A Novel Approach to Real-time Non-intrusive Gaze Finding

Li-Qun Xu, Dave Machin, Phil Sheppard

Advanced Perception Unit
British Telecom Laboratories
Martlesham Heath, Ipswich IP5 3RE, UK
[li-qun.xu|dave.machin|phil.sheppard]
@bt-sys.bt.co.uk

Abstract

We investigate a holistic approach to real-time gaze tracking by means of a well-defined neural network modelling strategy combined with robust image processing algorithms. Based on captured greyscale eye images, the system effectively learns the gaze direction of a human user by modelling *implicitly* corresponding eye appearance – the relative positions of the pupil, cornea, and light reflection inside the eye socket. In operation, the gaze tracker provides a fast, cheap, and flexible means finding the focus of a user's attention on any of the objects displayed on a computer screen. It works in an open-plan office environment under normal illumination without using any specialised hardware. It can be easily customised to a new user and integrated into an application system that demands an intelligent non-command interface.

1 Introduction

Gaze tracking is an interesting and challenging task across several disciplines including machine vision, cognitive science and human computer interactions [1]. The idea that a human subject's attention and interest reflected implicitly in his eye movements can be captured and learned by a machine which can then act accordingly on the subject's behalf is very appealing and natural, lending itself to many applications, prominently, in video conferencing [2] for focusing on interesting objects and transmitting only these images through the communication networks, design of new generation of *non-command* interface [3, 4] for computers to reach more wide users, and the study of human vision, cognition, and attentional processes [5] among many others. Traditional ways of gaze tracking use the so-called pupil-center/corneal-reflection method [6], which use controlled infrared lighting to illuminate the eye, computing the distance between the pupil centre (the bright-eye effect) and the small very bright reflection off the surface of the eye's cornea to find the line of sight on the display screen, through geometric projections. These methods normally involve specialised high speed/high resolution camera, controlled lighting source,

BMVC 1998 doi:10.5244/C.12.43

electronic hardware equipment, and are sometimes intrusive [7]. The user is often requested to remain motionless during the course of operation. As a result, the eye trackers are mostly used in a controlled laboratory environment for *passively* capturing, recording, and later on, playing back the overlaid time-stamped eye movement trajectories for analysis of fixation and saccade phenomena in connection with various psychophysical experimental tasks.

Recently, increasing demand on intelligent systems opens the need for more convenient, effective and natural ways of communication between people and computers. To a large extent, this requires us to expand the narrow-bandwidth channel from user to computer, which is currently operated through the low speed mouse and keyboard. Along with speech, gestures and other avenues [8], accurate extraction of eye movement information and the wise employment of it have been reckoned to play an essential role in forming such a fast and natural interface, which will have the ability to respond *actively* to a user's natural visual attention, see, for example, the work by Starker and Bolt [9] and Hansen et al. [10].

Our objective therefore is to look into new mechanisms for eye tracking and develop a flexible, cheap, and adequately fast eye tracker suitable for tasks of this nature, using the standard video conferencing equipment on a workstation and without resorting to any additional hardware and special lighting. In this paper, we present a neural network based real-time non-intrusive gaze tracker. The goal of this gaze tracker is to determine where the user is looking (within the boundary of a computer display) by the appearance of his eye images (watched by a monitoring camera). In theory, this task can be viewed as simulating a forward-pass mapping process from the (segmented) eye image space (e.g. the example shown in Figure 4) to a $2D$ coordinate space defined on the screen, though in practice the mapping function is a nonlinear and highly variable one subject to a variety of uncertainties. In Section 2, we describe the methodology and the system that employs a feed-forward neural network to model the aforementioned mapping process for gaze tracking, explaining the techniques used for each key component. In section 3, experimental studies are conducted, detailing the means of collecting correct training data and the strategy of training a large neural network. Section 4 outlines the features of the real-time gaze tracking system. Discussions of some important opening issues are given in Section 5.

