ARIA Adaptive Real-time Intelligence Assistant

A Multimodal AI Vision System | Complete Beginner's Manual

Table of Contents

---

1. What Is This Project?

ARIA is a real-time AI vision system that uses your webcam (or phone camera) to run **7 AI modules simultaneously** on live video. Each module is a separate AI feature that you can turn on or off at any time using your keyboard.

The system uses deep learning models neural networks trained on millions of images to understand what it sees in real time. You do not need any coding knowledge to **use** it. This document will guide you through everything.

What it can do at a glance:

- Detect and outline every object in the scene with colored masks
- Read your facial expression (Happy, Sad, Angry, etc.)
- Recognize hand gestures with a skeleton drawn on your hand
- Read handwritten text drawn in the air with your finger
- Map 468 landmarks across your face in 3D
- Control your computer entirely with hand gestures (volume, scroll, mouse, click)
- Alert you when you are drowsy or not paying attention

AI Coverage: ~92% every core feature is powered by a neural network or machine learning model.

---

2. How to Install & Run

Step 1 Install Python

Download Python 3.10 or 3.11 from https://python.org
During install, **check the box that says "Add Python to PATH"** .

Step 2 Install Tesseract OCR (for handwriting module)

Download from: https://github.com/tesseract-ocr/tesseract/releases
Install it. The default path is: `C:\Program Files\Tesseract-OCR\tesseract.exe`
If you install it somewhere else, update `TESSERACT_PATH` in `config.py` .

Step 3 Install all Python libraries

Open a terminal (Command Prompt or PowerShell) inside the project folder and run:

```
pip install -r requirements.txt
```

This installs all AI models and libraries automatically.

Step 4 Run the application

```
python main.py
```

A window will open showing your webcam feed. Press number keys (1–7) to activate modules.

---

3. Keyboard Controls Quick Reference

Key	Action
1	Toggle Object Detection + Segmentation
2	Toggle Facial Expression Recognition
3	Toggle Hand Gesture Recognition
4	Toggle Handwritten Text OCR
5	Toggle Face Mesh Overlay
6	Toggle Gesture-Controlled Computer (auto-enables key 3)
7	Toggle Drowsiness / Attention Detection
R	Rotate camera (0° → 90° → 180° → 270° → back to 0°)
V	Toggle voice announcements ON / OFF
A	Enable ALL modules at once
N	Disable ALL modules at once
Q	Quit the application

> **Tip:** You can run multiple modules at the same time. For example, press 1, 2, and 5 together for object detection + emotions + face mesh all running simultaneously.

---

4. Camera Setup (Laptop or Phone)

Open `config.py` and find this setting near the top:

```
CAMERA_SOURCE = 0
```

Using your laptop webcam (default):

```
CAMERA_SOURCE = 0   # 0 = first webcam, 1 = second webcam if you have two
```

Using your Android phone as a webcam:

1. Install the **IP Webcam** app (free on Google Play Store)
2. Open the app → scroll to bottom → tap **"Start Server"**
3. The app will show an IP address like `192.168.1.5:8080`
4. Change `config.py` to:

```
CAMERA_SOURCE = "http://192.168.1.5:8080/shot.jpg"
#                                ↑ replace with YOUR phone's IP and port
```

> The `/shot.jpg` endpoint fetches the current frame with **zero delay** (no lag).

Using DroidCam (alternative phone app):

```
CAMERA_SOURCE = "http://192.168.1.5:4747/video"
```

Fixing orientation (phone held in portrait mode):

Press **R** on your keyboard to rotate the image until it looks correct.

- Press once → 90° clockwise
- Press again → 180°
- Press again → 270°
- Press 4th time → back to normal (0°)

The current rotation angle is always shown in the **top-right corner** of the window.

---

5. Configuration Variables (Tuning Guide)

All settings are in `config.py`. These are the most important variables to change:

Camera Settings

```
CAMERA_SOURCE = 0          # 0 = laptop webcam, or URL string for phone
FRAME_WIDTH = 1280         # Camera resolution width (pixels)
FRAME_HEIGHT = 720         # Camera resolution height (pixels)
```

Confidence Thresholds

These control how "sure" the AI must be before showing a result.

