Design and Analysis of Fast Driver's Fatigue Estimation and Drowsiness Detection System using Android

Widodo Budiharto, Welly Dwi Putra, Yudha Pratama Putra
School of Computer Science, Bina Nusantara University, Jakarta-Indonesia
Email:wbudiharto@binus.ac.id

***Abstract*—The purpose of doing this research is to design an application for fast drowsiness detection system, so the rate of traffic accidents due to negligence of the driver can be lowered. This research uses several methods such as the method of analysis that includes literature study, design method and testing method. Then we design and test on how accurate the given data when compared to the real condition. The results achieved from the Drowsiness Detection System application design based on Computer Vision are the parameters which affect the performance of application such as distance and direction of camera to the eyes, and the ability to detect drowsiness around 90%. The conclusion obtained in this research is that application supports the theory that the average duration of eye closure which above 400 ms were classified into sleepy, while below 400ms, driver declared in normal state. With the ability to detect drowsiness of the drivers, this application can reduce the rate of traffic accidents every year, by giving the increasing awareness of the driver against driver's fatigue**

***Index Terms*—drowsiness detection system, computer vision, android, driver's fatigue**

I. Introduction

Nowadays, there are many artificial intelligence development performed to help the safety of human life. One of them is drowsiness detection system. Drowsiness is something that happened frequently. We can feel sleepy either when we do an activity or not. However, we must pay attention for that. Feeling sleepy when we are in certain condition can be dangerous and can bring great suffer to many people, for example when we are driving.

A lot of factors can cause people to feel sleepy such as lack of rest, driving a car at night, long distance driving, or there is no partner who can accompany the driver which lead him/her to become bored and sleepy. The general solution for this problem is to ask someone to accompany us while driving to prevent drowsiness. However this solution can't be done everyday because sometimes the driver must drive alone.

Many research about drowniess detection system have utilized artificial intelligence such as [8-11], but they were not implemented on mobile devices. Therefore, we propose a framework and software to measure fatigue and to detect drowsiness of the driver using low cost Android mobile devices. The software that we made is designed to give a driver the information of fatigue and give him/her a warning so it can prevent incidents like traffic accidents. The software is based on Android application in order to be implemented in real condition, so we can reach the purpose of this research[4].

II. Related Theory

A. Haar Cascade Classifier

Viola-Jones framework has been widely used by researchers in order to detect the location of faces and objects in a given image. Face detection classifiers are shared by public communities, such as OpenCV[1][2]. Haar Cascade Classifier use AdaBoost at every node in cascade to study high detection level with multi-tree classifier rejection level at every node in cascade. This algorithm combines some innovative features, such as:

1. Use haar-like input feature, threshold that is used to sum and differentiate square regions from image.
2. Integral image technique that enable fast computation for square regions or regions that is rotated 45 degree. This data structure is used to make computation from Haar-like input feature faster.
3. Statistical Boosting to make binary node classification (yes/no) that characterized with high detection level and weak rejection level.
4. Organizing weak classifier nodes from a rejection cascade. In other words, first group from the classifiers is selected so best detection in image region consist of an object although enabling many mistakes in detection; the next classifier groups are the second best detection with weak level rejection; and so on. In testing, an object can be known if that object makes it through all cascade[3]. Haar-like input feature that are used by classifier are:

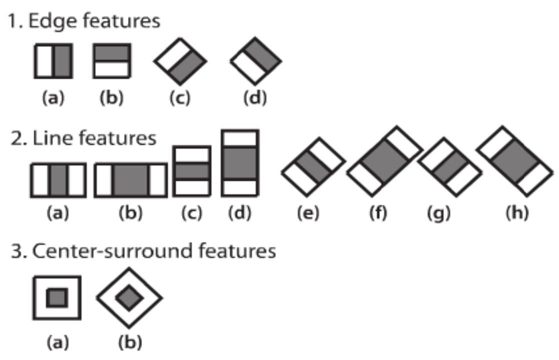


Figure 2 Haar-like Input Feature That are Used by Classifiers[3]

B. *Determination of Eye Condition*

To determine eyes' condition, first we have to get its position from eyelid. Eyelid is located between first intensity change and second intensity change show in the representation graphic below of intensity change[7].

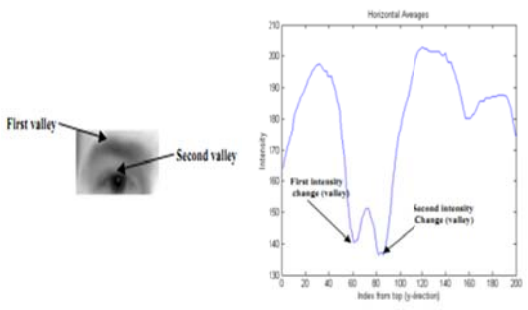


Figure 3 Graphic of Intensity Change

Therefore, to determine whether eye is open or not, it can be done with counting distance from both intensity change (valley). When eye is closed, the distance between two valleys will be bigger compared to when the eye is opened.

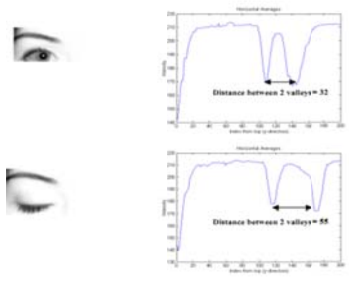


Figure 4 Comparison of Open Eye And Closed Eye condition

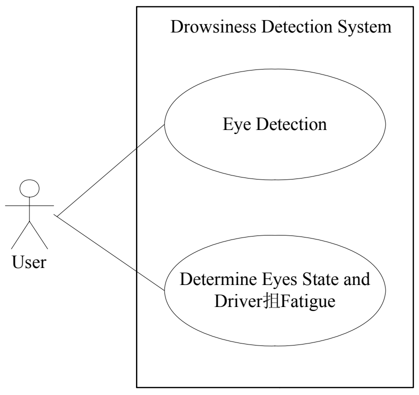
Based on Mandeep Singh's [9] research on the drowsiness parameter, the average duration of eye closure is 400ms and the minimum duration is 75ms. Therefore, if eye closure exceeds 400ms, then the person will be considered in drowsiness condition, as shown at table 1.

TABLE 1
DROWSINESS PARAMETER

Drowsiness	Description
Awake	Eye closure < $T_{drowsiness}$
Drowsy	Eye closure > $T_{drowsiness}$ and Eye closure < T_{skeep}
Sleeping	Eye closure >= T_{sleep}

III. DESIGN OF THE SYSEM

This Fatigue Estimation and Drowsiness Detection Application is built on Android Operation System, using Eclipse Juno with OpenCV library[6]. Capturing image is done with front camera of the mobile device. Below is the use case diagram of this application.



Drowsiness Detection System
Eye Detection
User
Determine Eyes State and Driver抯Fatigue

Figure 5 Use Case Diagram

Use case diagram (figure 5) description:

A. *Eye Detection*

In this section we perform eye detection from image that we have already captured. This eye detection is useful for determining fatigue level of the driver.

Flow of Events:

1. User opens application from mobile device.
2. Camera captures image.
3. System gets the image that was captured by camera.
4. System performs preprocessing to the image.
5. System detects eyes.

B. *Determine Eyes State and Driver's Fatigue*

In this section, system analyses eyes' condition and determine whether the driver is in drowsiness condition or not.

Flow of events:

1. System analyses eye region that has been detected.
2. System checks driver's condition.
3. System will give an output warning if the driver is in drowsiness condition.

This is the activity diagram of this application:

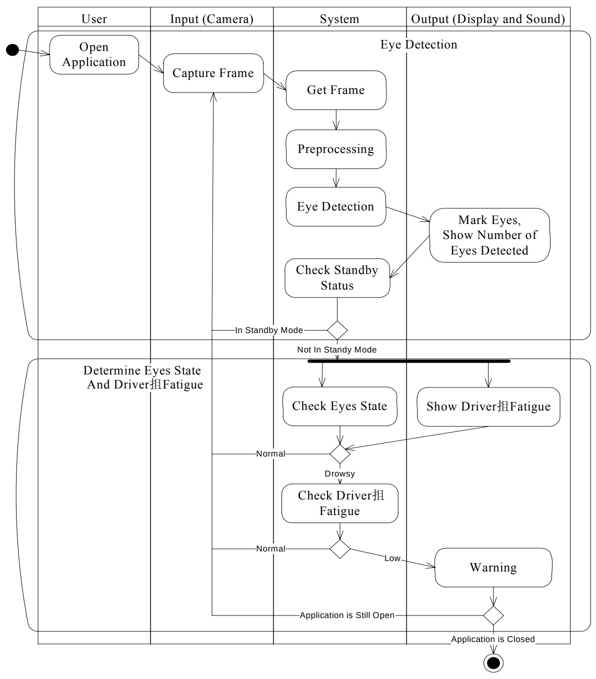


Figure 6 Activity Diagram

Activity diagram description (figure 6):

1. User opens drowsiness detection application.
2. Camera starts to capture image.
3. System takes image that has been captured by camera.
4. Then system performs preprocessing to the image. The preprocessing step will change RGB image to grayscale image.
5. After preprocessing, system performs eye detection of the image. This eye detection uses Viola-Jones methods. Template to do this eye detection, which has been prepared by OpenCV and is ready to be used, is called haarcascade_eye.xml.
6. If the eyes have been detected successfully, then green squares will be shown on the screen, indicating the eyes region.
7. System checks whether system is in standby status or not. If yes, then system will go back to process no 2. If not, then system will go to the next process.
8. After getting data of the eyes region, the system then checks eyes' condition (whether they are normal or not). System will also show user's fatigue level. To count how wide the eye is opened, system counts the ration of black pixel with white one. Eye region that needs to be counted is the middle region of eye.
   Afterward, system will do binarization on the eyes region. With binarization, the image will only consist of two colors, black and white. Eyeball area will be marked black while other parts , such as eyelid and eye bag, will be marked white.

After doing binarization, system will do an iteration at middle of eye region to count how much white and black pixels. After that, system will be able to count the ratio between black pixel and white pixel. After doing some experiments, we got to the conclusion that the ration must be higher than 20% for normal condition. If the system determinates that eye condition is not normal (closed/ only half opened), system will sound an alarm.

$$R = \frac{B}{B+W} \times 100\% \qquad (7)$$

if R > 20% then Normal
Else Not Normal

where:
R = ratio of black pixels
B = the number of black pixels
W = the number of white pixels

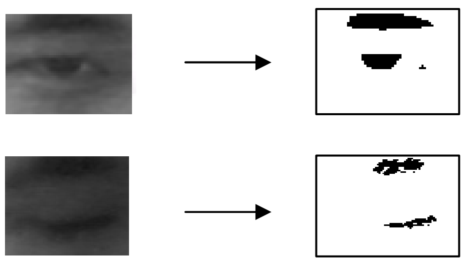


Figure 7 The Comparison Between Eye Opened Image And Eye Closed Images [12]

In Figure 7, we can see that in eye-opened image, the number of black pixels is higher than the number of black pixels in eye-closed image. Thus, determining eyes' condition can be done with counting ration of black pixel in the image. If user's eyes are normal(not in drowsiness condition), then system will go back to process no 2 while if user's eyes are not normal(drowsy), then system will decrease user fatigue level. If user fatigue level is still normal, system will go back to process no 2 while if user fatigue level is below normal, system will sound the alarm. If user does not close this application, the system will go back to process no 2 while if user close this application, system will be stopped.

IV. Experimental Result

We conducted the experiment using low cost Samsung Android phone, OpenCV for Android, Eclipse and Android SDK. The figure shown below is the user interface of the system:

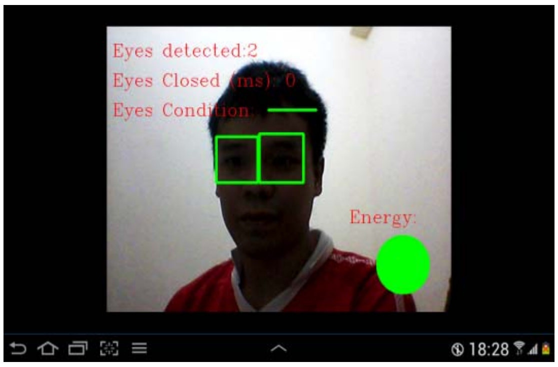


Figure 8 User Interface of Drowsiness Detection System

In Figure 8, there are information about how many eyes have been detected, for how long have eyes been closed, and eyes' condition (that is presented with green horizontal line). There are green squares that show eye regions that have been detected. User fatigue level is presented with green circle at the right corner. The green color shows that the user fatigue level is still high.

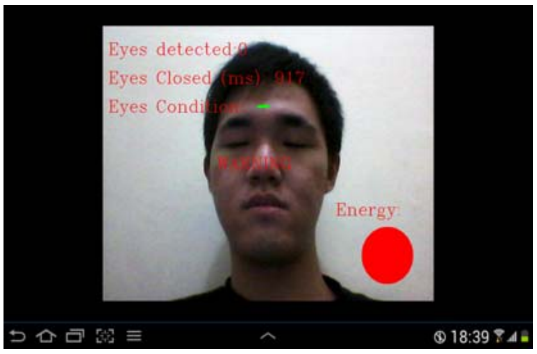


Figure 9 User Interface When User In Drowsiness Condition

In Figure 9, there are information about how many eyes have been detected, for how long have eyes been closed, and eyes' condition (that is presented with green horizontal line). Also there is the user fatigue level which is presented with red circle at the right corner. The red color shows that the user fatigue level is low. There is a warning sign on the middle of the screen. System will also sound the alarm to warn user.

4.1 Test 1

This test is done by 3 different people. The purpose of this test is to get the best range between camera and eyes so this application can detect eye better.

TABLE 2
THE RESULT OF TESTING 1

Sample	Person 1	Person 2	Person 3
Experiment 1 (40 cm)	Success	Success	Success
Experiment 2 (45 cm)	Success	Success	Success
Experiment 3 (50 cm)	Success	Success	Success
Experiment 4 (55 cm)	Success	Success	Fail
Experiment 5 (60 cm)	Fail	Success	Success

4.2 Test 2

This test is done by a sleepy person. The purpose of this test is to measure the accuracy of application in drowsiness detection. If the eye is closed for more than 400ms, then the driver is considered sleepy. Here is the table result.

TABLE 3
THE RESULT OF TESTING 2

Sample	Person 1
Experiment 1	535 ms
Experiment 2	781 ms
Experiment 3	666 ms
Experiment 4	386 ms
Experiment 5	744

4.3 Test 3

This test is done by a sleepy person. The purpose of this test is to measure accuracy of this application in drowsiness detection. The parameter used by this test is the condition of eye. If the eye is smaller than normal, then the driver is considered sleepy.

TABLE 4
THE RESULT OF TESTING 3

Sample	Person 1 Eye Condition
Experiment 1	Not Normal
Experiment 2	Normal
Experiment 3	Not Normal
Experiment 4	Not Normal
Experiment 5	Not Normal

4.4 Test 4

This test is done from below the eyes. The purpose of this test is to measure the success of application when detect driver's eyes.

TABLE 5
THE RESULT OF TESTING 4

Sampel	From Below
Experiment 1	Success
Experiment 2	Success
Experiment 3	Success
Experiment 4	Failed
Experiment 5	Success

Before detecting eyes, it's recommended to increase the image quality first. However, today's android technology can't support a lot of image processing because of the hardware limitation. Searching and implementing another algorithm that is faster and more accurate in drowsiness detection will need to be done. Development of this application in other platforms (such as Blackberry, Apple, and Windows Phone) is also important so this application can reach to more user.

V. CONCLUSION

The conclusions that can be drawn after doing analysis, design, test, and evaluation from driver's fatigue estimation and drowsiness detection system design are described as follow. With this application's accuracy rate in drowsiness detection, this application can be implemented in the real time to reduce traffic accidents rate due to drowsy drivers and it can also help drivers to stay awake when driving by giving a warning when the driver is sleepy. Factors such as distance and direction from camera to eyes will affect the image acquisition. Application hardly detects eyes when the distance is too near or too far. Application works its best when the distance from the eyes is around 50 cm. The best direction is from below eyes because the light comes from back. This application supports the theory that average eye closure time of sleepy person is more than 400ms. While the eye closure time of normal person is below 400ms.

ACKNOWLEDGMENT

We say thank you very much for the Bina Nusantara University that facilitated the facility to conduct this research.

REFERENCES

[1] Padilla R., Evaluation of Haar Cascade Classifiers Designed for Face Detection, Journal of WASET, Vol. 64, pp.362-365.

[2] Bradski, G., & Kaebler, A. 2008. Learning OpenCV. O'Reilly, USA. ISBN: 978-0-596-51613-0, pp:16.

[3] Viola P., Jones, M., Robust Object Detection, 2nd International Workshop on Computational Theories of Vision and sampling, Canada, 2001.

[4] Lee, W. M. 2011. *Beginning Android™ Application Development*. Wiley Publishing, Indiana. ISBN: 978-1-118-01711-1, pp: 2-7.

[5] OpenCV Dev Team. 2012. Introduction to OpenCV.

[6] http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/O4A_SDK.html (Accessed on November 7, 2012)

[7] Pamar, N. 2002. Drowsy Driver Detection System Design Project. Department of Electrical and Computer Engineering. Canada.

[8] Russel, J. S., & Norvig, P. 1995. Artificial Intelligence, A Modern Approach. Prentice-Hall, Inc , New Jersey. ISBN: 0-13-103805-2, pp. 4-8.

[9] Singh, M., & Kaur, G. 2012. Drowsy Detection On Eye Blink Duration Using Algorithm. International Journal of Emerging Technology and Advanced Engineering, Vol. 2, pp. 363 – 365.

[10] Uke, N. J., Thool, Dr. R. C., & Dhotre, P. S. 2012. Drowsiness Detection – A Visual System Driver Support. International Journal of Electronics Communication and Computer Engineering Vol. 3. pp. 29-33.

[11] Vats, E. M., Garg, E. & Anil. 2012. Detection and Security System for Drowsy Driver By Using Artificial Neural Network Technique. International Journal of Applied Science and Advance Technology Vol. 1, pp. 39-43.

[12] Yadav, V. Makhija, D. Savant, S. Dodani, N. & Theckedath, K. D. 2012. Driver Drowsiness Detection System. International Journal of Computer Applications (IJCA). pp. 11-13.

Widodo Budiharto is a senior Lecturer at Bina Nusantara Jakarta. He got PhD from Institute of Technology Sepuluh Nopember Surabaya (ITS)- Indonesia at 2011. His interests are in Computer Vision, Robotics and Mobile Devices Development.