Pedestrian Safety with Eye-Contact between Autonomous Car and Pedestrian

Kohei Arai[1], Akihito Yamashita[2], Hiroshi Okumura[3]
Dept of Information Science, Saga University, Saga City, Japan

***Abstract*—Method for eye-contact between autonomous car and pedestrian is proposed for pedestrian safety. The method allows to detect the pedestrian who would like to across a street through eye-contact between an autonomous driving car and the pedestrian. Through experiment, it is found that the proposed method does work well for finding such pedestrians and for noticing a sign to the pedestrians from the autonomous driving car with a comprehensive representation of face image displayed onto front glass of the car.**

***Keywords*—Autonomous driving car; eye-contact; self driving car; pedestrian safety; Yolo; OpenCV; GazeRecorder**

I. Introduction

In recent years, the world has become more interested in autonomous driving of cars. However, the realization of fully automatic driving has various problems in terms of safety, and can be mentioned as difficult problems. The International Transport Forum at the OECD is an intergovernmental organization with 54 member countries. It acts as a think tank for transport policy and organizes the Annual Summit of transport ministers. ITF is the only global body that covers all transport modes [1]. Centralized raw data fusion, neural networks for machine learning is developed for autonomous driving car [2]. 360-degree real-time perception of vehicle surroundings is also developed. The developed software system runs on a flexible yet automotive-grade hardware platform with state-of-the-art FPGA, SoC and safety processors.

An autonomous vehicle driving control system carries a large number of benefits, whether for the engineering industry or engineering education. The system discussed here provides the measurements obtained from vision namely, offset from the centerline at some look-ahead distance and the angle between the road tangent and the orientation of the vehicle at some look-ahead distance and these are directly used for control [3].

The hardware capabilities are already approaching the levels needed for well-optimized Autonomous Vehicle software to run smoothly. Current technology should achieve the required levels of computational power—both for graphics processing units (GPUs) and central processing units (CPUs)—very soon [4]. Cameras for sensors have the required range, resolution, and field of vision but face significant limitations in bad weather conditions. Radar is technologically ready and represents the best option for detection in rough weather and road conditions. Lidar systems, offering the best field of vision, can cover 360 degrees with high levels of granularity. Although these devices are currently pricey and too large, a number of commercially viable, small, and inexpensive ones should hit the market in the next year or two.

At present, connected vehicles equipped with a system that can communicate by IoT have appeared in order to ensure safety between vehicles. However, it does not hold between cars and pedestrians. In this study, we focus on eye contact with cars and pedestrians, which are not considered in the current autonomous driving, and aim to avoid danger and secure certain safety.

In this paper, from the background of the research in Section 1, we have presented the problems raised and the purpose of this research. In Section 2, we described trends in autonomous driving of automobile-related companies. The following section describes the system proposed in this paper. Then the following section gives an overview of this system followed by the conditions of eye contact. After that, image recognition, specifically pedestrian detection, and face and eye detection, as well as eye gaze estimation are described.

The experiments conducted in this study are described in the third chapter. Object detection using YOLO and the results are described. Then, the OpenCV face / eye detection procedure and results are followed together with the procedure and results of gaze estimation using GazeRecorder. After that, the discussion of the experiment is described. Final chapter gives the conclusions with some discussions.

II. Proposed Method

A. Proposed Method Overview

This section outlines the proposed system for securing pedestrian safety by eye contact detection in autonomous driving. This system projects the face of a person on the windshield of a car and sets eye contact with pedestrians. Therefore, we use an onboard camera to monitor pedestrians, detect pedestrians from the camera images, and confirm eye contact to avoid danger. Based on these, simulation is performed to verify whether safety can be maintained.

