Gaze Tracking Using A Regular Web Camera

David Wild
Department of Computer Science
Rhodes University
Grahamstown, South Africa 6139
Email: g09w0474@campus.ru.ac.za

***Abstract*—Traditional gaze tracking systems rely on either contact and invasive hardware, or expensive and non-standard hardware. To address this problem research has been done into systems that use only simple, non-invasive hardware to create gaze tracking systems. This system attempts to prove that is is possible to create a gaze tracking system using a regular web camera and the free, open source Computer Vision library OpenCV. This is achieved by researching various techniques required for such a system and then measuring the performance of the created system. Analysis of the results concluded that while the error of the system was greater than desired, in conjunction with the research done, the conclusion is that it is possible to create a robust system using simple hardware.**

***Index Terms*—Gaze Tracking, Face Tracking, Computer Vision, Feature detection, Eye Movement.**

I. Introduction

In his paper on non-invasive eye tracking technologies, Gao stated that gaze tracking technology is still in the "embryonic stages" [1] of development and is not ready to be used in daily lives. Despite this statement, there are already various systems available, like the Starburst system put forward in [2], and the Tobii system[1] mentioned in [3]. Though systems like these are available, we are yet to see a widespread usage of them because, as mentioned previously, the field is still in it's embryonic stages and is yet to become an adopted technological advance.

A possible reason for this lack of integration into commercial society is that many of the previous systems proposed, like the system in [4], are head mounted or invasive gear. This creates the need to research into non-contact systems, such as the system put forward in [5], which can perform at the same efficiency level. However, previous systems, like the system designed in [6], required the use of a large fixed infrared camera, an LED array as well as eye positioning cameras mounted on top of the screen to function efficiently. This need for extra equipment is still a problem for regular users since they do not wish to buy it.

However, the possibilities for use of a Gaze tracking system are immense. The obvious application of gaze tracking is for Human-Computer interactions, as a new system or as an aid to those with disabilities. [2] Other applications, that are not often thought of, include research into human behaviour. In his paper, [8], from 1996, Tock attempted to track and measure a motor vehicle driver's eyes for eyelid separation. This was to aid in a larger system that could detect when a driver was getting drowsy. This system could reduce the number of road accidents due to fatigue saving the lives of drivers. In [9], gaze tracking was used to study the user's visual behaviour when surfing the Internet. The applications of this research affect advertising, website design, improving usability in computer systems and optimising web browsers and websites for maximum user experience. An even more interesting study was presented in [10] where the researcher investigated the behaviour of the eyes while reading a book and from that information was able to infer information on how cognitive and lexical processing affects reading and learning. This could help develop cognitive development theories that aid in learning and teaching.

It is apparent that gaze tracking systems are beneficial but they are not developed to the point of commercial use yet. As such, this project developed a system to track the user's gaze within one of four regions in the screen. While this does not prove that a simple webcamera can be used for accurate gaze tracking, in conjunction with the research, it was found that this lack of accuracy is most likely due to the techniques used in the system rather than the hardware, and as such it is possible to create an accurate gaze tracker.

The remainder of this article continues by providing the necessary background information required to track the user's gaze in Section II and then explicitly goes through the design and implementation of the system in Sections III and IV respectively. Section V provides the results of the section and a comparison in techniques. Section VI summarises the conclusions made and Section VII discusses possible additional work to, or deriving from, the system.

II. Background Information

The first step to designing a gaze tracking system is to understand the requirements and the background information available.

A. Eye Structure and Movements

To accurately track the gaze of a person it is vital to know the basics of how the human eye works. This requires a knowledge of both the structure of the eye, as shown below in figure 1, as well as the movements of the eye.

[1]The website for the system is http://www.tobii.com/pceye
[2]As discussed in [7]

1) Eye Structure: The important factor to considering when looking at the structure of the eye is where the highest density area of nerve receptors are. On such area is the fovea. As shown in figure 1, the fovea is a spot on the back of the eye. Outside of the fovea the visual acuity drops to half or less that of the fovea and in [11] it was found that peripheral vision is not good enough to focus with or read text with. Considering that we wish to find the focus point of the vision the fovea then becomes the area to consider in our applications. It is mentioned in [12] that in order to focus on an object we will move our eyes such that the focus area is in the fovea area. [11]

