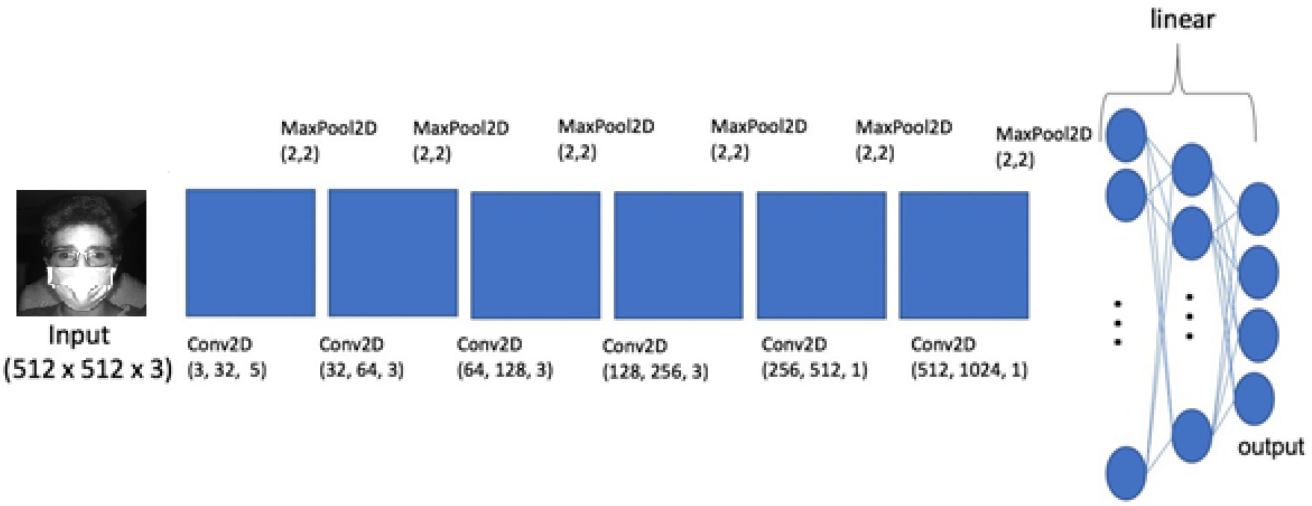

Figure 3. The architecture of the convolutional neural network architecture employed for detecting the pupil (x, y) coordinates of the driver.

pupils of the drivers. Due to the pandemic, we expect that most drivers would wear a mask while driving. To increase the robustness of the model for drivers wearing face masks, training data was augmented with such images. In the next section, the details of the ML algorithm will be presented.

3. Eye tracking system

Eye Tracking Components

We use two cameras to detect the driver's eyes in three dimensions. Each camera has its own active illumination to achieve optimum contrast. Visible filters were installed in the cameras to cut the visible light spectrum below 750 nm. The active illumination uses IR LEDs at 850 nm. Larger wavelength IR LEDs can also be used for illumination purposes. We used off-the-shelf USB 3.0 cameras with global shutter and collected data at various lighting conditions to test the eye tracking. We set the maximum exposure rate to 1 msec to minimize the motion blur. We optimized the IR illumination based on the ambient light to achieve maximum contrast. We detected the dark pupils in our eye tracking method. The eye tracking algorithms were run on an Intel microprocessor (i7) based PC.

Eye tracking algorithm

The model follows a basic Convolutional Neural Network architecture as shown in Figure 3. The convolutional layers are typically used in image processing, applying filters to each overlapping part or filter-sized patch of the input data. A convolution operation is defined with the function:

$$f_l^k(p,q) = \sum_a \sum_{x,y} I_a(x,y) \cdot e_l^k(u,v) \quad (1)$$

where, $I_a(x,y)$ is an element of the input image tensor I_a, which is element wise multiplied by $e_l^k(u,v)$ index of the k^{th} convolutional kernel k_l of the l^{th} layer [3].

The output feature-map of the k^{th} convolutional operation can then be expressed as:

$$F_l^k = [f_l^k(1,1), \ldots, f_l^k(p,q), \ldots, f_l^k(P,Q)] \quad (2)$$

The output of each pooling layer is represented as:

$$Z_l^k = g_p F_l^k \quad (3)$$

where g_p is max pooling in our work.

In-between each convolutional layer, the model applies a maximum pooling of 2x2 cells and reduces the dimensionality by a factor of 4. After flattening, the linear layers represent a fully connected deep neural network, resulting in the x and y coordinates of two pupils (4 output points).

