Implementasi Sistem Deteksi Mata Kantuk Pada Pengendara Mobil Dengan Metode *Eye Aspect Ratio* Dan *Facial Landmarks* Berbasis *Raspberry Pi 4B*

1st Dimas Maulana
Universitas Buana Perjuangan
Karawang, Indonesia
if19.dimasmaulana079@mhs.ubpkarawang.ac.id

2rd Deden Wahiddin
Universitas Buana Perjuangan
Karawang, Indonesia
deden.wahiddin@ubpkarawang.ac.id

3rd Santi Arum Puspita Lestari
Universitas Buana Perjuangan
Karawang, Indonesia
santi.arum@ubpkarawang.ac.id

***Abstract*—Angka kecelakaan lalu lintas di Indonesia meningkat setiap tahunnya. Kondisi pengemudi yang mengantuk merupakan faktor penting dalam terjadinya kecelakaan. Untuk menyelesaikan masalah ini, diperlukan sebuah alat yang secara otomatis bisa mendeteksi apakah pengemudi mobil sedang dalam keadaan mengantuk atau sadar. Tujuan dari penelitian ini adalah untuk mengidentifikasi tingkat kantuk pada pengemudi mobil menggunakan metode *Facial Landmarks* dan *Eye Aspect Ratio (EAR)*. Proses yang dilakukan melibatkan beberapa tahapan diantaranya, pengumpulan data, deteksi wajah *Facial Landmarks*, perhitungan *Eye Aspect Ratio (EAR)*, deteksi kantuk, dan pengujian. Proses dimulai dengan pengambilan gambar menggunakan kamera, selanjutnya melibatkan pemrosesan gambar menggunakan *Raspberry Pi 4b* untuk mengenali wilayah wajah. Setelah berhasil diidentifikasi, selanjutnya menerapkan metode *eye aspect ratio* untuk memeriksa kondisi mata. Sistem membaca jika mata terbuka, maka pengemudi dalam kondisi sadar dan tidak mengantuk. Jika mata tertutup dalam jangka waktu yang ditentukan, sistem mengenali bahwa pengemudi mengalami kantuk dan akan memberikan peringatan suara. Penelitian ini dilakukan 40 kali pengujian pada siang hari dan pada malam hari, 20 kali pengujian pada siang hari dan 20 kali pengujian pada malam hari. Hasil pengujian menunjukkan bahwa pada siang hari, akurasi mencapai 70%, sementara pada malam hari akurasi sebesar 35%.**

Kata kunci — Facial Landmarks, Eye Aspect Ratio, Mata Kantuk, Pengemudi.

I. PENDAHULUAN

Kecelakaan lalu lintas merupakan penyebab kematian tertinggi ke-8 di dunia (WHO, 2018). Masih menurut data WHO (2018), kecelakaan lalu lintas diperkirakan merenggut nyawa sebanyak 1,35 juta jiwa tiap tahunnya. Di Indonesia, berdasarkan data yang dirilis Korlantas Polri jumlah kasus kecelakaan lalu lintas sepanjang tahun 2018 adalah sebanyak 215.492 kasus, dengan jumlah korban meninggal dunia sebanyak 50.416. Faktor utama penyebab kecelakaan lalu lintas adalah *human error*, salah satunya ialah mengantuk. Studi yang dilakukan *American Automobile Association* (*AAA*) (2018) menunjukkan bahwa sekitar 10% kecelakaan lalu lintas yang terjadi diakibatkan oleh pengemudi yang mengantuk. Fakta-fakta di atas menunjukkan bahwa mengantuk merupakan salah satu penyebab utama kecelakaan lalu lintas.

Penyebab utama peningkatan angka kecelakaan kendaraan adalah human error. Kantuk adalah proses alami yang dipicu oleh tubuh memberitahu kita untuk tidur. Ada banyak faktor penyebab kantuk pada pengemudi, seperti kelelahan, mengemudi di malam hari, kebosanan saat bepergian dan yang paling umum biasanya kurang tidur, mengarah pada keadaan mengantuk saat mengemudi salah satu tindakan berbahaya. Karena dapat menyebabkan kecelakaan untuk diri sendiri atau untuk orang lain sehingga dapat menyebabkan kematian. Untuk dapat mengurangi jumlah kasus kecelakaan lalu lintas yang disebabkan oleh mengantuk, diperlukan adanya solusi, salah satunya adalah dengan membuat sistem yang dapat mendeteksi kantuk pada pengendara, tujuannya adalah agar pengendara yang terdeteksi mengantuk segera diberikan peringatan. Terdapat beberapa metode yang dapat digunakan untuk mendeteksi kantuk pada pengendara.[1]

Penelitian sebelumnya telah dilakukan Implementasi Sistem Deteksi Mata Kantuk Berdasarkan *Facial Landmarks Detection* Menggunakan Metode *Regression Tress*. Pada penelitian ini Sistem ini akan melakukan pemantauan kondisi pengemudi dengan cara melakukan perekaman wajah yang kemudian akan diproses dengan pengolahan citra digital menggunakan *metode regression trees* pada *Raspberry Pi*.[2] Penelitian selanjutnya yaitu Deteksi Kantuk Pengendara Roda Empat Menggunakan *Haar Cascade Classifier* dan *Convolutional Neural Network* (*CNN*). Pada penelitian ini tahapan yang dilakukan yaitu *Region of Interesting* (*ROI*) mata dengan menerapkan *Haar Cascade Classifier* dan mendeteksi kondisi mata terbuka dan tertutup menggunakan *Convolutional Neural Network*. Selanjutnya melibatkan deteksi tanda-tanda kantuk dengan menghitung durasi mata terpejam, dengan merujuk pada teori *microsleep*.[3]

Pada penelitian ini, tahapan yang dilakukan yaitu deteksi wajah menggunakan metode *Facial Landmarks*, deteksi dan pemberian label mata dengan kondisi terbuka dan tertutup menggunakan *Eye Aspect Ratio*. Berdasarkan latar belakang akan dibuat dataset sendiri menggunakan *Facial Landmarks* yang akan berisi data untuk mata yang terbuka dan tertutup. Dengan deteksi mata kantuk diharapkan mampu mengurangi tingkat kecelakaan yang sering timbul karena faktor kelalaian manusia dan dapat membantu mengawasi para pengemudi untuk lebih waspada dengan mempertimbangkan kondisi yang ada.