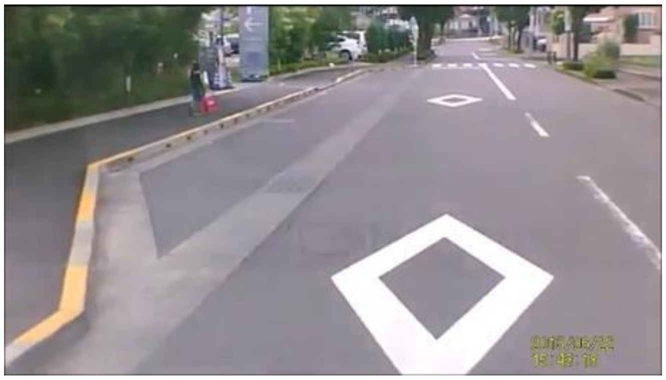
B. Eye-Contact

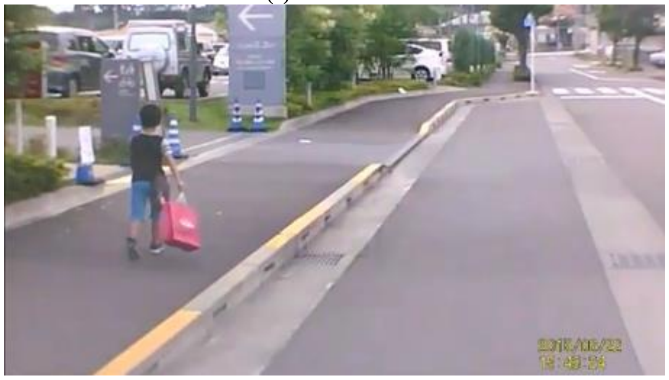
In this study, we define the conditions under which eye contact between a car and a pedestrian can be made as follows, and the situation is shown in Fig. 1 [1].

1) A pedestrian can be recognized from the on-board camera image (Fig. 1(a)).

2) It turns out that a pedestrian is going to cross (Fig. 1(b)).

3) Noticed the car and confirmed the driver (Fig. 1(c)).


(a) Condition#1


(b) Condition#2

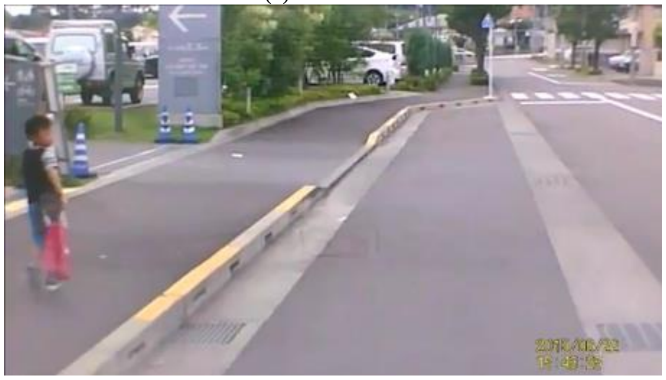

(c) Condition#3

Fig. 1. Conditions for Eye Contact.

The pedestrian can cross the street safely by satisfying the above conditions and giving an eye contact to the pedestrian from the car and giving an intention to stop (Fig. 2).

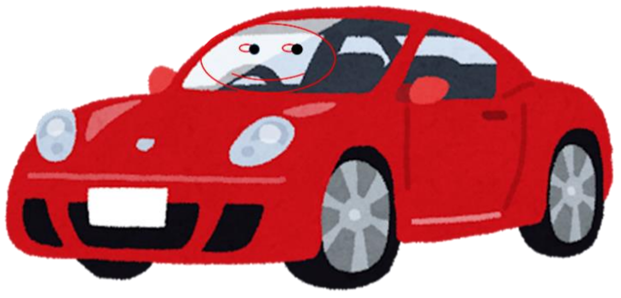


Fig. 2. Notifying a Sign to the Pedestrians from the Autonomous Driving Car with a Comprehensive Representation of Face Image Displayed onto front Glass of the Car.

C. Pedestrian Finding

In order to perform pedestrian recognition, we use an image recognition algorithm YOLO (You Only Look Once) [2] [3] that can be processed in real time by Joseph Redmon et al. In this research, we use Darknet [4] to implement YOLO.

D. Face and Eye Detection

We use the Haar-like feature classifier implemented in OpenCV (Open Source Computer Vision Library) [5] for face and eye detection. The features of various objects are learned in advance, and the xml file list [6] of the created feature data is shown in Table I.

In this research, we use "haarcascade_frontalface_alt.xml" and "haarcascade_eye.xml" for face detection and eye detection.

