Visual Perception for Aria Glasses

Neil Samant, Arshia Garg

WINLAB Summer Internship **2025**

Advised by Prof. **Yao Liu**

TABLE OF CONTENTS

01

Introduction

THE TEAM

Neil Samant

Arshia Garg

About Project Aria

Meta Research

Developed a cutting-edge platform designed to advance AI research in areas like computer vision.

Abilities

Stream data from multiple sensors, enabling real-time insight into human attention and behavior.

Software Development Kit

Provides APIs to help you create machine perception Python applications with Project Aria glasses.

02

Objectives

The Problem

- An issue most people in tech face that largely goes **unnoticed:**
    - Spending **too much time** in front of devices, **not taking enough breaks.**
    - 20-20-20 rule: every **20** minutes, look at an object **20** feet away for **20** seconds
    - **Eye strain** and **fatigue**

- The other side of it:
    - **Getting too distracted** while completing important tasks (studying, working)
    - Especially prevalent with reels, TikTok, etc.

10-Week Objectives (1/2):

- Create an efficient visual perception system with Aria Glasses
- Develop an **early version** of a **smart study/work “assistant”**
    - Monitors eye behavior with the aim of **reducing strain**, **encouraging breaks**, and **detecting distractions**

10-Week Objectives (2/2):

- Maintain **high accuracy** and close to real-time system **for several minutes**
- Run all inferences on the ORBIT testbed server to reduce the load on the AR glasses and speed up processing.

03

Testing and Development

Early Stages

4. Create an StreamingClient observer and attach it

Find more description of observer in the streaming code sample

```
class StreamingClientObserver:
    def __init__(self):
        self.images = {}

    def on_image_received(self, image: np.array, record: ImageDataRecord):
        self.images[record.camera_id] = image

observer = StreamingClientObserver()
streaming_client.set_streaming_client_observer(observer)
```

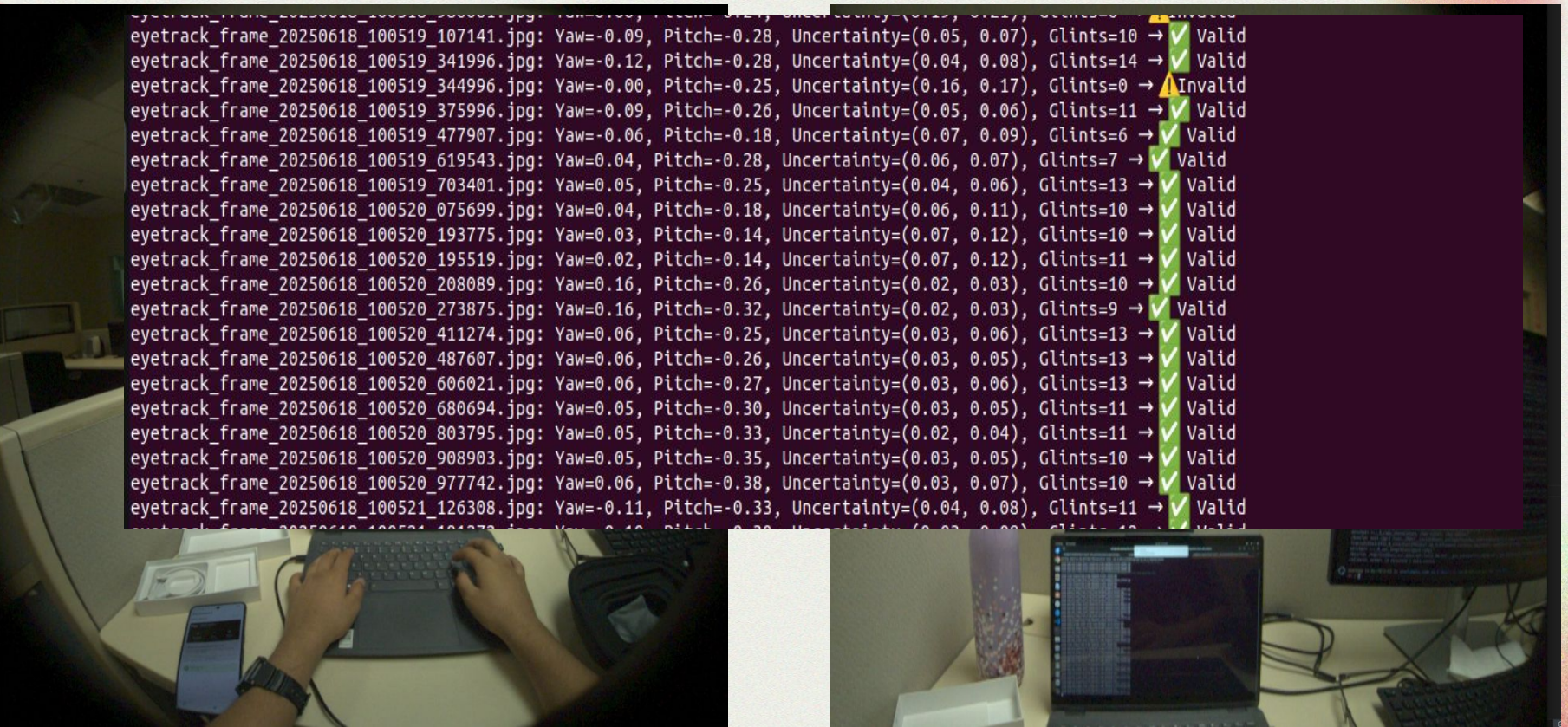
5. Start subscribing and listen to the live stream