II. TINJAUAN PUSTAKA

A. *Eye Aspect Ratio*

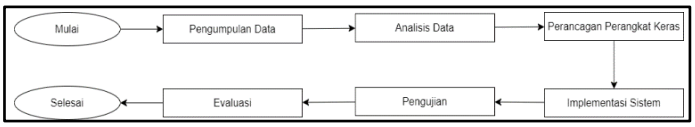
Eye Aspect Ratio adalah salah satu pustaka *dlib* yang tersedia untuk *Python*. pustaka ini digunakan untuk menentukan ambang mata. Pustaka ini adalah pendeteksi wajah yang telah dilatih sebelumnya berdasarkan histogram gradien berorientasi dan modifikasi *SVM (support vector machine*). Metode untuk *Object Recognition* rasio aspek mata adalah rasio tinggi mata terhadap lebar. Level mata adalah jarak antara kelopak mata atas dan bawah, lebar mata adalah jarak dari tepi kiri ke tepi kanan mata. Metode *Eye Aspect Ratio (EAR)* ini sering digunakan untuk menghitung kedipan mata seseorang setiap menitnya. Perhitungan *Eye Aspect Ratio (EAR)* ini dihitung berdasarkan koordinat mata kiri dan kanan yang terdapat pada *facial landmarks*. [4]

B. *Facial Landmarks*

Facial Facial landmark didefinisikan sebagai deteksi wajah berdasarkan titik-titik menonjol pada wajah yang dapat berfungsi sebagai titik jangkar pada gambar wajah. Hal tersebut berfungsi untuk menentukan bentuk biologis pada wajah manusia. Untuk membantu ekstraksi titik-titik pada wajah diperlukan sebuah tool atau metode seperti *Dlib Regression Tree* dimana metode tersebut melokalisasi 68 titik yang dilatih. Metode facial landmark dapat mendeteksi area wajah dengan mengalokasikan titik– titik pada wajah sehingga dapat dikenali oleh sistem. Metode *facial landmarks* dapat mendeteksi bagian pada wajah seperti mata, alis, mulut dan hidung.[5]

III. METODE PENELITIAN

Tahapan proses dalam penelitian ini dijalankan secara terstruktur guna mencapai hasil dan tujuan yang telah ditetapkan. Dimulai dari tahap pengumpulan data hingga evaluasi akhir, berikut merupakan tahapan proses penelitian ini:



Gambar 1. Prosedur Penelitian

Penelitian ini dimulai dengan melakukan pengumpulan data lalu dilanjutkan dengan analisis data, perancangan perangkat keras, implementasi sistem, setelah itu impelementasi sistem untuk pengujian dan terakhir yaitu evaluasi.

A. Pengumpulan Data

Pertama, Pada tahapan ini peneliti mencari dan mengumpulkan jurnal penelitian yang telah dilakukan untuk mengidentifikasi mata mengantuk dari berbagai peneliti dan beberapa metode yang berbeda. Diharapkan dari penelitian terdahulu ini, para peneliti akan mendapatkan pemahaman yang lebih baik tentang sistem yang akan dibuat kedepannya.

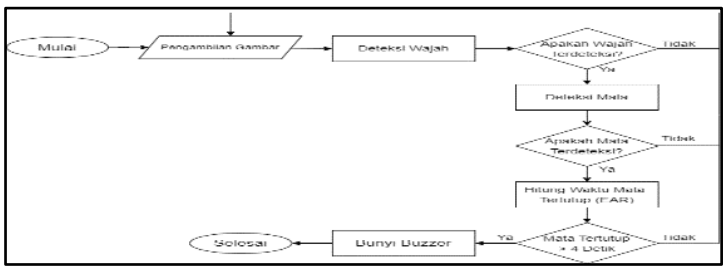
B. Analisis Data

Pada tahapan ini dilakukan dengan menganalisis kebutuhan data dengan cara memasukan *library dlib* untuk bisa menentukan *Facial Landmarks* yang bertujuan untuk mengidentifikasi bagian wajah dan mata dengan mengunduh *shape predictor* yang merupakan model.

C. Perancangan Perangkat Keras

Penelitian ini menggunakan beberapa perangkat keras diantaranya kamera dan *Raspberry Pi* yang nantinya akan disimpan pada dashboard mobil pada bagian pengemudi.

D. Implementasi Sistem



Gambar 2. *Flowchart* Impelementasi Sistem

Implementasi sistem dari Gambar 2. dapat dijelaskan sebagai berikut:

1. Pengambilan Gambar

   Proses pengambilan gambar dilakukan terlebih dahulu, pengambilan gambar dilakukan menggunakan webcam secara realtime dengan jarak pengambilan kamera dengan tempat duduk pengendara 30- 90 cm.

2. Deteksi Wajah

   Jika wajah tidak terdeteksi maka dilakukan pengambilan gambar kembali, atau jika wajah terdeteksi selanjutnya dilakukan deteksi mata.

3. Deteksi Mata

   Jika mata tidak terdeteksi maka dilakukan pengambilan gambar kembali, atau jika mata terdeteksi selanjutnya hitung waktu mata tertutup (*EAR*).