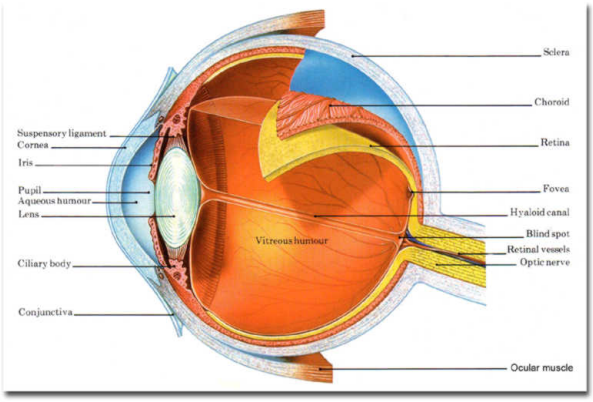

Fig. 1. Image of the Eye

The problem with using the fovea is that it is located at the back of the eye making it difficult to track. Instead the Pupil or Iris of the eye can be tracked and the focus point inferred from that. This is possible because the object of focus is kept in the centre of the eye so it is possible to find the focus point by looking at the outer layers of the eye and inferring the foveas focus point from those results. The area of interest thus becomes the Pupil or the Iris, as stated above. [12]

Another problem with Gaze tracking in general, as discussed in [11], is that within the fovea there is still the one degree of error. If the focus area is sufficiently small enough, then the focus point could shift without any eye movements since the focus point would still be within the fovea's one degree of vision. The consequence of this is that it is unlikely to better than one degree of error when measuring using gaze position. This is consistent with the errors found in [13], [5], [14], [15] and [16].

2) Eye Movements: Knowledge of eye movements is important in determining which information is relevant and what degree of accuracy is required for such a system. Human vision appears to be a smooth process with the eyes moving smoothly over the image to rest on a focus point but in [17], [11] and [12], eye movements are described as a series of sudden jumps with rest points between. The jumps are called *Saccades* and the rests are named *Fixations* [12].

Saccades are extremely quick jumps between focus points. It was found in [18] that the information gained during these jumps was ignored to maintain the perception of stability in the human eye. This effect is known as saccadic suppression [12]. This loss of information means that information is only gained during *fixations*.

Fixations are "a period of relative stability during which an object can be viewed" [11, p5] and as such we need only look at these events to gain enough information to track the gaze. Since *fixations* last many times longer than any other eye movements, a gaze tracking system does not need to track every movement of the eyes which lowers the restriction on the systems speed of capture and calculation.

B. Other Algorithms

Many other approaches to this type of system were examined and some examples taken from different approaches are given below.

1) Real-time gaze Tracking Algorithm with head movement Compensation (RTAC): The RTAC uses an infrared camera to increase the accuracy and speed up the system. The explanation of RTAC in [5] begins by referring to the "Pupil Center Corneal Reflection (PCCR) [5, p395]. According to the PCCR, gaze direction calculations firstly acquire the pupil-glint vector, and then use a gaze mapping function. The glint in referred to here is the reflection of the infrared light source in the eye. The pupil-glint vector is then the 2D vector between the glint and the center of the pupil. The algorithm then uses a mapping function to map this vector to a 3D gaze direction.

The algorithm then calculates the relative changes in head position in order to compensate for head movement during tracking. The compensation method employed by this algorithm uses two sets of equations: the calculation of the proportional change in magnification and then using these values to calculate the compensation value in the horizontal and vertical directions. [5, p397].

It was found that this algorithm had an accuracy rate of approximately one degree, and that there was little difference between different subjects. There was also little difference between different positions of the head found, though the error did increase when the head was moved near the boundaries of the cameras vision. [5]

2) Neural Networks (NN): The first step of using Neural networks, according to [15], is Face and Eye detection.

The eye and face are detected using the intensities of the pixels in a region and then each region is given a score. The scores are then arranged hierarchically and fed to an OpenCV library for detection. [15]

The next step is to reduce the ROI to a smaller picture. This picture is usually extremely small being less than a thousandth of the size and only containing the eye. From there the image is processed to make a clearer input for the neural network. The two processes applied are histogram equalisation that brightens the sclera and darkens the boundary of the eyes, and resizing the image to a smaller format using a bi-cubic interpolation. This reduces the pixel by fourfold which cuts the training time of the NN from several minutes to a time generally less than 60 seconds. [15]

The intensity of each pixel in the image is used as an input for the neural network. The neural network employed a feed forward, two-layer structure. The actual Neural network workings are dealing with Artificial Intelligence concepts rather than image processing and as such will be left out.[3] [15]

The error using the Neural Network approach in [15] was found to be approximately 1.5 degrees. The error found in [13] was similar but the difference in time between these two articles suggests an upper limit to the accuracy of this approach.