The client begins listening for incoming streaming data from the subscribed data types.

```
streaming_client.subscribe()
```

```
class StreamingClientObserver:
    def __init__(self):
        self.images = {}
        self.last_saved_time = {
            aria.CameraId.EyeTrack: 0,
            aria.CameraId.Rgb: 0,
        }

    def on_image_received(self, image: np.array, record: ImageDataRecord):
        now = time.time()

        # Only save if at least 0.5s has passed since last saved frame
        if now - self.last_saved_time[record.camera_id] < 0.3:
            return
        self.last_saved_time[record.camera_id] = now

        self.images[record.camera_id] = image
        timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S_%f")

        if record.camera_id == aria.CameraId.EyeTrack:
            eyetrack_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            filename = f"eyetrack_frame_{timestamp}.jpg"
            path = os.path.join("eyetrack_frames_final_2", filename)
            cv2.imwrite(path, eyetrack_rgb)
            print(f"Saved {path}")

        elif record.camera_id == aria.CameraId.Rgb:
            image_landscape_rgb = cv2.cvtColor(np.rot90(image, -1), cv2.COLOR_BGR2RGB)
            filename = f"rgb_frame_{timestamp}.jpg"
            path = os.path.join("output_frames_final_2", filename)
            cv2.imwrite(path, image_landscape_rgb)
            print(f"Saved {path}")
```

Eye Gaze Movements in Real Time



Mapping Eye Gaze to Object

- Used the **YOLO** v8 model to get **bounding boxes** of each object
- Yaw and pitch values were converted to the pixel in the RGB image