4. Hitung *EAR*

   Hitung waktu mata tertutup lebih dari 4 detik dengan *Eye Aspect Ratio*. Jika mata tidak tertutup, maka dilakukan pengambilan gambar kembali, atau jika mata tertutup lebih dari 4 detik, maka *Buzzer* akan berbunyi.

5. *Buzzer*

   Mata tertutup lebih dari 4 detik Buzzer akan berbunyi, dan selesai

Tabel 1. Hasil Deteksi

Level Kantuk	Deskripsi
Normal (Tidak mengantuk)	t_kedip < t_kantuk
Mengantuk	t_kedip > t_kantuk

Sebelum Pengujian kondisi kantuk dapat ditentukan berdasarkan informasi hasil deteksi apakah mata dalam keadaan terbuka atau tertutup. Pertama-tama, ketika mata tertutup, akan dianggap sebagai "kedipan" dan waktu akan diukur dalam satuan detik. Jika mata tertutup untuk kedua kalinya dalam waktu kurang dari 0.4 detik (t_kedip), maka ini akan dianggap sebagai mata tertutup. Namun, jika mata tertutup untuk kedua kalinya dengan waktu lebih dari 0.4 detik (t_kantuk) dan kurang dari 4 detik, maka kondisinya akan dianggap sebagai mata mengantuk. Jika terdapat hasil deteksi mata terbuka, maka penghitungan waktu akan diulang dari awal saat ada deteksi mata tertutup lagi.

IV. HASIL DAN PEMBAHASAN

Hasil yang diperoleh dari penelitian ini adalah pengembangan sistem yang mampu mengidentifikasi mata kantuk pada pengemudi mobil. Sistem ini menggunakan metode *Facial Landmarks* dan *Eye Aspect Ratio*, serta diimplementasikan pada Raspberry Pi model 4b. Tahapan proses perancangan sistem dimulai dari pengumpulan data, analisis data, perancangan perangkat keras, implementasi sistem, pengujian dan evaluasi. Berikut adalah penjelasan mengenai tahapan proses perancangan sistem ini:

A. Pengumpulan Data

Pengumpulan data dalam penelitian ini berupa library dlib dari data shape predictor yang di peroleh dari web (https://github.com/MortezaNedaei/Facial-Landmarks-Detection/blob/master/shape_predictor_68_face_landmarks.dat). Untuk membentuk model shape predictor ini menggunakan 1 foto wajah, kemudian model ini dilatih untuk menemukan 68 landmarks pada bagian wajah sebagai berikut:

Tabel 2. Pengumpulan Dataset Wajah

	**Bagian Wajah**	***Landmarks***
	*jaw*	0 - 17
	*right eyebrow*	17 - 22
	*left eye brown*	22 - 27
	*nose*	27 - 35
	*right eye*	36 - 42

	*left eye*	42 - 48
	*mouth*	48 - 68

Pada tahapan ini dilakukan dengan menganalisis kebutuhan data dengan cara melatih model shape predictor oleh *library dlib*. Untuk menemukan *landmarks* pada bagian wajah dibutuhkan beberapa tahapan diantaranya yaitu:

B. Analisis Data

1. Membuat Kode Pemetaan FACIAL_LndmRKS_IDXS

   Pertama yang harus dilakukan yaitu membuat kode pemetaan FACIAL_LANDMARKS_IDXS untuk menemukan 68 *landmarks*, dapat dilihat pada Gambar 3. berikut:

```python
from collections import OrderedDict

# define a dictionary that maps the indexes of the facial
# landmarks to specific face regions
FACIAL_LANDMARKS_IDXS = OrderedDict([
    ("mouth", (48, 68)),
    ("right_eyebrow", (17, 22)),
    ("left_eyebrow", (22, 27)),
    ("right_eye", (36, 42)),
    ("left_eye", (42, 48)),
    ("nose", (27, 35)),
    ("jaw", (0, 17))
])
```

Gambar 3. Kode Pemetaan untuk menemukan 68 Landmarks Wajah

2. Memvisualisasikan Gambar

   Selanjutnya yaitu memvisualisasikan masing – masing daerah landmarks wajah dan melapisi hasilnya pada input gambar, untuk melakukan hal ini, membutuhkan fungsi visualize_facial_landmarks yang sudah disertakan pada library imutils. Fungsi visualize_facial_landmarks membutuhkan dua argumen, diikuti dengan dua argumen opsional, dapat dijelaskan image yaitu gambar yang akan kita gunakan untuk menggambar visualisasi landmarks wajah, shape merupakan larik NumPy yang berisi 68 koordinat tengara wajah yang dipetakan ke berbagai bagian wajah, kemudian colors adalah daftar tupel BGR yang digunakan untuk memberi kode warna pada setiap wilayah landmarks wajah, dan alpha merupakan parameter yang digunakan untuk mengontrol opasitas overlay pada gambar asli. Kemudian pada baris 68 - 69 membuat dua salinan dari gambar *input*, dan membutuhkan salinan ini sehingga kita dapat menggambar hamparan semi-transparan pada gambar output. Kemudian pada baris 72 melakukan pengecekan untuk melihat apakah daftar warna tidak ada, dan jika ya, menginisialisasinya dengan daftar tupel BGR yang telah ditetapkan (ingat, *OpenCV* menyimpan warna/intensitas piksel dalam urutan *BGR*, bukan *RGB*).

```python
# show landmarks on the input image
def visualize_facial_landmarks(image, shape, colors=None, alpha=0.75):
    # create two copies of the input image -- one for the
    # overlay and one for the final output image
    overlay = image.copy()
    output = image.copy()
    # if the colors list is None, initialize it with a unique
    # color for each facial landmark region
    if colors is None:
        colors = [(19, 199, 109), (79, 76, 240), (230, 159, 23),
                  (168, 100, 168), (158, 163, 32),
                  (163, 38, 32), (180, 42, 220)]
```