A Neural Network was used differently in [20]. Initially the system used blink detection to pinpoint the location of the eyes. Once the eyes are found the Iris detection was done using a combination of edge detection and circular Hough transforms, followed by corner detection aimed at finding both the inner and outer corners. The Neural Network was only used to map the relation between the eye features and the gaze direction. This is a different approach than the one used in [15] in that the feature detection is done without the Neural Network. [20]

C. Summary

The structure of the eye suggests that tracking the Iris or pupil will be sufficient. Due to the nature of the eye movements and the effects of saccadic suppression it is only necessary to capture the gaze position of the user during a fixation and since this accounts for the majority of eye movement time the restrictions for real-time tracking are decreased significantly. Other algorithms that use different hardware and different techniques have been found to have excellent error rates but use extra equipment and more sophisticated techniques. The existence of these systems suggests that it is possible to create a robust system with a simpler webcamera if the differences in hardware can be compensated for.

III. Design and Tools

The aim of this project is to create a Gaze tracking system using simple techniques, a regular web camera and the Vision library OpenCV. Since the logical design of most Gaze Tracking systems is fairly standard, the overall design of this system is very similar to existing systems. However, the tools used in these systems have a large amount of variation in them. As such, the particular techniques used were chosen to be simple techniques for a baseline comparison.

This section will begin by detailing some design decisions, and move onto the logical and actual design specifics including brief discussions on the techniques used.

A. Design Decisions

Traditional, non-contact Eye tracking systems had several problems that need to be taken into consideration when calculating the focus point or when finding the eye. The first and foremost of these is head movement, as mentioned by [15]. Older eye tracking systems required the user to keep their head still in order to accurately capture the data. More comprehensive systems, like [5], [21] and [22] , take this problem into account in their algorithms. Interestingly, these algorithms all tend to incorporate infrared lighting. However, since this system is a simple system, it will not include head movement cancellation and will be limited as such.

B. General Algorithm

The general algorithm of the system is split into two parts: the continuous loop to calculate the gaze, and the main program which contains the loop.

1) Loop: The loop steps are the steps required to get all the necessary data from a single frame. The simplest form of this design is shown below: [4]

Fig. 2. Gaze Loop Steps: Logical Design

The first step is to get a single frame from the camera. Once the frame is retrieved the face is detected. Face detection is done in order to reduce the number of false positives from the eye detection. This is achieved by reducing the search area to only this area including the detected face. A requirement of this is that the face detection needs to be a quick process that does not add too much delay to the system. The next step is the eye detection. This step takes the area the face was found in and searches that area for the eyes. This step finds each eye separately in this system. Both eyes are used in order to increase the accuracy of the system. Once the eyes are obtained, the pupil and corners are needed.

The exact method of finding the pupil within the eye differs between algorithms, and this is the step requiring the most accuracy. There are some consistencies between different algorithms however. A particular technique is to detect circles in the image but this requires cleaning of the image first. Since no special lighting was used, some form of brightening of the image must occur in order to accentuate the differences between the pupil and the white of the eyes. It is also necessary to reduce the noise in the image. A threshold technique is useful to reduce the noise in the image. Edge detection is also usually a good method of accentuating that border around the iris. Once you have included enough noise reduction then it is possible to search the image for circles and find the circle of

[3]More information on NN can be found in [19]

[4]This step by step procedure is an adaption of design from [23]

the Iris. The center of the pupil can be approximated as the center of this discovered circle. [24]

The inner corner of the eye was chosen as the second feature to check. The reason for this is because this feature has been used successfully in [20]. The reason a second feature is required is to create a vector to track the changes in the eye when it moves within the socket. As such, this feature needs to be fixed relative to the movement in the eye. The nose was used in [3], but the inner corner was decided on in order to have separate points within the reduced region for each eye. Once the coordinates of both the eye and the corner are obtained for both eyes, then the loop steps are complete.

2) Main Program: The main program incorporates the loop steps to create the gaze tracking system. Initially a calibration program is run to get pupil and corner positions for certain fixed positions and then the main loop is entered.

The calibration program consists of the user looking at nine separate points in order. These nine points then provide a reference for calculating the area of the gaze later. Doing the calibration incorrectly however can cause large errors in the later gaze tracking.

Once this calibration is done the main system begins to operate. This system tracks the user's gaze and compares it to the points captured in the calibration to find the position on the screen of the user's gaze. A dot is drawn on the screen in one of four regions and the user can focus on that dot to check the results of the system.

