Journal of Technology and Informatics (JoTI)
Vol 1, No.1, Oktober Tahun 2019
E-ISSN 2686-6102 DOI: https://doi.org/10.37802/joti.v1i1.1

Implementasi Sistem Deteksi Mata Kantuk Berdasarkan *Facial Landmarks Detection* Menggunakan Metode *Regression Trees*

Andrea H.A.P. Perdana[1] **Susijanto Tri Rasmana**[2] **Heri Pratikno**[3]
Program Studi/Jurusan Teknik Komputer Institut Bisnis dan Informatika Stikom Surabaya
Email: [1] andrea.dcladea28@gmail.com, [2] susyanto@stikom.edu, [3] heri@stikom.edu

Abstrak: Kecelakaan lalu lintas merupakan salah satu dampak negatif dari kemajuan teknologi dibidang transportasi. Faktor manusia merupakan salah satu faktor dengan persentase tertinggi pada peningkatan kecelakaan lalu lintas. Salah satu contoh faktor manusia adalah kelelahan dalam berkendara. Kelelahan berkendara akan mengakibatkan pengemudi mengalami kantuk, oleh karena itu untuk meminimalisir kecelakaan yang dikarenakan kelelahan saat berkendara maka diperlukan sebuah sistem yang menggunakan *alarm* untuk mendeteksi mata kantuk secara *real time*. Salah satu penelitian sebelumnya untuk mendeteksi mata kantuk, sistem yang dibuat menggunakan metode segmentasi warna. Pada penelitian ini akan dibuat sistem deteksi mata kantuk yang berdasarkan *Facial Landmarks Detection* menggunakan metode *Regression Trees* yang diimplementasikan ke dalam *Raspberry Pi* 3 model B. *Input* dari sistem ini berupa video yang direkam dari PiCamera secara *real time*. *Output* dari sistem ini menggunakan *buzzer* sebagai *alarm* untuk memberikan peringatan bahwa pengemudi terdeteksi mengantuk. Sistem yang dibuat pada penelitian ini menunjukan bahwa dapat mendeteksi mata kantuk dengan baik. Berdasarkan hasil pengujian, sistem dapat mendeteksi mata sebesar 93.3%, kedipan mata sebesar 96.7%, dan sudut miring wajah sebesar 95%.

Kata Kunci: Mata Kantuk, *Facial Landmarks Detection*, *Regression Trees*.

Abstract: *Traffic accidents are one of the negative impacts of technological advances in the field of transportation. Human factor is one of the factors with the highest percentage in increasing traffic accidents. One example of human factors is fatigue in driving. Fatigue driving will cause the driver to experience drowsiness, therefore to minimize accidents due to fatigue while driving, a system that uses alarms is needed to detect sleepiness in real time. One of the previous studies to detect sleepy eyes, a system created using the color segmentation method. In this paper, a sleep eye detection system based on Facial Landmarks Detection will be made using the Regression Trees method that is implemented in the Raspberry Pi 3 model B. The input of this system is recorded video from PiCamera in real time. The output of this system uses the buzzer as an alarm to give a warning that the driver is detected drowsy. The system created in this study shows that it can detect eye sleep well. Based on The test results, the system can detect eyes by 93.3%, eye blinking by 96.7%, and face oblique angle of 95%.*

Keywords: *Eye Sleep, Facial Landmarks Detection, Regression Trees*

PENDAHULUAN

Kemajuan ilmu teknologi semakin berkembang pesat seiring berjalannya waktu. Perkembangan teknologi ini sangat membantu aktivitas sehari – hari di lingkungan masyarakat. Salah satu contohnya adalah perkembangan teknologi di bidang transportasi. Masyarakat semakin mudah untuk melakukan perjalanan jauh dengan menggunakan transportasi masa kini. Namun pada kemajuan teknologi ini, membawa dampak negatif bagi masyarakat di lingkungan sekitar salah satunya jumlah kecelakaan lalu lintas yang semakin meningkat di setiap tahunnya. Menurut Badan Penelitian dan Pengembangan Kesehatan, persentase kecelakaan transportasi darat meningkat dari

25.9% ditahun 2008 menjadi 47.7% di tahun 2013. Pada tahun 2010 hingga 2014, persentase korban meninggal akibat kecelakaan lalu lintas berkisar 17-22% [1].