E. Gaze Detection

In this research, safety of pedestrians is secured by eye contact detection. Therefore, in order to perform eye contact, it is considered essential to estimate the line of sight after detecting the face and eye from detection of the pedestrian. This time, we use a system called GazeRecorder [7] for eye gaze estimation. The main features of this system are shown below:

1) Gaze tracking using a webcam
2) There is no physical contact for gaze estimation
3) Real-time gaze tracking is possible
4) Dynamic heat map can be generated using an adaptive time window

Since real-time processing is important in this system, GazeRecorder is adopted for gaze estimation.

TABLE I. XML FILE LIST OF HAAR-LIKE FEATURE CLASSIFIER IMPLEMENTED IN OPEN CV

File Name
haarcascade_upperbody.xml
haarcascade_smile.xml
haarcascade_russian_plate_number.xml
haarcascade_righteye_2splits.xml
haarcascade_profileface.xml
haarcascade_lowerbody.xml
haarcascade_licence_plate_rus_16stages.xml
haarcascade_lefteye_2splits.xml
haarcascade_fullbody.xml
haarcascade_frontalface_default.xml
haarcascade_frontalface_alt2.xml
haarcascade_frontalface_alt_tree.xml
haarcascade_frontalface_alt.xml
haarcascade_frontalcatface_extended.xml
haarcascade_frontalcatface.xml
haarcascade_eye_tree_eyeglasses.xml
haarcascade_eye.xml

III. Experiment

A. Object Detection using YOLO

We implemented YOLO under the following environment.

- OS: Windows 10 (64 bit)
- CPU: Intel Core i5-5275U
- Memory: 8.00 GB
- Visual Studio 2015
- OpenCV 3.2
- Darknet

For the construction of the development environment, we referred to "Darknet[1] installed on windows 10 to perform object recognition and object discrimination" [8]. Although it affects the processing speed, this research does not use the GPU[2], and CUDA[3] as well as cuDNN[4] are not used because the processing is performed by the CPU.

B. Experimental Results for Pedestrian Detection

Fig. 3 shows the execution screen of YOLO in one scene of still image and moving image. Fig. 3(a) shows a screen shot of the process result of YOLO software operation while Fig. 3(b) shows an example of screen shot of the moving picture of object detection by YOLO. When processing with a still image, various objects other than human beings were detected. And even those who turned out to look backwards or from the image could be detected as humans. Also, it was found that the processing in the case of using the image in Fig. 3(c) takes approximately 2 seconds. When processing was performed using moving pictures, people and cars could be detected even though the image quality was poor. However, because processing was performed only by the CPU, it took more than 40 minutes to complete all processing even for a 30-second movie.

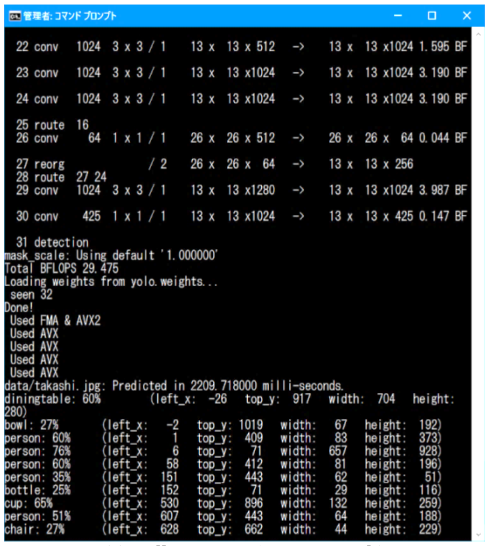
C. Face and Eye Detection by Python OpenCV

In this experiment, we use Anaconda [9], a Python distribution provided by Continuum Analytics, in order to use Python and, "Face Detection using Haar Cascades" [10] was referred to for face and eye detection. Fig. 4 shows the results of face and eye detection. The face and detected parts are enclosed in a yellow frame, and the eye and detected parts are enclosed in a light blue frame.

