

Perangkat Lunak Deteksi Kantuk Untuk Keselamatan Pengemudi Berbasis Pengolahan Citra Digital

Drowsiness Detection For Driver Safety Based On Digital Image Processing

1st Dhimas Pandu Yogaswara
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
dhimasswara@student.telkomuniversity.ac.id

2nd Muhammad Ikhsan Sani
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
ikhkansani@telkomuniversity.ac.id

3rd Marlindia Ike Sari
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
marlindia@telkomuniversity.ac.id

Abstrak-Di Indonesia kecelakaan lalu lintas sering sekali terjadi karena faktor utama dari pengemudi, Faktor yang menyebabkan kecelakaan diantaranya adalah kantuk, rasa lelah dalam mengemudi dapat menyebabkan kecelakaan pada kendaraan. Tujuan dari penelitian ini adalah untuk membantu mengurangi angka kecelakaan akibat kelalaian pengemudi dalam berkendara. Untuk mengurangi angka kecelakaan pada kendaraan maka dengan dibuatnya sistem pendeteksi kantuk ini dapat membantu para pengemudi agar lebih waspada ketika berkendara. Perangkat Lunak Deteksi Kantuk ini dirancang dengan menggunakan bahasa Python, dengan metode waterfall. Sistem ini dapat mendeteksi mata pengemudi untuk menentukan apakah pengemudi mengantuk atau tidak. Untuk menentukan kantuk sistem ini dapat menentukan rasio pada mata, untuk rasio mata dibawah 0.18 rasio pengemudi dapat dikatakan mengantuk atau mata dalam keadaan tertutup. Sistem yang dapat menentukan titik mata di rancang menggunakan metode facial landmarks agar dapat mudah menentukan titik mata pada wajah. Waktu yang ditentukan ketika mata tertutup adalah diatas 5 detik, jika mata menutup selama lebih dari 5 detik maka sistem baru berjalan sebagai pendeteksi kantuk.

Kata Kunci-kantuk, kecelakaan, python, kendaraan.

Abstract-In Indonesia, traffic accidents often occur because of the main factor of the driver. Factors that cause accidents include drowsiness, and fatigue while driving can cause accidents to vehicles. The purpose of this study is to help reduce the number of accidents due to driver negligence in driving. To reduce the number of vehicle accidents, the drowsiness detection system is made to help drivers be more alert when driving. This

Sleep Detection Software is designed using the Python language, with the waterfall method. This system can detect the driver's eyes to determine whether the driver is checking or not. To determine sleepiness, this system can determine the ratio of the eyes, for eye ratios below 0.18 the ratio of the driver can be said to be assessing or eyes in a closed state. The system that can determine the eye point is designed using the facial landmark method so that it can easily determine the eye point on the face. The time specified when the eyes are closed is more than 5 seconds, if the eyes are closed for more than 5 seconds, the new system runs as a sleep detector.

Keywords-sleepiness, accident, python, vehicle

I. PENDAHULUAN

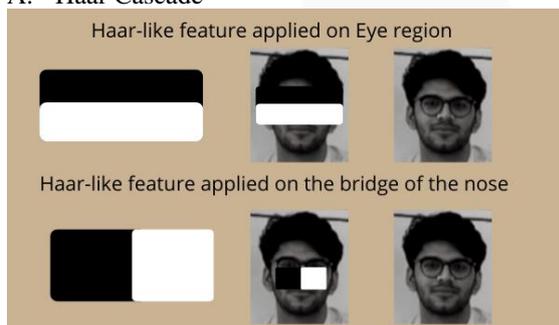
Seiring Di Indonesia kecelakaan lalu lintas sering sekali terjadi, terutama pada kendaraan roda empat. Ada beberapa faktor yang mempengaruhi kecelakaan di Indonesia, faktor yang sering terjadi yaitu karena kelalaian manusia itu sendiri. Namun ada beberapa faktor lain misal faktor kendaraan dan faktor lingkungan. Rasa kantuk yang terjadi Ketika mengemudi adalah hal yang paling sering membuat kecelakaan lalu lintas, kelelahan dalam perjalanan yang cukup jauh juga menimbulkan rasa kantuk yang sulit dikendalikan. Kelelahan pada tubuh manusia dapat diartikan sebagai dorongan biologis untuk melakukan istirahat dalam rangka pemulihan kondisi (Prabaswara, 2013)

Kelelahan pengemudi juga dapat diakibatkan karena fokusnya pengemudi dalam mengendarai kendaraan. Kelelahan tersebut menghasilkan dorongan pengemudi untuk tidur agar dapat memulihkan kelelahan yang dialami, sehingga timbulah rasa kantuk. Rasa kantuk dapat mengurangi

reaksi, konsentrasi dan menurunkan kewaspadaan saat mengemudi kendaraan, sehingga dapat menyebabkan kecelakaan lalu lintas, Berdasarkan informasi dari data KNKT (Komite Nasional Keselamatan Transportasi) terdapat 80% kecelakaan yang terjadi di jalan tol adalah akibat mengantuk dan letih. [1] Oleh karena itu, dibutuhkan sebuah sistem untuk mendeteksi pengemudi apakah sedang dalam keadaan mengantuk atau tidak, jika dalam keadaan mengantuk atau bahkan hampir tertidur sistem akan memberikan peringatan. Pengolahan citra digital dalam hal ini bisa dimanfaatkan untuk mengurangi tingkat kecelakaan pengemudi yang mengantuk. Algoritma yang ada dalam pemrograman python dan OpenCV bisa kita kelola untuk mendeteksi rasa kantuk pada mata, untuk menentukan bahwa objek yang terdeteksi sedang mengantuk maka perlu kita tentukan pada koordinat mata menggunakan metode facial landmarks. Jika mata tertutup dalam beberapa waktu maka sistem deteksi kantuk akan berjalan untuk mendeteksi besarnya koordinat mata, jika rasio mata di bawah 18 maka akan terdeteksi kantuk berdasarkan Eye Aspect Ratio. Berdasarkan faktor yang telah disebutkan kecelakaan bisa terjadi karena kondisi mengantuk, maka dari keadaan tersebut dapat diminimalisir dengan sistem yang akan mendeteksi rasa kantuk. Dengan adanya sistem pendeteksi kantuk maka diharapkan akan mampu mengurangi angka kecelakaan yang sering terjadi pada pengemudi dan dapat membantu pengemudi untuk tetap terjaga agar tidak mengantuk. Dalam hal ini maka di buatlah sistem deteksi kantuk untuk keamanan pengemudi.