- **Lower value** = shows more detections (may include wrong ones)
- **Higher value** = only shows confident detections (may miss some)

```
YOLO_CONFIDENCE_THRESHOLD = 0.4   # Object detection (range: 0.0 to 1.0)
                                   # 0.3 = detects more objects (noisier)
                                   # 0.6 = only very clear objects

FACE_CONFIDENCE_THRESHOLD = 0.3   # Emotion detection base threshold
                                   # Actual threshold in module is max(this, 0.40)

HAND_CONFIDENCE_THRESHOLD = 0.5   # Hand detection (range: 0.0 to 1.0)
                                   # Lower = detects hands from further away

OCR_CONFIDENCE_THRESHOLD = 30     # Handwriting OCR (range: 0 to 100)
                                   # Higher = only shows high-confidence text
```

Display Colors (BGR format Blue, Green, Red)

```
COLOR_OBJECT_DETECTION = (0, 255, 0)     # Green boxes around objects
COLOR_FACE_EXPRESSION  = (255, 100, 50)  # Blue-ish boxes around faces
COLOR_HAND_GESTURE     = (0, 200, 255)   # Orange boxes around hands
COLOR_OCR_TEXT         = (255, 0, 255)   # Magenta for recognized text
COLOR_FACE_MESH        = (0, 255, 200)   # Cyan-green for face mesh dots
```

> **How BGR works:** (0, 255, 0) = pure green. (0, 0, 255) = pure red. (255, 0, 0) = pure blue. Mix values 0-255 to get any color.

Hand Tracking Smoothing (in `modules/hand_gesture.py`)

```
self._smooth_alpha = 0.12    # One Euro Filter alpha  NOT directly useful to change
```

Instead, change the filter parameters here (line ~62):

```
self._lm_filter = HandLandmarkFilter(min_cutoff=1.0, beta=0.007)
```

- **`min_cutoff`** : Higher = less jitter when still, but slightly more lag at start of movement. Try `0.5` to `2.0` .
- **`beta`** : Higher = faster response when moving fast. Try `0.001` to `0.05` .

Mouse Smoothing (in `modules/gesture_control.py`)

```
self._smoothing = 0.25   # Mouse cursor smoothing (0.0 to 1.0)
                          # 0.1 = very smooth but laggy
                          # 0.5 = very responsive but shaky
```

Drowsiness Thresholds (in `modules/drowsiness_detector.py`)

```
 EAR_THRESHOLD = 0.22    # Eye Aspect Ratio  lower = eyes must be more closed to trigger
                         # Try 0.18 for stricter, 0.25 for more sensitive
DROWSY_TIME = 2.0        # Seconds eyes must stay closed before alarm (default: 2.0)
MAR_THRESHOLD = 0.6      # Mouth Aspect Ratio for yawn detection
```

---

6. Module 1 Object Detection + Segmentation

Key: Press `1` to toggle

What it does:

- Detects **80 types of everyday objects**: person, car, phone, laptop, chair, cup, dog, cat, bicycle, etc.
- Draws a **colored bounding box** around each detected object
- Draws a **semi-transparent colored mask** (segmentation) showing the exact shape of the object (not just a rectangle)
- Shows the object name and confidence percentage (e.g., "person 87%")

What you will see:

Each detected object gets its own color. The colored fill shows exactly which pixels belong to the object. The box stays even when objects partially overlap.

