Real-Time Eye Tracking Using a Smart Camera

Mehrube Mehrubeoglu*, Linh Manh Pham, Hung Thieu Le, Ramchander Muddu, and Dongseok Ryu

Texas A&M University-Corpus Christi, School of Engineering and Computing Sciences
Corpus Christi, Texas, USA
*Ruby.Mehrubeoglu@tamucc.edu

Abstract—**Real-time eye and iris tracking is important for hands-off gaze-based password entry, instrument control by paraplegic patients, Internet user studies, as well as homeland security applications. In this project, a smart camera, LabVIEW and vision software tools are utilized to generate eye detection and tracking algorithms. The algorithms are uploaded to the smart camera for on-board image processing. Eye detection refers to finding eye features in a single frame. Eye tracking is achieved by detecting the same eye features across multiple image frames and correlating them to a particular eye. The algorithms are tested for eye detection and tracking under different conditions including different angles of the face, head motion speed, and eye occlusions to determine their usability for the proposed applications. This paper presents the implemented algorithms and performance results of these algorithms on the smart camera.**

Keywords- real-time eye detection, real-time eye tracking, smart camera, LabVIEW, NI Vision Builder

I. Introduction

Eye tracking is the process of detecting the eye location across video frames to determine the direction of gaze. The motion of the eye relative to the head may also be of interest. Eye tracking is important for research and development areas such as visual systems, psychological analysis, cognitive science and product design. An eye tracking system is an integration of a set of devices and associated programs for measuring eye positions and eye movement, and correlating the results to the same eye across images acquired sequentially over time.

Many existing systems require PCs for vision processing [1], [2]. However, the PC is not adequate to withstand environmental factors in the actual field. In addition, a smart camera with its on-board data processing capabilities allows the execution of fast algorithms well suited for real-time tracking applications. Similar to industrial vision systems, DSP-based standalone vision equipment can provide optimized and rugged solutions specialized for individual applications in law enforcement, homeland security, medicine, and many other areas that require a compact system to work under a variety of conditions.

In this paper, a real-time standalone eye tracking system using a National Instruments Smart Camera is presented with new algorithms, building on our prior work [3]. A tracking algorithm optimized for the smart camera, and its implementation using LabVIEW 8.5.1 and NI Vision Builder AI 3.6 are discussed. The experimental evaluation of the algorithm is also described for eye detection and tracking under different conditions including different angles of the face, head motion, and eye occlusions to determine the usability of the system for real-time applications.

The remainder of this paper covers applications and related work. In Section II, the eye tracking methodologies and algorithms are described. Section III summarizes the smart camera implementation of the eye tracking system. Results are presented in Section IV. The conclusions can be found in Section V.

A. Applications

A real-time eye tracking system must be robust under a various conditions including motion speed, occlusions and face angles to be useful for specific applications. One such application involves an input device for paraplegic patients to help use computers [4], [5]. Eye tracking can serve military purposes by tracking a pilot's eye and developing appropriate digital visual interfaces for use in the cockpit [6]. Other applications include reader behavior testing when viewing websites [7], [8]. The results are important to determine gaze or reading patterns of the user. Personal preferences can be tested during shopping or reading brochures for interest groups to make more targeted products or advertisements based on the collected consumer data. Based on the analysis of the eye movement, physical state of a driver can be assessed to warn drowsy drivers [9]-[11]. Not only can the physical state of the human body but also the intent of a person can be examined [12]. Mental state or emotions of an individual such as anxiety, deceit, or hostility can be revealed by eye tracking, [13]-[15] which can then be used in homeland security applications. Although multiple approaches to building an eye tracking system exists, most work under the same principles, namely, detecting the same eye features across multiple image frames and correlating the results to a particular eye.