II. KAJIAN TEORI

A. Haar Cascade



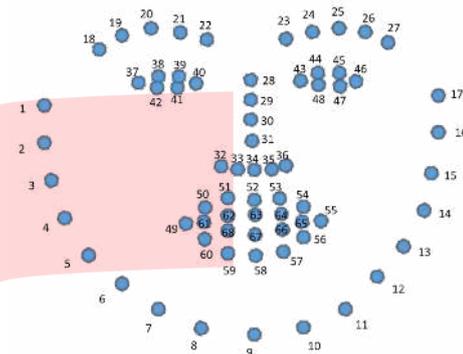
Algoritma Haar Cascade merupakan salah satu model machine learning yang sering kali digunakan sebagai Langkah awal pembuatan aplikasi object detection (terutama face recognition), dalam sebuah gambar maupun video. Berdasarkan gagasan Paul Viola dan Michael Jones yang berada dalam paper berjudul "Rapid Object Detection using a Boosted Cascade of Simple Features" (2001). Pada algoritma Haar Cascade menerapkan cascade fungsi untuk melatih gambar melalui 4 tahapan utama:

1. Menentukan Haar features.
2. Membuat gambar integral.
3. Adaboost training.

4. Melakukan klasifikasi dengan cascading Classifier [7].

Untuk mendeteksi wajah Algoritma Haar Cascade Classifier adalah algoritma yang digunakan untuk mendeteksi sebuah wajah. Algoritma tersebut dapat mendeteksi dengan cepat dan realtime sebuah benda termasuk wajah manusia. Algoritma Haar Cascade Classifier juga memiliki kelebihan perihal komputasi yang cepat karena hanya bergantung pada jumlah piksel dalam persegi dari sebuah image [9].

B. Dlib



Dlib merupakan suatu library machine learning yang ditulis dalam Bahasa pemrograman C++ dan ditunjukan untuk menyelesaikan masalah kehidupan sehari-hari. Dlib adalah library open source yang biasa dipakai untuk pengembangan software machine learning. Inti dari dlib adalah sebuah aljabar linier dengan Basic Linear Algebra Subprograms (BLAS). BLAS biasa digunakan untuk implementasi Bayesian network dan algoritma kernel-based untuk klasifikasi, clustering, deteksi anomaly, regresi dan feature ranking. Dlib mempunyai dua komponen penting yaitu aljabar linier dan machine learning tools.

Dlib menggunakan BLAS agar mendapatkan performa terbaik dan menambah kecepatan sebagai sebuah library. Dlib dapat menunjukan semua transformasi pada semua ekspresi dengan melibatkan BLAS yang dapat digunakan. Lalu BLAS dapat membuat pengguna melakukan penulisan atau penjumlahan dalam bentuk intuisi dan diberikan pada library [11].

C. Open CV



OpenCV (Open Source Computer Vision Library) adalah sebuah library open source yang dikembangkan oleh intel yang focus untuk menyederhanakan programing terkait citra digital. Di dalam OpenCV sudah mempunyai banyak fitur, antara lain: pengenalan wajah, pelacakan wajah,

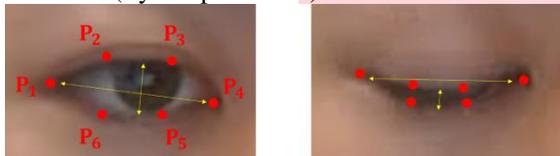
deteksi wajah, Kalman filtering, dan berbagai jenis metode AI (Artificial Intelligence). Dan menyediakan berbagai algoritma sederhana terkait Computer Vision untuk low level API.

OpenCV merupakan open source computer vision library untuk Bahasa pemrograman C/C++ dan telah dikembangkan ke python, java, matlab.

OpenCV mempunyai banyak fitur yang dapat dimanfaatkan, fitur-fitur tersebut antara lain :

1. Image and Video I/O
2. Computer Vision secara umum dan pengolahan citra digital
3. Modul Computer vision high level
4. Metode untuk AI dan machine learning
5. Sampling gambar dan transformasi
6. Metode untuk menciptakan dan menganalisa gambar biner.

D. EAR (Eye Aspect Ratio)

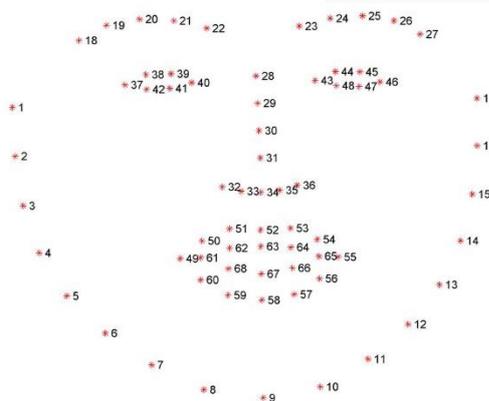


Eye Aspect Ratio merupakan salah satu library Dlib yang tersedia pada python, pada library ini digunakan untuk menentukan nilai ambang batas pada mata, Library ini merupakan detector wajah yang telah dilatih sebelumnya dengan berdasarkan pada modifikasi pada histogram gradien berorientasi dan menggunakan metode SVM (Support Vector Machine) untuk deteksi objek. Eye Aspect Ratio mempunyai perhitungan rumus sebagai berikut:

$$EAR = \frac{(|p2 - p6| + |p3 - p5|)}{2x|p1 - p4|}$$

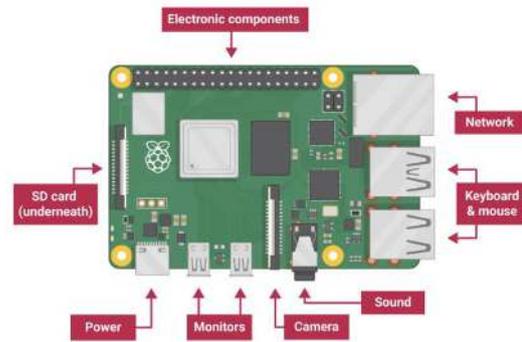
Berdasarkan teori, mata berkedip paling lama menutup yaitu 0.3 detik dan jika mengantuk mata akan tertutup lebih dari 5 detik [14].

E. Facial Landmarks



Facial Landmarks Detection adalah salah satu contoh keluaran terstruktur yang bertujuan untuk memprediksi bentuk geometri yang diperoleh dari sebuah data berupa citra wajah. Facial Landmarks adalah satu set poin penting yang ada pada citra wajah manusia. Jumlah landmark bergantung pada dataset atau aplikasi yang digunakan [16].

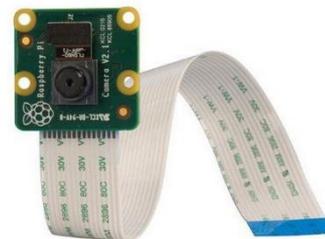
F. Raspberry Pi 4



Raspberry Pi adalah mini computer yang seukuran dengan kartu kredit yang dapat digunakan untuk menjalankan program perkantoran, permainan komputer, dan sebagai pemutar media hingga video beresolusi tinggi. Raspberry Pi dikembangkan oleh Yayasan nirlaba, Raspberry Pi Foundation.

Dalam sistem deteksi kantung Raspberry Pi digunakan sebagai alat yang akan menjalankan program kantung. Raspberry Pi dapat diprogram menggunakan Bahasa Python dan library pendukung OpenCV, oleh karena itu dalam pembuatan sistem deteksi kantung Mikrokontroler ini sangat bisa digunakan.