2 Methodology and System

There are two primary considerations or *constraints* on which the proposed method and the real time gaze tracking system are based. First, we are concerned with accurate gaze finding at close contact, typically the distance of a user facing a computer screen. In such a case, the appearance of an eye in the view of an observer (e.g. a camera) is believed to contain sufficient cues regarding where the user is looking. Second, this information would be less ambiguous and easier to extract if the user's head orientation generally conforms to the line of sight of his eyes.

The former is testified by our human-human communication experience, while the latter is introduced to avoid the over complicated situation of many-to-one mapping when a user tends to peep at an object on the screen while facing other directions. Despite these two assumptions, the actual relationship between an eye appearance and its corresponding

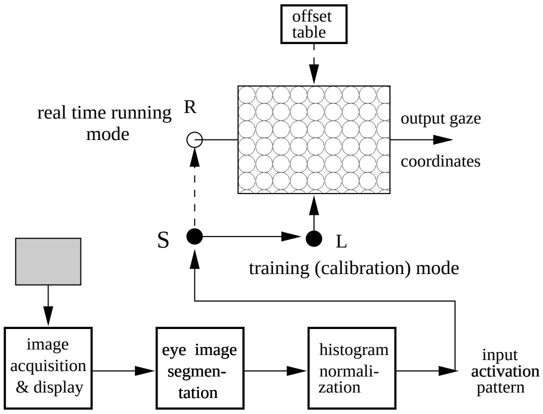

Figure 1: The neural network based gaze modelling/tracking system. The switch **S** is connected to node **L** in the training mode for simulating the forward-pass mapping function, or $\mathcal{F}$: eye images $\rightarrow$ gaze coordinates, and to node **R** in operational mode for real-time gaze tracking.

gaze point is very complicated and highly nonlinear. The complexity arises from uncertainties and noise encountered at every processing/modelling stage, especially, the errors in eye segmentation, the change in depth of the eye images relative to the camera due to head movements, the decorations around the eye such as wearing glasses or pencilled eyebrows, and the change of ambient lighting conditions among others.

We adopt a neural network centered methodology in an attempt to cope with the above problems and to model and generalise this mapping function. Especially, a gaze tracking system has been developed, employing robust image processing algorithms, efficient training procedures, carefully collected training examples and relevant domain knowledge. Figure 1 shows a schematic diagram of the system. We describe below the functions of several main components.

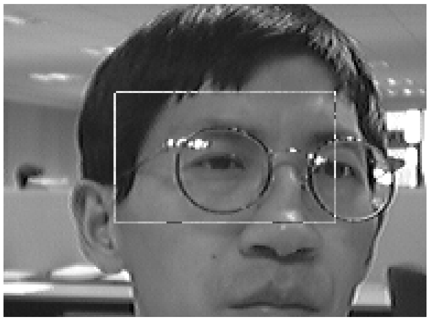

Figure 2: The snapshot of a user's head image (192 × 144) pixels taken in an open-plan office under normal illumination. The rectangular area (100 × 60) pixels defines a search window for eye image segmentation.

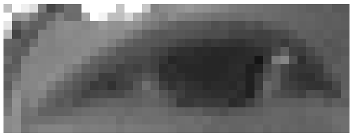

Figure 3: An automatically segmented eye image (40×15) pixels, containing the pupil, cornea, light reflection, and the eye socket.

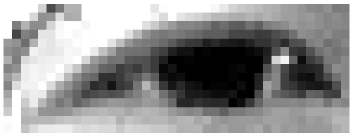

Figure 4: The histogram normalised eye image, showing marked improvements of the normalisation process in delineating important eye appearance features.

2.1 Eye image segmentation

The function of this module is first to detect the small darkest region in the pupil of the eye, and then to segment the proper eye image. For this purpose, a fixed search window of 100 by 60 pixels (shown in Figure 2) is started in the center part of the grabbed image. Inside this window, the image is *iteratively* thresholded, initially with a lower threshold T_0. Morphological filters (dilation and erosion) are used to remove noise or fill “gaps” of the generated binary image which is then searched pixel by pixel from top left to bottom right. Individual objects of pixel clusters are found and labelled using the 4-connectivity algorithm described in Jain et al. [11] A rectangular blob is used to represent each found object. Unless a reasonable number of objects of appropriate size are found, the threshold T_0 is increased by a margin to T_1, and the search process is repeated.