Gambar 4. Memvisualisasikan Landmarks Wajah

3. Mencari Entri Pada Kode Pemetaan FACIAL_LndmRKS_IDXS

   Selanjutnya menelusuri setiap entri dalam kamus FACIAL_LANDMARKS_IDXS. Untuk setiap wilayah ini mengekstrak indeks dari bagian wajah yang diberikan dan mengambil koordinat (x, y) dari larik bentuk NumPy. Selanjutnya pada baris 84 – 90 memeriksa apakah sedang memplot fungsi, dan jika demikian, perlu mengulang melalui titik-titik individu, menggambar garis yang menghubungkan titik-titik fungsi bersama-sama. Kemudian pada baris 93 - 95 menangani perhitungan convex_hull dari titik-titik dan menggambar hull pada overlay.

```python
    # loop over the facial landmark regions individually
    for (i, name) in enumerate(FACIAL_LANDMARKS_IDXS.keys()):
        # grab the (x, y)-coordinates associated with the
        # face landmark
        (j, k) = FACIAL_LANDMARKS_IDXS[name]
        pts = shape[j:k]
        # check if are supposed to draw the jawline
        if name == "jaw":
            # since the jawline is a non-enclosed facial region,
            # just draw lines between the (x, y)-coordinates
            for l in range(1, len(pts)):
                ptA = tuple(pts[l - 1])
                ptB = tuple(pts[l])
                cv2.line(overlay, ptA, ptB, colors[i], 2)
        # otherwise, compute the convex hull of the facial
        # landmark coordinates points and display it
        else:
            hull = cv2.convexHull(pts)
            cv2.drawContours(overlay, [hull], -1, colors[i], -1)
```

Gambar 5. Mencari Setiap Entri Pada FACIAL_LANDMARKS_IDXS

4. Membuat *Overlay* Transparan

   Selanjutnya dilakukan pembuatan *overlay* transparan dengan memanfaatkan fungsi cv2.addWeighted.

```python
    # apply the transparent overlay
    cv2.addWeighted(overlay, alpha, output, 1 - alpha, 0, output)
    # return the output image
    return output
```

Gambar 6. Kode Membuat Overlay Transparan

Selanjutnya metode cv2.addWeighted memerlukan enam argumen, pertama adalah overlay, gambar yang ingin kita "hamparkan" di atas gambar asli menggunakan tingkat transparansi alpha yang disediakan, kedua adalah transparansi alpha sebenarnya dari overlay. Semakin dekat alpha ke 1.0, overlay akan semakin buram. Demikian pula, semakin dekat alpha ke 0,0, overlay akan terlihat semakin transparan, ketiga untuk cv2.addWeighted adalah gambar sumber — dalam hal ini, gambar asli dimuat dari disk, nilai beta sebagai argumen keempat. Beta didefinisikan sebagai 1-alfa. Kita perlu mendefinisikan alpha dan beta sedemikian rupa sehingga alpha + beta = 1.0. Parameter kelima adalah nilai gamma — skalar ditambahkan ke jumlah tertimbang. *Gamma* bisa disebut sebagai konstanta yang ditambahkan ke gambar keluaran setelah menerapkan penambahan bobot. Setelah menerapkan fungsi visualize_facial_landmarks pada sebuah gambar dan titik-titik landmarks wajah yang terkait, hasilnya dapat dilihat pada Gambar 7. di bawah ini:

Gambar 7. Hasil Visualisasi Landmarks Wajah

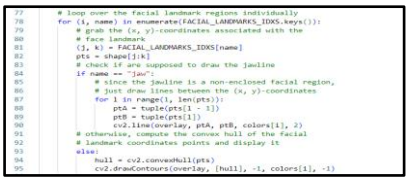
5. Mengekstrak Bagian Wajah

Selanjutnya pada Gambar 8. merupakan kode untuk mengimpor *library Python* yang dibutuhkan pada baris 1 - 5, kemudian pada baris 13 - 14 menginstal detektor wajah berbasis *HOG dlib* dan memuat prediktor *landmarks*, setelah itu melakukan prapemrosesan input gambar yang ada pada baris 21 – 22, dan pada baris 30 mendeteksi wajah dalam input gambar.

Gambar 8. Mengekstrak Bagian Wajah Dengan Dlib,OpenCV, dan Python

6. Menentukan Daerah *Landmarks*

Selanjutnya menentukan *landmarks* wajah untuk setiap wilayah wajah dari *ROI* dan mengubah 68 titik menjadi larik *NumPy*. Kemudian pada baris 38 mengulanginya untuk setiap bagian wajah, setelah itu menggambar nama atau label wilayah wajah yang ada pada baris 42 dan 43, dan menggambar masing-masing *landmarks* wajah sebagai lingkaran yang ada pada baris 46 - 47. Untuk mengekstrak setiap wilayah wajah, hanya perlu menghitung kotak pembatas dari koordinat (x, y) yang terkait dengan wilayah tertentu dan menggunakan pemotongan larik *NumPy* untuk mengekstraknya.



