Real-time Drowsiness Detection via Distributed Embedded Systems

Development of a non-invasive computer vision-based system for monitoring driver fatigue

Students:
Daniel Pipitone
Leonardo Cecchini

Introduction

1

2

Problem formulation

The Problem

- Impairment of reaction time & attention.
- High volume of road accidents.
- Need for early identification in monotonous tasks.

Technical Challenges

- Variable lighting conditions.
- Head poses and facial expressions.
- Hardware limitations (Embedded vs. Real-time)

Introduction

1

2

System Highlights & Methodology

Core Technology

Landmark extraction via MediaPipe

Behavioral analysis using EAR and MAR

Architecture

Modular and distributed system

Support for Standalone or Network-offloaded mode

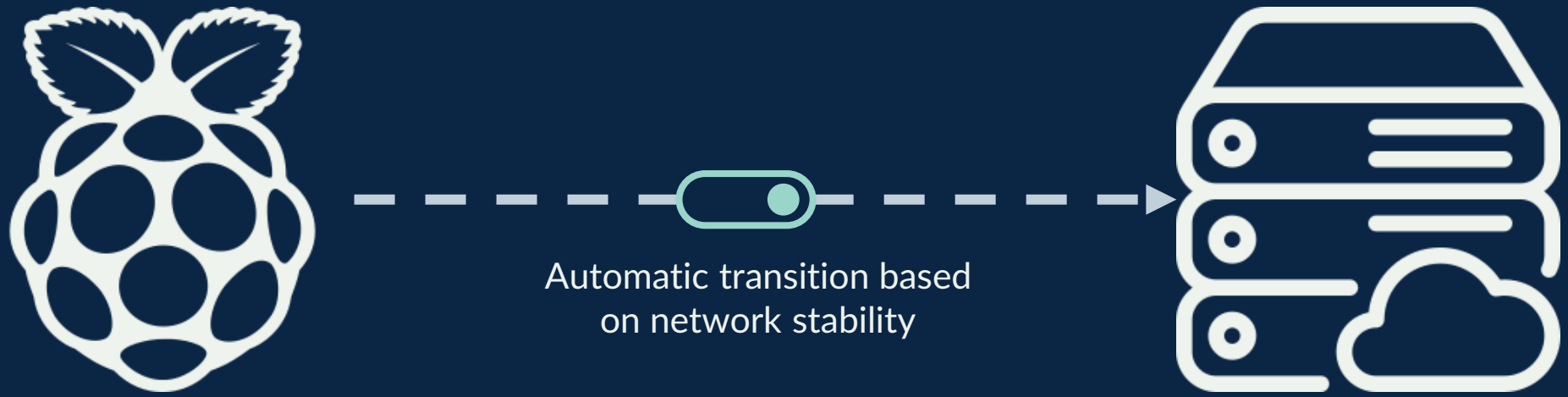
Value Proposition

Low-cost, non-invasive, and scalable compared to expensive physiological sensors.

System Architecture

Client-Server mode

High FPS • Rich UI (landmarks & metrics)