The number of blobs thus obtained are first merged when appropriate based on some adjacency requirements. Certain heuristics are then used to filter the blobs and identify the one most likely to be part of the pupil of the eye in that frame. The heuristics we found useful include:

- the number of detected pixels in each blob, roughly in the range $(15, 100)$,
- the position and value of the *single* darkest pixel in a blob,
- the ratio of the blob’s height to its width, approximately in the range $(0.33, 1.05)$,
- the knowledge of the relative eye position in the face, and
- the motion constraint that the eye movement is smooth and relatively small within two adjacent sampling frames.

A window around the found pupil is then expanded proportionally, based on local information, to the size of 40 by 15 pixels to contain the cornea and the whole eye socket. Figure 3 shows an example of the segmented right eye image.

This segmentation approach is not sensitive to the change of lighting conditions as long as the face is well lit (sometimes assisted by an ordinary desk lamp). It is not affected by the glasses the user wears either. But, the occasional strong reflections off the glasses and the appearance of the frame of the glasses in the segmented eye images due to the head moving away from the camera is generally harmful. They contribute to the burst of random noise disrupting the features in *activation patterns* (discussed shortly) to be sent to the purpose built neural network modelling system.

2.2 Histogram normalisation

The segmented 8-bit grayscale eye image needs to be properly preprocessed to have a value between -1.0 and 1.0 for each pixel. This is necessary for a neural network to discover the features inherent in the image and to learn to associate these features and their distributions with the correct gaze points on the screen, by means of adequate training stages.

This block takes as input the individual 40×15 8-bit grayscale image and computes its histogram that is normally a unimodal shape dominated by a main peak [1]. The lower and upper bounds or t_l and t_u of the histogram are then found. All pixels within the range of 5% of the upper bound are given a value 1.0, and those within the range of 5% of the lower bound are assigned a value of -1.0. An arbitrary pixel (t_p) falling within the 90% part of the bounds assumes a linearised value p between -1.0 and 1.0, or

$$\Delta t = 0.05(t_u - t_l) \quad \text{and} \quad p = -1 + 2\frac{t_p - t_l - \Delta t}{t_u - t_l - 2\Delta t}. \tag{1}$$

Other appropriate nonlinear transfer functions can also be used. The activation patterns thus generated together with associated properly coded output gaze points (discussed shortly) are ready for training a neural network. Figure 4 shows the same eye image as in Figure 3 after histogram normalisation. It demonstrates that the contrast between important features (the eye socket, pupil, the reflection spot) has been largely enhanced.

2.3 The Neural Network Modeller

A neural network shown in Figure 5 is adopted for learning the mapping function $\mathcal{F}$ based on representative training examples. The network has 600 input retina units, each receiving directly a normalised pixel value of the segmented eye image. As in [12], the hidden units are divided into two groups, designed for learning the respective features of input eye images representing the horizontal and vertical gaze cues. The hidden units are then connected to the corresponding group of output units which are specially organised to encode the horizontal and vertical position of a gaze point. All the hidden and output units assume a hyperbolic tangent transfer function $f(x) = (1 - e^{-x})/(1 + e^{-x})$ having an output value between ± 1.

Given a grid matrix, 50 by 40 say, partitioning a computer screen, the position of an arbitrary gaze point in this grid matrix can be a value between 0 and 49 along x direction and between 0 and 39 along y direction, with origin being in the top left corner $(0, 0)$ of the screen. Instead of using the commonly seen '1-out of -N' coding method for representing the desired activation pattern of a gaze point across the two groups of output units, respectively, we have adopted a Gaussian shaped coding method similar to the work by Pomerleau [13] on autonomous vehicle guidance. It is generally agreed that the '1-out of-N' coding method is more suitable for pattern classification tasks which require sharp definitive decision boundaries between different classes, while the mapping function simulation task of this study demands a gradual change in output representations when the data examples (eye appearance) in input data space exhibits slight difference. This preservation of topological relationship after data transformation (mapping) is the main concern in selecting an output coding mechanism.