D. Gaze Estimation by GazeRecorder

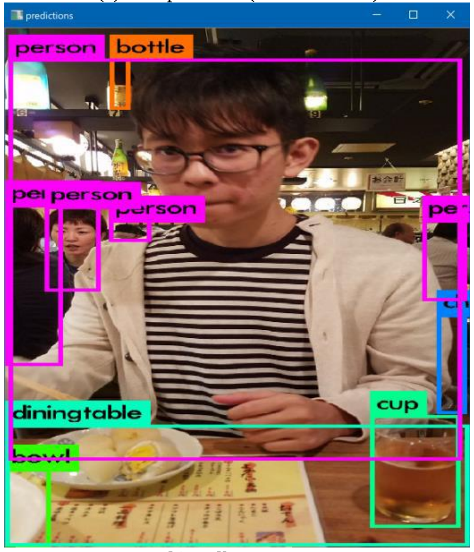
When GazeRecorder starts up, the screen shown in Fig. 5 is shown. Here, it is also possible to set execution results and eye tracking. GazeRecorder is downloadable software [11]. After the object detection [12], gaze is detected with GazeRecorder. After all, learning process on gaze detection can be done with anaconda [13].

When you click the icon that says "Start Cam", the webcam starts. After that, the screen "Look at dot" is displayed, so watch the white point at the center as shown in Fig. 6. Perform calibration by focusing on this point. And start tracking eye gaze.

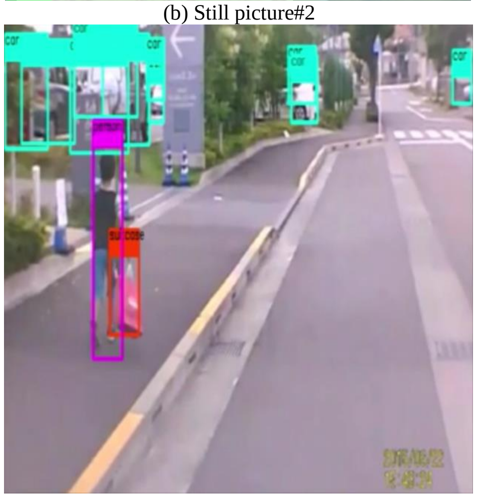


```
  22 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024 1.595 BF

  23 conv   1024  3 x 3 / 1    13 x  13 x1024   ->    13 x  13 x1024 3.190 BF

  24 conv   1024  3 x 3 / 1    13 x  13 x1024   ->    13 x  13 x1024 3.190 BF

  25 route  16
  26 conv     64  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x  64 0.044 BF

  27 reorg              / 2    26 x  26 x  64   ->    13 x  13 x 256
  28 route  27 24
  29 conv   1024  3 x 3 / 1    13 x  13 x1280   ->    13 x  13 x1024 3.987 BF

  30 conv    425  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 425 0.147 BF

  31 detection
mask_scale: Using default '1.000000'
Total BFLOPS 29.475
Loading weights from yolo.weights...
 seen 32
Done!
 Used FMA & AVX2
 Used AVX
 Used AVX
 Used AVX
 Used AVX
data/takashi.jpg: Predicted in 2209.718000 milli-seconds.
diningtable: 60%        (left_x:  -26   top_y:  917   width:  704   height:
280)
bowl: 27%       (left_x:   -2   top_y: 1019   width:   67   height:  192)
person: 60%     (left_x:    1   top_y:  409   width:   83   height:  373)
person: 76%     (left_x:    6   top_y:   71   width:  657   height:  928)
person: 60%     (left_x:   58   top_y:  412   width:   81   height:  196)
person: 35%     (left_x:  151   top_y:  443   width:   62   height:   51)
bottle: 25%     (left_x:  152   top_y:   71   width:   29   height:  116)
cup: 65%        (left_x:  530   top_y:  896   width:  132   height:  259)
person: 51%     (left_x:  607   top_y:  443   width:   64   height:  188)
chair: 27%      (left_x:  628   top_y:  662   width:   44   height:  229)
```

(a) Still picture#1(Process result)