B. Related Work

Many eye tracking systems are proposed in the literature and implemented in practice. Zhu and Ji describe a computer vision system based on active IR illumination for real-time gaze tracking and interactive graphic display [16]. Typical gaze tracking systems require a static head to work well, and require a complicated calibration process. In the system described by Zhu and Ji, the gaze tracker provides accurate gaze estimations without calibration and even with head-movements. A gaze calibration procedure is implemented that identifies the mapping from pupil parameters to screen coordinates using the generalized regression neural networks (GRNNs). The authors use pupil glint detection and tracking, gaze calibration and gaze mapping to accomplish eye tracking. In our system, we are interested in detecting the center of the eye. Calibration can be achieved by asking the user to gaze at two or three corners of the viewed screen to match camera view coordinates to screen coordinates. Villanuev *et al.* describe an eye tracking system whose objective is to identify user gaze direction [17]. Video-oculography (VOG) is used to provide the gaze position with high precision. VOG involves illumination and data acquisition to determine the point the subject is looking at through the captured images. One of VOG's advantages is as a solution for building nonintrusive systems which avoid the use of extra devices such as helmets and special glasses holding small cameras. Chen *et al.* describe an auto-stereoscopic display system for stereo visualization without the discomfort and inconvenience of wearing stereo glasses or head-mounted displays. In their project, the authors implement real-time tracking techniques which can efficiently provide user's eye position in images. These techniques include face detection using multiple Eigen spaces under various lighting conditions and fast block matching for tracking four motion parameters (X translation, Y translation, scaling, and rotation) from the user's face and eye regions. Although eye tracking is accomplished, the main purpose of the project is to provide auto-stereoscopic display for a stereo visualization system. The authors perform face detection first before eye tracking to increase the accuracy of the eye tracking algorithm [18]. Similarly in our application, no wearable devices are necessary. Eye tracking is achieved through eye detection algorithms implemented on the NI smart camera which has fast processing times suitable for real-time applications. Tracking is achieved by storing the location of previous matches in successive image frames. Fleck *et al.* describe the implementation of a particle filter for probabilistic real-time tracking in a smart camera using FPGA [19]. We use LabVIEW for the implementation of image processing algorithms for real-time eye detection that include pattern matching, edge detection, and computation and recording of eye center location for tracking. Garg *et al.* describe the implementation of PC-based iris detection algorithms in LabVIEW. In this paper, we use LabVIEW to implement eye detection algorithms on a smart camera.

II. METHODOLOGY

In this project, real-time eye detection and tracking are achieved with LabVIEW-based algorithms uploaded to a smart camera.

A. Real-time Eye Tracking with a Smart Camera

Real-time eye tracking algorithms were implemented in a smart camera (National Instruments) using LabVIEW 8.5.1 and NI Vision Builder AI 3.6 programming tools. Multiple methods exist for building real-time eye-tracking systems; typically webcams have been used as image sensors along with language tools such as OpenCV and Visual C as affordable solutions. However, the smart camera offers more flexibility and advantages in terms of hardware specification than a typical webcam; the smart camera has features of an independent computer; zooming can be adjusted with a variety of interchangeable lenses to match the requirements of an application. A smart camera contains a processor with one or multiple cores for processing algorithms faster. It also has internal memory such as ROM or RAM as well as an image sensor with an electronic shutter. In order to program for the smart camera, the software tools serving as the integrated development environment (IDE) for programming is required. The eye tracking system described in this work requires image processing pattern recognition algorithms. LabVIEW, a high-level graphical programming environment was chosen for programming and interfacing. Interfaces called virtual instruments (VIs) can be designed and controlled via GPIB, RS232, VXI or regular Ethernet interfaces. The eye tracking algorithm and testing process was developed. A built-in FTP server is used to connect to the internal storage of the NI smart camera directly. The TCP connection library from LabVIEW allows the smart camera to run on a local network similar to a regular computer.

B. Real-time Eye Tracking Algorithm

The camera is programmed to track the eye motion continuously until stopped by the user. Figure 1 shows the flowchart for the real-time eye tracking algorithm.

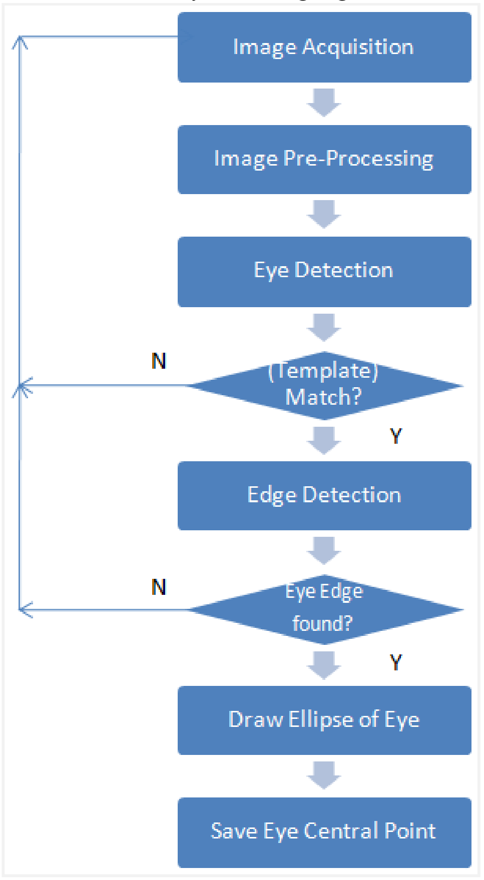


Figure 1. Flowchart of the Real-time Eye Tracking Algorithm

The flowchart of Figure 1 is explained below:

- Step 1 - Image Acquisition: The raw image is acquired automatically by the smart camera
- Step 2 – Image Pre-processing: The acquired raw image is converted to a gray-scale image, also automatically by the smart camera.
- Step 3 - Eye Detection: Initially the user's eye is detected using template matching. In template matching, a template from the S is compared with the given image using a matching metric. The matching metric provides a measure of similarity between the two templates. This similarity is converted into a numerical value as a score of the template match. A score of 1000 means an exact match with the tested template. Since, the images are obtained under different test conditions, the minimum threshold score is set around 600.
- Step 4 - Edge Detection: If the eye is detected, new Region of interest (ROI) which covers only the eye is extracted to reduce the processing area. An edge detection technique is applied to the new ROI to find points around the ellipse or circle of the eye.
- Step 5 - If at least 3 points (for circle) or 4 points (for ellipse) are found, the circle or the ellipse of the eye will be drawn. Otherwise, the current frame is skipped over with no match.
- Step 6 - If the ellipse/circle of the eye is drawn, then the coordinates of the center of the eye will be calculated in the camera's processor and saved to a spreadsheet for future reference. These coordinates are computed as the center of the rectangle bounding the detected eye.

This algorithm can be applied to any kind of smart camera with any programming language. The particular implementation of the experiment for this paper is discussed in Section IV.

III. Implementation

A. System Hardware

The NI smart camera is directly connected to a regular computer through an Ethernet cable in a local network. The data and control signal are sent back and forth in this cable in full-duplex form. The full-duplex form allows communication in both directions, which results in faster transmission. The computer has a Pentium 4 2.26GHz processor, an on-board 64 MB graphics card and 1.25 GB RAM. LabVIEW desktop interface is used to communicate with the smart camera. The experimental eye tracking system is shown in Figure 2.

The Main component of the eye tracking system is a NI 1762 smart camera with a Computar 12 mm lens and the

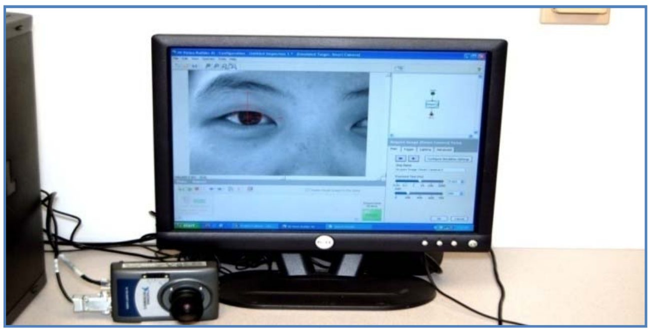


Figure 2. NI Smart Camera and Eye Tracking System

Gigabit Ethernet interface. The smart camera has dual processors. One is a 533 MHz PowerPC processor (Freescale). The other is a 720MHz DSP (Texas Instruments). These dual processors permit the smart camera to process data up to four times faster when applying algorithms such as pattern matching, optical character recognition, and data matrix code reading. The camera has a 1/3 inch Sony ICX 424 AL solid-state image sensor with square pixels for the acquisition of gray-scale images. In addition two separate 128 MB memory units exist on board the camera. One memory block is utilized for the system and the other is for firmware and job (data/program/spreadsheet) storage. Finally, the smart camera is equipped with a temperature sensor which shuts off at temperatures greater than 70^{o}C automatically, and resumes operation when a suitable operating temperature is reached again. The lens is attached to the camera through a C mount and allows high resolution images to be formed on the smart camera's sensor. Through the use of the lens and the aperture control, the amount of light and zoom can be adjusted.

B. Software and Design

Figure 3 shows the algorithms developed and implemented in Vision Builder AI 3.6 that is supported by the smart camera. In Figure 3, the first step corresponds to Step 1 and 2 in Figure 1. The edge detection and drawing the ellipse/circle representing the location of the eye is performed in "Draw Eye & Track" module.