Automatic transition based
on network stability

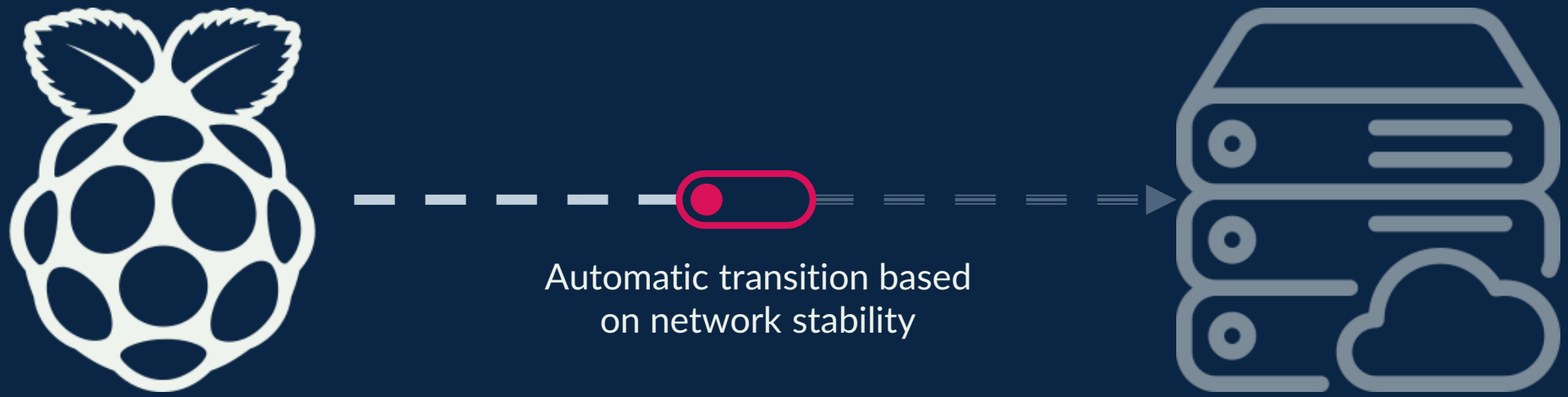
Raspberry Pi
Video Acquisition &
Send raw frames

Server
Landmarks Detection &
Drowsiness Detection

System Architecture

Standalone mode

Reduced FPS • Local processing • Rich UI (landmarks & metrics)



Automatic transition based
on network stability

Raspberry Pi
Video Acquisition &
Real-time detection

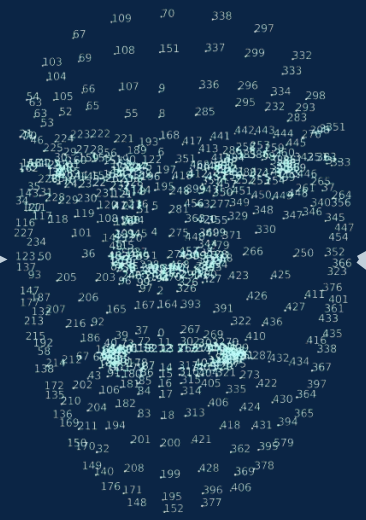
Server
- -

Facial Landmarks

Visual pipeline for extracting facial features needed for EAR and MAR calculation

Input Frame & Face Detection

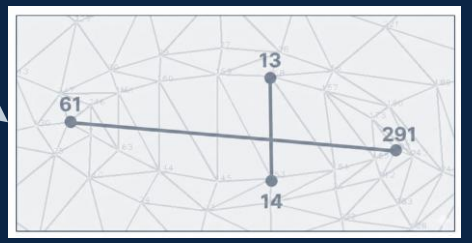
Landmark Extraction Model



Dense 3D Mesh Generation (468 pts)