The ML algorithm uses the well-known face landmark detection architecture for the DLib Dataset [4] which contains 6666 images of varying dimensions with 68 landmarks for each image. This dataset dad been automatically generated by applying DLib's pose estimation [5] on images from ImageNet [6] tagged as 'face' and had become widely used. The model we have employed in this study is very similar to that one, except that we have simplified that model so that we have not performed any rescaling. We only cropped the native resolution (640x512) of the camera to 512x512 pixels. Preserving the native resolution enabled the model to operate at a wider range as the center of pupil is not a large area in the images captured. Secondly, we have focused on only 2 key-points, right and left eye pupils, instead of the 68 landmark points annotated in DLib. Fewer targets made the model train and test faster. However, the image sizes are larger in our captured images; hence we have optimized the model depth accordingly.

More specifically, as shown in Figure 3, the CNN model employed has 6 convolutional layers with max-pooling, followed by 2 linear layers (first from 49K to 1K, and second from 1K to 4 (2x2) cells). Various levels of dropout are applied for each layer, ranging from 0.1 to 0.4, increasing by the layer. The model is trained for 10 epochs to avoid overfitting, with Smooth L1-Loss as the optimization criterion. We have also experimented with starting with a generic ResNet model, but training from scratch resulted in a sharper model, especially considering that the model is used in a custom application in a known setup of few drivers per car.

In order to make the model robust to mask wearing drivers, we have collected data from the same driver with and without a mask, and simply used data from both conditions while training.

Tracking Results

We trained the neural network with over 1000 images obtained from different people under different lighting conditions. One pupil detection example is shown in Figure 4.

Figure 4. Nighttime driving. There are multiple reflections from the glasses in this example. Red dots show the detected pupil positions.

We tested the algorithm under different lighting conditions and on drivers with and without glasses, as shown in Figure 5. The algorithm performed well under various lighting conditions. The IR illumination was adjusted to illuminate the driver's face based on the external light. The last image (on the right-hand side lower corner) shows the condition when the driver is wearing sunglasses during nighttime which is probably an extreme example of pupil obstruction. But IR illumination clearly illuminates the eyes behind the visible light blocking sunglasses. However, if the drivers wear IR blocking sunglasses, the cameras will not be able to detect the pupils and in this case the eye tracking will fail.

Additionally, we tested the algorithm using images from 10 different people whose face data was not included in the training set. The pupil locations were marked manually to establish the real pupil locations. Figure 6 shows the pixel error for 50 example images.

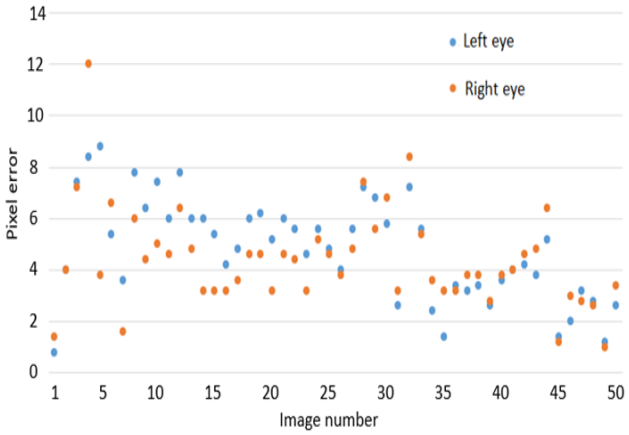

Figure 6. Left and right eye error for 50 images. (In this study, we evaluated images only from one camera to find the pupil location in 2D. However, one can easily employ off-the-shelf routines to find the location of the head position in 3D using images from two cameras.)

Our algorithm generated an average error of 5 pixels as shown in Figure 6. Each pixel in our setup corresponds to ~0.5 mm of spatial resolution for a nominal user-camera distance of ~60 cm. Therefore, our algorithm achieved an average of 2.5 mm error in