So the system will go through the calibration stage first, in order to provide reference points for the main detection, after which the main program begins running a continuous loop that compares the current points to the calibration points in order to calculate the gaze position.

IV. Implementation

This section will describe the techniques used for each step discussed in the design.

A. Facial Detection

The face detection is the first step in the implementation of the system. It was decided to use Haar feature detection for this step. OpenCV has predefined Haar Classifiers that come with the installation of the library, and these were used to detect the face. The image is grey scaled first to make the Haar detection more robust. A screenshot showing the detected face is shown below in figure 3:

Once the face is found the working area for the rest of the loop is reduced to the rectangle containing the face, as mentioned in section III.

B. Eye Detection

The eye detection is done in a similar manner using the built in Haar classifiers. The difference in this step is that it is run twice, once for each eye. It was found that this method of detecting the eyes was not as accurate and would often find the wrong eye. So the region of interest (ROI) was split into the two halves of the face in order to accurately detect the correct eye. Once the eye was found, the system cropped the image to only contain the eye, as shown in figure 4 below. This output is further cropped to remove the eyebrow as this interferes in the circle detection discussed below.

Fig. 3. Face detection Image

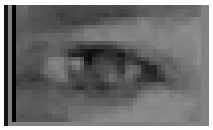

Fig. 4. Eye detection Image

C. Pupil Detection

The next step in the process is pupil detection. This is by far the most difficult part of the process. Figure 4 shows the image coming in to this function. As can clearly be seen, the pupil is not highlighted at all. It is even difficult to see the edge of the Iris correctly. This is partially due to the lighting in the testing room and an infrared system would clearly highlight the pupil. However, pre-processing the image can make a major improvement by highlighting the important information.

The pre-processing of the image runs through several stages before the circle detection to find the pupil occurs. The order is as follows:

1) Histogram Equalisation to accentuate the dark and light areas
2) A binary threshold is applied to reduce the noise and highlight the dark regions
3) The image is then smoothed to further reduce noise and make patterns more apparent

Once the pre-processing on the image is done, circle detection is used to find the edge of the Iris. The circle detection used is a function in OpenCV called HoughCircles that uses ellipse fitting. Only the strongest circle in the image is returned. Figure 5 below shows the detection of the pupil.

The center of the pupil is approximated to be the center of the detected circle.

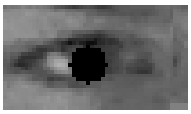

Fig. 5. Pupil Detected

D. Corner Detection

The corner detection works by equalising the histogram of the image again, to accentuate the edges, and then using the OpenCV method cvGoodFeaturesToTrack to find the strongest corners in the image. This function will find either corner of the eye however so the solution was to further reduce the ROI to only include the inner corner of the eye. The reductions for this were done separately for each eye. The output of this process is show below in figure 6.

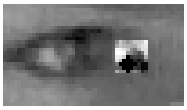

Fig. 6. Corner Detected

E. Transforming to Window Coordinates

The first step to converting the pupil and corner coordinates to window coordinates is to get the distances between the x and y coordinates of the corner and pupil. This is done before the main loop for the calibration coordinates to avoid repeatedly processing these constant values. The live data is then processed within the loop, and the distances are passed to a conversion function. This conversion function is what outputs the final coordinates.

The first part of the conversion function is to discover what region the eye is in by comparing the distance to the distances found in the calibration. This is a necessary step in order to know which points to use in the conversion. Once we have the region we calculate the ratio of the distances between the fixed points and the live point, as shown below in figure 7. The red dot in the figure represents the live point. The ratio is then multiplied by the window height and width to get the window coordinates which are then returned to the main function to use in the error recording.

V. Testing and Analysis

In testing the system several limitations were discovered. It was also found that the implementations of the pupil and corner detection were found to be too inaccurate for pinpoint gaze tracking but at least good enough to detect which region the user's gaze was in as shown in section V-A below.

A. System Output

An example of the console output is shown below in figure 8. As can be seen, the system outputs the region that the user has their gaze in. The rest of the console output is simple error checking the system runs through.