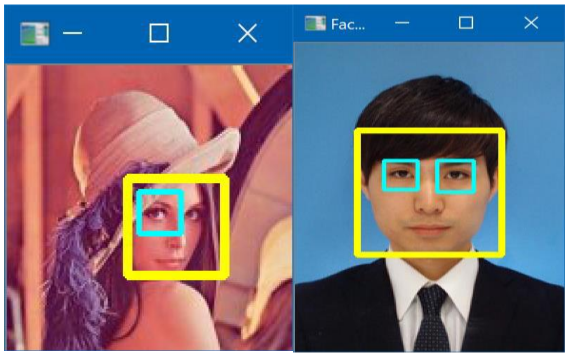
(b) Still picture#2



(c) Moving picture

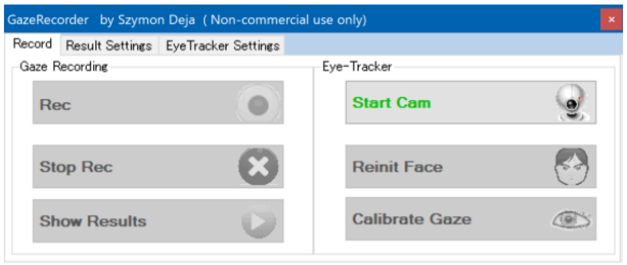
Fig. 3. Pedestrian Detection by YOLO.

1 https://pjreddie.com/darknet/
2 https://ja.wikipedia.org/wiki/Graphics_Processing_Unit
3 https://ja.wikipedia.org/wiki/CUDA
4 https://developer.nvidia.com/cudnn



(a) Lena (b) Co-author

Fig. 4. Face and Eye Detection.

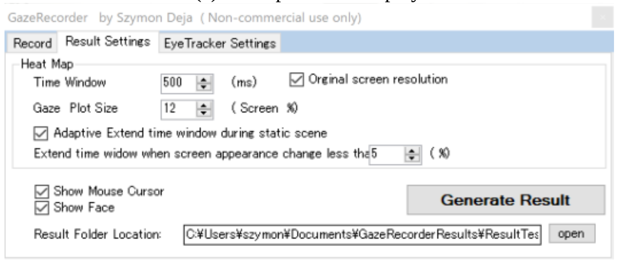

GazeRecorder by Szymon Deja (Non-commercial use only)

Record | Result Settings | EyeTracker Settings

Gaze Recording: Rec, Stop Rec, Show Results

Eye-Tracker: Start Cam, Reinit Face, Calibrate Gaze

(a) Star-up Screen Display.


GazeRecorder by Szymon Deja (Non-commercial use only)

Record | Result Settings | EyeTracker Settings

Heat Map

Time Window 500 (ms) ☑ Orginal screen resolution

Gaze Plot Size 12 (Screen %)

☑ Adaptive Extend time window during static scene

Extend time widow when screen appearance change less tha 5 (%)

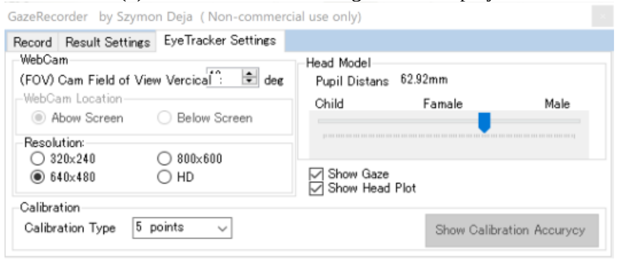
☑ Show Mouse Cursor

☑ Show Face

Generate Result

Result Folder Location: C:¥Users¥szymon¥Documents¥GazeRecorderResults¥ResultTes open

(b) Execution Result Setting Screen Display.


GazeRecorder by Szymon Deja (Non-commercial use only)

Record | Result Settings | EyeTracker Settings

WebCam

(FOV) Cam Field of View Vercical deg

WebCam Location: ◉ Abow Screen ○ Below Screen

Resolution: ○ 320x240 ○ 800x600 ◉ 640x480 ○ HD

Head Model

Pupil Distans 62.92mm

Child Famale Male

☑ Show Gaze

☑ Show Head Plot

Calibration

Calibration Type 5 points

Show Calibration Accuracy

(c) Result Setting Screen Display.

Fig. 5. Parameter Setting Screen Display of the Gaze Recorder.

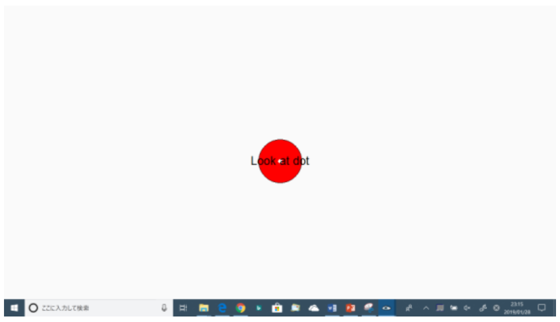


Fig. 6. Screen Display for Calibration.

Clicking on the “Rec” icon will start recording, and clicking on the “Stop Rec” icon will stop recording. The videos taken here are stored in an arbitrary directory, and a heat map is automatically generated.

As shown in Fig. 7, GazeRecorder shows that the face is captured in three dimensions. The light mesh area represents the entire eyeball, and the red circle indicates the pupil and the iris. Also, the light blue lines extending from both eyes indicate the pupil and the iris.

In the heat map, it shows whether a human is looking at the screen part, and also the place where the user is gazing is divided according to the degree of color as shown in Fig. 8, and the red part is the place that is most often seen. The purple point from the center of the screen to just above is what is actually tracking the eye gaze.

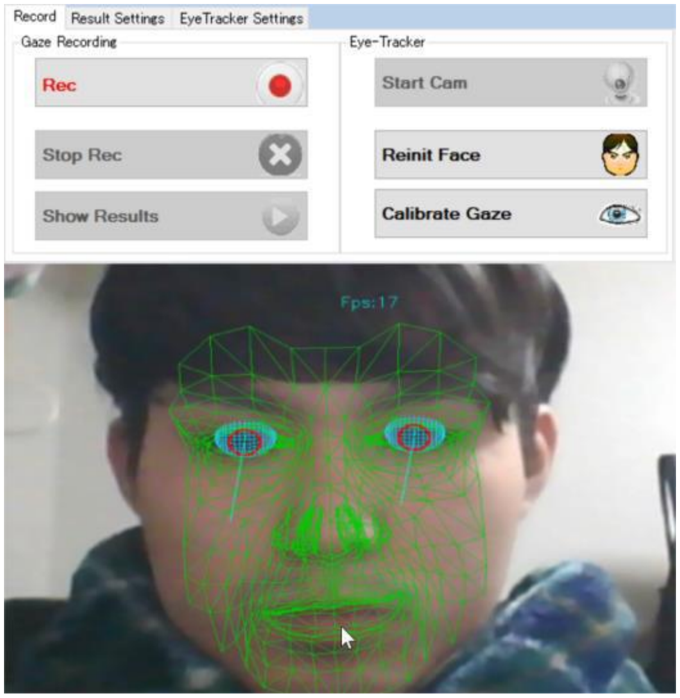


Fig. 7. Example of the Detected Line of Sight Vector and Gaze using Gaze Recorder.

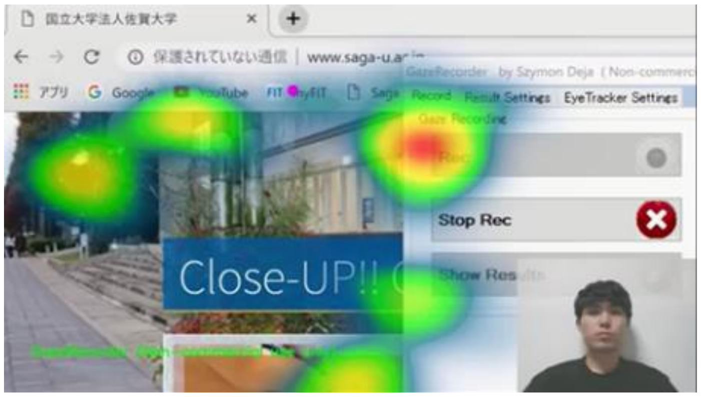


Fig. 8. Example of the Heat Map Derived from Gaze Recorder.

When performing eye tracking, the PC specifications described in the previous section enable real time eye tracking with almost no time lag.

Thus, the pedestrian gaze can be recognized. When the gaze to point to the autonomous car, then the car recognizes pedestrian's intention to across the street. Therefore, the car stops for the pedestrian with the sign shown in Fig. 2 for notifying the pedestrians with a comprehensive representation of face image displayed onto front glass of the car

IV. Conclusion

Method for eye-contact between autonomous car and pedestrian is proposed for pedestrian safety. The method allows to detect the pedestrian who would like to across a street through eye-contact between an autonomous driving car and the pedestrian. Through experiment, it is found that the proposed method does work well for finding such pedestrians and for noticing a sign to the pedestrians from the autonomous driving car with a comprehensive representation of face image displayed onto front glass of the car.