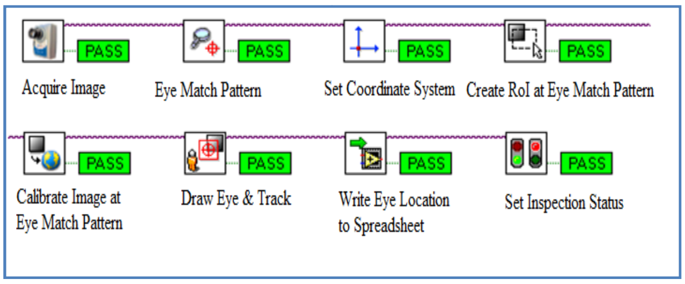


Figure 3. Eye Detection and Tracking Process

The following describes the eye detection and tracking process in detail:

- Step 1: A real-time eye image is captured using a smart camera interfaced to LabVIEW.
- Step 2: The eye is matched with a pre-stored eye pattern (only the best match is reported).
- Step 3: The coordinate system and origin are aligned with the captured eye image.
- Step 4: The Region of Interest is set to the matching eye area to reduce the area that is processed.
- Step 5: The image is calibrated to convert pixel coordinates to real-world coordinates.
- Step 6: An ellipse or circle is drawn based on the points extracted from the Region of Interest. Also the tracking line is drawn from a list of points that represent the center of the eye from successive frames.
- Step 7: The coordinates of the center of the eye are stored in a spreadsheet.
- Step 8: The inspection status is shown through pass (green) or fail (red) indicators.

In Figure 4, the block diagram of the real-time eye tracking algorithm in LabVIEW 8.5.1 is depicted. The block diagrams constitute the main graphical programming environment for the LabVIEW applications. Connections between virtual instruments (VIs) and analog input/output devices are created under LabVIEW 8.5.1 using the G programming language. In general, each block diagram is divided into modules corresponding to various program steps. Each module is called the appropriate subVIs serving as sub-functions. For example, in Figure 3, the module named "Eye Match Pattern," which applies the pattern matching algorithm to the gray-scale images with a given template, must call the subVIs "IVB Match Pattern Setup.vi," "IVB Match Pattern Exec.vi" and other related subVIs from the LabVIEW library called "IVB Match Pattern Code". In the figures, only the main modules are shown. Each subVI, in turn, can call its own subVIs. For example, module "Draw Eye and Track" calls "Edge Detection" and "Read the Eye Central Point from Spreadsheet" subVIs, respectively. The pseudocode for the "Read the Eye Central Point from Spreadsheet" subVI is described below:

```
WHILE eye is detected in the frame based on set criteria
    Calculate Center of eye in x
    Calculate Center of eye in y
    IF best eye match
        PUT (x,y) in Trajectory list
    END IF
END WHILE

IF the size of Trajectory list > Number of frames to track
    POP the last data point (x,y) from Trajectory
END IF

FOR Number of frames to track
    DRAW line for the Trajectory
END FOR
```

Figure 5 demonstrates the above pseudocode's implementation in LabVIEW. The eye center coordinates are stored in a list that is loaded into the smart camera's buffer for faster eye tracking. Then these points are written to a spreadsheet file in the internal disk of the camera for future reference. This file is retrieved from camera using FTP.

IV. Results and Analysis

A. The Accuracy of the Eye Tracking Algorithm

The results in Table I are obtained with real-time face data acquired over two minutes. The detection rate decreases as frame rate increases for all four test cases under ideal conditions, with head motion, with head angle to the camera and with partial occlusion of the eye region.