G. Raspberry Pi Cam



Raspberry Cam merupakan module dari Raspberry Pi yang dirancang untuk mengambil gambar dan video pada mikrokontroler Raspberry Pi. Pada sistem ini dibutuhkan Raspberry Pi Cam yang berfungsi untuk mengambil gambar untuk pengolahan citra digital. Raspberry Pi dapat mengambil gambar dengan resolusi yang dapat ditentukan, resolusi default dari Raspberry Pi Cam ini cukup tinggi dengan 5MP.

H. Buzzer dan Light Emitted Dioda



Buzzer merupakan komponen elektronika yang digunakan dalam sistem ini sebagai output, buzzer dapat menghasilkan suara dengan baik jika kondisi buzzer masih baik.

LED juga merupakan komponen elektronika yang dapat memberikan cahaya yang cukup terang, dan dalam sistem ini LED digunakan sebagai peringatan juga bahwa sistem telah aktif.

III. METODE

A. Gambaran Sistem Saat Ini

Sistem yang ada saat ini adalah pendeteksi kantuk menggunakan image processing sebagai metode untuk dapat mendeteksi wajah. Untuk dapat menentukan mata tertutup dan terbuka dihitung melalui nilai rasio yang berasal dari dataset facial landmarks. Untuk perhitungan nilai rasio dapat menggunakan rumus EAR agar dapat melihat nilai rasio yang ada pada landmark mata. Parameter yang digunakan sebagai penentu mata terbuka dan tertutup adalah melalui nilai rasio mata [17].

B. Identifikasi Kebutuhan Sistem

No	Kebutuhan Fungsional
1	Mendeteksi kantuk pada pengemudi menggunakan titik kordinat mata yang dibentuk oleh <i>facial landmarks</i> di dalam sistem pada Raspberry Pi 4
2	Membuat sistem pendeteksi kantuk
3	Sistem pendeteksi kantuk dapat menampilkan rasio
4	Sistem pendeteksi kantuk dapat mendeteksi mata

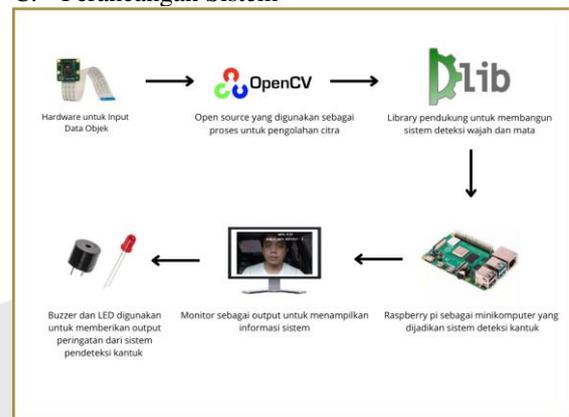
Pada Tabel 3-1 yang dibutuhkan pada fungsional adalah sistem pendeteksi kantuk yang

dapat mendeteksi kantuk pada mata pengemudi dengan titik kordinat pada mata menggunakan Raspberry Pi 4 yang dibantu oleh facial landmarks dalam membuat bentuk pola untuk mata.

No	Kebutuhan non Fungsional
1	Raspberry Pi sebagai mikrokontroler sistem pengolah citra
2	Raspberry Cam digunakan untuk menangkap video pengemudi
3	OpenCV sebagai <i>library</i> pendukung dalam pembuatan sistem
4	Bahasa yang digunakan Python
5	Monitor digunakan untuk menampilkan sistem
6	IR Proximity digunakan untuk mendeteksi jarak pengemudi dengan sistem
7	Buzzer dan LED digunakan sebagai <i>output</i> dari sistem

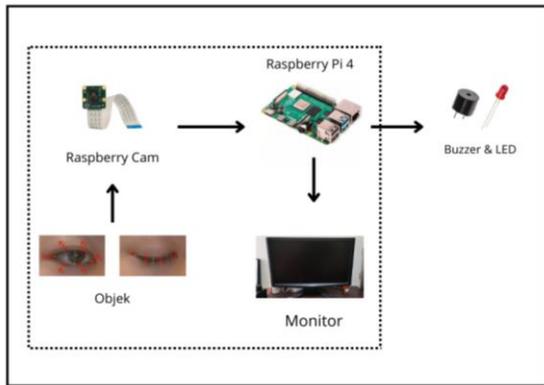
Pada Tabel 3-2 kebutuhan non fungsional adalah kebutuhan yang digunakan untuk membantu dalam perancangan sistem agar tersusun dengan baik. Raspberry Pi 4 digunakan sebagai alat utama yang akan dijadikan sistem dengan menggunakan bahasa python dan *library* open CV, lalu ada kamera Raspberry Pi yang digunakan untuk mengambil video untuk pengolahan citra, terakhir ada buzzer dan LED yang digunakan sebagai *output*.

C. Perancangan Sistem



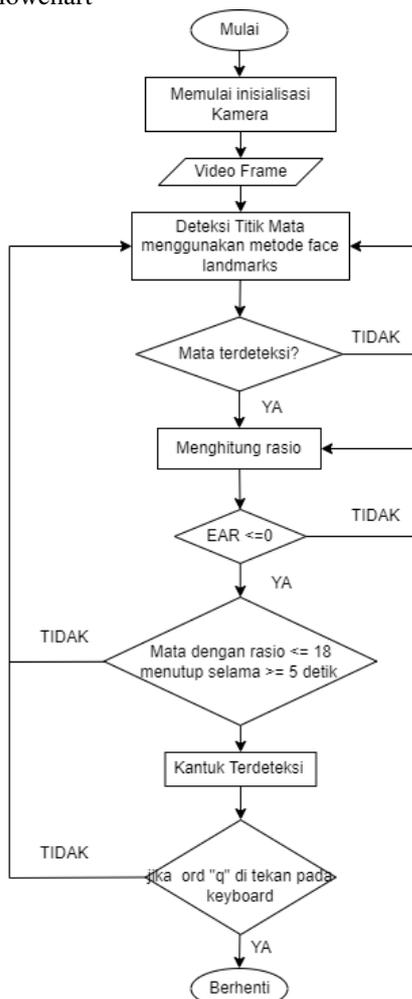
Perancangan sistem yang akan dibangun adalah menggunakan OpenCV sebagai pondasi untuk membangun sistem deteksi, lalu ada library pendukung DLIB untuk mendeteksi lebih rinci sebuah wajah. Pada library DLIB terdapat modul facial landmarks yang berfungsi untuk mendapatkan titik mata dari sebuah index array. Sistem dirancang menggunakan Raspberry Pi yang diprogram dengan sistem deteksi kantuk, sistem yang berjalan akan ditampilkan ke dalam sebuah monitor sebagai informasi sistem. Buzzer dan LED diprogram untuk memberikan output peringatan ketika sistem deteksi kantuk berjalan.