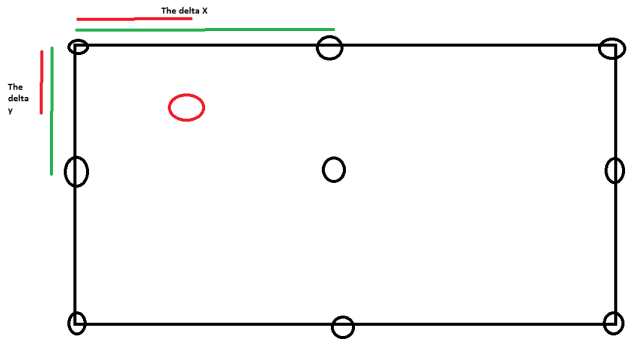

Fig. 7. Diagram of Distance Ratio's

```
The size of delta's is : 4
the size of myD is 4 and the size of calibD is 36
myD[0] 19 calibD[4] 17 myD[2] 9 calibD[6] 15
myD[1] 0 calibD[29] 4 myD[3] 7 calibD[31] 4
The gaze is found to be in region 4
```

Fig. 8. Sample Console Output

The actual error results, while still within the region, were far larger than expected. The possible reasons for these errors will be discussed in section V-B below.

B. Techniques Analysis

Several separate parts of the system were found to introduce inaccuracies in the results.

Firstly, the pupil detection method was found to be inaccurate to the point of finding the wrong point more than a third of the time. To counter this, the system used an averaging system to find where the majority of pupil center locations were found, in order to approximate the correct position from a set of points. While allowing the system to detect which region the gaze was in, this technique did not sufficiently decrease the error. A higher resolution camera could help with this but this could be resolved through techniques like interpolation to resize the image while retaining the data. Research into alternative techniques suggested using pixel intensities to find the pupil instead which could bypass the resolution problem entirely and increase the accuracy. This is one step that is assisted greatly by the use of an infrared camera.

Secondly, the corner as a feature to track introduced several problems in the implementation and further increased the error despite the alleviating steps taken. As previously mentioned, in [3] the systems uses the bridge of the nose as the second feature to track and this could remove several of the problems created and alleviate the error in the result. Alternatively, a better technique to detect the location of the corner could be implemented.

Other techniques also introduced limitations to the system that did not explicitly affect the error.

C. Limitations

In testing the system it was discovered that the Haar detection would be unable to find the face quite often for

seemingly no reason. Furthermore, the Haar detection could not detect the face if the user had darker skin than the features were trained for, which limits the possible users of the system. There are several other methods of face detection that could be used to replace this technique however so this does not affect the validity of a regular webcamera in a system like this. As previously mentioned, any head movements during the process will negatively affect the accuracy of the system but this can be compensated for with techniques similar to those mention in section II-B1.

VI. Conclusion

The results obtained show that the system was ineffective. As discussed above in section V this could be due to the error introduced through the specific techniques chosen. This suggests that with better techniques the efficiency could be increased.

Several alternative techniques to pupil detection are discussed in [23] and were found to be accurate enough for use in human-computer interfacing. These techniques, as well as the testing, were all done using regular webcameras, which suggests a system like the one this paper discusses is definitely possible to create. However, infrared cameras in conjunction with an infrared light have a better accuracy, on average, than those of a regular webcamera.

Through the research, it seems that the majority of the problems found in this system can be resolved or reduced using more complex techniques, that have already been tested to be accurate. As such, the conclusion of this paper is that it is possible to create an accurate Gaze tracking system using only a regular webcamera.

VII. Future Work

Future work to either confirm the conclusions of this paper, or to tackle further problems, include:

- Performing technique comparisons
- Removing the need for calibration
- Incorporating head movement cancellation
- Implementing a gaze tracker using a webcamera and a Raspberry Pi for mobility.
- Implementing the system on the GPU

The use of Gaze tracking in other fields, as discussed in section I, is not to be underestimated. As such, future work in this area could include work with other disciplines to research various activities.

References

[1] D. Gao, G. Yin, W. Cheng, and X. Feng, "Non-invasive eye tracking technology based on corneal reflex," *Procedia Engineering*, vol. 29, no. 0, pp. 3608 – 3612, 2012, 2012 International Workshop on Information and Electronics Engineering. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877705812005498

[2] D. Li, D. Winfield, and D. J. Parkhurst, "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops - Volume 03*, ser. CVPR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 79–. [Online]. Available: http://0-dx.doi.org.wam.seals.ac.za/10.1109/CVPR.2005.531

[3] I. Frieslaar, "Moving the mouse pointer using eye gazing," Published through Department of Computer Science, University of the Western Cape, 2011. [Online]. Available: http://www.cs.uwc.ac.za/~ifrieslaar/Doc.html

[4] D. Li, J. Babcock, and D. J. Parkhurst, "openeyes: a low-cost head-mounted eye-tracking solution," in *Proceedings of the 2006 symposium on Eye tracking research & applications*, ser. ETRA '06. New York, NY, USA: ACM, 2006, pp. 95–100. [Online]. Available: http://0-doi.acm.org.wam.seals.ac.za/10.1145/1117309.1117350

[5] Y. Huang, Z. Wang, and A. Ping, "Non-contact gaze tracking with head movement adaptation based on single camera," 2009.