Gambar 9. Kode Menentukan Wilayah Landmarks Wajah

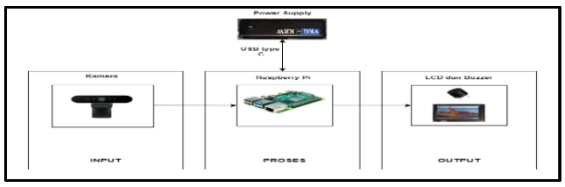
7. Menghitung Kotak Pembatas Daerah Wajah

Selanjutnya menghitung kotak pembatas wilayah ditangani pada baris 50 melalui cv2.boundingRect. Dengan menggunakan pemotongan larik *NumPy*, setelah itu dapat mengekstrak ROI pada Baris 51. Kemudian *ROI* diubah ukurannya menjadi lebar 250 piksel sehingga dapat memvisualisasikannya dengan baik. Pada baris 54 - 56 yaitu menampilkan wilayah wajah individu ke layar, sedangkan pada baris 59 – 61 menerapkan fungsi visualize_facial_landmarks untuk membuat overlay transparan untuk setiap bagian wajah. Dan hasil pelatihan model shape predictor dengan library dlib telah dibuat, maka bisa dilihat pada Tabel 2 merupakan hasil untuk menemukan 68 landmarks.

Gambar 10. Menghitung Kotak Pembatas Wilayah

C. Hasil Perancangan Perangkat Keras

Hasil dari perancangan perangkat keras meliputi input, proses dan output, Input pada sistem ini adalah citra wajah yang diambil oleh kamera, kemudian diambil kedipan pada bagian mata dan dilihat pergerakan pengedipan matanya. Proses pada sistem ini adalah *Raspberry Pi* untuk memproses citra yang dikirim oleh kamera. *Output* pada sistem ini adalah *LCD* dan *Buzzer* untuk peringatan berupa pesan *text* dan suara.



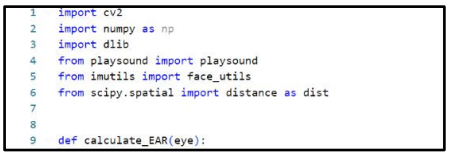
Gambar 11. Perancangan Perangkat Keras

D. Hasil Implementasi Sistem

Raspberry Pi menggunakan sistem operasi Raspbian, yang merupakan varian khusus dari *Linux Debian* yang dirancang khusus untuk perangkat *Raspberry Pi*.

1. *Library*

   Untuk pengenalan citra wajah dan mata menggunakan Bahasa pemrograman python dan membutuhkan beberapa library salah satunya yaitu opencv. Pada Gambar 12. merupakan beberapa library yang digunakan untuk membangun pengenalan citra wajah dan mata.



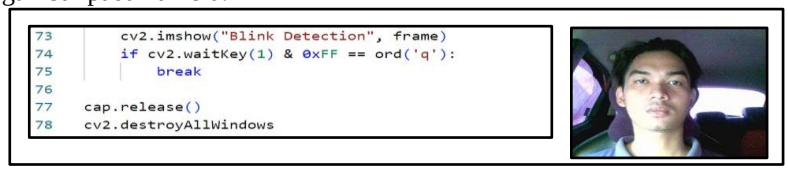
```
import cv2
import numpy as np
import dlib
from playsound import playsound
from imutils import face_utils
from scipy.spatial import distance as dist


def calculate_EAR(eye):
```

Gambar 12. *Library Python*

2. Menjalankan Kamera

   Selanjutnya pada Gambar 13. merupakan kode program untuk menjalankan kamera pada *open cv* berjalan dengan baik, dan hasil pengambilan gambar pada kamera.



```
    cv2.imshow("Blink Detection", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows
```

Gambar 13. Kode Membuka Kamera dan Hasil Pengambilan Gambar Pada Kamera

3. Deteksi Wajah

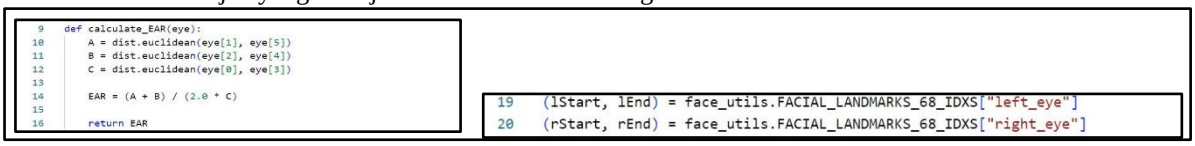
   Shape predictor atau bisa disebut prediktor *landmarks* yaitu fitur yang memprediksi koordinat x dan y dari gambar dengan memetakan beberapa titik koordinat yang disebut *landmarks*. prediktor ini dilatih untuk menemukan 68 facial landmarks bagian wajah tertentu, termasuk mata, hidung dan mulut. Prediktor mengambil gambar dan kotak pembatas dari wajah yang terdeteksi sebagai input dan mengembalikan prediktor orientasi wajah. Prediktor memerlukan file khusus dengan model yang terlatih yaitu "shape_predictor_68_face_landmarks.dat".

Gambar 14. Model *Shape Predictor dan* Landmarks Wajah Yang Terdeteksi

Selanjutnya untuk mendeteksi landmarks wajah menggunakan metode serupa. Pertama yang harus dilakukan yaitu membuat prediktor landmarks wajah dlib.shape_predictor dari library dlib. Kemudian setelah mengekstrak semua 68 landmarks bagian wajah, setelah itu mendeteksi satu per satu, dan membuat loop untuk mengekstrak semua 68 titik.

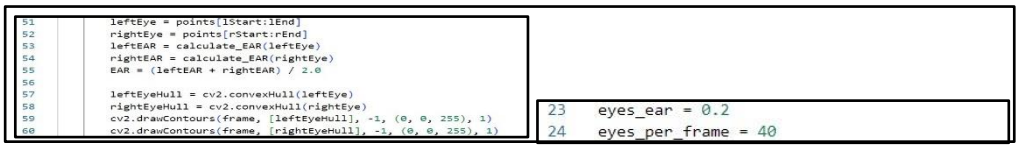
4. Deteksi Mata

Proses penghitungan mata mengantuk menggunakan *eye aspect ratio* dimulai dengan memasukan *library dlib* untuk dapat mengetahui *landmarks* wajah yang bertujuan untuk mendeteksi bagian mata.