[1] An eye image whose histogram does not satisfy some desired requirements is rejected as a false segmentation.

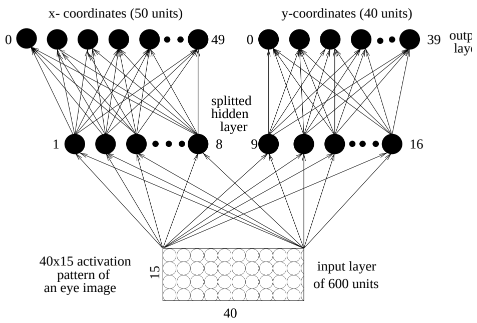

Figure 5: The neural network architecture employed for gaze modelling and tracking. The input retina units receive the normalised activation pattern of an eye image. The hidden units are split into two groups, and connected to the corresponding two groups of output units which encode, respectively, the horizontal and vertical position of a gaze point based on Gaussian coding of output activations. The links as shown are fully-connected.

In the experiments reported later, the Gaussian function used is of the form,

$$G(n - n_0) = -1. + 2. \exp(\frac{-(n - n_0)^2}{12.}), \tag{2}$$

with the standard deviation $\sigma = \sqrt{6}$. A paired (x, y) grid coordinates of a gaze point therefore give rise to two Gaussian shaped output activation patterns, taking values in the range between -1.0 and 1.0, one centered around unit $n_0 = x$ across the group of 50 output units representing horizontal axis and the other centered around unit $n_0 = y$ of the 40 output units for vertical axis. These two patterns concatenated together act as a *desired output* of the neural network system.

In decoding the outputs while testing the gaze tracking system, the Gaussian shaped activation pattern $G(n-n_0)$ is moved across the output units for x-coordinate by changing n_0 from 0 to 49 at a time, a least-square fitting procedure is performed at each unit position trying to match the actual output activation pattern. The peak of the Gaussian shaped pattern that achieves the smallest error determines the horizontal position of the gaze point. In the same way, the vertical position of the gaze point across the 40 output units for y-coordinate is found.

3 Experiments

3.1 Training Data Collection

This is essential to assure that the proposed gaze tracker learns the correct mapping function and generalise to real-time running situations. The procedures are as follows:

In a collection session, the user is prompted to visually track a blob cursor which moves across the screen within the partitioned grid matrix in either the horizontal or vertical zig-zag movements. At the same time, a video camera mounted along the side of the screen continuously grabs the head image sequence. For each frame a small patch of

40×15 pixels image containing appropriately the eye socket is then segmented. This eye image paired with the grid position (x, y) of the travelling cursor at that instant forms a raw training example. During the course of images collection, the user needs to satisfy some constraints as mentioned in Section 2. The segmentation algorithm can detect automatically those unwanted images when the eye blinks occur, and that contain eye brows, nostrils or left eyes.

3.2 Training of the Neural Network

For the data collected in the manner above, and preprocessed and coded according to Section 2, the back-propagation algorithm, see e.g. [14], has been modified to train the neural network. The cost function to be minimised is as usual the summed squared error (SSE). For stopping purpose, however, an evaluation criterion called *average grid deviation* (AGD) is introduced, which measures the average difference in grid units between the current predictions and the desired positions of the gaze for the entire training set excluding a few wildcards due to the user's unexpected eye movements. Note that the SSE and AGD do not always keep the same change directions.

In the following, we discuss a two phase strategy that was used to train this large neural network. It should be mentioned that though the number of training examples, normally between 2500 and 4000, is less than the number of free parameters (weights), 10,426 for the present case, the training strategy makes sure that the set of *effective* parameters are very well tuned, and neither under-fitting nor overfitting of the network occurred.

