Journal of Network and Computer Applications (2002) **25**, 179–196
doi:10.1006/jnca.2002.0142, available online at http://www.idealibrary.com on IDEAL®

A neural network based real-time gaze tracker*

Nischal M. Piratla and Anura P. Jayasumana†
Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523, USA

(Received 19 November 2001; accepted 29 July 2002)

A real-time gaze-tracking system that estimates the user's eye gaze and computes the window of focused view on a computer monitor has been developed. This artificial neural network based system can be trained and customized for an individual. Unlike existing systems in which skin color features and/or other mountable equipment are needed, this system is based on a simple non-intrusive camera mounted on the monitor. Gaze point is accurately estimated within a 1 in. on a 19-in. monitor with a CCD camera having a 640×480 image resolution. The system performance is independent of user's forward and backward as well as upward and downward movements. The gaze-tracking system implementation and the factors affecting its performance are discussed and analyzed in detail. The features and implementation methods that make this system real-time are also explained.

1. Introduction

In the process of looking at something, such as an image or an icon on a computer monitor, the eyes are pointed in a certain direction, thus indicating what is being focused upon. Because of physiological constraints, the humans can only observe their surroundings in detail with the fovea in this direction [5]. This feature of the human eye can be exploited for creating a human–computer interface in the place of the computer mouse, and a few researchers have considered it as an integral part of multimodal human–computer interaction [1,6]. In a multimodal interface, the function of gaze tracking systems can be either monitoring or responding based on the user's gaze [9,10]. In essence, the eyes cannot only replace the mouse but also act as smart monitoring and communicating agents.

The existing gaze-tracking systems, with the exception of a very few, demand cumbersome, mountable apparatus on user's head or restrict user's movement. In normal usage, a user makes very frequent head movements in all directions. Thus, the point of focus in the field of view, in our case the computer monitor, depends not just on the user's gaze direction, but also his face inclination, the distance between the eyes and the monitor, as well as many other parameters as outlined in the

*This research was supported in part by a grant from NSF via the Optoelectronics Computing Systems Center at Colorado State University.
†Corresponding author: E-mail: anura.jayasumana@colostate.edu

1084–8045/02/030179 + 18 $35.00/0

following sections. In current systems, special devices are often used to provide explicit information on parameters such as the distance between the user and the object, or to fix parameters such as the face inclination to a known constant value. The use of artificial neural networks (ANNs) allows the system to be more adaptive and can make eye trackers more non-intrusive [3]. This concept is used in a few non-intrusive systems that exist today. However, the accuracy and the real-time operation for large areas of view were difficult to achieve, as it involves feeding a large piece of the image to ANNs during training or usage.

We have developed an ANN-based gaze tracker, which requires no head mount apparatus or skin parameter dependence, making it more robust and user independent. The system, capable of operating in real-time is non-intrusive except for an ordinary spectacle frame. This frame has a strip with black and white bands mounted on it to facilitate the detection of eyes in a real-time manner. The strip also facilitates the estimation of several parameters of interest such as the inclination of the user's face and the distance at which it is from the monitor. This strip can also be replaced by a band that the user can wear. The spectacle frame is used only for the purpose of holding this strip. The point whether one should use face recognition and reach the eyes in the image or use a strip like this is much debatable. Although in the long run, as processing becomes faster, it will be possible to use face recognition, it is unlikely that such systems will become economical for wide-spread usage at least over the next few years. The robustness that comes with the strip is a major advantage over the skin parameter issues that change from user to user. Reference [2] presents a very efficient method of face recognition based on skin parameters. IBM's blue eyes project has a method to locate pupils in the image, but they have restricted image resolution that is captured to 320×240 due to real-time considerations. Blue eyes' pupil finder uses two infrared time-multiplexed light sources composed of two rings of 8 LEDs each, synchronized with the camera frame rate. With only a simple CCD camera, our system is much simpler. The use of a strip provides for real-time operation with low computation complexity, and the total cost of our system is much less than that of many systems that are being used today.

In Section 2, we discuss the existing eye-tracking systems. The parameters relating the user, the monitor and the gaze point, necessary to build an eye/gaze tracker are considered in Section 3. The system setup, the processing of image for extraction of parameters, the algorithms, and the factors that allow the real-time operation of the system are discussed in Section 4. The training procedure is also outlined in this section. Section 5 contains results obtained with the gaze tracker with respect to speed and accuracy. In the conclusion, we present the limitation and ongoing work related to this system.

2. Gaze tracking techniques

Though there are many eye-trackers today, none of them completely satisfy all the desirable usability requirements, especially the free movement of user and

non-intrusiveness. An ideal tracking device must offer an unobstructed field of view with good access to the face and head, make no contact with the subject (user), meet the practical challenge of being capable of artificially stabilizing the retinal image if necessary, offer good resolution (i.e. detect the smallest change in the eye position), offer good temporal dynamics and speed of response, possess a real-time response, measure all three degrees of angular rotation, be easily extendable to binocular recording, be compatible with the head and the body recordings, and very importantly be easy to use on a variety of subjects [8].

Many of the existing techniques for eye/gaze tracking use reflection as the primary criterion. Among these are limbus tracking, pupil tracking, corneal reflection and Purkinje image tracking. Other techniques exist that take advantage of the potential differences among the constituent parts of the eye [4]. In limbus tracking, the boundary between sclera (that is normally white) and iris (which is comparatively dark) is optically detected and tracked. This technique is based on the position and the shape of the limbus relative to the head, so either the head must be held quite still or the apparatus must be fixed on user's head, which makes it very intrusive. Pupil tracking is similar to limbus tracking except for the fact that a smaller boundary between iris and pupil is used for relative measurement. Even with this technique, the apparatus must be held completely still with respect to the head. Pupil tracking does have some advantages compared to limbus tracking. The pupil, being far less covered by the eyelids compared to the limbus, enables vertical tracking, and also the border of pupil with iris, being sharper, provides higher resolution. However, the forward and backward movements are not accounted for in these systems.