AI Model Used:

- **YOLOv8n-seg** (You Only Look Once version 8, nano, segmentation variant)
- File: `yolov8n-seg.pt` (auto-downloaded, ~7MB)
- Trained on: **COCO dataset** (118,000 images, 80 object categories)
- Speed: ~30ms per frame on a modern CPU

Libraries:

- `ultralytics` runs the YOLO model
- `opencv-python` draws boxes and masks on the frame

Variables to tune for big difference:

Variable	Location	Effect
`YOLO_CONFIDENCE_THRESHOLD`	`config.py`	Lower → more objects detected
Model file	`object_detection.py` line ~18	Change `yolov8n-seg.pt` → `yolov8s-seg.pt` for more accuracy (larger model)
`conf=0.4`	`object_detection.py`	Detection sensitivity

What can be added:

- **Object counting**: Count how many people are in frame
- **Distance estimation**: Estimate how far away an object is
- **Custom trained model**: Train YOLO on your own objects (e.g., detect products, tools)
- **Object tracking**: Give each object an ID and track it across frames (add `tracker=True`)

- **Alert system**: Trigger an alarm when a specific object appears (e.g., weapon, phone in exam)

---

7. Module 2 Facial Expression Recognition

Key: Press `2` to toggle

What it does:

- Detects faces in the frame
- Classifies the emotion of each face into **8 categories**: Anger, Contempt, Disgust, Fear, Happiness, Neutral, Sadness, Surprise
- Shows the main emotion label with **top-3 scores** (e.g., `Happiness (hap:72% neu:18% sur:10%)`)
- Uses temporal smoothing over 5 frames to prevent flickering

What you will see:

A colored box around your face with the detected emotion. The top-3 breakdown tells you what the model is "thinking" even when it's uncertain.

AI Models Used:

Emotion CNN:

- **EfficientNet-B0** (2022) a modern efficient neural network architecture
- Model file: `enet_b0_8_best_afew.onnx` (auto-downloaded from GitHub)
- Trained on: **AffectNet dataset** (~450,000 real-world facial images)
- Accuracy: ~80% on in-the-wild video (vs. 65% for the old FER library)

Face Detector:

- **OpenCV DNN SSD** (ResNet-10 Single Shot Detector)
- Model file: `res10_300x300_ssd.caffemodel` (auto-downloaded, ~10MB)
- Why DNN instead of MTCNN: 3× faster, better on side angles

Libraries:

- `hsemotion-onnx` runs the EfficientNet emotion model via ONNX Runtime
- `onnxruntime` fast neural network inference (no GPU required)
- `opencv-python` face detection and drawing

Variables for big difference:

Variable	Location	Effect
`confidence_threshold`	`face_expression.py` line ~39	Raise to 0.55 for fewer but more accurate results
`smooth_window=5`	`face_expression.py` line ~42	Increase to 8 for smoother labels, decrease to 2 for faster response
`MIN_MARGIN = 0.12`	`face_expression.py` line ~35	Raise to 0.20 to only show label when model is very confident
`_detect_every_n = 2`	`face_expression.py`	Change to 1 for every frame (slower), 3 for lighter CPU

What can be added:

- **Emotion timeline graph**: Plot emotion over time on screen
- **Multi-face support**: Already supported tracks up to N faces simultaneously
- **Valence/Arousal prediction**: Change model to `enet_b0_8_va_mtl` for mood intensity scoring
- **Screenshot on smile**: Auto-save a photo when Happiness is detected above 80%
- **Student attention monitor**: Log timestamps when student looks bored/sad

---

8. Module 3 Hand Gesture Recognition

Key: Press `3` to toggle

What it does:

- Detects your hand and draws a **21-point skeleton** on it (fingers, joints, wrist)
- Recognizes **13+ gestures** and counts fingers (0–5)
- Smoothed with **One Euro Filter** skeleton is rock-solid even when hand is perfectly still

Recognized gestures:

Gesture	What it looks like
Open Palm	All 5 fingers spread open
Fist	All fingers closed
Thumbs Up	Thumb pointing up, others closed
Thumbs Down	Thumb pointing down, others closed
Peace / Victory	Index + middle finger up (V shape)
Pointing Up	Only index finger up
OK Sign	Thumb and index touching in circle
Rock / Metal	Index + pinky up, others closed
Call Me	Thumb + pinky up
I Love You	Thumb + index + pinky up
Pinch	Thumb and index very close together
Count 0–5	Number of extended fingers

What you will see:

A colored hand skeleton with lines connecting each joint. A label shows the detected gesture and finger count (e.g., `Thumbs Up | Fingers: 1`).