based on which bounding box the gaze pixel was in

```
2025-07-30T11:09:04.124718,rgb_frame_20250730_110902_715750.jpg,laptop,0.907
2025-07-30T11:09:04.423215,rgb_frame_20250730_110903_041492.jpg,laptop,0.916
2025-07-30T11:09:04.687495,rgb_frame_20250730_110903_377298.jpg,laptop,0.929
2025-07-30T11:09:04.947544,rgb_frame_20250730_110903_699408.jpg,laptop,0.898
2025-07-30T11:09:05.207250,rgb_frame_20250730_110904_026695.jpg,laptop,0.914
2025-07-30T11:09:05.500970,rgb_frame_20250730_110904_429831.jpg,laptop,0.897
2025-07-30T11:09:05.844808,rgb_frame_20250730_110904_789009.jpg,laptop,0.898
2025-07-30T11:09:06.136852,rgb_frame_20250730_110905_229514.jpg,laptop,0.938
2025-07-30T11:09:06.391250,rgb_frame_20250730_110905_803816.jpg,laptop,0.906
2025-07-30T11:09:06.713172,rgb_frame_20250730_110906_120198.jpg,laptop,0.916
2025-07-30T11:09:06.996900,rgb_frame_20250730_110906_120198.jpg,laptop,0.916
2025-07-30T11:09:07.287428,rgb_frame_20250730_110906_674900.jpg,laptop,0.919
2025-07-30T11:09:07.701190,rgb_frame_20250730_110906_989120.jpg,laptop,0.933
2025-07-30T11:09:08.832558,rgb_frame_20250730_110907_377750.jpg,,
2025-07-30T11:09:09.408139,rgb_frame_20250730_110907_814754.jpg,,
2025-07-30T11:09:09.825576,rgb_frame_20250730_110907_814754.jpg,laptop,0.916
2025-07-30T11:09:10.237040,rgb_frame_20250730_110907_814754.jpg,laptop,0.916
2025-07-30T11:09:11.091396,rgb_frame_20250730_110908_784410.jpg,laptop,0.932
2025-07-30T11:09:12.080394,rgb_frame_20250730_110909_402388.jpg,laptop,0.943
2025-07-30T11:09:13.048581,rgb_frame_20250730_110909_788549.jpg,laptop,0.921
2025-07-30T11:09:13.665414,rgb_frame_20250730_110909_788549.jpg,laptop,0.921
2025-07-30T11:09:14.090903,rgb_frame_20250730_110910_372149.jpg,laptop,0.917
2025-07-30T11:09:14.537929,rgb_frame_20250730_110910_735405.jpg,tv,0.424
2025-07-30T11:09:15.115270,rgb_frame_20250730_110911_077871.jpg,tv,0.512
2025-07-30T11:09:15.698573,rgb_frame_20250730_110911_509359.jpg,laptop,0.915
2025-07-30T11:09:16.746571,rgb_frame_20250730_110912_067903.jpg,laptop,0.902
2025-07-30T11:09:17.124155,rgb_frame_20250730_110912_067903.jpg,laptop,0.902
2025-07-30T11:09:17.872795,rgb_frame_20250730_110912_067903.jpg,laptop,0.902
2025-07-30T11:09:18.377561,rgb_frame_20250730_110913_583381.jpg,laptop,0.925
```

```
0: 640x640 3 persons, 1 bottle, 1 chair, 2 laptops, 1 mouse, 1 keyboard, 2 cell phones, 5.1ms
Speed: 3.0ms preprocess, 5.1ms inference, 1.3ms postprocess per image at shape (1, 3, 640, 640)
{'label': 'laptop', 'confidence': 0.94}
INFO:     127.0.0.1:33206 - "POST /analyze HTTP/1.1" 200 OK
temp_eyetrack_frame_20250730_110929_921176.jpg: Yaw=0.11, Pitch=-0.20, Yaw from=0.09 to 0.13, Pitch from=-0.23 to -0.15Glints=11 → ✅ Valid

0: 640x640 3 persons, 1 bottle, 1 tv, 2 laptops, 1 keyboard, 2 cell phones, 5.0ms
Speed: 3.0ms preprocess, 5.0ms inference, 1.3ms postprocess per image at shape (1, 3, 640, 640)
{'label': 'laptop', 'confidence': 0.944}
INFO:     127.0.0.1:33208 - "POST /analyze HTTP/1.1" 200 OK
temp_eyetrack_frame_20250730_110930_272555.jpg: Yaw=0.17, Pitch=-0.31, Yaw from=0.15 to 0.20, Pitch from=-0.33 to -0.28Glints=14 → ✅ Valid

0: 640x640 3 persons, 1 bottle, 1 tv, 2 laptops, 1 keyboard, 2 cell phones, 5.0ms
Speed: 3.0ms preprocess, 5.0ms inference, 1.3ms postprocess per image at shape (1, 3, 640, 640)
{'label': 'laptop', 'confidence': 0.944}
INFO:     127.0.0.1:33210 - "POST /analyze HTTP/1.1" 200 OK
temp_eyetrack_frame_20250730_110930_583671.jpg: Yaw=0.17, Pitch=-0.31, Yaw from=0.15 to 0.20, Pitch from=-0.33 to -0.28Glints=12 → ✅ Valid

0: 640x640 3 persons, 1 bottle, 1 tv, 2 laptops, 1 keyboard, 2 cell phones, 5.0ms
Speed: 3.0ms preprocess, 5.0ms inference, 1.3ms postprocess per image at shape (1, 3, 640, 640)
{'label': 'laptop', 'confidence': 0.944}
INFO:     127.0.0.1:33216 - "POST /analyze HTTP/1.1" 200 OK
temp_eyetrack_frame_20250730_110930_896808.jpg: Yaw=0.17, Pitch=-0.31, Yaw from=0.15 to 0.20, Pitch from=-0.33 to -0.28Glints=13 → ✅ Valid

0: 640x640 3 persons, 1 bottle, 1 tv, 2 laptops, 1 keyboard, 2 cell phones, 5.0ms
Speed: 3.0ms preprocess, 5.0ms inference, 1.3ms postprocess per image at shape (1, 3, 640, 640)
{'label': 'laptop', 'confidence': 0.944}
INFO:     127.0.0.1:33224 - "POST /analyze HTTP/1.1" 200 OK
temp_eyetrack_frame_20250730_110931_306229.jpg: Yaw=0.23, Pitch=-0.28, Yaw from=0.22 to 0.26, Pitch from=-0.30 to -0.26Glints=13 → ✅ Valid
```

Moving Workflow to ORBIT

- The entire pipeline was run on a local machine
- **Needed better speeds** - CPU was not good enough
- Solution: **certain ORBIT nodes had NVIDIA GPUs**
- Have both the **Eye Gaze model** and **YOLO** run on a server running on the node

```python
def call_analyze_route(eye_path, rgb_path):
    try:
        with open(eye_path, "rb") as f1, open(rgb_path, "rb") as f2:
            response = requests.post(
                "http://localhost:8080/analyze",
                files={"eyetrack": f1, "rgb": f2}
            )
            response.raise_for_status()
            return response.json()
    except Exception as e:
        print(f"Analyze API call failed: {e}")
        return None
```