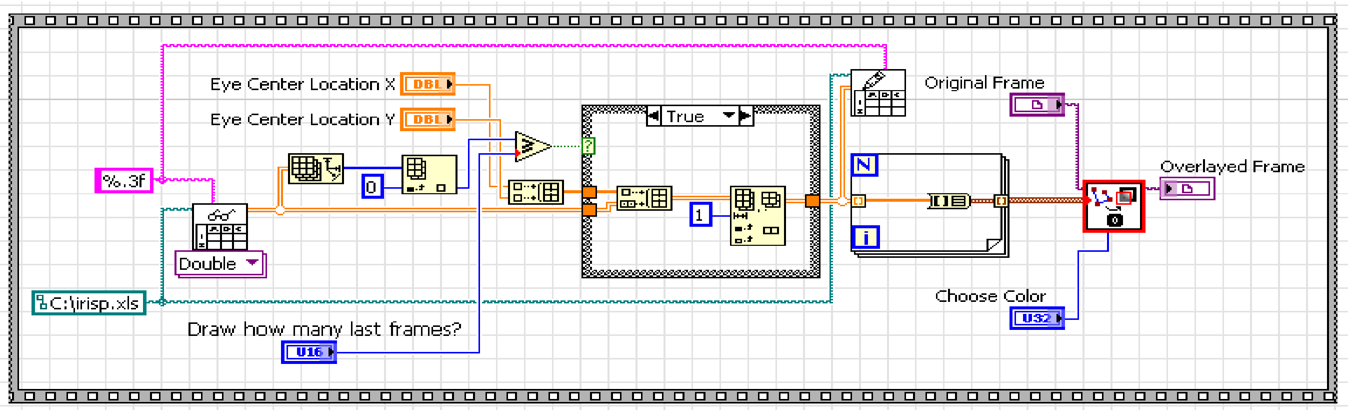


Figure 4. LabVIEW Block Diagram of Eye Detection & Tracking Process

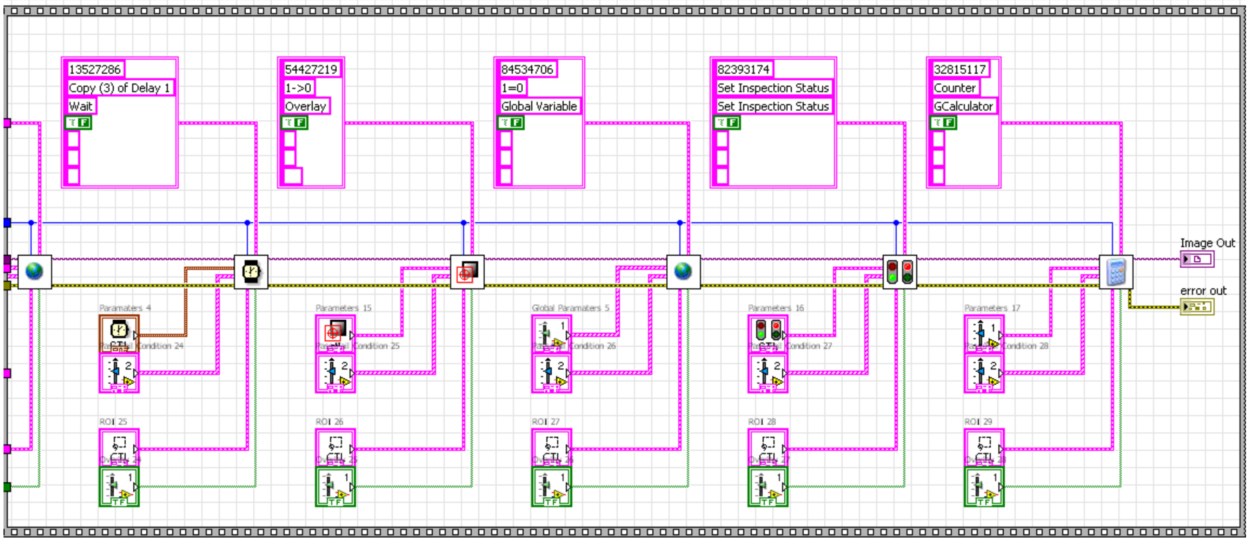


Figure 5. Block Diagram of "Read the Eye Central Point from Spreadsheet" subVI

Formula for average detection rate is given as

$$\text{Average detection rate} = \frac{\#\text{ of frames eye is detected}}{\text{total }\#\text{ of frames}}$$

Table I shows the correct eye detection rates reported for each case corresponding to increasing frame rates. The results are retrieved from the last step in "Set inspection Status" and user-defined variables of LabVIEW block diagram shown in Figure 3. Under ideal conditions with the face directly in front of the camera, eye detection rates are above 93% even at the highest frame rates. When the face is moved, the detection performance is reduced but is still above 87%. The detection rate decreases significantly (down to 70.21% for the same frame rate) if the face is at an angle of about 45^{o} to or looking away from the camera. Partial occlusions further reduce detection rates (worst tested case 66.49%), and therefore, require more elaborate algorithms than what have been implemented.