D. Diagram Blok Sistem yang Akan Dibangun



Pada Gambar 3-2 untuk proses yang dihighlight menjelaskan sistem deteksi kantuk bekerja untuk menangkap objek mata agar dapat diproses sistem. Kamera Raspberry Pi sebagai input, Raspberry Pi 4 sebagai sistem dari deteksi kantuk, dan layer monitor sebagai output untuk menampilkan informasi dari sistem deteksi kantuk. Sistem akan melakukan proses untuk mendeteksi sebuah kantuk melalui nilai rasio, jika rasio di bawah 0.18 dan waktu rasio tetap berada di bawah 0.18 dengan waktu di atas 5 detik maka sistem akan memberikan peringatan kepada pengemudi sebagai kondisi mengantuk.

E. Flowchart



Pada flowchart di atas adalah proses sistem

yang bekerja sebagai deteksi kantuk dengan beberapa proses yang dilalui, dan kondisi yang membutuhkan objek mata untuk menentukan apakah objek mengantuk atau tidak, kondisi mengantuk ketika rasio terdeteksi di bawah 0.18 rasio pada mata pengemudi dan waktu mata tertutup dengan waktu lebih dari 5 detik

F. Kebutuhan Perangkat Keras

No	Nama Perangkat	Fungsi
1	Raspberry Pi 4	Sebagai mikrokontroler yang akan dijadikan sistem
2	Raspberry Pi Cam	Sebagai alat input untuk menangkap video objek
3	Monitor	Untuk menampilkan informasi sistem
4	Buzzer dan LED	Berfungsi untuk memberikan output sistem

G. Kebutuhan Perangkat Lunak

No	Nama Perangkat	Fungsi
1	Thony Pyhton IDE	Sebagai text editor untuk menyusun program sistem deteksi kantuki
2	Sistem Operasi Raspbian	Sistem Operasi untuk menjalankan Raspberry Pi 4
3	Open CV	Library pengolahan citra yang dapat membantu membuat program deteksi kantuk
4	Dlib	Library python untuk membantu membangun landmark pada wajah agar terbentuk deteksi mata

Pada Tabel 3-4 adalah daftar perangkat lunak pendukung yang digunakan untuk dapat membangun program deteksi kantuk pada Raspberry Pi.

IV. HASIL DAN PEMBAHASAN

A. Implementasi

Pada implementasi proyek akhir ini akan dijelaskan bagaimana program Deteksi Kantuk akan diimplementasikan pada Raspberry Pi 4 untuk dijadikan sistem yang dapat mendeteksi kantuk secara realtime. Berikut merupakan langkah implementasi yang dilakukan.

B. Implementasi Pembuatan Program Deteksi Kantuk

1. Import Library yang di butuhkan

```

from scipy.spatial import distance
from imutils import face_utils
import numpy as np
import dlib
import cv2
import time
from threading import Thread
import RPi.GPIO as GPIO
from time import sleep
import math

```

Library pendukung untuk pembuatan program Deteksi kantuk menggunakan beberapa library.

- Scipy.spatial = Library Untuk menemukan jarak Euclidean
- Imutils = Library yang membantu untuk membentuk scaling pada wajah
- Threading = Library untuk memproses lebih cepat program yang dieksekusi
- Numpy = Library yang membantu untuk membuat proses array atau numerik
- Dlib = Untuk membantu membangun dan menganalisa bagian wajah, serta memiliki module didalamnya untuk membentuk landmark pada wajah
- Cv2 = Library untuk mengolah data citra digital secara real time

2. Inisialisasi untuk menentukan ambang batas

```

#aspek rasio minimal mata di bawah, alarm akan
ditriggered
EYE_ASPECT_RATIO_THRESHOLD = 0.18
#rasio mata di bawah batas untuk memanggil alarm
EYE_ASPECT_RATIO_CONSEC_FRAMES = 5

```

Untuk menentukan rasio mata tertutup pada program perlu dibuat deklarasi untuk nilai ambang batas saat mata tertutup dan frame yang berjalan untuk mengambil nilai ambang batas. Pada variable `eye_aspect_ratio_threshold` adalah untuk memberikan nilai batas ketika nilai rasio mata menurun, dan variabel `eye_aspect_ratio_consec_frames` adalah untuk memberikan nilai frame agar variabel ambang batas berjalan sesuai dengan frame yang ditentukan.

C. Variabel Pengambilan Dataset Haarcascade

```

face_cascade =
cv2.CascadeClassifier("haarcascades/haarcascade_fronta
lface_default.xml")

eye_cascade =
cv2.CascadeClassifier("haarcascades/haarcascade_eye.xm
l")

```

Variabel pengambilan dataset haarcascade adalah untuk membentuk deteksi wajah menggunakan modul haarcascade yang akan dijadikan sebagai pola untuk mencari titik mata menggunakan facial landmarks.

D. Fungsi EAR untuk Menentukan nilai rasio mata

```

def eye_aspect_ratio(eye):
    A = distance.euclidean(eye[1], eye[5])
    B = distance.euclidean(eye[2], eye[4])
    C = distance.euclidean(eye[0], eye[3])

    ear = (A+B) / (2*C)
    return ear

```

Fungsi `eye_aspect_ratio` adalah variabel yang digunakan untuk menghitung dataset mata untuk dijadikan nilai rasio yang nanti akan digunakan sebagai parameter untuk menentukan mata tertutup dan terbuka. Pada variabel A dan B baris ke dua dan ke tiga adalah dataset yang akan menghitung nilai mata pada garis vertical dan variabel C adalah dataset yang akan menghitung nilai mata pada garis horizontal.

E. Inisialisasi Pemanggilan Variabel Prediktor Wajah Facial Landmarks

```

#Detektor dan prediktor wajah, dari library lib
print("[INFO] menunggu facial landmark prediktor...")
detector = dlib.get_frontal_face_detector()
predictor =
dlib.shape_predictor("models/shape_predictor_68_face_l
andmarks.dat")

#indeks landmark wajah untuk mata kiri dan kanan
(lStart, lEnd) =
face_utils.FACIAL_LANDMARKS_IDXS['left_eye']
(rStart, rEnd) =
face_utils.FACIAL_LANDMARKS_IDXS['right_eye']

```

Untuk mendeteksi area mata diperlukan dataset facial landmarks yang digunakan sebagai prediktor untuk menentukan daerah wajah yang akan kita jadikan parameter. Pada baris 3 dan 5 adalah variabel yang akan memanggil dataset facial landmarks untuk bisa dijadikan program yang dapat mendeteksi area wajah.

Variabel pada baris 9 dan 11 berfungsi untuk memanggil nilai dataset dari facial landmarks untuk dijadikan index yang akan menentukan titik mata kiri dan kanan pada wajah.