Faktor tertinggi penyebab peningkatan jumlah kecelakaaan lalu lintas adalah faktor manusia, dimana memiliki persentase 69.7% [2]. Salah satu contoh faktor manusia adalah kelelahan dalam berkendara. Pada [3] lebih dari 25% penyebab kecelakaan merupakan kelelahan yang mengakibatkan pengendara mengalami kantuk saat sedang berkendara. Menurut data Badan Pusat Statistik Indonesia (2016), jumlah kecelakaan dari tahun 2012 hingga 2016 berkisar 100.000 kejadian dengan kerugian hingga 200 juta rupiah.

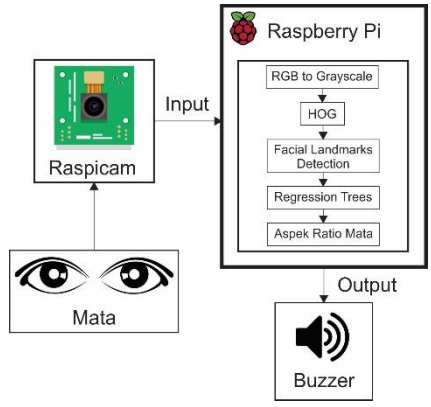
Kantuk adalah suatu kondisi manusia yang disebabkan oleh kurangnya istirahat atau tidur. Manusia dewasa membutuhkan tidur selama 8 jam setiap malamnya agar mencapai kinerja optimal [4] . Kekurangan tidur dapat mengakibatkan kantuk sehingga terjadi peningkatan jumlah kedipan mata hingga terjadi adanya *microsleeps* [5]. Pada kondisi mengantuk, seseorang akan mengalami peningkatan 20% dari frekuensi kedipan mata per menit. Selain itu, seseorang akan mengalami *microsleeps* dengan durasi penutupan mata berkisar 0,5 detik atau lebih [3].

Hingga saat ini, khususnya penelitian deteksi mata kantuk menggunakan pengolahan citra digital sudah banyak dilakukan.

Berdasarkan permasalahan – permasalahan tersebut, maka penelitian selanjutnya akan dibuat sebuah sistem deteksi mata kantuk. Sistem ini akan melakukan emantauan kondisi pengemudi dengan cara melakukan perekaman wajah yang kemudian akan diproses dengan pengolahan citra digital menggunakan metode *regression trees* pada *Raspberry Pi*.

METODE PENELITIAN

Pada sistem deteksi mata kantuk ini menggunakan *Pi Camera* untuk mengambil data berupa citra gambar yang kemudian diproses oleh *Raspberry Pi* menggunakan metode *regression trees* berdasarkan *facial landmarks detection* sehingga dapat mengetahui hasil data merupakan mata yang mengantuk atau tidak. Apabila hasil proses terdeteksi mata kantuk, maka sistem akan mengaktifkan *buzzer* untuk memberikan peringatan kepada pengemudi.



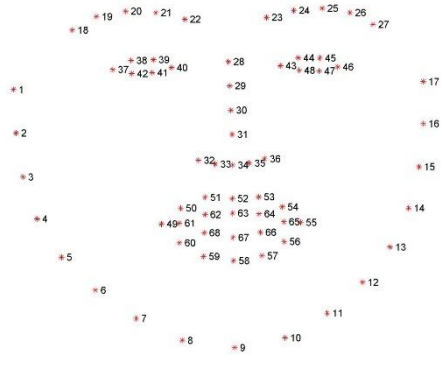
Gambar 1. Blok Diagram Sistem Mata Kantuk

Histogram of Oriented Gradients

Histogram of Oriented Gradients merupakan metode yang digunakan untuk mendeteksi suatu objek citra gambar. Untuk memperoleh suatu hasil, citra gambar akan dibagi menjadi beberapa sel dan setiap selnya akan dihitung sebagai *histogram of oriented gradients.*

Facial Landmarks Detection

Facial landmarks detection merupakan keluaran terstruktur yang memiliki tujuan untuk memprediksi bentuk geometri yang diperoleh dari sebuah data berupa citra wajah. Jumlah titik *landmark* bergantung pada dataset yang digunakan. Pada sistem deteksi mata kantuk ini, menggunakan salah satu model *facial landmarks detection* yaitu 68 *landmarks*.