TABLE I. AVERAGE % DETECTION RATES IN DIFFERENT CASES FOR DATA COLLECTED OVER TWO MINUTES

Test Case / Frame Rate	Ideal Conditions	Moving Face	Eye at an Angle	Partial Occlusion
30 fps	97.57	92.06	78.23	74.21
35 fps	96.88	91.64	76.54	72.58
40 fps	95.89	90.83	75.41	72.31
45 fps	95.21	90.14	73.39	70.67
50 fps	94.39	89.89	71.7	68.25
55 fps	94.21	88.95	70.98	67.34
60 fps	93.93	87.37	70.21	66.49
Average	95.44	90.13	73.78	70.26

Figure 6 shows the result of the real-time eye detection algorithm under varying real-time input image conditions. Each case is explained as follows: **1.** (top left) Detection of the eye in the ideal case, with face directly in front of the camera, no motion, no occlusion. The user is staring at the camera lens, and the red colored circle is the location of detected eye; **2.** (bottom left) Detection of the eye when the face is moved (tilted); **3.** (top right) Detection of the eye when the face is turned away from the camera lens and is at an angle; **4.** (bottom right) Detection of the eye when an occlusion (in this case, eye glasses) exists around the eye region.

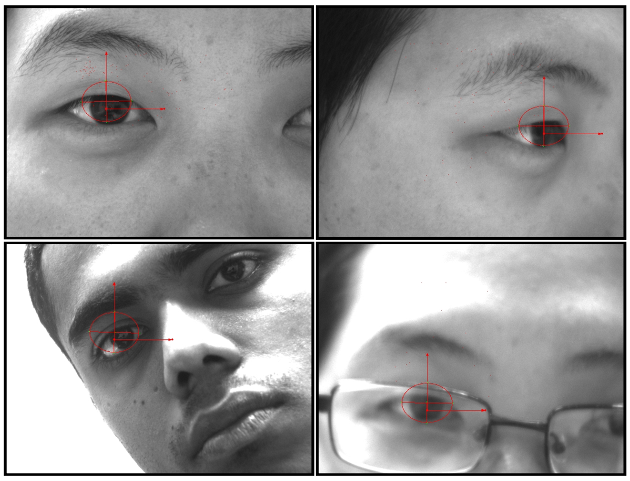


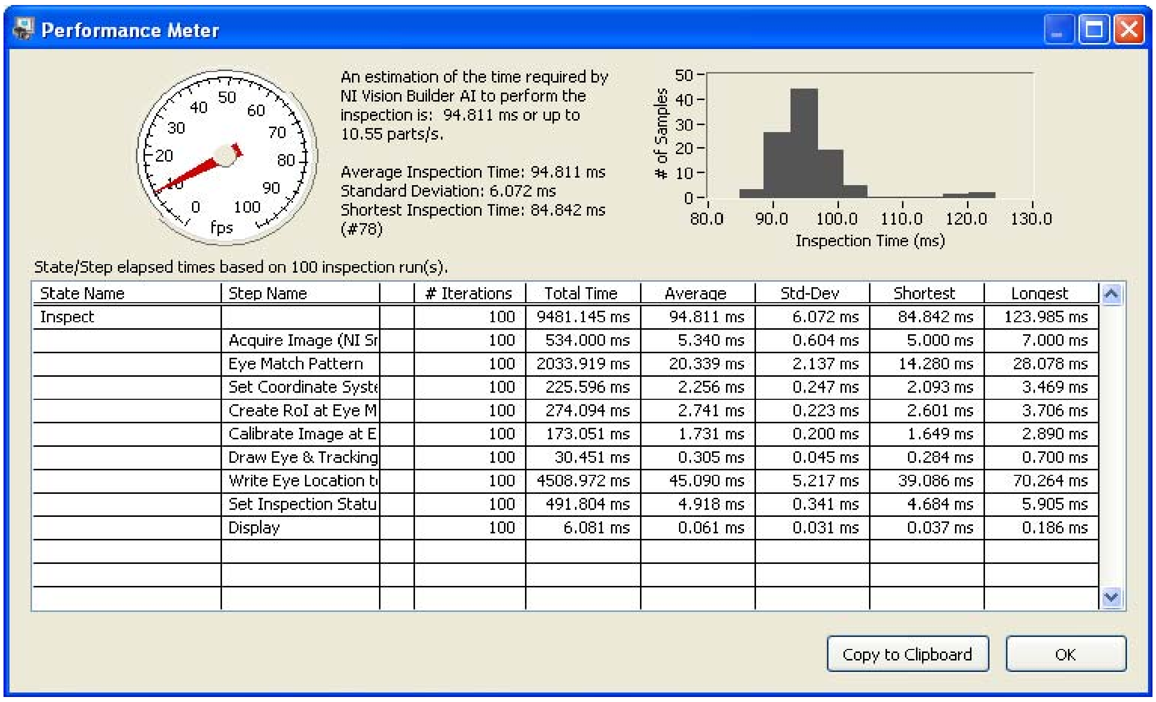
Figure 6. Real-time Eye Detection with Images Captured by Smart Camera

B. *Speed of Eye Tracking Algorithm*

The testing for the speed of the algorithm involves positioning the tester (user) in front of the mounted camera with a fixed distance. The distance between the mounted smart camera and the tester is about 1.8 to 2 feet. The algorithm is limited to up to 45^{o} of rotation of the head to the left or right.