Eyes

Mouth



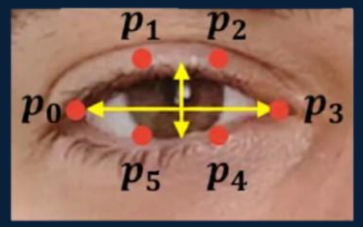
Drowsiness Metrics

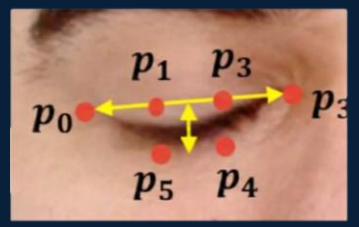
Eye Aspect Ratio

$$\frac{\|p_1 - p_5\| + \|p_2 - p_4\|}{2\|p_0 - p_3\|}$$

Normal state: $EAR > \tau_{eye}$

Drowsiness detected: $EAR < \tau_{eye}$ **per N frame**



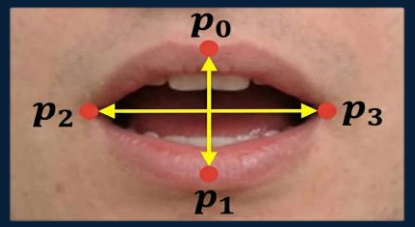


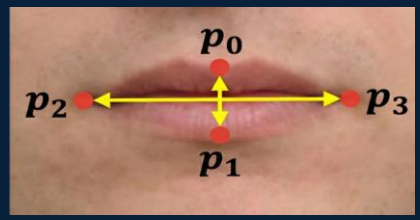
Mouth Aspect Ratio

$$\frac{\|p_0 - p_1\|}{\|p_2 - p_3\|}$$

Normal state: $MAR > \tau_{mouth}$

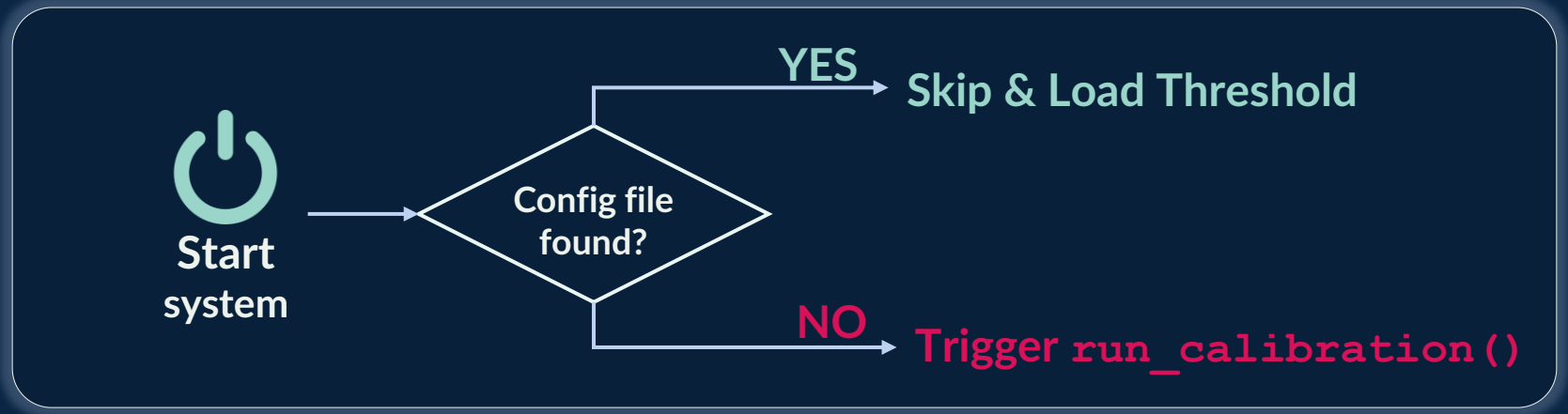
Yawning detected: $MAR < \tau_{mouth}$ **per M frame**





Source: Eye blink detection using facial landmarks

EAR Personalization



`run_calibration()`

User positioning & Preview

EAR Data Collection

For 10 seconds

Threshold Generation

$$\overline{\text{EAR}}_{\text{samples}} \times 0.85$$

15% less than the average sample collected during calibration

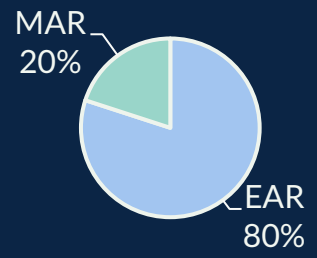
2

3

4

Drowsiness Risk Scoring

$$Final\ Score = 0.8 \cdot EAR_{tot} + 0.2 \cdot MAR_{tot}$$



$$EAR_{tot} = 0.5 \cdot S_{ear} + 0.5 \cdot D_{ear}$$

- **Value Score (S_ear):** Measures the severity of the closure
- **Duration Score (D_ear):** Assesses the frequency of events based on total_drowsy_events.