AI Model Used:

- **MediaPipe HandLandmarker** (Google, 2023)
- Model file: `hand_landmarker.task` (auto-downloaded, ~9MB)
- Architecture: BlazePalm (palm detection) + Hand Landmark CNN
- Output: 21 landmarks in (x, y, z) coordinates
- Speed: ~8ms per frame the fastest module in the system

Libraries:

- `mediapipe` Google's ML framework for hand tracking
- `opencv-python` drawing the skeleton
- `utils/one_euro_filter.py` adaptive smoothing filter

One Euro Filter explained (for tuning):

This is the smoothing algorithm that makes the skeleton stable. It's in `modules/hand_gesture.py` :

```
self._lm_filter = HandLandmarkFilter(min_cutoff=1.0, beta=0.007)
```

- **`min_cutoff`** : Jitter suppression when still. Higher = more stable. Default: `1.0`
- **`beta`** : Responsiveness when moving fast. Higher = less lag. Default: `0.007`

Good combos:

- Max stability: `min_cutoff=2.0, beta=0.003`
- Max responsiveness: `min_cutoff=0.5, beta=0.05`
- Balanced (default): `min_cutoff=1.0, beta=0.007`

What can be added:

- **Two-hand detection**: Model already supports it (currently only first hand is smoothed)
- **Custom gesture training**: Record your own hand poses and train a classifier

- **Sign language recognition**: Add a classifier on top of landmarks for ASL/BSL
- **Gesture to text typing**: Map gestures to keyboard keys

---

9. Module 4 Handwritten Text OCR

Key: Press `4` to toggle

What it does:

- Tracks your **index finger tip** in real time
- As you move your index finger with other fingers **closed**, it draws a line following your fingertip
- When you lift your hand away, it reads the drawn text using **OCR**
- Recognized text is shown on screen and announced via voice
- **Clear canvas**: Make a fist (all fingers closed) to erase and start again

How to use it step by step:

1. Press `4` to enable
2. Raise your **index finger only** (other fingers curled)
3. Move your finger in the air to draw letters
4. Lower your hand fully the text is recognized and displayed
5. Make a **fist** to clear the canvas

AI/Technology Used:

- **Tesseract LSTM OCR Engine** (Google, 2023 build)
- Tesseract is an open-source OCR engine using a trained LSTM (Long Short-Term Memory) neural network
- Language: English (can be changed to 100+ languages)

Libraries:

- `pytesseract` Python wrapper for Tesseract
- `opencv-python` canvas drawing and fingertip tracking
- `mediapipe` provides finger landmark positions
- `utils/one_euro_filter.py` smoothed finger tracking

Variables for big difference:

Variable	Location	Effect
`self._history_len = 20`	`handwriting_ocr.py` line ~70	Reduce to 10 for faster cursor response, increase to 30 for more smoothing
`self._min_move_px = 8`	`handwriting_ocr.py`	Dead zone increase to draw thicker strokes, decrease for finer control
`OCR_CONFIDENCE_THRESHOLD = 30`	`config.py`	Raise to 60 for only very clear text recognition

What can be added:

- **Multi-language OCR**: Change `lang='eng'` to `lang='ara'` for Arabic, `lang='urd'` for Urdu, etc.
- **Mathematical formula recognition**: Use MathPix or pix2tex model
- **Save drawing**: Auto-save the canvas image when recognized
- **Color switching**: Different finger gestures change the drawing color

---

10. Module 5 Face Mesh Overlay

Key: Press `5` to toggle

What it does:

- Detects your face and maps **468 individual landmark points** across it
- These points cover eyes, eyebrows, nose, lips, cheeks, jaw, and forehead
- Points are connected with lines to show the 3D structure of the face
- The mesh updates in real time as you move your head in any direction

What you will see:

A glowing cyan-green mesh stretched across your face like a 3D wireframe model. Every subtle expression changes the positions of these 468 points.

Why it's useful:

The face mesh provides the **foundation** for the Drowsiness module (Module 7). The eye and mouth landmark positions are used to calculate EAR (Eye Aspect Ratio) and MAR (Mouth Aspect Ratio). It can also be the basis for lip-sync, AR face filters, or 3D avatar control.