Started with small random weights each having a value between $-\epsilon$ and ϵ, this strategy consists of a fast search phase followed by a fine tuning phase :

- In the first phase, the network is updated once for every few tens of training examples (typically, U_0 is between 10 and 30) which are drawn *at random* from the entire training data set. A *nominal* learning rate $r = r_0$ and a momentum factor $m = m_0$ are adopted in training, which means that, for each connection weight w_i, the actual learning rate used for updating its value varies, and is much smaller, equal to the nominal learning rate divided by the fan-in of the unit to which w_i is connected. A small offset $\sigma = \sigma_0$ is added to the derivative of each unit's transfer function to speed up the learning process. This is especially useful when a unit's output approaches one of the two saturation limits, -1 or 1, of the hyperbolic tangent function. Besides, for each input training pattern we have added random Gaussian noise corresponding to 5% of the size of each retina input, i.e. the actual input x_a received by each unit is

$$x_a = x_o(1 + 0.05g) \tag{3}$$

  where x_o is the original normalised input and g the Gaussian variable with zero mean and unit variance. This is particularly effective for overcoming the overfitting problem in training a neural network and achieving better generalisation performance. In so doing, the neural network, albeit over ten thousands weights, would always approach a satisfactory solution after between 50 and 80 training epochs.

- In the second fine tuning phase, we update the network weights once after presenting the whole training set. The nominal learning rate r_1 to use is proportionally

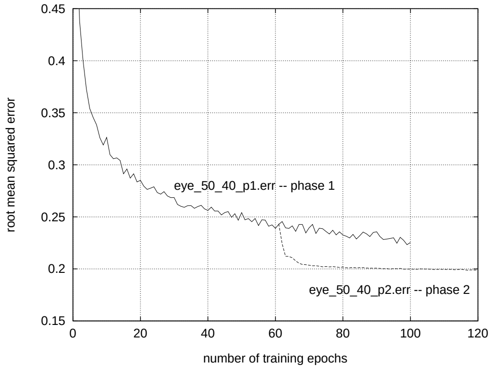

Figure 6: The root mean square errors in phase 1 and 2 versus number of training epochs in one typical training trial of the neural network. Phase 2 started with weights obtained after 60 training epochs in phase 1.

much smaller than that in the first phase, and a slightly smaller magnitude of Gaussian noise, around 3% of each retina input, is used. After about 30 epochs, the system can settle down to a very robust solution.

Repeated training trials, using different initial random seeds, on data examples from several users have demonstrated consistently good performance of the training strategy. A range of values can be used for the parameters without affecting the training performance. In the experiments reported, $\epsilon = 0.1$, $r_0 = 0.4$, $m_0 = 0.6$, $\sigma_0 = 0.05$, $U_0 = 20$, and $r_1 = 0.002$, were used. Figure 6 shows a trial learning result for the user BA. The original data were collected in two horizontal and two vertical cursor running sessions, respectively. The cursor is confined to only travel within the top-right 40×30 area – an application relevant part of the screen – of the entire 50×40 grid matrix. So, each running session can ideally provide $1,200$ data examples. The total number of examples successfully collected for the four sessions is $3,906$, and the number of training examples used in obtaining Figure 6 is $3,000$. The rest 906 examples, which were randomly selected from the whole data examples available, were used to examine the learning performance and to find most appropriate stopping point. Table 1 gives some performance measurements. In this trial, the weights saved at the 60th epochs of training phase 1 are loaded for further refinement in training phase 2. It can be seen that this overall strategy leads to rapid reduction in training error which then settles down to a stable status allowing for no further overfitting of the neural network. In practice, the weights obtained at the end of the $60 + 20 = 80$th training epoch are used to drive the real-time gaze tracker.

4 A Real-time Gaze Tracking System

Once the training (calibration) process is finished and an optimised weight set is loaded into the system, the gaze tracker will be ready to run. It constantly outputs the (x, y) gaze coordinates whenever it captures an valid eye image, or reports a failure when no eye is detected.

The system works in an open plan office environment with a video camera mounted

Training epochs	50	60	+10	+20	+30	+40
r.m.s.(tr)	0.254	0.239	0.204	0.202	0.201	0.200
AGD (tr)	1.432	1.387	1.078	1.057	1.052	1.046
AGD (te)	1.860	1.824	1.616	1.612	1.619	1.619

Table 1: The performance of the neural network versus number of training epochs in one trial. The size of the training set (tr) is 3000 and that of the test set 906. The first two columns (50,60) give the measurements in training phase 1, and the rest show the results in training phase 2 with extra training epochs.

on the right hand side of the display screen to continuously monitor the user's face. There is no additional hardware and special lighting source involved. The user sits comfortably at a distance of about 22 to 25 inches away from the screen, he is allowed to move head freely while looking at the screen, but needs to keep his head within the field of view of the camera and his face within a search window overlaid on the captured head image.

The system now works at about 20 Hz in its stand-alone mode on a SunUltra-1 Workstation. To gauge its performance, the average prediction accuracy on a separately collected test data set is about 1.5 degrees, or around 12 mm apart on the computer screen. One can also test the system interactively by clicking a mouse button while looking at a highlighted grid point, the predicted gaze point will be shown on the screen. As the gaze prediction error is distributed non-uniformly over the screen, an *offset table* is introduced to adjust predictions in those badly performed areas in real-time running situation. The process of acquiring this offset table can be achieved interactively or automatically.

5 Discussion

We have developed a real-time non-intrusive gaze tracking system based on powerful neural network modelling techniques. In contrast with other gaze tracking systems, the current system works in an office environment under normal illumination, without resorting to any specialised hardware and controlled lighting source. The system can be customised to individual users and particular applications. The system now runs at about 20 Hz in its stand-alone mode on a Sun Ultra-1 station, despite that the maximum capture rate of the video card (frame grabber) is 25 fps. It has been developed with a variety of applications in mind, especially, the design of multimodal interface, providing focus of attention in video conferencing, disambiguating verbal as well as nonverbal information.

There are a few issues currently under study, including alternative training techniques to allow for rapid customisation of the system to different users; tracking and modelling head orientation to allow for larger head movements; and modelling user's attentional behaviours such that the interface can make more human-like responsive judgement.

References

[1] B.M. Velichkovsky and J.P. Hansen. New technological windows into mind: There is more in eyes and brains for human-computer interaction. *Technical Report*. Unit of Applied Cognitive

Research, Dresden University of Technology, 1996.

[2] J. Yang, L. Wu and A. Waibel. Focus of attention in video conferencing. *Technical Report, CMU-CS-96-150*, School of Computer Science, Carnegie Mellon University, June 1996.

[3] Jakob Nielsen. Noncommand user interfaces. *Communications of the ACM*, **36**(4), 83-99, 1993.

[4] R.J.K. Jacob. Eye tracking in advanced interface design. In W. Barfield and T. Furness (eds.) *Advanced Interface Design and Virtual Environments*. Oxford University Press, 1995.

[5] W.H. Zangemeister, H.S. Stiehl, C. Freska (eds). *Visual Attention and Cognition*. North-Holland: Elsevier Science B.V.: Amsterdam, 1996.

[6] D. Cleveland and N. Cleveland. Eyegaze eyetracking system. *Proc. of 11th International Forum on New Images*, Monte-Carlo, January 1992.

[7] D. Stampe. Heuristic filtering and reliable calibration methods for video based pupil-tracking systems. *Behaviour Research Methods, Instruments, & Computers*, **25**(2), 137-142, 1993.

[8] R.J. Mammone (ed.) *Artificial Neural Networks for Speech and Vision*. Chapman & Hall: London, 1994.

[9] I. Starker and R.A. Bolt. A Gaze-responsive self-disclosing display. *ACM CHI'90 Conference Proceedings: Human Factors in Computing Systems.*, Seattle, Washington, 3-9, 1990.

[10] J.P. Hansen, A.W. Andersen, and P. Roed. Eye-gaze control of multimedia systems. In Y. Anzai, K. Ogwa and H. Mori (eds). *Symbiosis of Human and Artifact*, Elsevier Science, 1995.

[11] R. Jain, R. Kasturi, and B.G. Schunck. *Machine Vision*, McGraw-Hill and MIT Press, 1995.

[12] S. Baluja and D. Pomerleau. Non-intrusive gaze tracking using artificial neural networks. *Technical Report CMU-CS-94-102*, Carnegie Mellon University, 1994.

[13] D.A. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishing, 1993.

[14] C. Bishop. *Neural Network for Pattern Recognition*. Oxford University Press, 1995.