The Purkinje images are the four reflections that occur on the boundary of eye lens and cornea when light (usually infrared) is shone on user's eye. These four reflections are from air–cornea interface, liquid–cornea interface, liquid–lens interface, and lens–aqueous humor interface. The disadvantage with this technique is that the fourth Purkinje image is rather weak, so surroundings lighting must be heavily controlled and also as in the previous case lateral movement is not allowed. The user is required to place his chin on an apparatus throughout the usage period. Another technique [3] uses corneal reflection with image of the eye (typically of size 40 by 15 pixels) centered at glint, to feed an ANN and the output of the network is a set of display coordinates. This is non-intrusive by nature. However, this technique needs a good view of the eye and the presence of reflection by cornea at the same time, which is difficult to attain unless the eye is in the line of light. The size of neural network is also very large as can be seen by the size of the input set (600 units). Though other techniques like wearing special contact lenses [4] provide more accurate tracking, they need wiring apparatus, which complicates non-laboratory usage. There is also a high risk of high-frequency electromagnetic fields affecting the health of the subject. However, the above methods do provide a clear set of parameters that are very useful in implementing gaze-tracking systems.

3. Eye movements

The information required for tracking the gaze includes the pupil position, and the head position in the three-dimensional space. It is also necessary to track the variation of these parameters while the system is in use. One changes the head position very often while sitting in front of a computer. There is no one to one relation between the above parameters and the gaze point, and these parameters differ from subject to subject as well.

Eye movements are caused by several factors other than saccades that are the principal method for moving the eyes to different parts of visual scene. Saccades are fast conjugate changes of eye position between fixations. Convergence motion, i.e. the variation of the distance between eyeballs, depends on how near the visual attention target is, i.e. the eyeballs move closer to each other as the target becomes closer (target being of same size). The motion of eyeball within the eye-socket, which can span the two extremes (left and right) in horizontal direction as shown in Fig. 1(a), is an essential parameter for horizontal tracking. When the user looks down, the eyelids normally cover most of the irises. Pupil is an inner circular area that is quite small compared to the iris and does not get covered by the eyelids completely. Thus, pupils are preferable for tracking the eye motion compared to eyeballs as a whole. Moreover, the vertical motion of the pupils is not as large as its horizontal motion. Vertical tracking is quite difficult in these cases except for the large vertical movement of eyes that happens only if we look far up or far down from the line of sight with zero inclination. We address this problem by considering the amount of opening of eyelids associated with the vertical motion. This motion of eyelids was not acceptable in previous systems as the eyelids partially covered the eyeball, the image or the horizontal displacement (in pixels) of which was used to track the gaze. Eyelid movements in vertical direction as seen in Fig. 1(b), provide a wider range of values corresponding to the vertical motion than that can be obtained just by monitoring the pupil movement.

The movement of the head affects the effectiveness of the gaze-tracking system more than any other parameter. All the other parameters can be tracked only if the head is fixed or there is a reference with respect to which the variations of iris/pupil

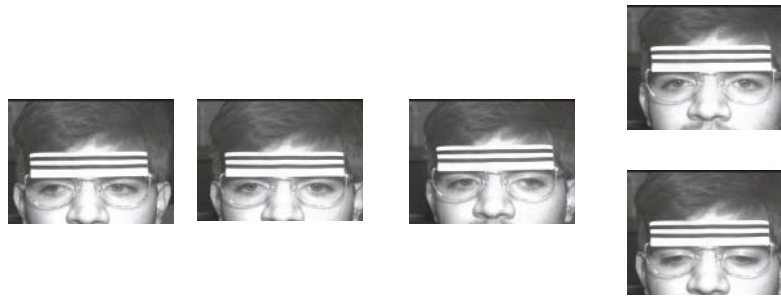


Figure 1. The Variation of (a) eyeball position during horizontal motion, with user looking right, straight and left, and (b) eyelid position during vertical motion with user looking up and down.

movement can be computed. Initial gaze-tracking systems, e.g. limbus tracking, did not allow for the head movement. Some multimodal systems could handle this problem successfully [13] based on statistical color models, which are used, for example, in face recognition to obtain a frame of reference. In our system, this frame of reference is created using a strip fixed on a spectacle frame worn by the user. It also makes it convenient to account for tilt and user's distance from the screen as discussed later.

4. System setup and implementation

The neural network based gaze-tracking system consists of a computer with Pentium (II) CPU chip of 266 MHz clock, a Coreco frame grabber (model: F/64 ISA) placed in EISA slot of the system, a Pulnix progressive scan CCD camera (model: TM 9701) interfaced to the frame grabber board and a lamp (Fig. 2). The choice of the equipment is made targeting at low-cost and embedded systems. The captured image of the user is shown on the monitor in this figure, which is the case during sample gathering for training, but not under normal usage phase of the eye-tracker. The frame grabber grabs the image of the subject sitting in front of the system at 30 frames/s, through a CCD camera placed over the monitor. A small table lamp that is placed above the monitor is used to illuminate the edge between iris and pupil that is difficult to detect under poor ambient light. If there is enough illumination, this lamp can be removed from the setup. It also ensures that we have some degree of consistency in background lighting during different times of the day. Another option

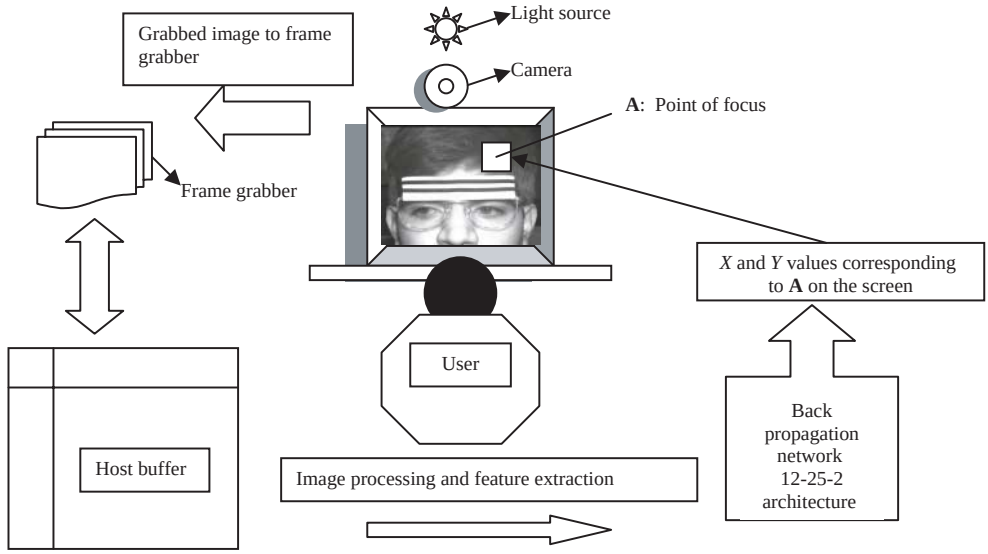


Figure 2. The gaze-tracker system setup.

was to keep the camera below the monitor, i.e. on user's desk. But when the camera is lower than the monitor, the plane of which an image is grabbed is more inclined when compared to the earlier option. This would make the input–output relation more non-linear and the computation of inputs more complex.

