Synthesis: A Multimodal Pipeline for Automated Egocentric Dataset Creation and Annotation

Matteo Breganni - 869549

1 Introduction

Machine Learning has advanced rapidly thanks to large, annotated datasets. However, creating such data, especially for egocentric vision, is time consuming and costly, limiting progress in applications such as assistive technologies and activity recognition. This thesis presents a multimodal pipeline using Meta's Project Aria glasses that enables automated, hands-free annotation by combining audio, visual, eye gaze, and motion data. Users label objects simply by looking and speaking, leveraging natural behavior to reduce annotation effort. The pipeline integrates WhisperX for speech transcription, YOLO-World for open-vocabulary detection, and SAM2 for pixel-level segmentations. A user-friendly interface supports parameter tuning, batch processing, and visualization, making the tool accessible and efficient.

Evaluated using IoU against ground truth, the system shows strong accuracy, with error and computational analysis revealing performance bottlenecks. Results demonstrate its efficiency and suitability for real-world deployment. This work contributes both a practical tool for rapid dataset creation and a methodological exploration of modern vision models in egocentric, multimodal settings, paving the way for next-generation assistive technologies and intelligent activity recognition systems.

2 State of the Art Components

2.1 Meta's Project Aria

Project Aria (Fig 1) is a research platform for egocentric AI, centered on sensor-rich glasses that captures synchronized multimodal data into a VRS file. The device includes an RGB camera, two side cameras, dual eye-tracking sensors, a 7-microphone array, IMUs, GNSS, and environmental sensors. Due to power constraints, no on-device processing is allowed. Meta provides Project Aria Tools, including cloud APIs, open models, datasets, and a Python library for streamlined egocentric AI research.

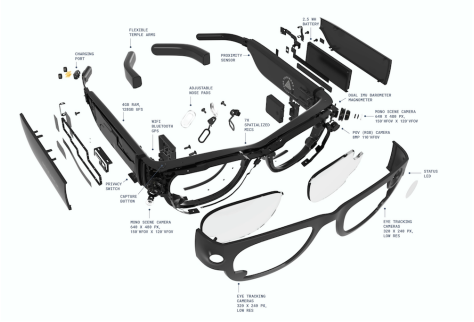

Figure 1: Project Aria Glasses

2.2 Whisper and WhisperX

Whisper is a speech recognition model trained on 680,000 hours of weakly supervised web-scraped audio-transcript pairs. It requires no fine-tuning and supports a wide range of tasks, including multilingual transcription and translation.

WhisperX improves Whisper for long-form audio by using VAD to segment speech at natural pauses, then processing segments in batches for speed. It applies phoneme-level alignment to generate precise word-level timestamps without retraining the model. This way it achieves up to 12× faster inference.

2.3 YOLO-World

YOLO-World is a real-time open-vocabulary detector based on YOLOv8 that uses natural language to detect objects without fixed categories. It combines a CLIP text encoder with a RepVL-PAN module to fuse text and visual features during training, enabling language-guided detection. At inference, text embeddings are re-parameterized into convolutional weights, removing the need for real-time language encoding and preserving YOLO's speed. This makes it fast, flexible, and ideal for dynamic detection tasks.

2.4 SAM and SAM2

The Segment Anything Model (SAM) introduces a promptable framework for image segmentation, generating masks from points or boxes. SAM's modular design supports real-time interaction and outputs multiple masks to handle ambiguity. SAM2 extends SAM to video with temporal memory for coherent spatio-temporal segmentation, maintaining a memory bank of past objects and prompts. It works on both video and images, delivering faster, more accurate, and temporally consistent results than SAM.

3 Pipeline Design and Implementation

3.1 Recording Procedure and UI Tool

The dataset is created by having users wear the Project Aria Glasses to record sessions where they naturally interact with their environment, looking at objects and naming them in everyday speech. The system supports conversational, unscripted speech in any language.

Due to battery and thermal limits, long recordings are not feasible. Instead, the pipeline processes multiple shorter sessions in batch and combines them into a unified dataset, ensuring scalability.

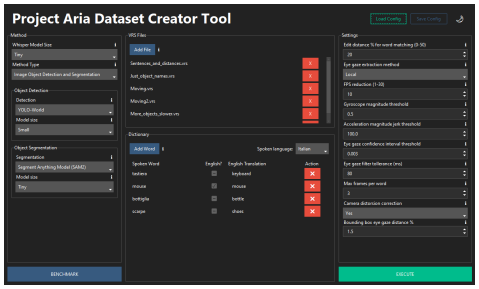

Figure 2: UI Tool's Main Page

A dedicated UI tool (Fig 2) was developed, allowing users to configure, load and save pipeline settings for reproducible, batch processing of recordings. The interface allows users to choose processing modes, model sizes, and parameters, including a dictionary system to support multilingual recordings and category labels.