$$MAR_{tot} = 0.5 \cdot S_{mar} + 0.5 \cdot D_{mar}$$

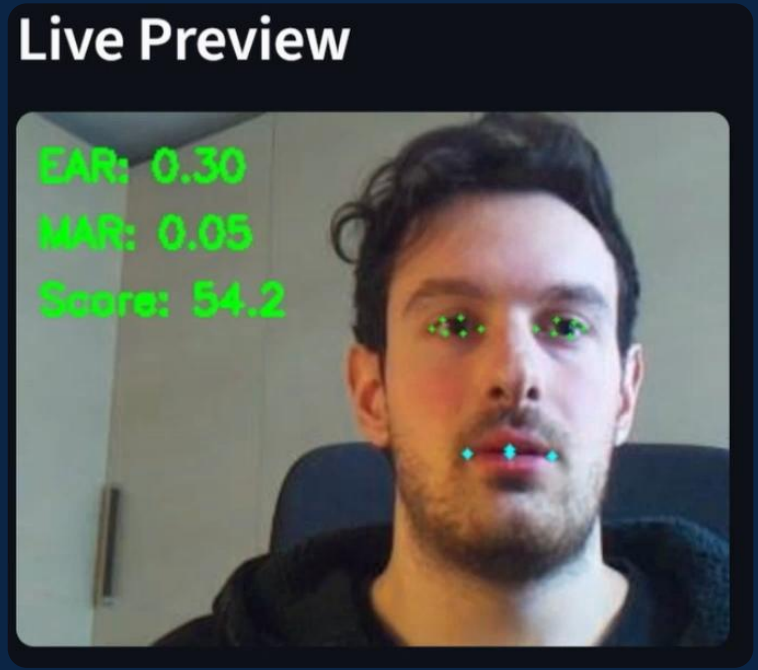
- **Value Score (S_mar):** Quantifies the amplitude of yawning beyond the threshold
- **Duration Score (D_mar):** Increases proportionally to the total number of yawns detected.

Cumulative Logic: The score depends not only on the duration of the single event, but on the total number of events (*total_events*).

Persistence: This means that the perceived risk increases if micro-sleeps are frequent, even if brief.

Algorithm

```
For each frame:
    Detect face and landmarks
    If landmarks are valid:
        face_detected = True
        Compute EAR and MAR
        Update EAR and MAR counters
        If EAR below personalized threshold for N frames:
            Trigger drowsiness event
        If MAR above threshold for M frames:
            Trigger yawning event
    Else:
        face_detected = False
        Update face lost counte
        If face lost persists beyond threshold
            Trigger "face lost" alert
```



Prototype

Software Module Allocation

MODULE	RASPBERRY Pi	PC SERVER
Video Acquisition	✓	-
Mediapipe Face Mesh	⇢	✓
EAR/MAR Computation	⇢	✓
Visualization Dashboard	⇢	✓
Event Logging	✓	✓

Performance

System Monitoring Methodology

CPU & RAM Profiling

- **Library**: `psutil`
- **CPU Metric**: Sum of the usage of each core
- **RAM Metric**: Percentage of virtual memory used.

Thermal Monitoring

- **Library**: `gpiozero`.
- **Source**: SoC internal sensor.
- **Metric**: Temperatures in C°