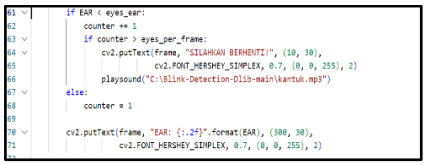
Gambar 15. Perhitungan *Eye Aspect Ratio* (*EAR*) dan Program *Landmarks* Mata Kanan dan Kiri

Pada Gambar 15. merupakan perhitungan *eye aspect ratio* dan program *Landmarks* mata kanan dan kiri. Untuk menentukan deteksi mata, pertama yang harus dilakukan adalah mendapatkan *Euclidean* antara 2 koordinat mata, dengan mengambil koordinatnya, setelah itu akan mendapatkan landmarks vertical mata, dan kemudian akan mendapatkan landmarks mata horizontal dengan menggunakan algoritma yang sama, setelah mendapatkan koordinat, hitung *eye aspect ratio* (*EAR*), dan terakhir mengembalikan *eye aspect ratio* (*EAR*). Kemudian untuk membuat program *landmarks* pada mata kanan dan kiri ini menggunakan *library imutils* face_utils.FACIAL_LANDMARKS_68_IDXS berisi *indexs* yang telah ditentukan sebelumnya bervariasi dalam beberapa fitur pada bagian wajah. Variabel (*lStart*, *lEnd*) menyimpan awal dan indeks orientasi wajah terakhir untuk mata kiri. dan variabel (*rStart*, *rEnd*) menyimpan awal dan indeks akhir landmarks wajah di mata kanan. Indeks tersebut kemudian digunakan untuk mengekstrak landmarks wajah untuk mata dari objek bentuk yang dikembalikan oleh dlib pendeteksi *landmarks* wajah. Setelah *landmarks* wajah dihapus, mereka dipindahkan ke *eye aspect ratio* (*EAR*) berfungsi untuk menghitung rasio bentuk mata (*EAR*) yang digunakan untuk mendeteksi kantuk. Pada Gambar 17. di bawah ini menunjukkan mengantuk dan tidak mengantuk yang terdeteksi:



Gambar 16. Eye Aspect Ratio dan Kode Menentukan Ambang Batas dan Frame

Selanjutnya pada Gambar 16. Merupakan *Eye Aspect Ratio dan* Kode untuk menentukan ambang batas dan frame. Untuk *EAR* merupakan kode landmarks yang terdeteksi oleh fungsi *library dlib detector* untuk menemukan mata pada gambar. Kode ini menggunakan indeks landmarks wajah yang sesuai dengan mata kiri dan kanan, disimpan dalam variabel *lStart*, *lEnd*, *rStart*, *rEnd*, untuk mengekstrak koordinat mata kiri dan kanan dari bentuk objek dan menyimpannya dalam variabel *leftEye* dan *rightEye*. Selanjutnya untuk menentukan nilai ambang batas mata yaitu mata (0.20) yang ada pada kode baris 23, kemudian sistem akan mendeteksi kantuk ketika pengguna menutup matanya selama frame pada video bernilai (40).



Gambar 17. Kode Menghitung Ambang Batas *Eye Aspect Ratio*

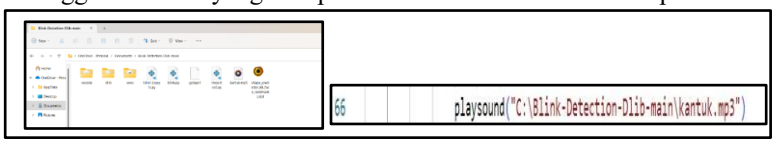
Kode pada Gambar 17. di atas memeriksa apakah *eye aspect ratio* yang disimpan dalam variabel "*EAR*", adalah kurang dari ambang batas yang ditentukan, disimpan dalam konstanta "eyes_ear". Jika aspek rasio kurang dari ambang batas, itu berarti mata tertutup, menunjukkan bahwa pengemudi tersebut mengantuk. Ketika rasio aspek kurang dari ambang batas, kode meningkatkan nilai penghitung variabel “counter” sebesar 1. Jika variabel *counter* mencapai nilai ambang batas yang ditentukan "eyes_per_frame", artinya mata sudah tertutup selama angka tertentu frame berturut-turut, menunjukkan bahwa orang tersebut mengantuk.

	Mengantuk
	Tidak Mengantuk

Gambar 18. Deteksi Mata *Eye Aspect Ratio*

5. File Audio (*Buzzer)*

Selanjutnya memasukan kode notifikasi berupa audio untuk mata yang telah teridentifikasi mengantuk. Sebelumnya siapkan file audio, pada sistem ini menggunakan file yang bertipe MP3 dan bernama "kantuk.mp3".

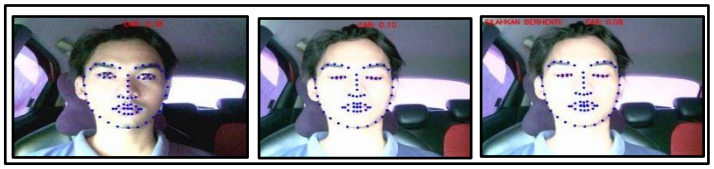


Gambar 19. File Audio dan Kode Pemanggil Audio

Setelah file audio tersedia, selanjutnya melakukan kode untuk pemanggilan file audio yang akan digunakan. kode memutar file suara dengan bantuan *library playsound*. Pada baris 23 – 24 menunjukan bahwa jika nilai "ear" lebih kecil dari nilai ambang batas (0.20) ketika pengguna menutup matanya selama frame pada video bernilai (40) maka sistem akan membaca pengguna mengantuk dan sistem akan memutar file suara bernama "kantuk.mp3" yang ada pada line 66. Setelah semua kode telah berfungsi dan tidak ada error yang terdeteksi, maka hasil program identifikasi mata kantuk bisa dijalankan.

6. Hasil Program

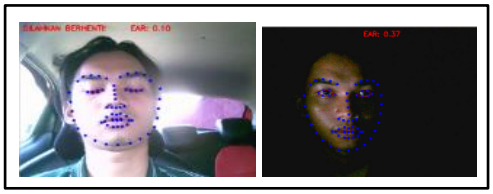
Selanjutnya pada Gambar 20. merupakan hasil program yang sudah dijalankan, pada diagram yang memiliki tanda warna merah adalah sebuah hasil dari *library dlib* yang berguna sebagai *eye aspect ratio*, dan diagram yang memiliki tanda warna biru adalah hasil dari *library dlib* yang berguna sebagai facial landmarks. Ketika mata dalam keadaan tertutup dan kemudian aplikasi sedang aktif, proses yang akan dijalankan framework dan juga akan bekerja pada *EAR* di video seperti yang ditampilkan pada Gambar 19. tersebut, ketika sampai pada nilai yang telah ditentukan, sistem akan menampilkan peringatan pesan "SILAHKAN BERHENTI".



Gambar 20. Hasil Program Saat Dijalankan

E. Pengujian

Pengujian ini telah dilakukan sebanyak 40 kali pada siang hari dan malam hari, 20 kali pengujian pada siang hari, dan 20 kali pengujian pada malam hari, dengan jarak pengemudi menuju kamera sejauh 50 cm. Contoh gambar pengujian ini dapat dilihat pada Gambar 21. di bawah ini.



Gambar 21. Contoh Gambar Pengujian

Tabel 3. Pengujian

Percobaan	Kondisi Mata	EAR	Jeda Waktu (s)	Audio	Kondisi	Keterangan
1.	Tertutup	0,10	4,04	Bunyi	Siang hari	Sesuai
2.	Tertutup	0,09	4,05	Bunyi	Siang hari	Sesuai
3.	Tertutup	0,11	4,09	Bunyi	Siang hari	Sesuai
4.	Tertutup	0,14	4,51	Bunyi	Siang hari	Sesuai
5.	Terbuka	0,37	-	Mati	Siang hari	Tidak Sesuai
6.	Tertutup	0,04	4,37	Bunyi	Siang hari	Sesuai
7.	Terbuka	0,34	-	Mati	Siang hari	Tidak Sesuai

8.	Tertutup	0,13	4,81	Bunyi	Siang hari	Sesuai
9.	Tertutup	0,13	4,81	Bunyi	Siang hari	Sesuai
10.	Terbuka	0,36	-	Mati	Siang hari	Tidak Sesuai
11.	Tertutup	-	-	Mati	Siang hari	Tidak Sesuai
12.	Tertutup	0,10	4,74	Bunyi	Siang hari	Sesuai
13.	Tertutup	0,09	4,23	Bunyi	Siang hari	Sesuai
14.	Tertutup	-	-	Mati	Siang hari	Tidak Sesuai
Percobaan	**Kondisi Mata**	**EAR**	**Jeda Waktu (s)**	**Audio**	**Kondisi**	**Keterangan**
15.	Tertutup	-	-	Mati	Siang hari	Tidak Sesuai
16.	Tertutup	0,06	4,42	Bunyi	Siang hari	Sesuai
17.	Tertutup	0,07	4,87	Bunyi	Siang hari	Sesuai
18.	Tertutup	0,11	4,56	Bunyi	Siang hari	Sesuai
19.	Tertutup	0,16	4,34	Bunyi	Siang hari	Sesuai
20.	Tertutup	0,10	4,09	Bunyi	Siang hari	Sesuai
21.	Tertutup	-	-	Mati	Malam hari	Tidak Sesuai
22.	Tertutup	-	-	Mati	Malam hari	Tidak Sesuai
23.	Tertutup	0,18	4,93	Bunyi	Malam hari	Sesuai
24.	Terbuka	0,34	-	Mati	Malam hari	Tidak Sesuai
25.	Tertutup	-	-	Mati	Malam hari	Tidak Sesuai
26.	Tertutup	0,11	4,84	Bunyi	Malam hari	Sesuai
27.	Tertutup	0,13	4,72	Bunyi	Malam hari	Sesuai
28.	Terbuka	0,37	-	Mati	Malam hari	Tidak Sesuai
29.	Terbuka	0,30	-	Mati	Malam hari	Tidak Sesuai
30.	Tertutup	0,15	4,86	Bunyi	Malam hari	Sesuai
31.	Tertutup	0,16	-	-	Malam hari	Tidak Sesuai
32.	Terbuka	0,40	-	-	Malam hari	Tidak Sesuai
33.	Tertutup	0,15	-	-	Malam hari	Tidak Sesuai
34.	Tertutup	0,22	-	-	Malam hari	Tidak Sesuai
35.	Tertutup	0,16	4,12	Bunyi	Malam hari	Sesuai
36.	Tertutup	0,15	4,76	Bunyi	Malam hari	Sesuai
37.	Terbuka	0,20	-	-	Malam hari	Tidak Sesuai
38.	Tertutup	0,10	-	-	Malam hari	Tidak Sesuai

39.	Tertutup	0,12	-	-	Malam hari	Tidak Sesuai
40.	Tertutup	0,18	4,15	Bunyi	Malam hari	Sesuai

F. Evaluasi

1. Hasil Pengujian Siang Hari

Setelah dilakukan pengujian sebanyak 20 kali bahwa nilai yang ditemukan untuk kantuk yang teridentifikasi pada siang hari sebanyak 14. Dengan demikian, tingkat keakuratan rata-rata untuk mengidentifikasi mata kantuk pada pengemudi adalah sebagai berikut:

$$Accuracy\ (Siang\ Hari) = \frac{14}{20}\ x\ 100\%) = 70\%$$
$$Error\ = \frac{13}{20}\ x\ 100\%\ = 60\%$$

Hasil dari pengujian pada siang hari kantuk dapat terdeteksi dengan tingkat accuracy berhasil 70% dan tingkat accuracy error sebanyak 30%.