The smart camera then acquires the tester's face image for processing.

Figure 7 shows the time needed to process each image processing step by the smart camera for the first frame. Based on these results, processing steps that need optimization can be determined and addressed. The entire process takes an average of 94.811 ms. The longest time for an image frame processing step is "Write Eye Location to the Spreadsheet File" (Figure 3) with an average of 45.090 ms. The actual processing time of the algorithm without writing to the spreadsheet file is 94.811 - 45.090 = 49.721 ms. The performance is expected to increase with the use of a faster computer which will result in faster data transfer times between the computer and the smart camera.



An estimation of the time required by NI Vision Builder AI to perform the inspection is: 94.811 ms or up to 10.55 parts/s.

Average Inspection Time: 94.811 ms
Standard Deviation: 6.072 ms
Shortest Inspection Time: 84.842 ms (#78)

State/Step elapsed times based on 100 inspection run(s).

State Name	Step Name		# Iterations	Total Time	Average	Std-Dev	Shortest	Longest
Inspect			100	9481.145 ms	94.811 ms	6.072 ms	84.842 ms	123.985 ms
	Acquire Image (NI Sr		100	534.000 ms	5.340 ms	0.604 ms	5.000 ms	7.000 ms
	Eye Match Pattern		100	2033.919 ms	20.339 ms	2.137 ms	14.280 ms	28.078 ms
	Set Coordinate Syste		100	225.596 ms	2.256 ms	0.247 ms	2.093 ms	3.469 ms
	Create RoI at Eye M		100	274.094 ms	2.741 ms	0.223 ms	2.601 ms	3.706 ms
	Calibrate Image at E		100	173.051 ms	1.731 ms	0.200 ms	1.649 ms	2.890 ms
	Draw Eye & Tracking		100	30.451 ms	0.305 ms	0.045 ms	0.284 ms	0.700 ms
	Write Eye Location t		100	4508.972 ms	45.090 ms	5.217 ms	39.086 ms	70.264 ms
	Set Inspection Statu		100	491.804 ms	4.918 ms	0.341 ms	4.684 ms	5.905 ms
	Display		100	6.081 ms	0.061 ms	0.031 ms	0.037 ms	0.186 ms

Figure 7. Time per Inspection Step Using Benchmark Tool of NI Vision Builder AI 3.6

Figure 8 depicts the result of real-time eye tracking algorithm implemented in NI Vision Builder AI 3.6. The red lines keep track of eye movement. Only the data for the last 30 frames are shown in the figure. The end points of each line segment are retrieved from the list which was originally populated directly from the camera memory. The tracking algorithm still works well when an occlusion such as a pair of glasses exists around the eye region.

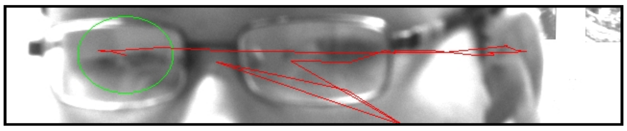


Figure 8. Real-time Eye Tracking

V. Conclusion

Real-time eye detection and tracking have been developed not for a PC but for use in a standalone smart camera with DSP capabilities. The tracking algorithm is optimized and simplified for the NI Smart Camera. The eye has been detected at varying real-time input conditions, and the developed GUI interface is used to visualize the trajectory by connecting the eye center coordinates through red line segments across frames. The experimental results revealed that eye tracking accuracy decreases with increasing frame rates. Especially, eye at an angle and occlusion showed a drop in successful detection rates that were notably lower than the ideal input conditions. The limiting factor that affects the speed of the algorithm is the transfer time for the detected eye center coordinates from the smart camera to a spreadsheet file on the PC which is the protocol used for the testing stage. A faster PC is expected to significantly reduce the data transfer time. Alternatively, transferring the data after each session instead of after each data point can be considered. Data transfer is for historic data collection only; all the data processing is performed on the smart camera. Future work involves more robust algorithms to improve the performance of the system under varying conditions.

Acknowledgment

The authors thank Mr. John Gonzalez for the customized camera mount, and the Office of Graduate Studies for their sponsorship of student participation in this conference. This project was partially supported by Texas Research and Development Fund.