AI Model Used:

- **MediaPipe FaceMesh** (Google, 2023)
- Architecture: BlazeFace (fast face detector) + FaceMesh CNN
- Landmarks: 468 (x, y, z) normalized coordinates
- Supports multi-face detection

Libraries:

- `mediapipe` face mesh model
- `opencv-python` drawing the mesh points and connections

Variables for big difference:

Variable	Location	Effect
`COLOR_FACE_MESH`	`config.py`	Change color of the mesh dots
Max faces	`face_mesh.py` ~line 40	`max_num_faces=1` → change to 4 for multi-person
Confidence	`face_mesh.py` ~line 42	Lower → detects faces at more angles

What can be added:

- **AR face filters**: Place virtual glasses, hat, or mask using landmarks
- **Head pose estimation**: Calculate head pitch/yaw/roll angles
- **Blink counter**: Count blinks using eye aspect ratio
- **Lip sync**: Map lip landmarks to animate a 3D avatar

---

11. Module 6 Gesture-Controlled Computer

Key: Press `6` to toggle (automatically also enables Module 3 hand gestures)

What it does:

Controls your real computer using only hand gestures visible to the camera. No touching the keyboard or mouse needed.

Full gesture command map:

Gesture	What to do	System Action
☝ **Pointing (index finger up)**	Point index finger, move hand	**Moves the mouse cursor**
▯ **Pinch**	Bring thumb + index tip very close together	**Left click**
▯ **Open Palm**	Open all 5 fingers and move hand up	**Scroll up**
▯ **Open Palm**	Open all 5 fingers and move hand down	**Scroll down**
▯ **Thumbs Up**	Thumb up, others closed	**Play / Pause media**
▯ **Fist**	Close all fingers (hold 1 second)	**Mute / Unmute**
✌ **Peace Sign**	Index + middle up (hold 2 seconds)	**Screenshot** (saved to `screenshots/` folder)

How to use the virtual mouse:

1. Press `6` to enable (hand gesture also turns on automatically)
2. Hold up only your **index finger** you'll see a **"MOUSE"** label and crosshair on screen
3. Move your index finger the **real mouse cursor** follows your finger
4. To **click**: bring your **thumb** very close to your **index fingertip** (pinch together)
5. To **stop** mouse mode: open your full palm or make a different gesture

Technologies Used:

- `pyautogui` moves the real mouse, sends keyboard keys, takes screenshots
- `pycaw` Windows volume control (if available)
- `mediapipe` hand landmark positions
- `utils/one_euro_filter.py` smooth cursor movement

Mouse smoothing variables (in `gesture_control.py`):

```
self._smoothing = 0.25    # 0.1 = very smooth/slow, 0.5 = fast/shaky
```

Dead zone (in `_move_mouse` method):

```
if dx < 5 and dy < 5: return    # Change 5 to larger number to need bigger movements
```

What can be added:

- **Right click**: Middle finger + thumb pinch gesture
- **Drag and drop**: Pinch and hold while moving
- **Zoom in/out**: Two-hand pinch/expand gesture
- **Custom key bindings**: Map any gesture to any keyboard shortcut

---

12. Module 7 Drowsiness & Attention Detection

Key: Press `7` to toggle (works best when Module 5 Face Mesh is also enabled)

What it does:

- Monitors your **eyes** and **mouth** continuously using face mesh landmarks
- Calculates **EAR** (Eye Aspect Ratio) a number that drops when eyes close
- Calculates **MAR** (Mouth Aspect Ratio) a number that rises when you yawn
- Tracks an **Attention Level %** gauge (0% = asleep, 100% = fully alert)
- Triggers a **flashing red alert** and **voice warning** if drowsiness is detected for 2+ seconds

What you will see:

- A green attention bar showing your alertness (like a battery indicator)
- EAR and MAR values displayed in real time
- If you close your eyes for 2+ seconds: the screen flashes red and you hear "Wake up! Drowsiness detected"
- Yawn detected: yellow text on screen

How it works (the science):

Eye Aspect Ratio (EAR):

```
EAR = (vertical eye distance 1 + vertical eye distance 2) / (2 × horizontal eye distance)
```

When the eye is open, EAR ≈ 0.30. When fully closed, EAR ≈ 0.0. Threshold is 0.22.

Mouth Aspect Ratio (MAR): Similar calculation for the mouth. Large open mouth = yawn = MAR > 0.6.

AI Technology Used:

- **MediaPipe FaceMesh** landmarks provides the 468 3D face points
- The EAR/MAR calculations are done with pure math on those landmarks (no separate model needed)

Libraries:

- `mediapipe` face mesh landmarks
- `pyttsx3` offline text-to-speech voice warning
- `opencv-python` drawing the gauges and alert overlay

Variables for big difference:

In `modules/drowsiness_detector.py` :

```
 EAR_THRESHOLD = 0.22    # Lower = eyes must be more closed to trigger (less sensitive)
                          # Higher = triggers even with slightly droopy eyes (more sensitive)
                          # Recommended range: 0.18 to 0.28

DROWSY_TIME = 2.0         # Seconds of closed eyes before alert fires
                          # Reduce to 1.0 for faster alert, increase to 3.0 for more tolerance

MAR_THRESHOLD = 0.6       # Mouth openness for yawn. Lower = detects smaller yawns
```

What can be added:

- **Sound alarm**: Play an audio file instead of/in addition to voice
- **Email/SMS alert**: Notify someone when drowsiness is repeatedly detected
- **Log events**: Save timestamps of all drowsiness events to a file
- **Head nodding detection**: Also detect head dropping forward using head pose estimation

---

13. AI Models & Libraries Reference

AI Models Summary

Module	Model Name	Architecture	Training Data	Accuracy	Size
Object Detection	YOLOv8n-seg	CSP-DarkNet + FPN	COCO 118k images	mAP 36.8	7MB
Face Expression	enet_b0_8_best_afew	EfficientNet-B0	AffectNet 450k	~80%	Auto
Face Detection	res10_300x300_ssd	ResNet-10 SSD	WIDERFACE	~90%	10MB
Hand Gesture	hand_landmarker.task	BlazePalm + Landmark CNN	Internal Google	~95%	9MB
Face Mesh	FaceMesh (MediaPipe)	BlazeFace + 468-pt CNN	Internal Google	Sub-cm	3MB
Handwriting OCR	Tesseract LSTM	LSTM RNN	Various scripts	~85% (print)	~30MB

Libraries Installed

Library	What it does	Version
`opencv-python`	Video capture, image drawing, DNN face detection	≥4.8
`mediapipe`	Hand tracking, face mesh (Google's ML kit)	≥0.10
`ultralytics`	Runs YOLOv8 object detection/segmentation	≥8.0
`hsemotion-onnx`	EfficientNet-B0 emotion recognition via ONNX	≥0.3
`pytesseract`	Python wrapper for Tesseract OCR	≥0.3
`pyttsx3`	Offline text-to-speech (works without internet)	≥2.90
`pyautogui`	Controls real mouse cursor and keyboard	≥0.9
`pycaw`	Windows system volume control	≥2024
`comtypes`	Windows COM interface (required by pycaw)	≥1.4
`numpy`	Fast numerical array operations	≥1.24

Library	What it does	Version
`Pillow`	Image processing	≥10.0
`torch` / `torchvision`	PyTorch deep learning (used by YOLO internally)	≥2.0
`tensorflow`	Used internally by some models	≥2.15

---

14. Project File Structure

```
ARIA/
│
├── main.py                    ← Entry point  run this to start the app
├── config.py                  ← All settings  edit this to tune the system
├── requirements.txt           ← All Python libraries needed
│
├── modules/                   ← One file per AI module
│   ├── object_detection.py     ← Module 1: YOLOv8 detection + segmentation
│   ├── face_expression.py      ← Module 2: EfficientNet emotion recognition
│   ├── hand_gesture.py         ← Module 3: MediaPipe hand tracking + gestures
│   ├── handwriting_ocr.py      ← Module 4: Air-writing + Tesseract OCR
│   ├── face_mesh.py            ← Module 5: MediaPipe 468-point face mesh
│   ├── gesture_control.py      ← Module 6: Virtual mouse + system control
│   ├── drowsiness_detector.py  ← Module 7: EAR/MAR drowsiness alerts
│   ├── voice_feedback.py       ← Text-to-speech announcements
│   ├── scene_describer.py      ← Describes what is visible in the scene
│   ├── activity_recognition.py ← Detects body activities
│   └── relationship_analyzer.py← Analyzes spatial relationships between objects
│
├── utils/                     ← Helper utilities
│   ├── camera_reader.py        ← Threaded camera reader (zero lag for IP cameras)
│   ├── one_euro_filter.py      ← Adaptive smoothing filter for landmarks
│   ├── ai_dashboard.py         ← On-screen AI status dashboard
│   ├── drawing.py              ← Drawing helpers (bounding boxes, panels)
│   ├── performance.py          ← FPS counter and latency tracker
│   ├── object_tracker.py       ← Object ID tracking across frames
│   └── ...
│
├── models/                    ← Downloaded model files (auto-created)
│   ├── hand_landmarker.task    ← MediaPipe hand model
│   ├── deploy.prototxt         ← DNN face detector config
│   └── res10_300x300_ssd.caffemodel  ← DNN face detector weights
│
├── screenshots/               ← Auto-saved screenshots (from peace gesture)
├── yolov8n-seg.pt             ← YOLOv8 segmentation model weights
└── yolov8n.pt                 ← YOLOv8 detection model weights (fallback)
```

---

15. Common Errors & Fixes

"Could not open camera!"

- Make sure no other program is using the webcam (close Zoom, Teams, etc.)
- Try changing `CAMERA_SOURCE = 1` instead of `0`

"Tesseract not found" / OCR doesn't work

- Install Tesseract from the link in Section 2
- Update `TESSERACT_PATH` in `config.py` to match where you installed it

High latency with phone camera

- Make sure your phone and laptop are on the **same WiFi network**
- Use the `/shot.jpg` snapshot URL format (not `/video`)
- Move closer to the WiFi router

Emotion detection shows nothing

- Make sure your face is well-lit (face a window or lamp)
- Look straight at the camera extreme side angles reduce accuracy
- The module needs ~0.5 seconds to warm up when first activated

Hand skeleton shakes/wobbles

- This is rare with the One Euro Filter but can happen in very low light
- Make sure the camera FPS is at least 15 (check the FPS counter top-left)
- Try increasing `min_cutoff` from `1.0` to `1.5` in `hand_gesture.py`

Voice announcements not working

- Make sure your system sound is not muted
- `pyttsx3` uses Windows SAPI5 this is built into Windows, no install needed
- Try pressing **V** to toggle voice off and on again

"Module not found" errors after install

- Run `pip install -r requirements.txt` again
- Make sure you are in the correct project folder when running

---

16. What Can Be Added Next

Here are the most impactful features that could be added to each module:

Feature	Difficulty	Impact
Age & Gender estimation on detected faces	Easy (add DeepFace)	High
Custom object detection (train on your own images)	Medium	Very High
Two-hand gesture control (currently only one hand is smoothed)	Easy	High
Sign language to text (classify hand poses into alphabet)	Medium	Very High
Multi-language OCR (Urdu, Arabic, Chinese)	Easy (change Tesseract lang)	High
AR face filters (virtual glasses, hat using face mesh)	Medium	High
Attention/engagement analytics for classrooms	Medium	Very High
Phone notification when drowsiness detected (via Telegram Bot)	Medium	High
Real-time scene narration using GPT/LLM API	Easy (add API call)	Very High
Posture detection (add MediaPipe Pose for full body)	Easy	High
Gaze direction (where are you looking on screen)	Hard	High
Emotion-controlled music (play calm music when stress detected)	Medium	Fun

---

Documentation written for ARIA Adaptive Real-time Intelligence Assistant System uses ~92% AI-powered components across 7 real-time modules