3.2 Proposed Pipeline

The **speech processing** stage uses WhisperX to transcribe the VRS audio into word-level segments with timestamps and confidence scores. To support multilingual input, keywords are matched against the original-language dictionary entries. A configurable length-normalized edit distance threshold improves matching robustness, allowing minor model mistakes and grammatical variations.

The **frame extraction and preprocessing** stage begins with applying devignetting masks to correct brightness falloff in the front RGB camera. For each matched keyword, multiple frames are captured within

the speech interval to enhance visual robustness. Frames are extracted at a reduced frame rate determined by the user-defined setting, to reduce the computational load on the following stages. Optionally, camera distortion correction can be applied to remove the "fisheye" effect, using calibration data generated from using checkerboard patterns, which estimates intrinsic parameters and distortion coefficients.

The **eye gaze data extraction** stage processes images from the glasses' grayscale eye cameras, choosing to use either a local open model or Meta's Machine Perception Services API. The local model offers near real-time performance but lacks depth estimation and calibration, while MPS provides depth via gaze ray triangulation and support for in-session calibration at the cost of significantly longer processing times. The pipeline uses yaw and pitch angles from the selected model, along with uncertainty bounds, to represent gaze direction, and projects these onto the front camera frame. If lens distortion correction is applied, gaze coordinates are transformed using the same calibration data to maintain spatial alignment.

The **frames filtering and selection** stage improves data quality by removing low-quality frames. Using IMU data, motion-based filtering computes gyroscope and smoothed jerk magnitudes to detect and exclude frames with excessive or abrupt motion. Separately, eye gaze filtering removes frames with unreliable gaze estimates by analyzing the confidence interval widths of yaw and pitch angles, excluding those that exceed a user-defined threshold. After filtering, a user-defined number of high-quality frames per keyword are selected, evenly spaced across the speech interval to maximize visual diversity and robustness.

The **object detection** stage uses YOLO-World to detect and localize objects corresponding to user-defined keywords. For each keyword given as prompt, selected frames are processed to generate bounding boxes and confidence scores, which are then refined by merging overlapping detections or those aligned with eye gaze to reduce redundancy. In the final selection, only bounding boxes spatially consistent with gaze (either inside or within a configurable distance threshold) are considered, and the highest confidence box is chosen, prioritizing gaze alignment when segmentation is requested by the user.

The **object segmentation** stage generates pixel-accurate masks using either SAM or the recommended SAM2. Segmentation can be performed in three modes: standalone gaze-driven, detection-guided, or video-based. The gaze-only approach uses eye gaze as a point prompt but suffers from inaccuracies due to gaze misalignment and ambiguity in mask selection, as SAM/SAM2 outputs multiple candidates (whole, part, subpart), with the highest-confidence mask chosen automatically. Detection-guided segmentation, which uses the bounding box from the prior detection stage as a spatial prompt and conditionally incorporates gaze (if it's within the bounding box), significantly improves accuracy by eliminating the ambiguity issue. Video object segmentation, supported only by SAM2, enables temporal tracking but was found to offer no meaningful improvement over frame-by-frame processing due to gaze-driven errors, prompt overwriting behavior, and high computational cost, leading to its deprioritization. Ultimately, detection-guided image segmentation is recommended for producing high-quality, reliable annotations efficiently.

3.3 Results Interface and Dataset Output

The results interface provides real-time feedback and tools to debug the pipeline after execution. The right panel displays progress and timing metrics, while the left offers visualization buttons to explore data like extracted frames, filtering outcomes, audio transcriptions, and detection or segmentation results. Finally, users can export results in two formats: (1) a structured folder-based output with detections split by category and saved as cropped bounding box images, or segmentation masks isolated on transparent

backgrounds; and (2) a full-frame export with COCO-style JSON annotations that include bounding boxes, segmentation masks, and associated metadata.

4 Evaluation and Benchmarking System

The benchmarking process begins by selecting a reference dataset that can be created through a Python notebook as a subset of the COCO dataset. Users verbally annotate images displayed on a monitor while recording with Project Aria Glasses. During annotation, the system shows one image at a time with the target category label, prompting the user to look at the object, speak its name, and advance to the next image. This method presents challenges in temporal synchronization and spatial correspondence due to lens distortion and perspective differences between the displayed image and the recorded frame.

The synchronization system aligns the on-screen benchmark with the Aria recording using a 1-second green flash. A local timestamp is recorded when the flash appears, and the pipeline detects it in the video by identifying a peak in green channel change relative to red and blue, accounting for auto-exposure. This is a simple, yet robust method to align the image that is shown with what is being recorded.

To make the recording's frames and the dataset's images comparable, the recording goes through a pipeline that first corrects fisheye distortion using calibrated parameters, then localizes the displayed image in the frame using SIFT-based feature matching and homography estimation, followed by perspective rectification to restore frontal view geometry.

Evaluation can be performed manually or automatically: manual mode lets users visually assess predictions alongside ground truth with IoU guidance (Fig 3), while automatic mode uses user-defined IoU thresholds to classify detections and segmentations automatically. Both methods report per-class and overall accuracy, mean IoU, and size-weighted IoU to emphasize larger objects.

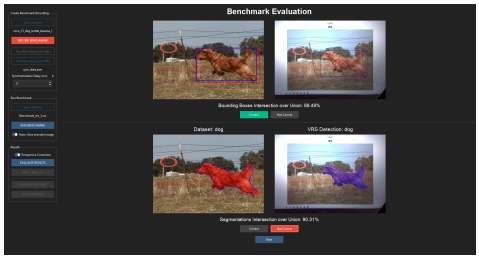

Figure 3: Manual Evaluation UI

After evaluation, failed predictions can be diagnosed by reviewing intermediate results for each incorrect annotation. By assigning errors to specific pipeline parts, failure rates can be computed for each stage, revealing bottlenecks and guiding improvements. This transforms the benchmark into a diagnostic tool that explains why failures occur, not just how often.

5 Results Evaluation

The evaluation results demonstrate the performance of the multimodal pipeline across two benchmark configurations, `benchmark_small` and `benchmark_large`, using different model sizes while keeping all other parameters constant. The primary metrics include overall accuracy, per-class detection and segmentation performance, mean Intersection over Union (IoU), and error type classification. As shown in the results (Table 1), benchmark large, which employs larger models for object detection and segmentation, achieves a higher overall accuracy of 83.00% compared to 78.00% in `benchmark_small`. Improvements are most notable in challenging classes like banana, where accuracy increases from 50% to 70%, indicating that model scaling benefits objects with higher visual variability or lower contrast.

Table 1: Object Detection and Segmentation Results Across Benchmarks

Object Class / Metric	**Detection**	**Segmentation**
	`small` | `large`	`small` | `large`
banana	9/20 | 14/20	10/20 | 14/20
bird	20/20 | 19/20	19/20 | 19/20
bottle	16/20 | 17/20	15/20 | 16/20
dog	20/20 | 20/20	19/20 | 19/20
tv	18/20 | 19/20	15/20 | 15/20
Overall Accuracy	**83/100** | **89/100**	**78/100** | **83/100**
Average IoU	**87.60%** | **90.47%**	**85.36%** | **86.21%**
Weighted Avg IoU	**90.43%** | **92.33%**	**86.56%** | **85.75%**

Error analysis (Table 2) reveals that the most common failure mode in `benchmark_small` is missing bounding boxes, accounting for 13.13% of cases, which drops significantly to 7.07% in benchmark large. This near 50% reduction highlights the substantial impact of using a larger object detection model, which also improves detection average IoU from 87.60% to 90.47%. However, this gain comes with a computational cost (Table 3): object detection runtime increases by approximately 80%, from 9.74s to 17.57s, which is not a lot relative to the other stages. The improved bounding box detection directly enhances segmentation accuracy, rising from 78.00% to 83.00%. Additionally, better-aligned bounding boxes provide more precise prompts to the segmentation model, reducing spatial errors in the resulting masks.

In contrast, upgrading the segmentation model yields only marginal improvements: average IoU increases slightly from 85.36% to 86.21%, while causing a 5x increase in processing time, from ∼44s to ∼230s. This suggests strongly diminishing returns: the larger segmentation model adds significant latency with minimal accuracy gain, especially since much of the improvement in segmentation can be attributed to better detection rather than superior mask generation. Most wrong segmentations actually come from mistakenly segmenting the screen where the benchmark is run, instead of the tv shown in the image.

Table 2: Error Rate Classification Across Benchmarks

Error Type	**small**	**large**
Speech to text	0.00% (0/100)	0.00% (0/100)
Extracted frames	0.00% (0/100)	0.00% (0/100)
Filtering	1.00% (1/100)	1.00% (1/100)
Eye gaze accuracy	0.00% (0/99)	0.00% (0/99)
Wrong bounding box	3.03% (3/99)	3.03% (3/99)
Missing bounding box	13.13% (13/99)	7.07% (7/99)
Perspective correction	0.00% (0/99)	0.00% (0/99)
Wrong segmentation	6.97% (6/86)	6.52% (6/92)

Table 3: Execution Time Comparison

Task	**small**	**large**
Speech to text & matching	43.43 s	45.54 s
Frames extraction	49.14 s	48.87 s
Eye gaze extraction	116.16 s	0.04 s
Frames selection	0.56 s	0.68 s
Object Detection	9.74 s	17.57 s
Object Segmentation	44.29 s	230.22 s
Total	**263.41 s**	**343.01 s**

6 Conclusion

The multimodal pipeline successfully demonstrates an effective approach to automating egocentric dataset annotation by integrating speech, eye gaze, and visual data from wearable sensors. The system achieves high accuracy and IoU scores, while highlighting limitations and possible future improvements. Beyond annotation, the system holds promise for future real-time assistive applications, such as aiding visually impaired users by providing context-aware, semantic understanding of what users see and say.