Domains with CUDA Capabilities

Domain	Node: CUDA Card(s)
sb4.orbit-lab.org	node2-1: GeForce GT 640 GDDR5
sb9.orbit-lab.org	node1-1 to node1-8: Tesla V100
grid.orbit-lab.org	node21-1 to node21-7: Tesla K80 node21-9 to node21-32: Tesla P100
⇨COSMOS	see COSMOS testbed for details

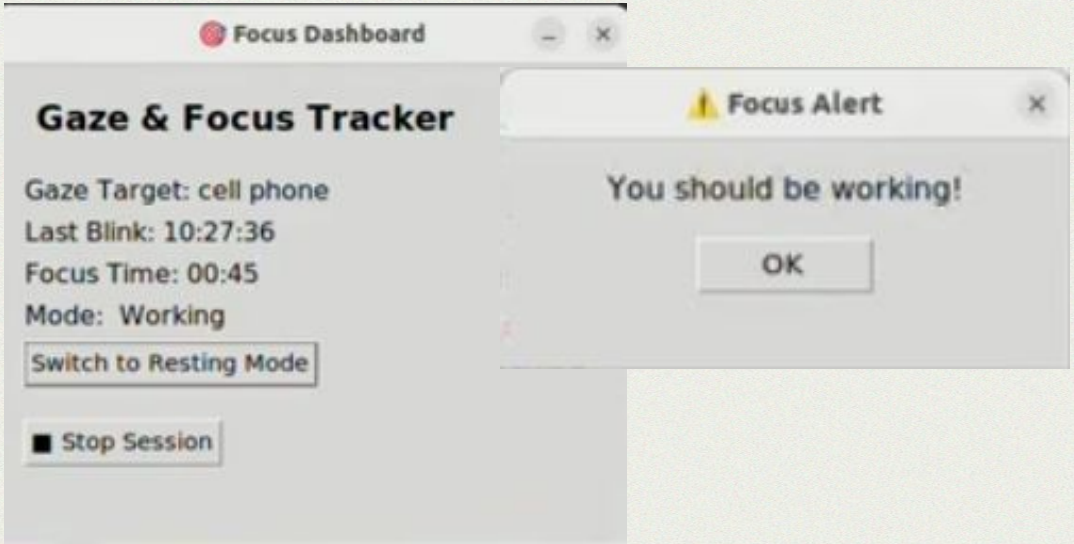
Dashboard and Session Setup (1/2)

- **Working Mode**:
  - Scaled 20/20/20 rule to **15 seconds** of continuously looking at laptop
  - Displays popup to take a break

- **Resting/Break Mode:**
  - If user is taking a break - should not be looking at their laptop (except to switch modes)
  - Popup after **2 seconds** to alert user

Dashboard and Session Setup (2/2)

- **Phone Distraction**:
  - If user is looking at phone while working/studying: **could indicate distractions.**



04

Workflow

Dashboard logs object and shows alerts

Images streamed through Wi-Fi and saved on disk

RGB and ET frames sent through SSH tunneling

ET model gets yaw and gaze estimate -> converts to pixel

YOLO returns object bounding boxes

Object returned based on which bounding box the pixel is in

Technologies Used

Python

For **PyTorch**, **Tkinter**, **OpenCV**, and interactions with **ORBIT**, Project Aria **SDK**

Project Aria ET Model

A **PyTorch**-based model developed by **Meta Reality Labs Research** to generate eye gaze predictions.

YOLO v8

Deep Learning model from **Ultralytics** that detects objects in each RGB frame

Google Pixel 8 Pro

Phone to connect to the glasses via **Aria Companion App**

05

Results

KEY NUMBERS

Average Accuracy

95%

(detecting the exact point at which a user is looking at, and mapping that to the correct object)

Average Confidence

93%

YOLO's confidence in the object it is detecting

Pipeline Latency

(per image)

9-15 ms

Time taken from capturing eye + RGB frames to producing a result (object label and overlay)

Alert Response Time

300 ms

Time between detecting prolonged focus and showing a popup warning

06

Future Plans

Future Work

Use Meta Aria Gen 2 Glasses

Increased battery use, more sensors, features such as **microphone** and **speaker**

Advanced Applications

Adaptive gaze-based UIs, posture correction alerts using **IMU** and **SLAM** data

Dashboard Upgrades

Personalized **visual heatmaps**, detailed post-session **focus** and **strain analytics**

Questions?

Check out our website:
https://winlab2025-ariaglasses.vercel.app

Website QR Code: