A HEURISTIC APPROACH FOR GAZE TRACKING USING SETS

Gaurav Agarwal and Prasoon Nigam
Signal Processing Lab, Department of Electrical Engineering,
Indian Institute of Technology, Kanpur, India. PIN – 208016

e-mails: gaaga@indiatimes.com , prasoonnigam@hotmail.com

ABSTRACT

This paper discusses the implementation and results of a new algorithm for gaze tracking. It uses "*sets*" for the detection of eye-balls and *principal component analysis* for determining the direction of gaze. It has an auto-exposure algorithm which automatically sets the camera parameters for the best results.
*Keywords:*Gaze Tracking , Purkinje's Image,PCA .

1. INTRODUCTION

Gaze information plays an important role in identifying a person's focus of attention. The information can provide useful communication cues. For example, it can used to identify where a person is looking at, and what he/she is paying attention to. A person's gaze direction is determined by two factors viz. the *orientation of the head*, and the *orientation of the eyes*. While the orientation of the head determines the overall direction of the gaze, the orientation of the eyes is determining the exact gaze direction and is limited by the head orientation.
Possibly applications include:

- a computer mouse could be replaced by a vision system which moved the cursor where ever the user looked on the screen.
- A robot could learn about a new environment from an instructor who simply looked at obstacles and identified them.
- by observing operator's movements work areas could be redesigned more ergonomically. A portable robotic video camera could be built to focus and film where ever the operator looked.

In this paper, focus was on monitoring a user's eye gaze, that is, estimating where a user is looking at on a screen based on information of the user's eye gaze.

2. Methodology

Before going into methodology, a term *Purkinje's Image* needs to be defined**.** When light is shone into the user's eye, several reflections occur on the boundaries of the lens and cornea, those are the Purkinje images (see Figure 1). The first Purkinje image is also called the *glint*, and this can be video-recorded or photographed using an camera/sensor as a very bright spot and a less bright disc, respectively.

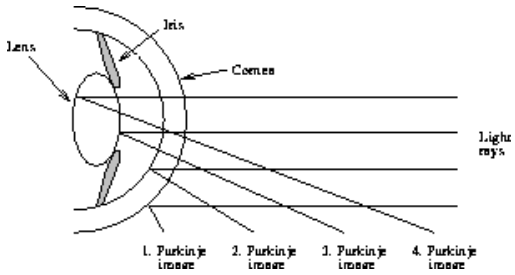

Figure 1. *The Purkinje images*

Coming back to methodology, the whole process can be divided in four major steps:

- **Setting the camera exposure** (which is shutter duration in our case) is optimally set using an auto-exposure algorithm [1], which is briefly described in the section 3.
- **Approximately locating the eyes using the pukinjees image** and

hence forming a “BOX” around the region of the eyes. Train the net with initial input.

- **Processing the image in the box to locate the direction of gaze** (Left, Right, Top, Bottom or Close). The method used for this process is Principal Component Analysis (PCA)[2], because of its robustness in classifying a given input image into the given classes (which in our case were five in number).At the end of this stage, the direction of gaze has been determined.

- **Using the gaze information for the required purpose**. For example this information can be used to drive a mouse icon for hands free surfing.

Figure 2 (the flow-chart), gives a clear picture of the overall process.

3. Adjusting the Exposure

Exposure is adjusted so that a pre-fixed threshold values can be used at the later stage for the best results. Adjusting the exposure is possible, if the brightness of the image lies within an acceptable range .The auto-exposure algorithm takes care of this.

Intensity histogramming of the image is done.Then the histogram is scanned from highest to lowest intensity until the number of pixels exceeds a fixed percentage of the total number of pixels in the image.

The intensity of that slot in the histogram is assumed to be the highest value to be exposed.Now treating this value of brightness as the maximum allowed brightness the intensities are scaled down .

4.Locating the Eyes

It should be mentioned that the method used for locating eyes is robust yet simple. The various step involved in this are as follows:
4.1 Capture the image after the auto-exposure setting is complete.
The result will be something like as in Figure 3.
4.2 Locate the pukinjees image.
4.2.1 Smoothen & edge detect the image using Canny filter.
The advantages of this method is that

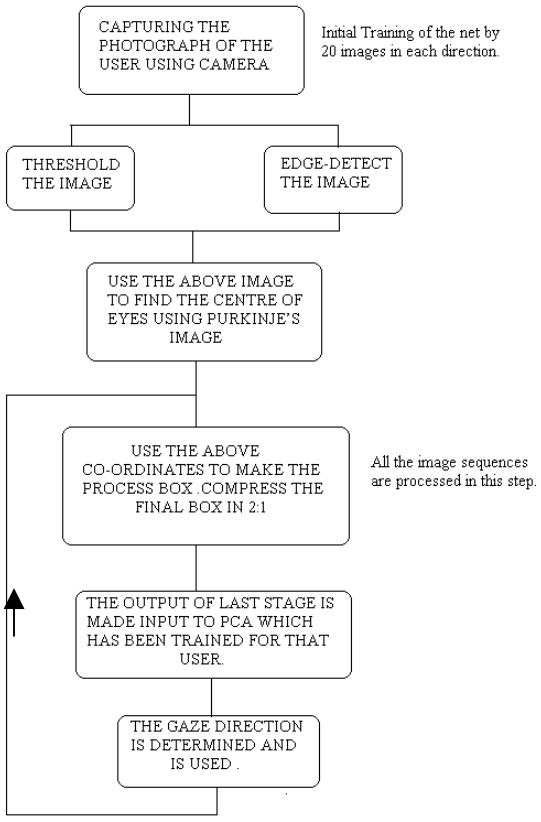

Figure 2. *The Complete process*

the purkinjees images are obtained in more or less circular and smooth form, which gave excellent result for the later stages. The disadvantage of this method was that it is a time intensive process. But as this is required only for initial adjustments, it is acceptable. The canny filter used has sigma= 1.5 and cut-off=60.The result is shown in Figure 4.
4.2.2 Threshold the image using a suitable cutoff. The alternative of 4.2.1 is simple thresholding. The advantage of this method is that it is very quick but it gives a noisy output. Since the assumption is that the face is uniformly lit, so the noise is not a problem. The threshold value of 140 gave excellent result (a binary image) which is shown in *Figure 5*.

4.2.3 Forming "SET"
In the obtained binary image, purkinjees image appear as a two distinct and isolated white region *with approximately the same*

Figure 3.*Input*

Figure 4.*After passing through canny filter(sigma=1.6) and finally thresholding(cut_off =60)*

Figure 5.*Simple thresolding (cut-off=140)*

*white are and row level.*Then "SETS" are are regions of *all connected white points* in a given binary image as shown in *Figure 6* which has 3 sets.

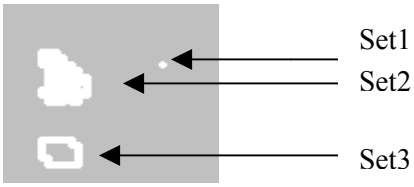

Figure 6. Sets

The centroids of all the sets are found. Now the problem is reduced to identifying the two sets, corresponding to the two purkinje's image.

4.4 The following five search are used to pinpoint purkinje's images.These functions were implemented as C functions.

4.4.1) *Search_0(flag, upper, lower)*

The role of this function is to find out those sets out of the total sets that have a row level difference between *lower* to *upper*. Search_0(1,50,0) will find out those sets whose centroid are displaced by 20 pixels. or less from some other set. This search is based on the assumption that the difference between the level of eyes are not more than 50 pixels when somebody is asked to keep his head straight.

4.4.*2 Search_1(int flag,int upper,int lower)*

The role of this function is to find out those sets out of the total sets that have column difference between *lower* and *upper* .
if *flag* =0 the function performs the required filtering for whole of the set search space, if *flag* = 1 the function performs filtering only on those sets that have *possible[]* value of 1. *upper* denotes the upper limit of column level difference.*lower* denotes the lower limit on column level difference. Search(1,200,120) will find those sets that have some other set between 200 and 120 pixel difference.This is useful because the distance between the eyes , when a user is sitting at a comfortable distance lies between this range (Assumption : Distance of the face from the setup is roughly constant).

4.4.3 *Search_2(flag, upper, lower)*

This search puts a threshold on the size of the set .It returns those set whose size is between *lower* and *upper*. For eg. , Search(1,70,5) will choose only those sets that have area between 5 pixel and 70 pixels.This function is like our noise filter and immensely reduces the search space.

4.4.4) *rectangle(flag, x, y, width,height)* This function restricts its search in a box centered (*x,y*) of width *width* and height *height.(*Assumption: The user sits in a manner that his face is more or less near the centre of the image.)
e.g. rectangle(1,200,300,100,150) - This function restricts its search in a box centered (200*,300*) of width 100 and height 150.This will return all the sets lying between 100<x<300 and 150<y<450.
4.4.5) *Search_0_1_2(flag , upper1, lower1, upper2, lower2, upper3, lower3)* This filter uses the properties of Search_0 , Search_1 , Search_2.

5.Box-Creation

The purkinjees are located with 99.9% accuracy in all the test cases and a square of size d/2 is created with the center at purkinjees images where 'd' is the distance between the purkinjees image. This assures good result in our algorithm.

The results of this step are shown in *Figure 6* , *Figure 7* , *Figure 8* and *Figure 9* for various types of background (black, general or white).


Figure 6.Black Background

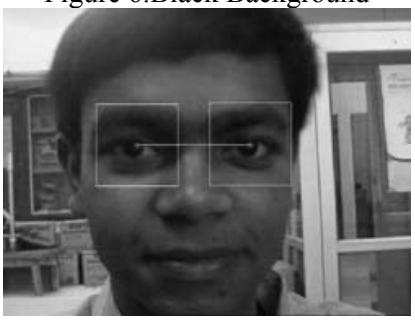
Figure 7.General Background


Figure 8. White Background


Figure 9.Black Background

6.Processing the "Box" using trained PCA to find the Gaze.
Principal component analysis was done using a software names PCA[**2**]. The input to the PCA is 64X64 pixel "Box" (eg *Figure 10* & *Figure 11*).The box is re-sized to the above size. For training, initially ,about 50 images (10 images for each direction) is fed into the network with the labelled images i. e. L,R,U,D,C,F.Here each term refer to one state of the eye viz. left,right,up,down,close,front. The user looks in the above state one after the another for training. The input rate of image is 10 frames per second , so training takes less than 10 seconds. After the training is over , the sysytem is ready for gaze tracking.The output would again be either of L,R,U,D,C,F.


Figure10.Front (F)

Figure 11.Left(L)

The simulation results for various test cases are listed in the table below:

S.No.	User	Eye	Output Correct/Total	% Result
1.	Black Background	R	57/57	100%
2.	Black Background	L	52/54	96.3%
3.	With Spectacles	L	51/55	92.3%
4.	White Background	L	41/44	93.2%
5.	Black Background	R	21/23	91.3 %
6.	Noisy Background	R	14/23	60. 9 %
7.	A's training wt. Used for B.	L for user A	24/44	45.45%

Table 1. RESULTS
training set = 50 images

The method was very successful and gave results above 90% (Table 1).To make a special mention:-

Many of the user will be wearing spectacles so we experimented with users with spectacles (for example S.No 3 in Table 1) .Above 92% accuracy of detection was achieved . (see *Figure 12.* & *Figure 13*.)

Figure 12.With Spectacles

Figure 13.With Spectacles

6.CONCLUSION

The result obtained substantiates the effectiveness of our method which involved a blend of heuristic(search functions) and rigorous approach(pca). This method can be used for design of intelligent systems based on gaze tracking.

7. REFERENCES

1. Mike Caplinger and Mike Malin , Autoexposure Algorithm. http://www.msss.com/http/near_cal/fs_algorithms/autoexposure/autoexposure.html

2. http://www.elec.gla.ac.uk/~romdhani

3. Jie Yang and Alex Waibel , "Gaze Tracking for Multimodal Human Computer Interaction", *Proceedings of WACV'96.*

4.Anil K. Jain , Fundamentals Of Digital Processing , Prentice Hall (PHI).