FPS Calculation

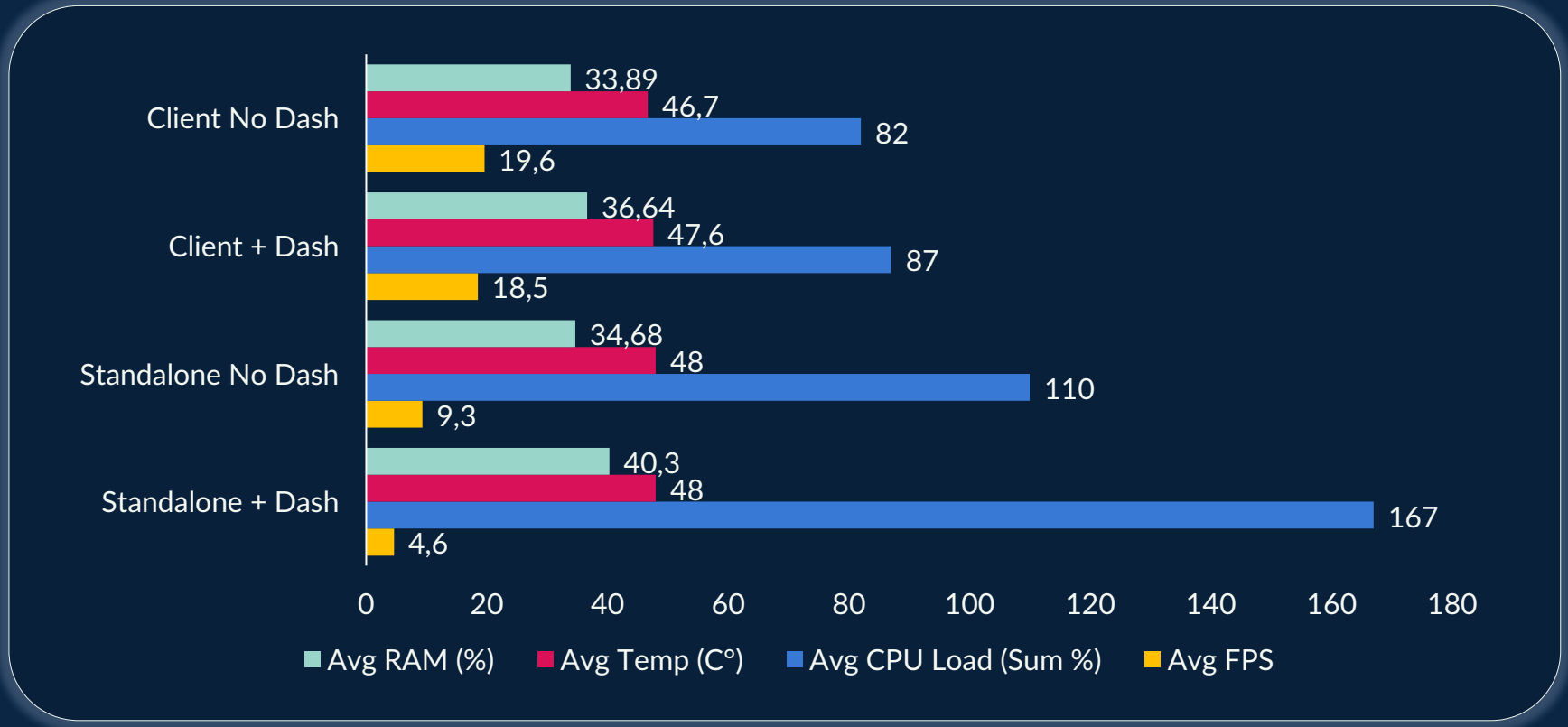
Formula: $FPS = \frac{\text{frame\_count}}{\text{elapsed\_time}}$

calculated by dividing the number of frames processed by the elapsed time.

Resource Optimization

- **Update Policy**: Periodic refresh every 1s
- **Trigger**: Operation performed when `frame_count % config. CAMERA_FPS == 0`

Comparative Performance Analysis



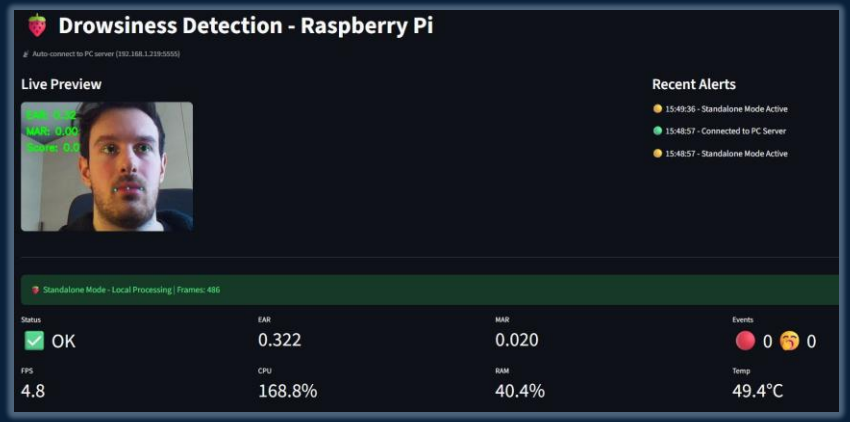
Computational offloading in Client mode provides a +300% increase in frame rate while reducing CPU load by 50%