It is found that when performing pedestrian detection with YOLO and eye tracking with GazeRecorder together with face and eye detection with OpenCV, the PC specifications described in the previous section enable real time eye tracking with almost no time lag.

Thus, the pedestrian gaze can be recognized. When the gaze to point to the autonomous car, then the car recognizes pedestrian's intention to across the street. Therefore, the car stops for the pedestrian with the sign shown in Fig. 2 for notifying the pedestrians with a comprehensive representation of face image displayed onto front glass of the car.

Future research works are as follows:

In this experiment, just visible camera is used for pedestrian detection and face/eye detection as well as gaze estimation. Further experiments have to be done with Near Infrared: NIR cameras and NIR Light Emission Diode: LED for same objectives in a bad weather condition and in night time operations.

Acknowledgments

Authors would like to thank to Prof. Dr. Osamu Fukuda of Saga University for his valuable comments and suggestions for this study.

References

[1] International Transport Forum 2 rue André Pascal 75775 Paris Cedex 16 Franc https://cyberlaw.stanford.edu/files/publication/files/15CPB AutonomousDriving.pdf.

[2] Marcel Zolg, Techniocal University of Munich, https://www.mentor.com/mentor-automotive/autonomous

[3] KHALID BIN ISA Department of Computer Engineering, Faculty of Electric and Electronic Engineering, University College of Technology Tun Hussein Onn, 86400 Parit Raja https://www.ijee.ie/articles/Vol21-5/Ijee1674.pdf

[4] By Kersten Heineke, Philipp Kampshoff, Armen Mkrtchyan, and Emily Shao https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/self-driving-car-technology-when-will-the-robots-hit-the-road

[5] https://www.youtube.com/watch?v=QdYMOZRGdMw

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection"

[7] YOLO | Joseph Redmon, https://pjreddie.com/darknet/yolo/

[8] Darknet | Joseph Redmon, https://pjreddie.com/darknet/

[9] OpenCV (Open Source Computer Vision Library), https://opencv.org/

[10] OpenCV Face detection（Haar-like feature extraction）|Qiita, https://qiita.com/hitomatagi/items/04b1b26c1bc2e8081427

[11] GazeRecorder | GazeRecorder, http://gazerecorder.christiaanboersma.com/gazerecorder/

[12] Object detection, ryuji shirata，"Darknet on windows10, http://weekendproject9.hatenablog.com/about.

[13] Anaconda and JetBrains Join Forces to Launch 'PyCharm for Anaconda' https://www.anaconda.com/.

Authors Profile

Kohei Arai, He received BS, MS and PhD degrees in 1972, 1974 and 1982, respectively. He was with The Institute for Industrial Science and Technology of the University of Tokyo from April 1974 to December 1978 and also was with National Space Development Agency of Japan from January, 1979 to March, 1990. During from 1985 to 1987, he was with Canada Centre for Remote Sensing as a Post Doctoral Fellow of National Science and Engineering Research Council of Canada. He moved to Saga University as a Professor in Department of Information Science on April 1990. He was a councilor for the Aeronautics and Space related to the Technology Committee of the Ministry of Science and Technology during from 1998 to 2000. He was a councilor of Saga University for 2002 and 2003. He also was an executive councilor for the Remote Sensing Society of Japan for 2003 to 2005. He is an Adjunct Professor of University of Arizona, USA since 1998. He also is Vice Chairman of the Commission-A of ICSU/COSPAR since 2008. He received Science and Engineering Award of the year 2014 from the minister of the ministry of Science Education of Japan and also received the Bset Paper Award of the year 2012 of IJACSA from Science and Information Organization: SAI. In 2016, he also received Vikram Sarabhai Medal of ICSU/COSPAR and also received 20 awards. He wrote 34 books and published 520 journal papers. He is Editor-in-Chief of International Journal of Advanced Computer Science and Applications as well as International Journal of Intelligent Systsems and Applications. http://teagis.ip.is.saga-u.ac.jp/