2. Hasil Pengujian Siang Hari

Pengujian pada malam hari terdapat 7 nilai teridentifikasi dari 20 pengujian dengan tingkat akurasi dan rata – rata sebagai berikut:berikut:

$$Accuracy\ (Malam\ Hari) = \frac{7}{20}\ x\ 100\% = 35\%$$
$$Error\ = \frac{13}{20}\ x\ 100\%\ = 60\%$$

Pengujian pada malam hari kantuk dapat terdeteksi dengan tingkat *accuracy* berhasil sebanyak 35% dan tingkat *accuracy error* sebanyak 65%.

3. Kekurangan

Pada saat pengujian dilakukan pada malam hari, kamera yang digunakan untuk mendeteksi mata mengantuk seringkali menghadapi kendala dalam hal kinerjanya. Kondisi pencahayaan yang kurang pada malam hari dapat menghambat kemampuan kamera untuk mengenali tanda-tanda mata mengantuk dengan akurat. Sehingga mengakibatkan hasil deteksi yang kurang optimal. Pada saat proses berlangsung pada *Raspberry Pi*, sering kali terjadi peningkatan suhu yang dapat menyebabkan panas pada perangkat tersebut.

V. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian deteksi mata kantuk pada pengendara mobil dengan metode *eye aspect ratio dan facial landmarks*, dapat diambil kesimpulan bahwa penelitian ini berhasil dilakukan pada perangkat keras *raspberry pi 4b* menggunakan kamera, sementara audio dapat dihubungkan langsung ke audio mobil atau ke audio *portable* menggunakan *bluetooth* pada *raspberry pi 4b*. dan untuk mengidentifikasi pengemudi yang mengantuk dilakukan dengan memanfaatkan parameter berupa kondisi mata tertutup dalam rentang waktu yang telah ditentukan. Ketika sistem mendeteksi bahwa mata pengemudi tertutup selama periode waktu yang telah ditetapkan, sistem akan mengenali kondisi tersebut sebagai tanda pengemudi mengantuk.

Hasil dari pengujian sistem deteksi mata kantuk pada pengendara mobil dengan metode *eye aspect ratio* dan *facial landmarks* pada siang hari mendapatkan tingkat *accuraccy* 70% dan tingkat *error* 30%, Sedangkan hasil dari pengujian pada malam hari mendapatkan tingkat *accuracy* 35% dan tingkat *error* 65%.

Saran yang diberikan pada penelitian ini untuk upaya pengembangan penelitian selanjutnya agar lebih optimal yaitu kamera pendeteksi mata mengantuk untuk pengemudi dapat ditingkatkan dengan fitur malam dan piksel yang lebih tinggi agar kamera dapat mendeteksi mata mengantuk pengemudi dengan lebih akurat di malam hari, saran selanjutnya pada saat pengerjaan sistem pada kendaraan mobil diharapkan alat disertai dengan pendingin karena penggunaan yang tertunda akan menyebabkan alat menjadi panas, saran terakhir untuk pengendara mobil diharapkan dapat memasang alat pendeteksi kantuk untuk mengurangi angka kecelakaan.

PENGAKUAN

Naskah ilmiah ini merupakan bagian dari penelitian Tugas Akhir yang dilakukan oleh Dimas Maulana dengan judul Deteksi Mata Kantuk Pada Pengendara Mobil Dengan Metode *Eye Aspect Ratio* Dan *Facial Landmarks*. Penelitian ini dibimbing oleh **Bapak Deden Wahiddin, M.Kom** dan **Ibu Santi Arum Puspita Lestari, M.Pd.**

DAFTAR PUSTAKA

[1] C. Aj. Saputra, D. Erwanto, en P. N. Rahayu, “Deteksi Kantuk Pengendara Roda Empat Menggunakan Haar Cascade

Classifier Dan Convolutional Neural Network", *JEECOM J. Electr. Eng. Comput.*, vol 3, no 1, bll 1–7, 2021, doi: 10.33650/jeecom.v3i1.1510.

[2] Andre Hartoko Aji Putra Perdana, Susijanto Tri Rasmana, en Heri Pratikno, "Implementasi Sistem Deteksi Mata Kantuk Berdasarkan Facial Landmarks Detection Menggunakan Metode Regression Trees", *J. Technol. Informatics*, vol 1, no 1, bll 1–9, 2019, doi: 10.37802/joti.v1i1.1.

[3] R. T. Puteri en P. Utaminingrum, "Deteksi Kantuk Menggunakan Kombinasi Haar Cascade dan Convolutional Neural Network", *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol 4, no 3, bll 816–821, 2020.

[4] M. A. Maulla, "Deteksi microsleep berbasis eye aspect ratio menggunakan histogram oriented gradient+ support vector machine linier untuk memicu alarm", 2022, [Online]. Available at: http://digilib.uinsby.ac.id/id/eprint/52395.

[5] R. P. H. Sejati en R. Mardhiyyah, "Deteksi Wajah Berbasis Facial Landmark Menggunakan OpenCV Dan Dlib", *J. Teknol. Inf.*, vol 5, no 2, bll 144–148, 2021, doi: 10.36294/jurti.v5i2.2220.