The steps the system goes through are illustrated in Figs. 3 and 4, and listed below:

(i) The camera grabs an image with resolution 640×480 of the subject sitting in front of the system.

(ii) The image, which consists of the subject with spectacle frame along with the strip over it (Fig. 3), is processed to detect the black and white bands on the strip and the edges of the strip at lower ends. The coordinates of these edges are the first four features for the neural network. See points $P_1(X_{P_1}, Y_{P_1})$ and $P_2(X_{P_2}, Y_{P_2})$ in Fig. 3.

(iii) With an offset from the computed edge points of the strip, two windows (W1 and W2) are selected as shown in Fig. 3, for detection of eyeballs and computation of the center of the each of them, as well as computation of displacements of eyelids from these centers. The eyeball center's coordinates $P_3(X_{P_3}, Y_{P_3})$ and $P_4(X_{P_4}, Y_{P_4})$ as shown in Fig. 3, are the next four input features for the neural network. Eyelid distances are measured only along the vertical line through eyeball centers, thus giving displacements from these centers. They account for four values $Y_{P_3\mathrm{U}}$ (upper eyelid for eye in window W1), $Y_{P_3\mathrm{L}}$ (lower eyelid for eye in window W1), $Y_{P_4\mathrm{U}}$ (upper eyelid for eye in window W2) and $Y_{P_4\mathrm{L}}$ (lower eyelid for eye in window W2) (Fig. 3).

(iv) These twelve values $(X_{P_1}, Y_{P_1}, X_{P_2}, Y_{P_2}, X_{P_3}, Y_{P_3}, X_{P_4}, Y_{P_4}, Y_{P_3\mathrm{U}}, Y_{P_3\mathrm{L}}, Y_{P_4\mathrm{U}}, Y_{P_4\mathrm{L}})$ obtained by processing the subject's image stepwise are the twelve inputs to an ANN. The X and Y coordinate pair of the point the user is looking at on the screen (A in Fig. 2) is the output of the neural network. Treating this point as a center, a window of size 80×80 is popped up, which almost all the time covers the actual point of focus.

High-speed access to the memory has been a key concern in the design of frame buffers [9]. Real-time systems dealing with large images are mostly slowed down

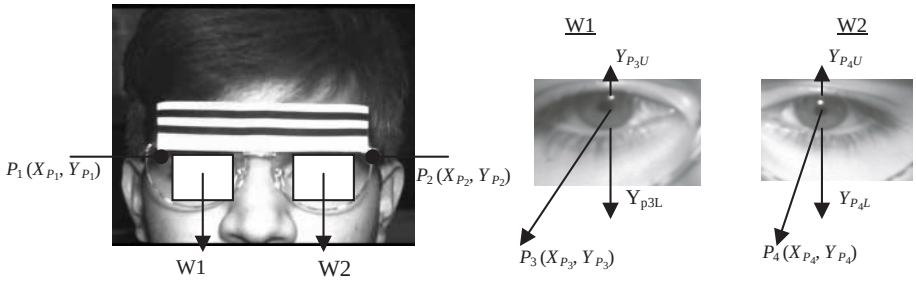


Figure 3. Processing stages and the inputs for the neural network.

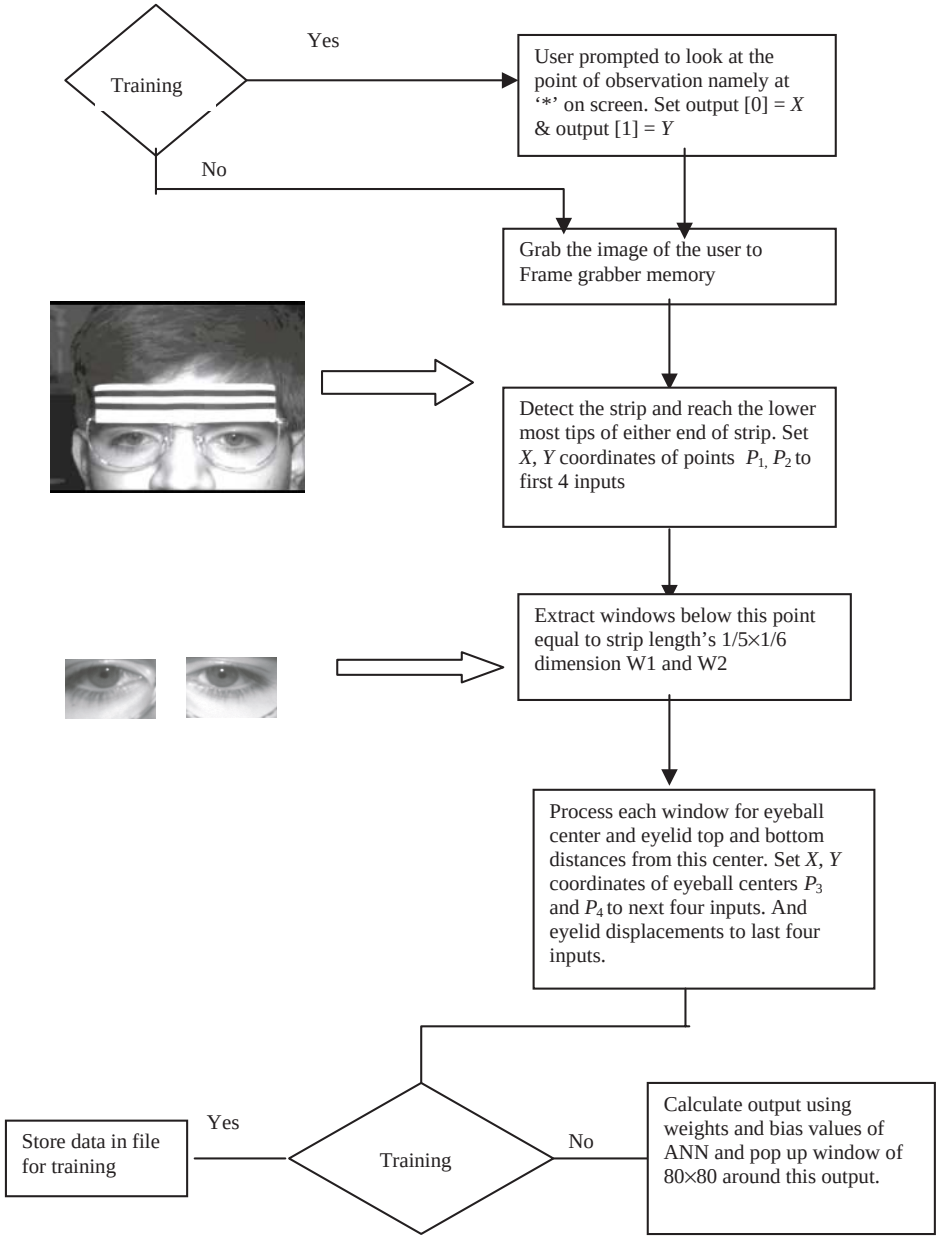


Figure 4. Flow diagram of the system implementation.

due to the interrupts generated while accessing a frame buffer. Therefore to make the system real-time, only the parts of image, which needed to be processed, were selectively transferred to the host memory. There was almost a 90% reduction in time when the image was processed in host memory after the transfers compared to processing in the frame buffer.

4.1 *Computation of reference points*

Normally, while using a system, a person moves from left to right or *vice versa* quite frequently. Such an important feature is taken into much less consideration in many of the eye tracking systems. This movement of head causes the eye to fall anywhere in the grabbed image of size 640×480. The position of the eyeball cannot by itself be a tracking parameter, i.e. we cannot say that the subject is looking at the extreme left or the extreme right just because his/her eyeballs are at the corner of the eye-sockets. The motion of the eyeballs should be taken with reference to the position of the head. This reference should provide a measure of where user's face is with respect to the center of the monitor. The strip, which was discussed earlier, provides this reference and the edge points of this strip provide the reference points. This strip contains white–black–white–black–white bands so that it can be detected and processed easily for edge points.

The *X*–*Y* format gray scale image of size 640×480 is too large to process for these reference points. Therefore, vertical line arrays of the image spaced by a fraction of the width of the image ($1/6 \times 640$ pixels in this case) are transferred to the host buffer. This fraction is reached by moving towards and away from the monitor. It is observed that if the part of the image containing the user's face is less than 1/3rd of the total image, he is much farther from the monitor and presumably, not using the system. However, this fraction is configurable. The buffered gray level image lines are then searched from top to bottom for band shifts of white and black using a technique similar to run length coding. If first line array does not show any such pattern then a second line is transferred to host buffer and this may be repeated till the sixth line array.

The search process is quite simple due to the fixed geometric pattern of the strip. A simple algorithm traverses the strip vertically and jumps from band to band as each band is detected. The first band detection determines the size of jump. There are five bands and a failure at any jump, triggers the search process again from that point. The choice for the number of bands was set to five because of the possible white–black–white in the surroundings. Five such changes are very rare in the ambiance. Selecting seven bands increases the time of processing and moreover five are sufficient to meet the requirement. The first band is chosen white so that the algorithm does not spend lot of time processing hair of the subject, assuming majority have higher gray level hair. The last strip is white to avoid the detection stopping on eyebrows with the similar assumption.

To avoid failure in strip detection due to unusual surrounding light conditions or imperfections in the strip, even after six vertical line arrays are processed, the algorithm is rerun, with an offset of 10 pixels from the first line of the array. Once the last white band is detected, a horizontal line array spanning the image at this vertical level is transferred to buffer. This buffer is then scanned in horizontal directions to reach the band edges i.e. points P_1 and P_2 in Fig. 5(a). To ensure success, a second scan and third scan are performed by taking another horizontal

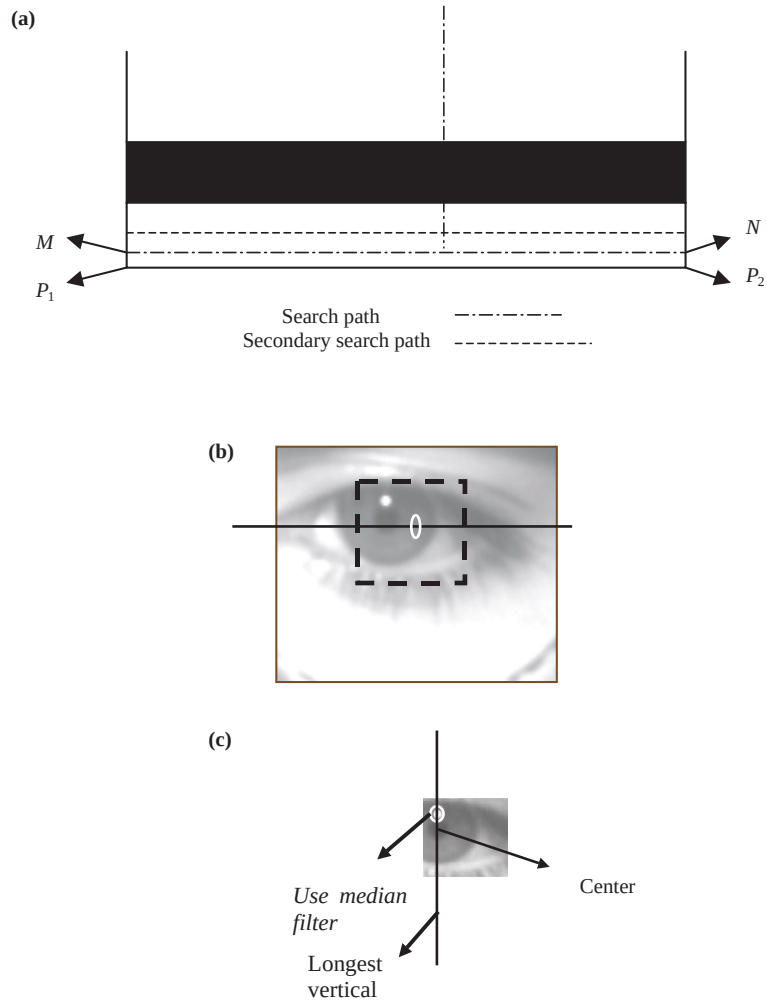


Figure 5. (a) Computation of the reference points; (b) darkest horizontal computation; (c) center of pupil computation.

line array of pixels that are at a distance of 5 pixels from the first, as indicated by the dashed line in Fig. 5(a), leading to points *M* and *N*. With a tilted face, sometimes the white may not stay horizontal and in these cases second or third scans help reach the edges.