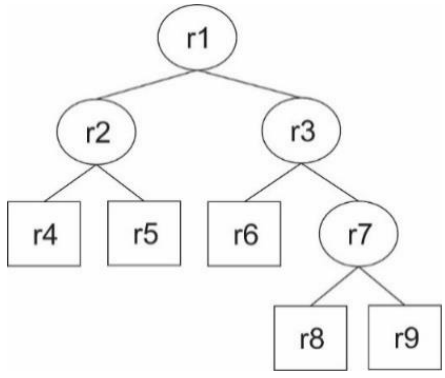
Gambar 2. Penanda Koordinat Bentuk Wajah

Regression Trees

Classification and Regression Trees merupakan salah satu metode dari teknik pohon keputusan. Pada [6] *regression trees* merupakan metode dengan variabel respon yang memiliki bentuk kontinu, sedangkan *classification trees* merupakan metode dengan variabel respon yang memiliki bentuk skala kategorik.

Regression trees merupakan metode percabangan rekursif biner dimana kumpulan data (disebut simpul) dicabang menjadi 2 bagian cabang (disebut anak simpul) hingga menghasilkan simpul yang tidak dapat dicabang lagi (disebut simpul akhir).

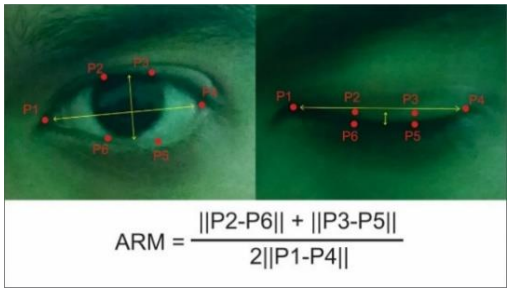
Gambar 3. Struktur *Regression Trees*



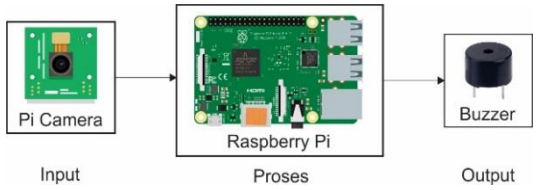
Aspek Rasio Mata

Aspek rasio mata digunakan untuk menentukan kedipan suatu mata. Rumus perhitungan aspek rasio mata ditunjukan pada gambar 4.

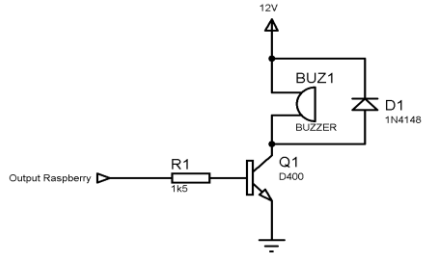
Gambar 4. Perhitungan Aspek Rasio Mata



Perancangan Perangkat Keras



Gambar 5. Blok Diagram Perangkat Keras Sistem Deteksi Mata Kantuk

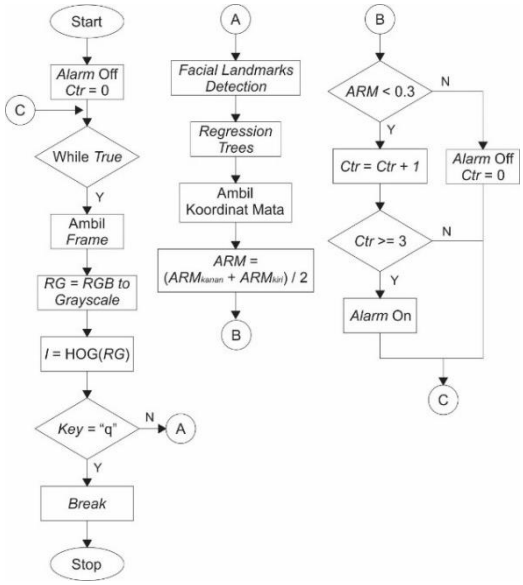


Gambar 6. Rangkaian Output Raspberry Pi 3

Dari gambar 6, dapat dijelaskan bahwa data citra gambar yang diperoleh dari kamera akan diproses oleh *Raspberry Pi* menggunakan *Python* dan *OpenCV* dengan metode *regression trees* untuk menentukan koordinat mata. Setelah menemukan koordinat mata, maka nilai koordinat mata akan diolah menggunakan rumus aspek rasio mata untuk menentukan hasil kedipan mata. Apabila hasil aspek rasio mata yang diperoleh berada dibawah *threshold* yang telah ditentukan oleh penulis selama kurun waktu 1 detik, maka *buzzer* akan aktif sebagai notifikasi *alarm*. Pada sistem ini, *buzzer* diletakkan di GPIO pin 18.

Perancangan Program

Flowchart Sistem Deteksi Mata Kantuk



Gambar 7. *Flowchart* Sistem Deteksi Mata Kantuk

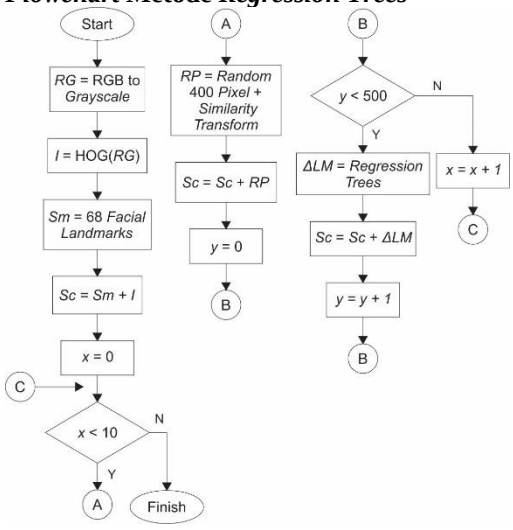
Pada flowchart diatas dapat dijelaskan bagaimana sistem deteksi mata kantuk bekerja. Awal mula sistem mendeklarasikan *alarm* pada kondisi *off* dan *counter* = 0. Sistem akan melakukan *infinite looping* terus menerus hingga mendapatkan *input* keyboard berupa karakter "q" dari pengemudi. Kemudian sistem mengambil 1 *frame* dari kamera untuk diproses lebih lanjut. Proses selanjutnya adalah mengubah data citra gambar pada *frame* yang telah diambil dari RGB menjadi *grayscale*.

Dari citra gambar *grayscale* akan diolah untuk mengambil area wajah pengemudi menggunakan metode *Histogram of Oriented Gradients* untuk memperkecil proses komputasi selanjutnya. Batas koordinat tersebut kemudian akan digunakan untuk pembatas pemetaan 68 penanda koordinat bentuk wajah. Kemudian titik – titik koordinat tersebut akan digeser untuk pemetaan bentuk wajah menggunakan *looping regressor* selama beberapa kali. *Looping regressor* ini digunakan untuk memberikan penanda piksel pada citra wajah dengan menggunakan metode *regression trees*.

Kemudian sistem mengambil koordinat mata pada 68 penanda koordinat bentuk wajah dimana mata memiliki koordinat 37 hingga 42 untuk mata kiri dan 43 hingga 48 untuk mata kanan. Nilai koordinat (x,y) pada koordinat mata akan diolah menggunakan rumus aspek rasio mata untuk menentukan kedipan mata.

Apabila nilai ARM kurang dari 0.3 selama 1 detik atau pada sistem memiliki nilai counter lebih besar sama dengan 3, maka *alarm buzzer* akan berbunyi selama mata nilai ARM kurang dari 0.3. Ketika nilai ARM lebih besar dari 0.3, maka *alarm buzzer* akan mati dan *counter* akan direset.

Flowchart Metode *Regression Trees*



Gambar 8. *Flowchart Regression Trees*

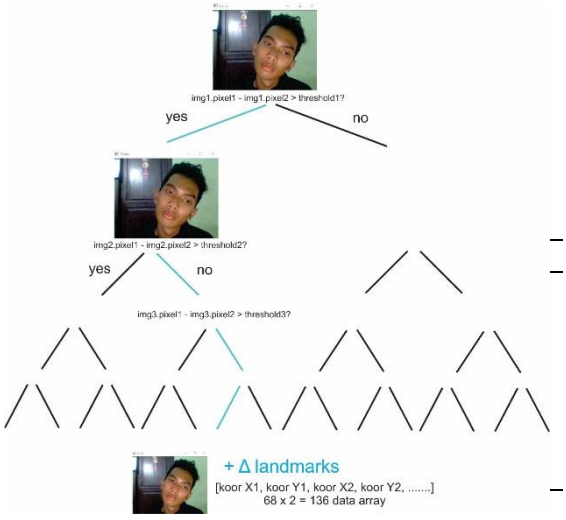
Flowchart Regression Trees mula – mula mengambil data citra gambar yang sudah dikonversi ke citra gambar *grayscale*. Kemudian citra gambar tersebut diolah menggunakan metode *histogram of oriented gradients* untuk memperkecil area komputasi. Area komputasi ini berbentuk persegi yang berada di area wajah pengemudi. Pada area persegi ini akan dipetakan 68 penanda koordinat bentuk wajah yang akan digunakan untuk mendeteksi bagian – bagian yang ada pada wajah pengemudi.

Kemudian dilakukan *looping* X sebanyak 10 kali untuk memberikan hasil yang regresi yang maksimal pada 68 penanda koordinat bentuk wajah. Pada setiap *looping* X, sistem akan melakukan penempatan 400 piksel secara acak menggunakan *similarity transform* dari citra gambar *Sm* ke citra gambar *Sc*. Untuk *looping* X=1 digunakan untuk *data training* yang pertama. Selanjutnya untuk *looping* X>=2 hingga X<=10 digunakan untuk *data training*

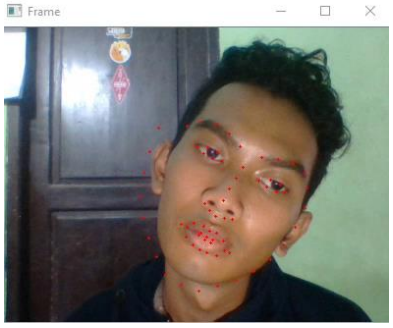
with dlib untuk membandingkan beberapa data yang sudah ada pada *library dlib*.

Dari citra gambar *Sc*, sistem akan melakukan *looping* Y yang digunakan untuk membandingkan selisih perbedaan piksel dengan nilai *threshold* pada setiap percabangannya. Hasil dari perbandingan ini memiliki 16 kemungkinan yang akan dijadikan nilai dari selisih *landmark* untuk setiap *looping* Y. Kemudian citra gambar *Sc* akan ditambahkan dari nilai hasil selisih *landmark*.

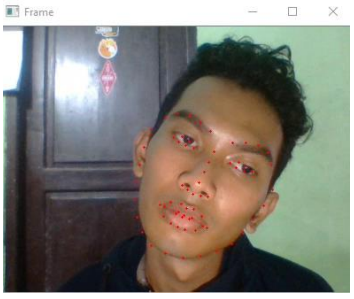
Data *Training* 1



Gambar 9. Data *Training* 1



Gambar 10. Citra Gambar Pertama



Gambar 11. Citra Gambar Hasil

Dari gambar 9, dapat dijelaskan bahwa *data training* ini merupakan implementasi dari metode *regression trees* dimana pengambilan keputusan menggunakan beberapa percabangan kondisi sehingga membentuk pohon keputusan. Mula – mula pada citra gambar melakukan 400 *random pixel*. Kemudian dari 400 *pixel* tersebut, dilakukan *random* kembali untuk diambil 2 data *pixel* untuk diolah lalu dibandingkan dengan *random threshold* citra gambar. Pada sistem ini memungkinkan menghasilkan 16 hasil data Δ *landmarks*. Hasil data tersebut berupa koordinat titik 68 *facial landmarks* yang terdiri dari x dan y, sehingga pada array data Δ *landmarks* memiliki 136 data.

Data *Training with Dlib*

Data training with Dlib secara konsep sama dengan *data training* 1, hanya saja *threshold* yang digunakan bukan dari hasil *random threshold* namun diambil dari maksimum korelasi perbandingan dengan *Helen Data* yang ada pada *Dlib.*

Tabel 1. Perhitungan *Threshold With Dlib*

f1-f1	f1-f2	…	f400-f400	Gt - Sc	rand (136)	d
p1,i1	p2,i1	…	p16000, i1	Δ1	Δr	d1
p1,i2	p2,i2	…	p16000, i2	Δ2	Δr	d2
…	…	…	…	…	…	…
p1,in	p2,in	…	p16000, in	Δn	Δr	dn

Dari tabel perhitungan *threshold with dlib* diatas, dapat dijelaskan bahwa Δ1 diambil dari operasi pengurangan citra gambar antara *ground truth* dengan *current shape*. Hasil tersebut

kemudian akan dikalikan vektor dengan random *threshold* yang menghasilkan nilai d. Selanjutnya untuk data ke dua hingga data N, dilakukan dengan cara yang sama. Kemudian dilakukan operasi perhitungan maksimum korelasi dari nilai d.