F. Perulangan untuk menjalankan fungsi EAR dan deteksi wajah

```

for face in faces:
    shape = predictor(gray, face)
    shape = face_utils.shape_to_np(shape)

    #Array koordinat Mata kiri dan Mata kanan
    leftEye = shape[lStart:lEnd]
    rightEye = shape[rStart:rEnd]

    #Hitung rasio aspek kedua mata
    leftEyeAspectRatio = eye_aspect_ratio(leftEye)
    rightEyeAspectRatio =
    eye_aspect_ratio(rightEye)

    eyeAspectRatio = (leftEyeAspectRatio +
    rightEyeAspectRatio) / 2

```

Perulangan di atas berfungsi untuk menjalankan prediktor wajah yang akan diubah menjadi nilai array menggunakan library numpy, pada baris 7 dan 8 adalah variabel array yang akan dijadikan perhitungan untuk menampilkan nilai rasio mata kiri dan kanan. Untuk baris 14 adalah rumus untuk menghitung nilai array yang telah dijadikan variabel pada baris sebelumnya untuk mendapatkan nilai rasio mata kiri dan kanan.

G. Variabel untuk membentuk garis pada mata

```
#inisialisasi untuk menghilangkan perbedaan kontur
cembung dan menggambar bentuk mata di sekitar mata
leftEyeHull = cv2.convexHull(leftEye)
rightEyeHull = cv2.convexHull(rightEye)
cv2.drawContours(frame, [leftEyeHull], -1, (0,
255, 0), 1)
cv2.drawContours(frame, [rightEyeHull], -1,
(0, 255, 0), 1)
```

Pada baris 3 dan 4 adalah variabel untuk mengambil fungsi array yang telah ditentukan nilai rasionya lalu dipanggil oleh variabel selanjutnya agar dapat dibentuk garis kontur pada mata. Pada baris 5 dan 6 adalah fungsi untuk membentuk garis pada mata menggunakan convexhull yang dapat membentuk kontur berdasarkan bentuk mata.

H. Kondisi untuk menentukan mata tertutup dan terbuka

```
if(eyeAspectRatio < EYE_ASPECT_RATIO_THRESHOLD):
    COUNTER += 1
    #Jika tidak, frame lebih besar dari nilai
    if COUNTER >=
EYE_ASPECT_RATIO_CONSEC_FRAMES:

cv2.putText(frame, "KANTUK DETEKSI AKTIF", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)

        if not ALARM_ON:
            TOTAL +=1
            ALARM_ON = True
            GPIO.output(buzzer,GPIO.HIGH)
            GPIO.output(led,GPIO.HIGH)

        else:
            COUNTER = 0
            ALARM_ON = False

            GPIO.output(buzzer,GPIO.LOW)
            GPIO.output(led,GPIO.LOW)

cv2.putText(frame, "MATA:
{:.2f}".format(eyeAspectRatio), (350, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
cv2.putText(frame, "JUMLAH MATA TERTUTUP :
{}".format(TOTAL), (250, 70),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
```

Variabel yang digunakan untuk menentukan kondisi adalah variabel yang sebelumnya telah ditentukan nilainya sebagai parameter untuk menentukan mata tertutup dan terbuka. Pada kondisi baris 1 adalah variabel untuk menentukan nilai ambang batas, jika nilai tersebut kurang dari 0.18 maka kondisi akan berjalan pada variabel selanjutnya untuk menentukan kondisi mata tertutup dan akan menampilkan peringatan berupa tulisan pada monitor . Jika kondisi tidak kurang dari nilai ambang batas 0.18 maka program akan tetap berjalan dengan mendeteksi mata terbuka. Pada baris akhir adalah program untuk menampilkan nilai rasio berdasarkan perhitungan EAR sebelumnya menggunakan dataset facial landmarks, program tersebut akan menampilkan nilai rasio pada monitor.

I. Fungsi untuk menampilkan sistem pada monitor

```
#Menampilkan video monitor
cv2.imshow("Sistem Deteksi Kantuk", frame)
key = cv2.waitKey(1) & 0xFF
if key == ord("q"):
    break
```

Pada baris 2 adalah variabel yang berfungsi untuk menampilkan sistem pada layar monitor, program yang dibutuhkan adalah cv2.imshow untuk

menampilkan realtime video dari kamera yang akan menampilkan dalam bentuk desktop dengan title sistem deteksi kantuk.

Untuk baris 3 adalah fungsi untuk mematikan sistem dengan menggunakan shortcut pada keyboard dengan menekan huruf q pada keyboard.

J. Program Akhir untuk pembersihan sistem yang berjalan

```
#Finish Code
video_capture.release()
cv2.destroyAllWindows()
```

Program untuk membersihkan sistem ketika sistem telah mati adalah menggunakan variabel video.capture.release() yang berfungsi untuk membersihkan sistem pada kamera agar kamera dapat menutup.

Lalu variabel cv2.destroyAllWindows() berfungsi untuk memberhentikan semua program yang berjalan setelah keluar dari program.

K. Pengujian

Pada pengujian, akan dijelaskan apa saja yang telah diuji dalam sistem deteksi kantuk untuk menentukan apakah sistem yang telah dibuat dapat berjalan semua dengan baik atau tidak. Pengujian yang akan dilakukan adalah sebagai berikut:

1. Pengujian Landmark Wajah dan Mata
2. Pengujian Rasio Mata
3. Pengujian Deteksi Kantuk Pada Objek

1. Pengujian Landmark Wajah dan Mata

a. Tujuan Pengujian

Pada pengujian landmark wajah dan mata dilakukan untuk mengetahui titik wajah yang akan dijadikan parameter untuk menentukan sistem kantuk. Pengujian ini juga akan menunjukkan landmark wajah yang dapat terdeteksi dengan sistem untuk pemrosesan gambar. Untuk menjadikan sistem deteksi kantuk parameter yang diambil adalah titik mata, dengan menggunakan metode facial landmark kita bisa menentukan titik yang sesuai untuk dijadikan parameter.

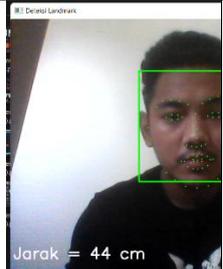
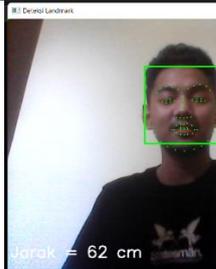
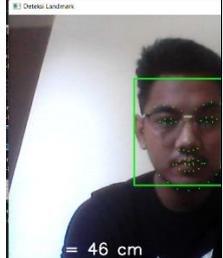
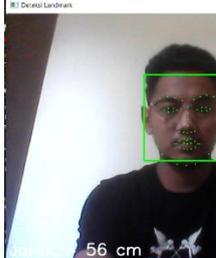
b. Skenario Pengujian

Skenario pengujian dilakukan dengan mencoba mendeteksi wajah dan mata menggunakan metode facial landmark. Untuk pengujiannya dilakukan dengan mendeteksi area wajah menggunakan kacamata dan tidak menggunakan kacamata. Jarak yang diambil dalam pengujian ini adalah 45 cm dan 60 cm dari kamera terhadap objek.

c. Hasil Pengujian

1) Pengujian Landmark Wajah

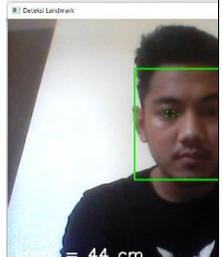
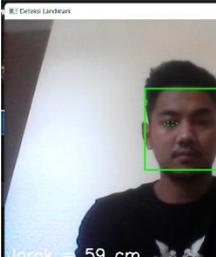
Jarak 45 cm	Jarak 60 cm	Keterangan
Tanpa Kacamata	Tanpa Kacamata	Terdeteksi

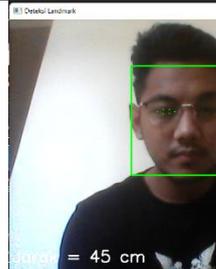
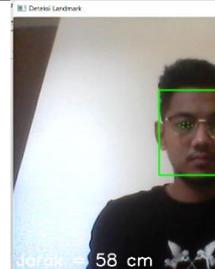
		
Menggunakan Kacamata	Menggunakan Kacamata	Terdet
		

Pengujian *landmark* wajah dilakukan untuk mengetahui titik-titik yang dapat dideteksi oleh sistem, untuk melakukan deteksi wajah perlu menggunakan module dari *facial landmark*. Pada bagian wajah yang dideteksi kita akan mengambil titik mata saja untuk dijadikan parameter kondisi dalam pengujian rasio sistem. Index array yang diambil untuk menampilkan semua titik pada mata ketika mendeteksi adalah pada nilai 1 sampai 68. Pengujian yang dilakukan berhasil mendeteksi wajah dengan dua parameter yaitu dengan menggunakan kacamata dan tidak menggunakan kacamata. Untuk jarak pengujian adalah pada jarak 45 cm dan 60 cm dari kamera terhadap objek.

2) Pengujian *Landmark* Mata Kanan

Tabel 4-1 Pengujian *Landmark* Mata Kanan

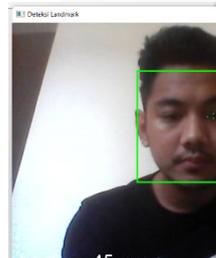
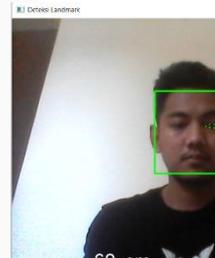
Jarak 45 cm	Jarak 60 cm	Keterangan
Tanpa Kacamata	Tanpa Kacamata	Terdeteksi
		
Menggunakan Kacamata	Menggunakan Kacamata	Terdeteksi

		
--	---	--

Setelah melakukan pengujian pada *landmark* wajah, pengujian selanjutnya adalah mendeteksi *landmark* mata kanan. Untuk mengetahui apakah deteksi *landmark* dapat berjalan baik, maka dilakukan pengujian pada masing-masing mata. Index *landmark* mata kanan diambil dari nilai index array 36 sampai 42. Dari pengujian di atas, *landmark* mata kanan berhasil terdeteksi dengan baik, dalam pengujian dilakukan dengan dua parameter juga yaitu menggunakan kacamata dan tidak menggunakan kacamata. Untuk jarak pengujian yang dilakukan adalah pada jarak 45 cm dan 60 cm dari kamera terhadap objek yang dideteksi.

3) Pengujian *Landmark* Mata Kiri

Tabel 4-2 Pengujian *Landmark* Mata Kiri

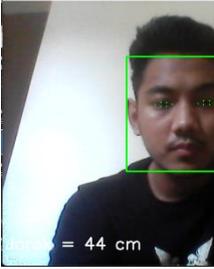
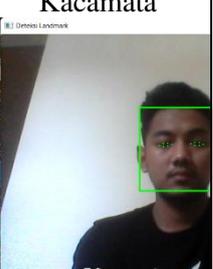
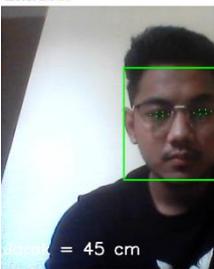
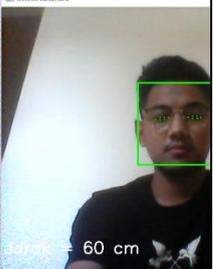
Jarak 45 cm	Jarak 60 cm	Keterangan
Tanpa Kacamata	Tanpa Kacamata	Terdeteksi
		
Menggunakan Kacamata	Menggunakan Kacamata	Terdeteksi

Pada pengujian sebelumnya telah diuji untuk mendeteksi mata bagian sebelah kanan, langkah selanjutnya adalah mencoba mendeteksi mata pada bagian kiri, pada pengujian deteksi mata bagian sebelah kiri terlihat normal sama seperti pengujian mata bagian sebelah kanan. Agar dapat mendeteksi mata bagian sebelah kiri maka perlu mengambil index array dari nilai 42 sampai 48. Untuk pengujian mata sebelah kiri juga dilakukan dengan menggunakan dua parameter, parameter yang

dilakukan adalah menggunakan kacamata dan tidak menggunakan kacamata. Pada pengujian jarak sama seperti sebelumnya adalah mengambil jarak 45 cm dan 60 cm dari kamera terhadap objek.

4) Pengujian *Landmark* Mata Kanan dan Mata Kiri

Tabel 4-3 Pengujian *Landmark* Mata Kanan dan Mata Kiri

Jarak 45 cm	Jarak 60 cm	Keterangan
<p>Tanpa Kacamata</p> 	<p>Tanpa Kacamata</p> 	Terdeteksi
<p>Menggunakan Kacamata</p> 	<p>Menggunakan Kacamata</p> 	

Pengujian terakhir yang dilakukan adalah dengan menguji *landmark* pada kedua mata. Pengujian ini dilakukan agar dapat melihat sistem dalam mendeteksi *landmark* pada bagian kedua mata. Untuk menampilkan *landmark* pada kedua mata index yang diambil adalah 36 sampai 48 dalam array *facial landmark*. Pada pengujian mendeteksi *landmark* kedua mata parameter yang dilakukan adalah menggunakan kacamata dan tidak menggunakan kacamata.

d. Analisis Pengujian

Dalam pengujian untuk mendeteksi *landmark* wajah dan mata dapat berjalan dengan kondisi normal. Pada pengujian *landmark* wajah kita mengambil semua index array yang ada pada *facial landmark* yaitu pada nilai 1 sampai 68 agar dapat mengetahui titik wajah yang berhasil dideteksi. Untuk pengujian mata dilakukan dengan tiga kondisi, index array yang diambil dari *landmark* mata kanan adalah 36 sampai 42, pada mata kiri diambil index array dari nilai 42 sampai 48, dan untuk menampilkan kedua mata dilakukan dengan mengambil nilai index dari 36 sampai 48. Dari pengujian yang dilakukan kita dapat menentukan parameter untuk membuat kondisi kantung pada sistem dengan *landmark* mata.

2. Pengujian Rasio Mata

a. Tujuan Pengujian

Pengujian rasio mata bertujuan untuk melihat apakah sistem dapat mendeteksi rasio mata dengan keadaan mata terbuka dan dengan keadaan mata tertutup. Untuk keadaan rasio mata normal pada sistem terdapat pada rata-rata 0.30 rasio dan untuk rasio mata tertutup terdapat rata-rata pada rasio di bawah 0.18. Rasio mata akan ditampilkan pada tabel untuk menunjukan berapa rasio mata saat terbuka dan rasio mata pada saat tertutup.

b. Skenario Pengujian

Pada pengujian ini dilakukan dengan kondisi objek yang berbeda untuk melihat rasio mata yang akan ditentukan oleh sistem. Kondisi yang dilakukan adalah dengan menggunakan kacamata dan tidak menggunakan kacamata. Adapun pengujian tempat yang dilakukan adalah di dalam mobil dengan kondisi waktu dan cahaya yang berbeda. Untuk tempat dan waktu dilakukan dengan tiga kondisi, yaitu pada pagi hari pukul 06.30 dengan intensitas cahaya 199,8 lux, pada siang hari pukul 13.45 dengan intensitas cahaya 989,0 lux dan pada malam hari pukul 22.00 dengan intensitas cahaya 0,6 lux.

Adapun pengujian ini dilakukan dari beberapa responden dengan kriteria berbeda. Kriteria responden dapat dijelaskan di bawah:

- 1) Orang ke-1, umur 22, tinggi 165 cm
- 2) Orang ke-2, umur 21, tinggi 171 cm
- 3) Orang ke-3, umur 21, tinggi 170 cm
- 4) Orang ke-4, umur 21, tinggi 169 cm

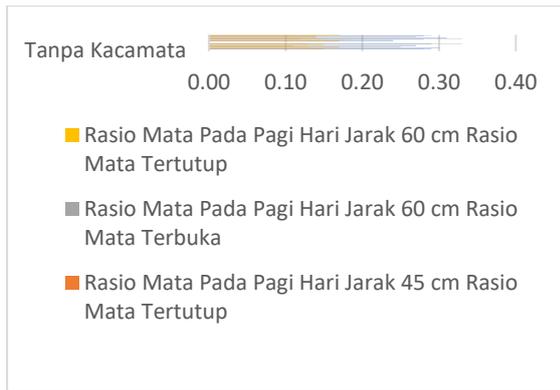
L. Hasil Pengujian

1. Pengujian Rasio Mata pada Pagi Hari

Responden	Rasio Mata Pada Pagi Hari			
	Jarak 45 cm		Jarak 60 cm	
	Rasio Mata Terbuka	Rasio Mata Tertutup	Rasio Mata Terbuka	Rasio Mata Tertutup
Tanpa Kacamata				
Orang ke-1	0.29	0.15	0.28	0.17
Orang ke-2	0.30	0.17	0.25	0.13
Orang ke-3	0.27	0.11	0.27	0.15
Orang ke-4	0.29	0.13	0.33	0.15
Menggunakan Kacamata				
Orang ke-1	0.24	0.17	0.24	0.18
Orang ke-2	0.28	0.12	0.33	0.15
Orang ke-3	0.31	0.14	0.27	0.14
Orang ke-4	0.28	0.14	0.29	0.17

Pada Tabel 4-5 adalah data rasio yang berhasil terdeteksi ketika mata terbuka dan tertutup, dan data yang diambil adalah pada pagi hari.

a. Grafik Pengujian Rasio Mata pada Pagi Hari



Pada Gambar 4-1 merupakan hasil grafik yang didapat dari nilai rasio pada tabel data rasio pagi hari.

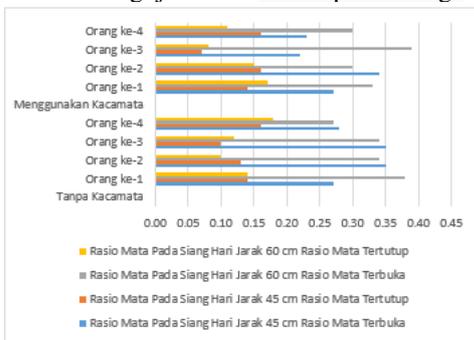
b. Pengujian Rasio Mata pada Siang Hari

Tabel 4-6 Pengujian Rasio Mata pada Siang Hari

Responden	Rasio Mata Pada Siang Hari			
	Jarak 45 cm		Jarak 60 cm	
	Rasio Mata Terbuka	Rasio Mata Tertutup	Rasio Mata Terbuka	Rasio Mata Tertutup
Tanpa Kacamata				
Orang ke-1	0.27	0.14	0.38	0.14
Orang ke-2	0.35	0.13	0.34	0.10
Orang ke-3	0.35	0.10	0.34	0.12
Orang ke-4	0.28	0.16	0.27	0.18
Menggunakan Kacamata				
Orang ke-1	0.27	0.14	0.33	0.17
Orang ke-2	0.34	0.16	0.30	0.15
Orang ke-3	0.22	0.07	0.39	0.08
Orang ke-4	0.23	0.16	0.30	0.11

Pada Tabel 4-6 merupakan data nilai rasio ketika mata tertutup dan terbuka yang diambil pada siang hari.

c. Grafik Pengujian Rasio Mata pada Siang Hari



Pada Gambar 4-2 adalah hasil grafik yang dapat ditampilkan dari data pengujian rasio pada siang hari.

M. Analisis Pengujian

Hasil analisis yang dapat dijelaskan adalah ketika mata terbuka rasio yang dapat terdeteksi adalah di atas 0.20 rasio, dan rasio ketika mata tertutup adalah di bawah 0.18 rasio. Pada nilai di bawah 0.18 rasio sistem dapat mendeteksi mata dengan kondisi mengantuk. Untuk hasil dari pengujian rasio, waktu yang efektif untuk dapat mendeteksi rasio adalah pada pagi hari dan siang hari. Pada malam hari sistem tidak dapat mendeteksi apapun dari objek, maka dari itu rasio pada mata tidak dapat terdeteksi untuk menentukan kantuk.

1. Pengujian Deteksi Kantuk Pada Objek

a. Tujuan Pengujian

Pengujian ini dilakukan untuk mengetahui apakah sistem dapat mendeteksi kantuk pada objek atau tidak, dan apakah sistem dapat mendeteksi kantuk dalam beberapa kondisi, seperti pada kondisi cahaya gelap atau terang, di dalam mobil, dan jarak yang telah ditentukan.

b. Skenario Pengujian

Skenario pengujian sistem deteksi kantuk dilakukan di dalam mobil. Dengan melakukan perbedaan kondisi sistem ini juga diuji dengan jarak yang telah ditentukan, jarak yang pertama ditentukan dengan jarak 60 cm, dan jarak yang kedua ditentukan dengan jarak 45 cm.

1) Skenario Pengujian Cahaya di dalam Mobil

Pagi Hari Pukul 06.30	als-tcs3701 199,8 lux
Siang Hari Pukul 13.45	als-tcs3701 0,6 lux
Malam Hari Pukul 22.00	als-tcs3701 989,0 lux

Pada pengujian dilakukan terlebih dahulu pengujian cahaya pada tempat yang akan digunakan untuk pengujian objek yang akan diuji, berdasarkan Gambar 4-3 pengujian dilakukan dengan cahaya yang berbeda dengan kondisi pada pagi hari pukul 06.30 dengan intensitas cahaya 199,8 lux , pada siang hari pukul 13.45 dengan intensitas cahaya 989,0 lux dan malam hari pukul 22.00 dengan intensitas cahaya 0,6 lux.

2) Skenario Peletakan Sistem



Gambar 4-1 Skenario Peletakan Tempat

Pada skenario peletakan sistem, dilakukan di dalam mobil untuk dapat menentukan apakah sistem dapat berjalan sesuai tempat untuk implementasi atau tidak. Pada skenario ini peletakan tempat dilakukan dengan dua kondisi, pada kondisi pertama peletakan sistem ditempatkan pada dashboard mobil, dan pada kondisi kedua peletakan ditempatkan pada spion tengah mobil. Untuk peletakan sistem pada spion tengah mobil hanya dilakukan beberapa hasil pengujian dikarenakan

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil pengujian dan penelitian Proyek Akhir ini, penulis dapat mengambil kesimpulan sebagai berikut.

1. Sistem Deteksi Kantuk dibuat dengan mini komputer Raspberry Pi 4 dengan menggunakan Bahasa pemrograman Python dengan beberapa library pendukung seperti OpenCV, DLIB, dan Imutils.
2. Metode Facial Landmarks digunakan untuk dapat mendeteksi titik-titik pada wajah, untuk landmark yang dipakai adalah titik mata kanan dan mata kiri. Landmark mata yang berhasil terdeteksi akan dijadikan parameter untuk pengukuran rasio yang dapat menentukan kantuk pengemudi.
3. Kondisi nilai rasio untuk dapat menentukan kantuk adalah pada nilai di bawah 0.18 rasio, jika sistem mendeteksi mata dalam keadaan rasio di atas 0.18 maka sistem tidak dapat menentukan kondisi itu sebagai kantuk.
4. Sistem pendeteksi kantuk dapat mendeteksi objek dengan normal pada keadaan cahaya di atas 150 lux, jika cahaya berada di bawah 150 lux sistem tidak dapat mendeteksi objek dengan normal. Untuk peletakan sistem pendeteksi ditempatkan pada dashboard dan spion tengah mobil, namun peletakan yang efektif adalah pada dashboard mobil, karena jika jarak kurang dari 30 cm pada peletakan di spion tengah, sistem tidak dapat mendeteksi objek.website.

B. Saran

Adapun saran dari sistem pendeteksi kantuk yaitu :

1. Penggunaan kamera infrared pada sistem sangat diperlukan untuk mendeteksi objek pada malam hari.
2. Menggunakan lebih banyak parameter untuk mendeteksi kantuk, seperti menggunakan tingkat lebar mulut.
3. Menghubungkan hasil data sistem ke dalam sebuah website atau database.

4. Menjadikan sistem pendeteksi kantuk dapat diimplementasikan pada kendaraan umum.

REFERENSI

- [1] berita nasional Populer, Viva.co.id, "KNKT Data kecelakaan akibat mengantuk dan leleh".
- [2] K. C. Daniel Siswanto, ST.,MT, Romi Loice, ST., MT., "PERANCANGAN ALAT DETEKSI KANTUK DAN ANALISIS TINGKAT KANTUK PENGEMUDI BUS MALAM X," 2014.
- [3] P. R. Indonesia *et al.*, "Sistem pendeteksi kantuk menggunakan webcam dan Raspberry Pi," no. 09.
- [4] R. T. Puteri and F. Utamingrum, "Deteksi Kantuk Menggunakan Kombinasi Haar Cascade dan Convolutional Neural Network," vol. 4, no. 3, pp. 816–821, 2020.
- [5] M. A. Maulana, J. T. Elektro, F. T. Industri, and U. I. Indonesia, "Deteksi Kantuk Pada Pengendara Roda Empat Melalui Citra Wajah Menggunakan Metode Facial Landmark," 2022.
- [6] "5 Kondisi Penyebab Sering Mengantuk yang Jarang Diketahui." <https://www.alodokter.com/sering-mengantuk-karena-waktu-tidur-terganggu-mungkin-ini-penyebabnya>
- [7] D. C. J. M. Pardede, A. M. Rumagit, S. Tangkawang, and G. Kaunang, "Deteksi Pengendara Mengantuk Menggunakan Metode Eye Tracking Berbasis Raspberry Pi".
- [8] "Viola Jones Algorithm and Haar Cascade Classifier." <https://towardsdatascience.com/viola-jones-algorithm-and-haar-cascade-classifier-ee3bfb19f7d8>
- [9] S. Abidin, "Deteksi Wajah Menggunakan Metode Haar Cascade Classifier Berbasis Webcam Pada Matlab," *J. Teknol. Elekterika*, vol. 15, no. 1, p. 21, 2018, doi: 10.31963/elekterika.v15i1.2102.
- [10] "Face Detection and Landmarks using dlib and OpenCV." <https://vigneshs4499.medium.com/face-detection-and-landmarks-using-dlib-and-opencv-8c824f50cc78>
- [11] N. Wahyudiana and S. Budi, "Perbandingan Performa Pre-Trained Classifier dLib dan HAAR Cascade (OpenCV) Untuk Mendeteksi Wajah," *J. Strateg.*, vol. 1, no. November, p. 376, 2019.
- [12] "Python: Real Time Face Detection Menggunakan Library OpenCV." <https://bndrsnvtch.medium.com/python-real-time-face-detection-menggunakan->

- library-opencv-a10ccf3c14cf
- [13] “Eye fatigue estimation using blink detection based on Eye Aspect Ratio Mapping(EARM).”
<https://www.sciencedirect.com/science/article/pii/S2667241322000039>
- [14] H. Classifier, “Haarcascade Classifier Dan Eye Aspect Ratio Untuk,” no. Ciastech, pp. 437–444, 2021.
- [15] “Facial landmarks 68 with dlib, OpenCV, and Python.”
<https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/>
- [16] Andre Hartoko Aji Putra Perdana, Susijanto Tri Rasmana, and Heri Pratikno, “Implementasi Sistem Deteksi Mata Kantuk Berdasarkan Facial Landmarks Detection Menggunakan Metode Regression Trees,” *JoTI*, vol. 1, no. 1, pp. 1–9, 2020, doi: 10.37802/joti.v1i1.1.
- [17] D. A. Navastara, W. Y. M. Putra, and C. Fatichah, “Drowsiness Detection Based on Facial Landmark and Uniform Local Binary Pattern,” *J. Phys. Conf. Ser.*, vol. 1529, no. 5, 2020, doi: 10.1088/1742-6596/1529/5/052015.