REFERENCES

[1] R. Garg, V. Gupta, and V. Agrawal, “Efficient iris recognition method for identification,” *in Proc. ICUMT,* pp.1-6, 2009.

[2] C. A. Perez, V. A. Lazcano, P. A. Estévez, and C.M. Held, “Real-time iris detection on faces with coronal axis rotation,” *in Proc. SMC* (7) , pp.6389-94, 2004.

[3] M. Mehrubeoglu, H. T. Bui, and L. McLauchlan, “Real-time iris tracking with a smart camera,” *Proceedings of SPIE*, vol. 7871, Real-Time Image and Video Processing, 2011, N. Kehtarnavaz and M. F. Carlsohn, Eds., 787104, Feb. 2011.

[4] Frangeskides and A. Lanitis, “A Hands-Free Non-Invasive Human Compuer Interation System,” *Advances in Systems, Computing Sciences and Software Engineering,* pp. 235-242, 2006.
(DOI: 10.1007/1-4020-5263-4_38)

[5] C. Mauri and T. Granollers, “Computer vision Interaction for People with Severe Movemtn Restrictions,” *Human Technology*, 2(1), pp. 38-54, April 2006.

[6] W. Haibo, X. Chengqi, L. Qing, “The Eye Movement Experiment and the Usability Evaluation of the Fighter Cockpit Digital Interface,” *2nd International Conf. Information Engineering and Computer Science (ICIECS) 2010,* pp. 1-4, 2010. (10.1109/ICIECS.2010.5678205)

[7] Q. Li, L. Sun and J. Duan, “Web Page Viewing Behaviorof Users: An Eye-Tracking Study,” *Proc. of ICSSSM '05, International Conf. on Services Systems and Services Management,* pp. 244-249, 2005.

[8] W. Lu, M. Li, S. Lu, Y. Song, J. Yin, and N. Zhong , “Impact of Information Overload for Visual Search on Web Pages: An Eye-tracking Study,” *Proc. 2010 IEEE/ICME International Conference on Complex Medical Engineering, July 13-15, 2010, Gold Coast, Australia.*

[9] R. Garg, V. Gupta and V. Agrawal, “A Drowsy Driver Detection and security system,” in *Proc. ICUMT*, pp.1-8, 2009.

[10] M. J. Flores, J. M. Armingol, and A. D. L. Escalera, “Real-Time Warning System for Driver Drowsiness Detection Using Visual Information,” *presented at Journal of Intelligent and Robotic Systems,* pp.103-125, 2010.

[11] Q. Ji and X. Yang, “Real-Time Eye, Gaze, and Face Pose Tracking for Monitoring Driver Vigilance,” *Real-Time Imaging,* Vol. 8, Elsevier Science Ltd., pp. 357–377, 2002.

[12] S. Milekic, “The more you look the more you get: Intention-based interface using gaze-tracking,” in *Trant, J.(Des.) Museums and the Web2002: Selected Papers from an Int. Conf., Archives and Museum Informatics*, 2002, pp. 1–27.

[13] J. Orozco, F. X. Roca, and J. Gonzàlez, “Real-time gaze tracking with appearance-based models,” *Machine Vision and Applications,* 20(6): 353-364, 2009.

[14] J. C. Kircher, A. E. Cook and D. J. Hacker, “Deception Detection using Oculomotor Movements,” United States Patent Application Publication, Pub. No.: US 2010/0324454 A1,Pub. Date: Dec. 23, 2010.

[15] I. Cohen, N. Sebe, L. Chen, A. Garg, T. S. Huang, “ Facial expression recognition from video sequences: temporal and static modeling,” *Comput. Vision Image Understand* **91**(1–2), pp. 160–187, 2003.

[16] Z. Zhu, and Q. Ji, “Eye and gaze tracking for interactive graphic display,” *Machine Vision and Applications*, pp. 139-148, July 2004.

[17] A. Villanuev, R. Cabeza, and S. Porta, “Gaze tracking system model based on physical parameters,” *International Journal of Pattern Recognition and Artificial Intelligence*, 21(5), 2007.

[18] Y. Chen, C. Su, C., Chen, Y. Hung, and C. Fuh, “Video-based Eye 1.Tracking for Autostereoscopic Displays,” *Opt.. Eng.*, 40, pp. 2726 - 2734 , 2001.

[19] S. Fleck, S. Lanwer, and W. Straßer, “A smart camera approach to real-time tracking,” *13th European Signal Processing Conference (EUSIPCO 2005).*