HASIL DAN PEMBAHASAN

Hasil Perbandingan Metode *Histogram of Oriented Gradients* Dengan *Haar Cascade*

Untuk mengetahui perbandingan akurasi deteksi, *frame per second*, dan tingkat ketahanan pergerakan antara metode HOG dengan *Haar Cascade*.

Tabel 2. Hasil Perbandingan HOG dengan *Haar Cascade*

Parameter	HOG	*Haar Cascade*
Frame Per Second	2-3 fps	5-6 fps
Menoleh dengan Sudut 45°	Berhasil	Gagal
Menoleh dengan Sudut 90°	Gagal	Gagal
Kemiringan Wajah	Berhasil	Gagal
Tingkat Ketahanan Pergerakan	Berhasil	Berhasil

Dari tabel di atas, dapat dijelaskan bahwa metode *Histogram Of Oriented Gradients* memiliki tingkat akurasi deteksi yang lebih baik daripada *Haar Cascade* namun memiliki kelemahan pada *frame per second* dikarenakan komputasi yang terlalu banyak dan kompleks. Sehingga dari tabel tersebut dapat disimpulkan bahwa pada sistem ini lebih baik menggunakan metode *Histogram Of Oriented Gradients* karena memiliki tingkat deteksi yang lebih akurat.

Hasil Pengujian Tingkat Akurasi Deteksi Mata

Pada pengujian tingkat akurasi deteksi mata bertujuan untuk mengetahui apakah sistem deteksi mata kantuk ini dapat mendeteksi mata dalam beberapa kondisi secara akurat.

Tabel 3. Hasil Pengujian Tingkat Akurasi Deteksi Mata

Data Ke-	*Frame 1*	*Frame 2*	*Frame 3*
1	Berhasil	Berhasil	Berhasil
2	Berhasil	Berhasil	Berhasil
3	Berhasil	Berhasil	Gagal
4	Berhasil	Berhasil	Berhasil
5	Berhasil	Gagal	Berhasil
6	Berhasil	Berhasil	Berhasil
7	Berhasil	Berhasil	Berhasil
8	Berhasil	Berhasil	Berhasil
9	Berhasil	Berhasil	Berhasil
10	Berhasil	Berhasil	Berhasil

Dari hasil tabel diatas, dapat dijelaskan bahwa dari 30 data *sample* uji coba ditemukan beberapa data yang tidak dapat mendeteksi mata. Sehingga persentase keberhasilan adalah 96.67%.

Gambar 12. Contoh Gagal Dari Hasil Pengujian Tingkat Akurasi Deteksi Mata

Gambar 13. Contoh Berhasil Dari Hasil Pengujian Tingkat Akurasi Deteksi Mata

Hasil Pengujian Tingkat Sensitif Kedipan Mata

Pada pengujian tingkat sensitif kedipan mata bertujuan untuk mengetahui apakah sistem dapat mendeteksi kedipan mata, mata menuju tertutup, maupun mata setengah tertutup. Pada pengujian ini, apabila mata menuju tertutup maka sistem akan melakukan perhitungan pada *counter*.

Tabel 4. Hasil Pengujian Tingkat Sensitif Kedipan Mata

Data	*Frame* 1	*Frame* 2	*Frame* 3
Data 1	Berhasil	Berhasil	Gagal
Data 2	Berhasil	Berhasil	Berhasil
Data 3	Berhasil	Berhasil	Berhasil
Data 4	Berhasil	Berhasil	Berhasil
Data 5	Berhasil	Berhasil	Berhasil
Data 6	Berhasil	Berhasil	Berhasil
Data 7	Berhasil	Berhasil	Berhasil
Data 8	Berhasil	Berhasil	Berhasil
Data 9	Berhasil	Berhasil	Berhasil
Data 10	Berhasil	Berhasil	Berhasil

Dari hasil tabel diatas, dapat dijelaskan bahwa dari 30 data *sample* uji coba ditemukan beberapa data yang tidak dapat mendeteksi kedipan mata. Sehingga persentase keberhasilan adalah 96.67%.

Gambar 14. Contoh Gagal Dari Hasil Pengujian Tingkat Sensitif Kedipan Mata

Gambar 15. Contoh Gagal Dari Hasil Pengujian Tingkat Sensitif Kedipan Mata

Hasil Pengujian Sudut Miring Wajah

Pada pengujian sudut miring wajah bertujuan untuk mengetahui apakah sistem dapat mendeteksi wajah dan mata ketika seorang pengendara sedang menggerakan kepalanya ke beberapa sudut dari arah kamera.

Tabel 5. Hasil Pengujian Sudut Miring Wajah

Data	*Frame* 1	*Frame* 2
Data 1	Gagal	Berhasil
Data 2	Berhasil	Berhasil
Data 3	Berhasil	Berhasil
Data 4	Berhasil	Berhasil
Data 5	Berhasil	Berhasil
Data 6	Berhasil	Berhasil
Data 7	Berhasil	Berhasil
Data 8	Berhasil	Berhasil
Data 9	Berhasil	Berhasil
Data 10	Berhasil	Berhasil

Dari hasil tabel diatas, dapat dijelaskan bahwa dari 20 data *sample* uji coba ditemukan beberapa data yang tidak dapat mendeteksi kedipan mata. Sehingga persentase keberhasilan

Gambar 16. Contoh Gagal Dari Hasil Pengujian Sudut Miring Wajah

Gambar 17. Contoh Berhasil Dari Hasil Pengujian Sudut Miring Wajah

Hasil Perbandingan *Running Program* di *Raspberry Pi* Dengan *Personal Computer*

Perbandingan *running program* di *Raspberry Pi* dengan *Personal Computer* bertujuan untuk mengetahui apakah sistem yang berjalan pada *Raspberry Pi* dapat dikatakan *real time*, dimana akan dibandingkan dengan *Personal Computer* yang sistemnya sudah berjalan secara *real time*.

Tabel 6. Hasil Perbandingan Kedipan Mata di *Raspberry Pi* Dengan *Personal Computer*

Data Ke-	PC	*Raspberry*	*Delay* (detik)	Ket
1	Terdeteksi	Terdeteksi	0.42	Ok
2	Terdeteksi	Terdeteksi	0.27	Ok
3	Terdeteksi	Terdeteksi	0.28	Ok
4	Terdeteksi	Terdeteksi	0.36	Ok
5	Terdeteksi	Terdeteksi	0.49	Ok
6	Terdeteksi	Terdeteksi	0.31	Ok
7	Terdeteksi	Terdeteksi	0.34	Ok
8	Terdeteksi	Terdeteksi	1.17	Ok
9	Terdeteksi	Terdeteksi	1.33	Ok

Tabel 7. Hasil Perbandingan Deteksi Kantuk I di *Raspberry Pi* Dengan *Personal Computer*

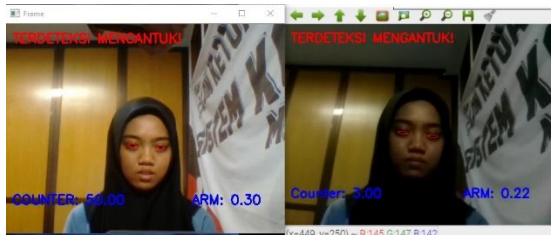
Data Ke-	PC	*Raspberry*	*Delay* (detik)	Ket
1	Terdeteksi	Terdeteksi	0.11	Ok
2	Terdeteksi	Terdeteksi	0.13	Ok
3	Terdeteksi	Terdeteksi	0.16	Ok
4	Terdeteksi	Terdeteksi	0.18	Ok
5	Terdeteksi	Terdeteksi	0.21	Ok
6	Terdeteksi	Terdeteksi	0.11	Ok
7	Terdeteksi	Terdeteksi	0.2	Ok
8	Terdeteksi	Terdeteksi	0.18	Ok
9	Terdeteksi	Terdeteksi	0.14	Ok

Tabel 8. Hasil Perbandingan Deteksi Kantuk II di *Raspberry Pi* Dengan *Personal Computer*

Data Ke-	PC	*Raspberry*	*Delay* (detik)	Ket
1	Terdeteksi	Terdeteksi	0.14	Ok
2	Terdeteksi	Terdeteksi	0.17	Ok
3	Terdeteksi	Terdeteksi	0.09	Ok
4	Terdeteksi	Terdeteksi	0.12	Ok
5	Terdeteksi	Terdeteksi	0.27	Ok
6	Terdeteksi	Terdeteksi	0.05	Ok
7	Terdeteksi	Terdeteksi	0.26	Ok
8	Terdeteksi	Terdeteksi	0.2	Ok
9	Terdeteksi	Terdeteksi	0.22	Ok

Dari ketiga tabel di atas, dapat dijelaskan bahwa perbandingan pengujian kedipan mata pada *Raspberry Pi* dengan *Personal Computer* menghasilkan *output* yang sama dengan persentase 100%, namun *Raspberry Pi* memiliki *delay* dengan rata-rata berkisar 0.55 detik terhadap *Personal Computer*. Sama halnya dengan pengujian deteksi kantuk, dimana *Raspberry Pi* menghasilkan *output* yang sama dengan *Personal Computer* yang memiliki persentase 100% dan rata-rata *delay* 0.16 detik.

Gambar 18. Contoh Hasil Perbandingan Kedipan Mata di *Raspberry Pi* (Kanan) Dengan *Personal Computer* (Kiri)



Gambar 19. Contoh Hasil Perbandingan Deteksi Kantuk di *Raspberry Pi* (Kanan) Dengan *Personal Computer* (Kiri)

KESIMPULAN

Dari perancangan *program* hingga pengujian yang telah dilakukan, maka dapat disimpulkan sebagai berikut:

1. Metode untuk mendeteksi wajah pada sistem deteksi mata kantuk secara akurat dapat menggunakan *Histogram of Oriented Gradients*, dimana metode ini memiliki komputasi yang cukup banyak sehingga proses yang dilakukan *Raspberry Pi* cukup lama.
2. Sistem deteksi mata kantuk yang berjalan pada *Raspberry Pi* dapat dikatakan cukup *real time*. Sistem ini memiliki *delay* berkisar 0.16 detik dari sistem yang dijalankan pada *personal computer* yang dapat dikatakan *real time*.
3. Sistem deteksi mata kantuk yang berjalan pada *Raspberry Pi* dapat beroperasi dengan baik sesuai dengan sistem yang berjalan pada *personal computer* dengan persentase sebesar 100%.

SARAN

Untuk menghasilkan keluaran dengan *frame per second* yang tinggi secara *real time* dibutuhkan perangkat keras yang memiliki *microprocessor* dengan *clockspeed* yang lebih tinggi daripada *Raspberry Pi 3* Model B.

DAFTAR PUSTAKA

[1] Djaja, S., Widyastuti R., Tobing K., Lasut D., dan Iriant J. 2016. *Gambaran Kecelakaan Lalu Lintas Di Indonesia 2010-2014.* Jurnal Nasional Ekologi Kesehatan. Vol. 15, No. 1

[2] Badan Pusat Statistik. 2016. *Statistik Transportasi Darat 2016.* Badan Pusat Statistik Indonesia. ISSN: 2598-5612

[3] Bergasa L.M., Nuevo J.U., Sotelo M. A., Barea R., dan Lopez. 2008. Visual Monitoring of Driver Inattention. International Journal of Studies in Computational Intelligence. Vol. 132, No.19

[4] Mahachandra, M., Yassierli, Iftikar Z., Sutalaksana, dan Kadarsah S. 2011. *Sleepiness Pattern of Indonesian Professional Driver Based on Subjective Scale and Eye Closure Activity*. International Journal of Basic & Applied Sciences IJBAS-IJENS. Vol. 11, No. 06

[5] Caldwell, John A., & Caldweel, J.Lynn. (2003). Fatigue in Aviation : A Guide to Staying Awake at The Stick. Farnham : Ashgate Publishing

[6] Breiman et.al. 1993. Classification and Regression Tress. New York : Chapman and Hall