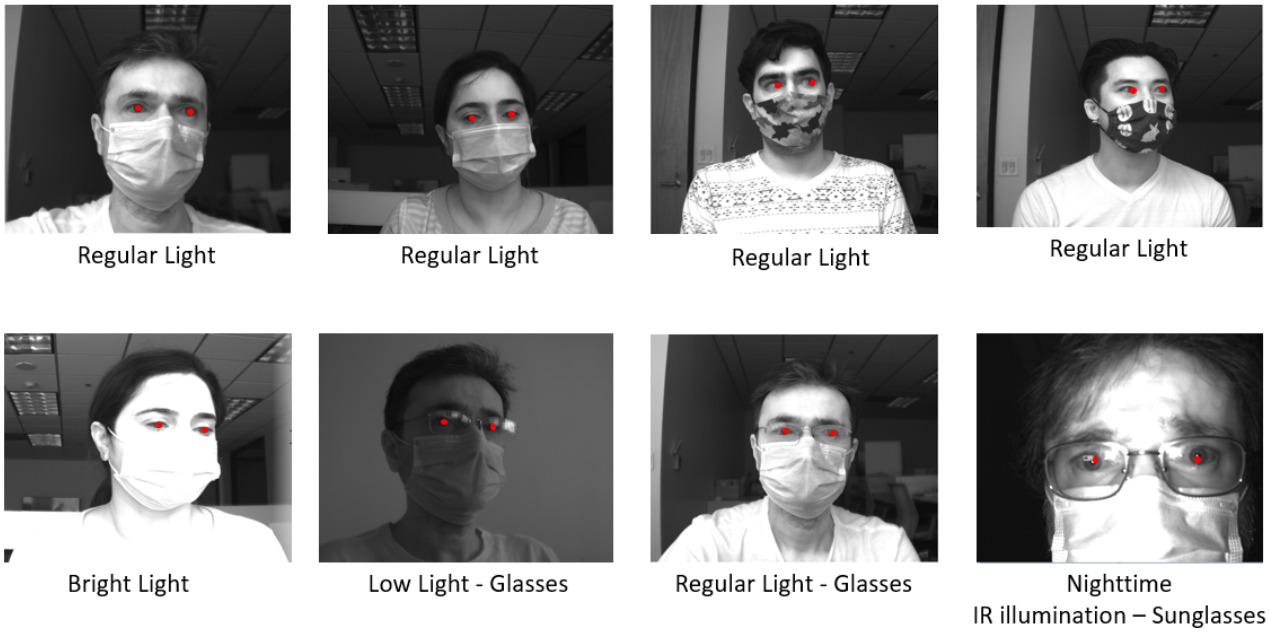

Figure 5. Different lighting conditions and glasses with male and female drivers. Red dots show the detected pupil locations. The above images were obtained in an office environment. However, regular driving conditions are not too different.

pupil detection when the drivers were not in the training data set. The error can be reduced by further optimizing the ML algorithm parameters.

4. Discussion and/or Conclusion Section

AR HUD applications demand accurate pupil location detection at relatively high frame rates. Previously developed driver monitoring systems focus on pupil dilation to predict the drowsiness level of the driver [7]. State of the art eye tracking systems on the other hand, use reflections from the cornea and computer vision methods [8] for accurate detection of pupils. Typical reported frame rates for driver monitoring systems are around 30 fps. Moreover, these methods are not suitable under all driving conditions. Especially, under bright sun light, it is very difficult to differentiate IR reflections from the pupil.

In this paper, we demonstrated ML-based eye tracking for our 3D holographic AR HUD. The algorithm used CNN networks. The training data set included mask wearing users. Our method worked under various lighting conditions. We found that the 750 nm visible filter works well with 850 nm IR illumination. Using IR illumination allowed to detect pupils of drivers that were wearing sunglasses. The pupils appeared as dark spots in the images. Our algorithm can achieve average error of 2.5 mm which can be further improved by tuning the neural network parameters. However, if the same driver data is used in training, much smaller error can be achieved for the same driver.

Another important parameter of eye tracking systems is the frame rate. In this study we achieved 100 fps (~2 msec for image acquisition from the camera, ~8 msec for calculation time) for eye tracking. Typical refresh rate of a HUD display (LCD or DMD based) is 60 Hz or faster for reduced color break-up. For seamless augmented reality applications, the eye tracking should be able to achieve similar speeds to be able to compensate for rapid driver motion while driving.

5. References

1. Urey, H et al. Computational Holographic Displays for 3D AR HUD using Freeform Optics. SID Vehicle Displays and Interfaces 2020, 2020
2. Kazempourradi S, Ulusoy E, Urey H. Full-color computational holographic near-eye display, Journal of Information Display, 20:2, 45-59, DOI: 10.1080/15980316.2019.1606859.
3. Khan A, Sohail A, Zahoora U, Qureshi AS. A Survey of the Recent Architectures of Deep Convolutional Neural Networks. Artificial Intelligence Review, 2020
4. http://dlib.net/
5. https://blog.dlib.net/2014/08/real-time-face-pose-estimation.html
6. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et. al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 2015, Sep., 115(3)
7. Dasgupta A, George A, Happy SL, Routray A, Shanker T. An on-board vision-based system for drowsiness detection in automotive drivers. Int. J. Adv. Eng. Sci. Appl. Math. (April-September 2013) 5(2-3) 94-103.
8. Liu X, Xu F, Fijumura K, Real-time eye detection and tracking for driver observation under various light conditions. Intelligent Vehicle Symposium, 2002. IEEE Volume: 2.