[6] T. Ohno and N. Mukawa, "A free-head, simple calibration, gaze tracking system that enables gaze-based interaction," in *Proceedings of the 2004 symposium on Eye tracking research & applications*, ser. ETRA '04. New York, NY, USA: ACM, 2004, pp. 115–122. [Online]. Available: http://doi.acm.org/10.1145/968363.968387

[7] C. Mauri and J. Lors, "Computer vision interaction for people with severe movement restrictions."

[8] D. Tock and I. Craw, "Tracking and measuring drivers' eyes," *Image and Vision Computing*, vol. 14, no. 8, pp. 541 – 547, 1996, 6th British Machine Vision Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0262885696010918

[9] G. Buscher, E. Cutrell, and M. R. Morris, "What do you see when you're surfing?: using eye tracking to predict salient regions of web pages," in *Proceedings of the 27th international conference on Human factors in computing systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 21–30. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518705

[10] E. D. Reichle, A. Pollatsek, D. L. Fisher, and K. Rayner, "Toward a model of eye movement control in reading," *PSYCHOLOGICAL REVIEW*, vol. 105, no. 1, pp. 125–157, 1998.

[11] R. J. K. Jacob, "Eye tracking in advanced interface design," 1995.

[12] K. Rayner, "Eye movements in reading and information processing: 20 years of research," *Psychological Bulletin*, pp. 372–422, 1998.

[13] S. Baluja and D. Pomerleau, "Non-intrusive gaze tracking using artificial neural networks," Pittsburgh, PA, USA, Tech. Rep., 1994.

[14] C. H. Morimoto and M. R. Mimica, "Eye gaze tracking techniques for interactive applications," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 4 – 24, 2005, special Issue on Eye Detection and Tracking. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314204001109

[15] W. Sewell and O. Komogortsev, "Real-time eye gaze tracking with an unmodified commodity webcam employing a neural network," in *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, ser. CHI EA '10. New York, NY, USA: ACM, 2010, pp. 3739–3744. [Online]. Available: http://0-doi.acm.org.wam.seals.ac.za/10.1145/1753846.1754048

[16] Z. Zhu, Q. Ji, and S. Member, "Novel eye gaze tracking techniques under natural head movement," 2009. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.143.7178

[17] R. J. K. Jacob, "The use of eye movements in human-computer interaction techniques: What you look at is what you get," *ACM Transactions on Information Systems*, vol. 9, pp. 152–169, 1991.

[18] S. Klingenhoefer and F. Bremmer, "Saccadic suppression of displacement in face of saccade adaptation," *Vision Research*, vol. 51, no. 8, pp. 881 – 889, 2011, perception and Action: Part II. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0042698910005730

[19] M. Husken, C. Igel, and M. Toussaint, "Task-dependent evolution of modularity in neural networks," *Connection Science*, vol. 14, p. 2002, 2002.

[20] D. Torricelli, S. Conforto, M. Schmid, and T. DAlessio, "A neural-based remote eye gaze tracker under natural head motion," *Computer Methods and Programs in Biomedicine*, vol. 92, no. 1, pp. 66 – 78, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169260708001521

[21] W. Z. Huang Ying and T. Xuyan, "A real-time compensation strategy for non-contact gaze tracking under natural head movement," *Chinese Journal of Electronics*, vol. 19, pp. 446–450, July 2010.

[22] S. Kawato and N. Tetsutani, "Detection and tracking of eyes for gaze-camera control," *Image and Vision Computing*, vol. 22, no. 12, pp. 1031 – 1038, 2004, proceedings from the 15th International Conference on Vision Interface. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885604000769

[23] K. P. Ciesla Michal. Eye pupil location using webcam. Department of Physics, Astronomy and Applied Computer Science, Jagiellonian University, Reymonta 4. Accessed on 31 October 2012. [Online]. Available: http://arxiv.org/ftp/arxiv/papers/1202/1202.6517.pdf

[24] CodeBox. (2010, November) opencv - iris detection. Accessed on 26 October 2012. [Online]. Available: http://codingbox.net/jinstall/index.php?option=com_content&view=section&layout=blog&id=3&Itemid=16