4.2 *Computation of eyeball center*

The eyeballs are always below the reference points in the image. Two windows W1 and W2, the size of which corresponds to a fraction of the length of the strip, are selected. The windows are transferred (Figs. 3 and 4) into the host buffer for processing. It is observed that, a window with a horizontal length of 1/6th the length

of the strip and vertical length of 1/5th the length of the strip will cover the eyes and eyelids even for a person with big eyes. A margin is left at the top and sides to account for the shadows caused by the strip. There is no loss of information due to this selection. The fact that pupils are always darker on a gray scale compared to the skin is used for processing of these buffers. These windows are searched for the highest cumulative value of darkness in horizontal level and the center of mass of the darkest horizontal line falls near the eyeball as in Fig. 5(b). Ideally it should be the center. But due to the noise in the image and the reflections, this is not always true. Hence a smaller window is chosen around this center of mass for further processing. Once again, to account for the adaptive nature, the strip length is used in computing the size of this smaller window. On an average, an image of pupil falls inside a square window of edge length equal to 1/300th the strip length. To make the system more user independent and robust for all subjects, a window of twice this size is chosen around the center of mass for center of the pupil detection. In this window the darkest vertical image line is found. The mid-point of this darkest vertical line is the center of subject's eyeball in the image as indicated in Fig. 5(c).

4.3 *Computation of gap between eyelids*

Upper and lower eyelids have a varying distance between them along the periphery of the eye; thus the measurement of gap between eyelids also needs a reference so that this value can be used as a reliable parameter. We took X-coordinates X_{P_3} and X_{P_4} at the eyeball center as vertical references in windows W1 and W2, respectively. Eyelids always stay over the eyeballs when a user is looking at the monitor, irrespective of the point of focus on the screen. Therefore, if we move away from the center of pupil in the image, the end of darker (i.e. lower) gray level values of eyeballs is the indication of beginning of eyelids of a subject. The eyeball's gray level value depends on the subject and the ambient lighting conditions.

Moreover, the lamp that is placed over the monitor for illumination to differentiate pupil from iris of that pupil, causes reflection on the eyeball. Therefore, before these beginning points of eyelid peripheral are computed, a 3×3 median filter is applied to the smaller windows, in which pupil centers were detected as described in Section 4.2. The median of gray level values for these small windows is approximately equal to the gray level values of eyeballs, as most of the area of these windows is covered by the eyeball in the image. The computation of eyelids is done by traversing in the image memory array along the vertical reference, in both directions, till a gray level less than or equal to the median value of respective smaller window ceases to exist.

4.4 *Training of the ANN*

A resolution of 640×480 is used for the monitor display to map the outputs, making the possible outcomes of the neural network around 307 200 in number. Although, 640×480 is referred to here, the resolution (in cm) of the eyetracker is independent of the monitor resolution. To account for the non-linearity, a 3-layer back

propagation network (BPN) with a log-sigmoid activation function was used. A 25 neuron hidden layer is used in between the two output neurons for horizontal and vertical coordinates on the screen and twelve inputs discussed earlier.

The system being discussed works for up–down, left–right and front–back movements of the user. To supervise learning and to cover all these possibilities, the subject is required to give training samples involving all these motions. During this phase, the captured image is displayed as shown in Fig. 6(a). The monitor is divided into nine parts and the subject has to provide training samples by keeping the image of the strip center on the spectacle frame somewhere in the region for all of these nine regions. This covers up–down and left–right movements. For front and back movements the user may provide three to five different front and back movements keeping the strip in one of these nine pseudo regions.

During the training process, a point of attention, indicated by * in Fig. 6(b), is popped on the screen. Fig. 6(b) shows more than one point to illustrate the place-holders for these points. However, these points are popped only one at a time during the sample gathering process. The user looks at the point and the image is grabbed. The point moves to a new position with every sample taken. Samples are taken with user gazing at different cross points of a grid with 40 by 40-pixel spacing.

For each of the nine pseudo regions we have 12×16 samples, and for each sample we have 12 inputs as discussed earlier. Due to the real-time processing achieved by buffer transfers, the inputs can be processed very quickly. The frame rate is 30 and we can get 30 samples in a second. However, it will not be of much

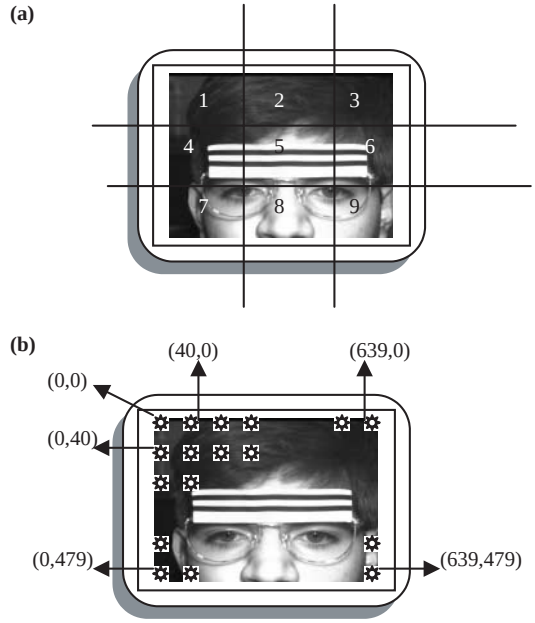


Figure 6. Sample collection for training (a) the nine pseudo regions for frame placement, and (b) the grid of gaze points spaced apart by 40 pixels horizontally and vertically.

use to see the user's gaze point for as much as 30 times in a second for PC applications. We took samples every 200 ms, i.e. 5 times a second to make sure the gaze point info is retained. For a given strip position in one of the nine regions, the collection of samples as discussed above takes approximately 1 min. But to make the system more robust and perform to account for head motions of the user, we take more samples with different head movements. Therefore for nine different strip positions (head moving up, down and left and right) and five front and back movements, it takes approximately 12 min. This would be the real conservative approach. The user has an option to add samples for the strip positions after a break, i.e. he/she can provide one set of samples for one strip position and provide the next set sometime later. Also for one strip position the strip need not have exactly the same position for all the points. The user can move a little to left or to right, as long as the strip center is in that pseudo region.

5. Analysis and results

5.1 *Efficiency of the algorithms*

Though the identified tracking features are the ones that are available in the image without any spectral or other kind of processing, it is still hard to detect them from the user's image. The strip detection is a simple search for a known geometrical pattern and hence requires a very little computational effort to reach the edge points. The detected edges as discussed earlier are the true bottom corners of the strip in the user's image. However, the detected eyeball centers may not be the exact centers at times. The algorithms are dependent on the distribution of gray levels, which in turn depend upon the ambient light variation in the field of view of the CCD camera. This variation can result in multiple light distribution patterns over user's face resulting in a complex grabbed image in term of gray level variation. Non-uniform distribution of light causes shadows and partially darkened images. The lamp over the monitor does help to provide a uniform pattern but some variation in light still exists due to shadows of the surrounding objects.

Fig. 7 contains a graphical representation of the errors of eyeball center estimation, and eyelid edges, i.e. the difference in the values obtained using the algorithm and the actual eyeball centers and eyelid peripherals measured physically. The error is shown in terms of pixels. The dashed lines in these graphs are the actual points in the user's image and normal lines show the points detected by the algorithm. During the extraction of the detected values for these figures, a point of attention was moved along horizontal direction for variation in X values of eyeball center, and for eyelid displacement one such point is moved in vertical direction. In Fig. 7, first four graphs correspond to the eyeball movements and the next two correspond to upper eyelid displacements. The mean and standard deviations in absolute error in computation of features for 100 random points are listed in Fig. 8.

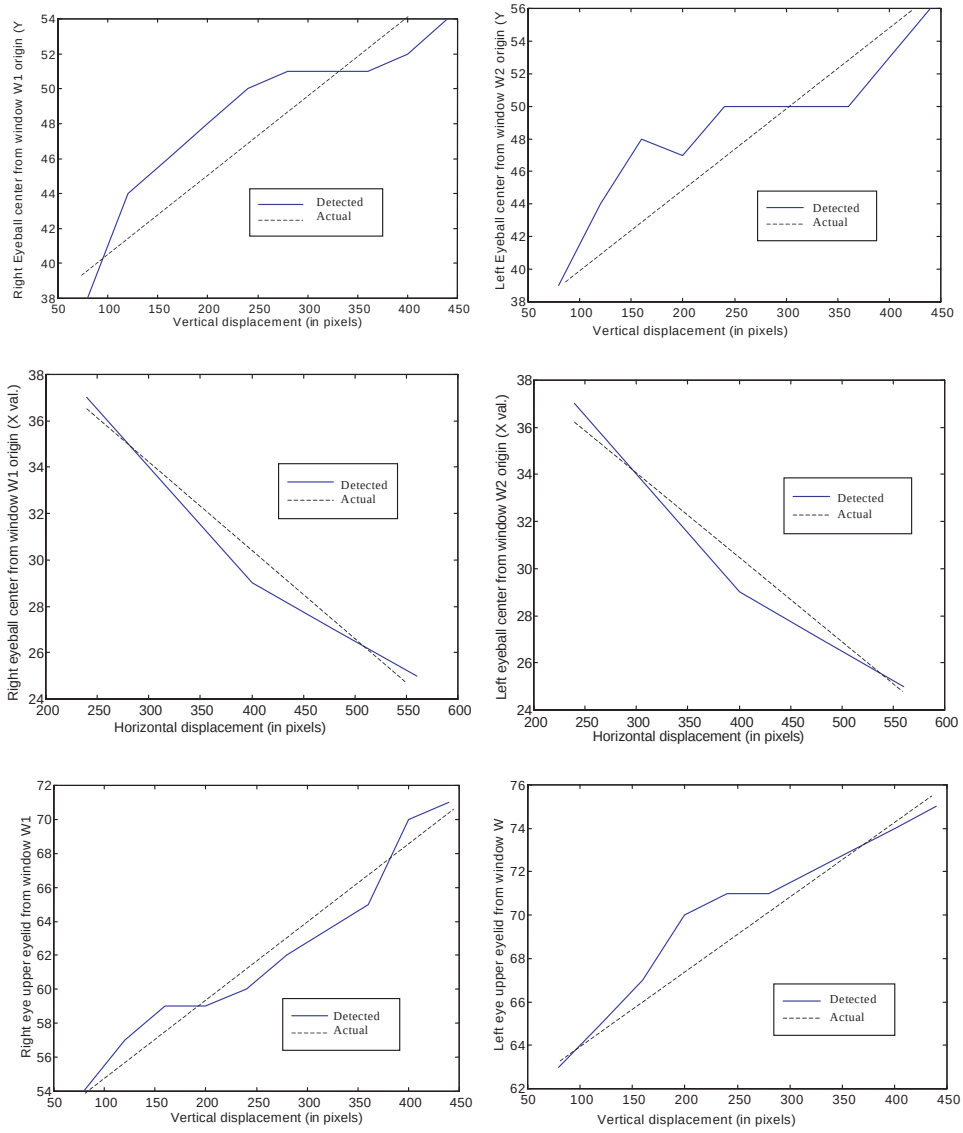


Figure 7. Accuracy of algorithms used for computing eyeball centers and eyelid displacements.

From these graphs it can be observed that the vertical values, i.e. *Y* values have a higher error. The user's eyelid peripheral is not a discrete line and there is a gradient of gray level near this peripheral. This gradation causes an error in the detection of the eyelid displacements. Also, the eyelid displacements have a range, which is much lower than that of the value for center of the eyeball, i.e. the amount of movement between eyelids is less than the eyeball movement for the same distance, say 40 pixels.

Feature	Mean Error	Std. deviation
Right pupil center *X* value	1.57	0.70
Right pupil center *Y* value	1.75	0.71
Right upper eyelid displacement	1.7	0.75
Right lower eyelid displacement	1.25	0.64
Left pupil center *X* value	1.32	0.58
Left pupil center *Y* value	1.67	0.70
Left upper eyelid displacement	1.60	0.68
Left lower eyelid displacement	1.30	0.59

Figure 8. Mean and standard deviations in the computation of features for 100 random points.

5.2 *Experimental results*

After approximately 16 000 iterations, when there is no change in the third decimal of sum-squared error and sum-squared error is around 0.3, the training process is terminated. With the weights and bias values obtained after the completion of the training, for a given input, the neural network provides the point of focus that falls almost accurately in a 80×80 window. The size of the window explains that there can be a $+40$ to -40 error in X and Y values of the output, which correspond to a point on the monitor. It is quite interesting to note that a window of 80×80 pixels i.e. 2×2 square inches is very small compared to the actual size of the screen.

All the above results have been verified by having three users, A, B and C, evaluate the system. User A, a dark eyed male provided training samples for all nine pseudo regions, i.e. the conservative approach as discussed in Section 4.4 and also one sample set by moving far away from the screen. User B is a brown-eyed female who provided training samples for all the nine pseudo regions but none for forward and backward movements. User B's eyes are also larger in size compared to an average user. This size variation and color variation, subjected the system to the test of robustness with a variation in the size of the user's eyes and gray scale of eye image in the system. User C provided training samples only for 6 pseudo regions but none for forward and backward movements. The effect of having less number of sample sets was evident in the results obtained for user C. All the users started with a distance of approximately 40 cm and moved freely in all directions.

After the training of the neural networks, the users A, B and C used the system with their strips in random regions, i.e. they were allowed to move their head in all regions irrespective of whether they have had training sample sets in that region. The distance from the screen was anywhere between 30 and 60 cm.

Each user was asked to look at random points on the screen and remember this point of actual focus. Then the user's image was grabbed while looking at this point. The image is analyzed for features using the methods we discussed in earlier

sections. But this time the inputs were fed to the neural network, whose weights were drawn from the results of corresponding user's training sample sets. The outputs of the neural network are X and Y coordinates on the screen that corresponds to the user's point of focus based on learning performed by previous training sets. These outputs X and Y were then used to pop up multiple windows as shown in the Fig. 9. The user was then asked to identify the actual point of focus (which he looked at while image is being grabbed) on the screen. If the point was within the 80×80 size window then it was treated as the one, belonging to this window and all higher windows. If the point was in 120×120 size window but not in 80×80 window then it was treated as one belonging to 120×120 window and higher but not 80×80 and similar classification was followed for 160×160 and 200×200 size windows. In this analysis we considered 200×200 to be the maximum acceptable size and rejected the points, which had an error higher than $+$ or -100 and fell outside this window. These high magnitude error points of focus were treated as outputs corresponding to system failure.

Each user tested the system for 100 random points on the screen and compared them to the computed point of focus, after identifying the actual points of focus in these windows. The results were interesting and are tabulated below in Fig. 10.

From the tabulated results, we observe that user A got the best results compared to user B or C. From this, we can state that the user who provides samples in a conservative manner can expect best results. It can be recollected that user A gave

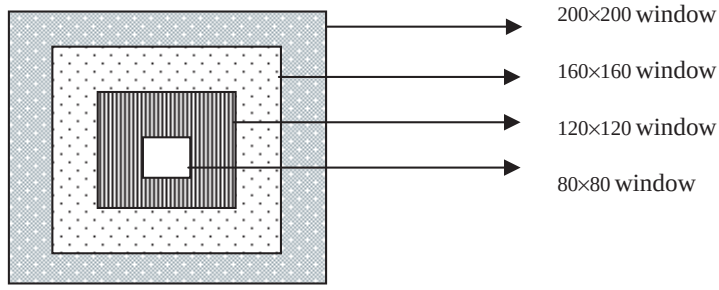


Figure 9. Multiple windows that are popped up on the screen during system use.

Window Size/	80×80	120×120	160×160	200×200	Failure
User A	84 %	95%	99%	100%	-
User B	78%	96%	98%	98%	2%
User C	72%	79%	82%	90%	10%
W/o System	2.1%	4.7%	8.3%	13%	-

Figure 10. Summary of the results obtained for Users A, B and C and for the case where there is no eye tracking, i.e. with random guessing.

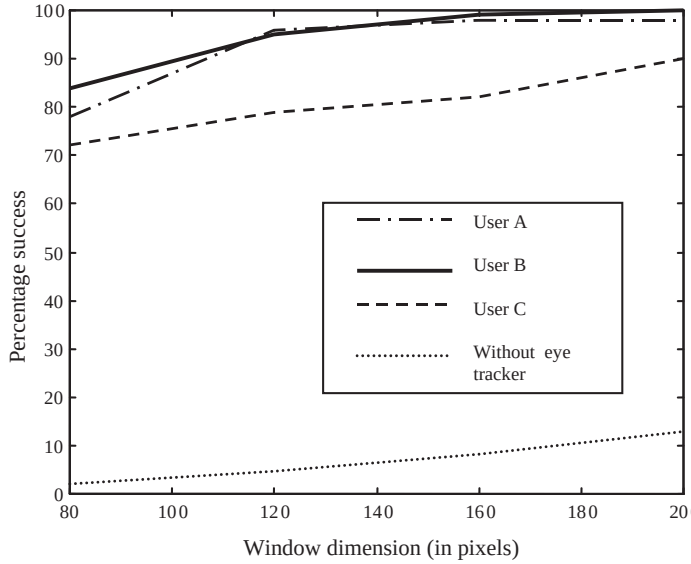


Figure 11. Window size vs Percentage success for users A, B, C, and for the case where eye tracker is not used (i.e. random guessing).

samples in all nine pseudo regions and for one backward movement. User B has given all nine pseudo regions but none for forward or backward movements. User C provided just six pseudo regions and ended up finding only 72% of the points in the 80×80 window. However, if a user feels that he does not move side-wards while using the system, she/he can achieve very good results with only a few training samples. This will also reduce the number of iterations for convergence (around 5000 as seen in user C's case).

To evaluate the effectiveness of the gaze tracker, we compared it with random guessing, indicated by 'w/o system' in Fig. 10. The probability that a point lies in a given 80×80 window at random is very low. Let the screen be divided into 80×80 windows then we have 48 ($640 \times 480/80 \times 80$) such windows and the probability of a point falling in one such window at random is 1/48, which is approx. 2.1%. This goes to 4.7% for 120×120 window, 8.3% for 160×160 window and 13% for 200×200 window. To summarize in Fig. 11, the effectiveness of gaze tracker for users A, B and C are against window sizes. However, it should be remembered that A had a more complete training set compared to that provided by B, and B provided a better set than did C. The case in which the eye tracker is not used is also shown in Fig. 11.

6. Conclusion

We have presented the implementation of a non-intrusive gaze tracker using neural networks. Feature identification and extraction for gaze tracking uses simple image

processing techniques thus making it possible to do gaze tracking in real-time. The features are then fed to an ANN and the network learns to work during normal operation generating the point of focus of the user.

This technique, as observed from results, not only accommodates the motion in user's head, but also does it accurately in real-time. One does not need to detect the face for reaching the eyes as the strip mounted on the frame worn by the user provides a reference. The strip not only helps make the system more general purpose, but also makes it more robust, i.e. if two faces are in the field of view the one with reference frame is treated as the user. The size of the window 80×80 is quite small as far as the screen is concerned, but it is larger than any typical icon. A smaller window size with better accuracy can be achieved by using higher resolution cameras instead of 640×480. The real-time tracking of the system also makes it useful for everyday usage at work. The lighting does not matter once the parameters are detected making it light independent for reasonable lighting conditions. However, system cannot provide an output if user is out of the field of view of camera. Our successful real-time implementation using a low-end PC is an indication that the computation complexity of this gaze tracking system is such that it can be implemented on a state-of-the art micro-controller, thus lending itself to many low-cost applications.

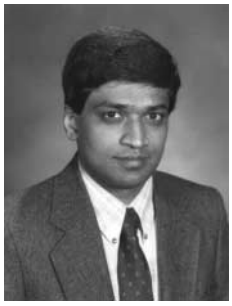
Though the system is developed by keeping the mouse replacement in mind, the potential applications of this system are numerous. To list a few, it may be used as a mode to interface with computer for physically challenged. There is also empirical evidence that eye tracking based interface actually can be quite faster than traditional interfaces [12]. This system can also be used to implement smart displays, which can be tailored to user's current point of attention, for example, to provide higher resolution within the field of focus. Within an application such as a telecubicle, where the displayed data is accessed over a network, this can help provide the user an apparent resolution far higher than what would be displayed at uniform resolution. Future has much in store for eye trackers once they become much more accurate while staying non-intrusive.

References

1. H. Ando, Y. Kitahara & N. Hataoka 1994. Evaluation of Multimodal Interface Using Spoken Language and Pointing Gesture on Interior Design System. In *Proceedings of the ICSLP'94*, Yokohama, Japan, September 1994, Vol. 2, pp. 567–570.
2. V. Bakic & G. Stockman 1998. Real-time Tracking of Face Features and Gaze Direction Determination. In *Proceedings of the WACV '98, Fourth IEEE Workshop on Applications of Computer Vision*, pp. 256–257.
3. S. Baluja & D. A. Pomerleau 1994. Non-intrusive Gaze Tracking Using Artificial Neural Networks. In *Advances in Neural Information Processing Systems*. San Mateo, CA: Morgan Kaufmann Publishers, pp. 753–760.
4. J. Gips, P. Olivieri & J. Tecce 1993. Direct Control of the Computer Through Electrodes Placed Around the eyes. In *Proceedings of the Fifth International Conference on Human*–Computer Interaction, Orlando, FL: Elsevier, pp. 630–625.

5. A. J. Glenstrup & T. Engell-Nielsen 1995. Eye Controlled Media: Present and Future State, Thesis with the Department of Information Psychology at the University of Copenhagen, Denmark, May, pp. 2–3.
6. S. Nakagawa & J. X. Zhang 1994. An Input Interface with Speech and Touch Screen. *Transactions of the Institute of Electrical Engineering of Japan* **114-C**(10) 1009–1017.
7. N. M. Piratla 1999. Design and Performance of Non-intrusive Gaze Tracker, MS Thesis, Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO.
8. D. Scott & J. M. Findlay 1992. Visual Search, Eye Movements and Display Units, Human Factors Report. University of Durham, UK.
9. R. F. Sproull 1987. Frame-buffer Display Architectures. In *Readings in Human–Computer Interaction* (R. M. Baecker & W. A. Buxton, eds.). Morgan Kaufmann Publishers, Inc, pp. 342–355.
10. R. Stiefelhagen & J. Yang 1997. Gaze Tracking for Multimodal Human–Computer Interactions. In *Proceedings of the ICASSP' 97*.
11. M. T. Vo & A. Waibel 1993. A Multi-modal Human–computer Interface: Combination of Gesture and Speech Recognition. Adjunct *Proceedings of the Inter CHI'93*, Amsterdam, The Netherlands, April 26–29.
12. C. Ware & H. H. Mikaelian 1987. An Evaluation of an Eye Tracker as a Device for Computer Input. In *Proceedings of the SIGCHI + GI' 87*, Toronto.
13. J. Yang, R. Stiefelhagen, U. Meier & A. Waibel 1998. Visual Tracking for Multimodal Human Computer Interaction. *Summary CHI 98*, pp. 353–354.

Nischal M. Piratla received his BTech degree in Electronics and Communications from Jawaharlal Nehru Technological University, Hyderabad, India in 1996 and MS degree in Electrical Engineering from Colorado State University, Colorado, USA in 1999. He worked as a R&D Engineer in CMC (India) Ltd. (1996–97), Seagate Technology (1999–2000) and Avaya Inc. (2000–2002). He is currently pursuing his PhD program at Colorado State University. His research interests are in the area of VoIP, QoS and Performance of Computer Networks.



Anura Jayasumana is a Professor in Electrical and Computer Engineering at Colorado State University and holds a joint appointment in Computer Science. His areas of expertise include Networks, Protocol Implementation, Performance Modeling, Protocols and Applications for Next Generation Internet, Optical Networks and Design and Design for Testability of VLSI. He received the PhD (1984) and MS (1982) degrees in Electrical Engineering from Michigan State University, and BSc (First Class Honours) in Electronics and Telecommunications Engineering from University of Sri Lanka, Moratuwa. He has served extensively as a consultant to industry. Awards and recognitions he had received include the Outstanding Faculty of the Year Award from American Electronics Association (1990), Award for Outstanding Academic Achievement at Michigan State University (1982, 1983), and the Award for the Best Student in Electrical Engineering at University of Sri Lanka (1978).