Dashboards Demo

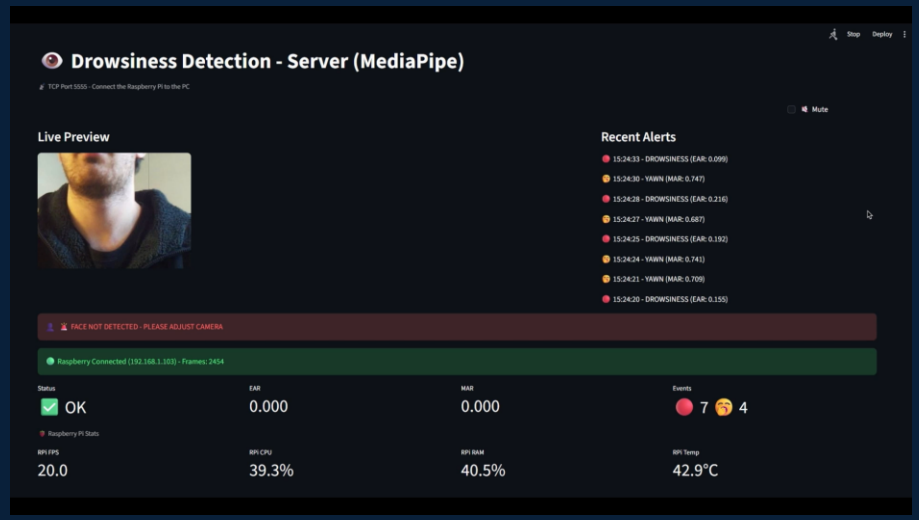
Raspberry

Client-server mode

Standalone mode

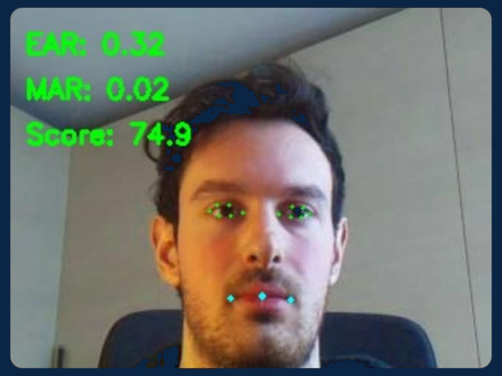


Server



Status & Alerts

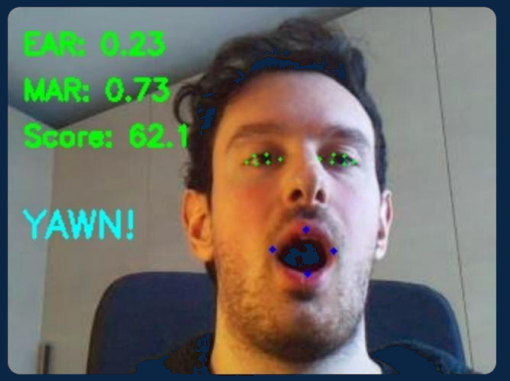
Normal



Drowsiness

No-Face

Yawn



7

References

- Soukupova, Tereza, and Jan Cech. "Eye blink detection using facial landmarks." 21st computer vision winter workshop, Rimske Toplice, Slovenia. Vol. 2. 2016.
- Lugaresi, Camillo, et al. "Mediapipe: A framework for building perception pipelines." *arXiv preprint arXiv:1906.08172* (2019).
- Evaluation of Techniques for Ocular Measurement as an Index of Fatigue and as the Basis for Alertness Management