Point, Detect, and Discover

Real-Time Object Awareness and Annotation using the Meta Quest

ARUSHI JELOKA, KRISHNA PANCHAPAGESAN, and KRITI SHARMA, University of Washington

1 INTRODUCTION

Augmented Reality (AR) bridges digital information with physical environments, enhancing user interaction and understanding. Our project, *Point, Detect, and Discover*, aims to advance AR usability by combining precise hand-tracking, real-time object detection, and informative annotations directly within the user's field of view. Motivated by real-world applications in education, accessibility, and everyday convenience, our system transforms passive observation into active learning.

1.1 Contributions

- Integrated YOLO object detection with Meta Quest Pro using a screen capture approach on a Macbook to overcome direct video feed limitations.
- Implemented on-device inference using LLaVA-34B for accurate, contextual annotations displayed within AR (this had to be heavily quantized to run on our computer). We also peformed various inference optimizations to reduce time-to-response, both on the YOLO side as well as the llm.
- Used Unity + Depth Sensing via the Passthrough API to create annotations on objects which stick to objects in space. We also created a "box" in which objects in physical space could be put in such that the YOLO would run on it.

2 RELATED WORK

We've seen a lot of work in this space, notably by Meta in delivering Mixed reality experiences that enable people to understand the space around them. However, most of these demos were on Orion, their new AR proof of concept. Given that, we based our work off the following related work.

2.1 On-device Object Detection and Annotation in the Hololens2

Real-time AR object detection and labeling have been explored extensively, especially in the Hololens2. Notably, **IntelligentEdgeHOL** implemented object recognition on the HoloLens, demonstrating feasibility in mixed-reality contexts. A demo of this in action can be seen here. The github repo can be found here. The takeaway here for us is that simple object detection was possible almost a decade ago, but there's been little advancement in this space. We wanted to use the additional compute available on these newer devices along with higher performing models to deliver better information on objects in a viewer's space as well as enable the viewer to create their own annotations.

Authors' address: Arushi Jeloka, arushi.jeloka@uw.edu; Krishna Panchapagesan, kpanchap@uw.edu; Kriti Sharma, krits29@uw.edu, University of Washington.

2.2 Pairing Object Detection Models with vLLMs

One of the challenges that arises when using vLLMs is the aspect of constraining the potential video space into meaningful chunks that we could run inference on. We found an application of this in the recent Treehacks 2025 winning project **HawkWatch**, which effectively combined YOLO with large vision-language model to detect events, inspiring our use of YOLO alongside LLaVA. By doing this we were able to only process video streams which contained meaninful objects. See the github repo here.

3 METHOD

Our AR annotation framework consists of three core components:

1. Real-time Object Detection: Utilizing YOLO, the system continuously processes captured screen frames to detect and track objects.

2. Gesture-based Selection: Hand-tracking via Meta's SDK enables intuitive gesture-based interactions, including gaze-based object selection and pointing gestures.

3. Contextual Annotation: Once an object is selected, cropped image data is passed to an on-device LLaVA model (34B) to generate annotations comprising object name, usage, and price range. The user can also click on any object in passthrough to add a text annotation which remains in place.

4 IMPLEMENTATION DETAILS

4.1 Hardware and Software

The project employs a Meta Quest 3 headset and a local workstation for development. The headset provides the AR environment and initial user interactions, while the workstation handles computationally intensive tasks such as running deep learning models. Unity was used to populate the components needed to add annotations into the pass-through world such as a virtual keyboard, an input field, and text boxes which contain the annotations.

4.2 Challenges with Direct Video Feed

Initially, our goal was to utilize the direct video feed from the Meta Quest Pro headset to perform object detection. However, we encountered significant SDK limitations and API access constraints that prevented direct camera feed utilization. As a workaround, we implemented a robust screen-capture method leveraging Python libraries:

```
from mss import mss
import numpy as np
import cv2
from ultralytics import YOLO
```

The implementation captures the entire screen area in real-time using `mss`, which efficiently grabs frames at a high frame rate. Each captured frame is then resized and processed through the YOLO object detection model. YOLO identifies objects within the screen capture, providing bounding box coordinates for each detected object.

Upon detection, each object is cropped from the frame using the bounding box coordinates. These cropped images are encoded into a PNG format and further encoded into a base64 string to facilitate integration with the LLaVA-34B vision-language model running locally on-device. The core inference method is implemented as follows:

```
def run_llava_inference_from_crop(cropped_image):
    _, buffer = cv2.imencode('.png', cropped_image)
   image_b64 = base64.b64encode(buffer).decode('utf-8')

    prompt = (
        "Describe this object in three parts:\n"
        "1. What is the object?\n"
        "2. What could it be used for?\n"
        "3. What is its typical price range?\n"
        "ONLY RESPOND in JSON with keys:

        object_name, object_usage, object_pred_price."
    )

    response = client.chat.completions.create(
        model="llava:34b",
        messages=[{"role": "user", "content": [
            {"type": "text", "text": prompt},
            {"type": "image_url", "image_url":
         {"url": f"data:image/png;base64,{image_b64}"}}
        ]}]
    )
    # JSON parsing omitted for brevity
```

The LLaVA model processes each encoded image alongside the provided textual prompt, returning structured annotations in JSON format. These annotations include detailed information about the object's identity, possible uses, and estimated price range. This structured JSON response, combined with the original YOLO-generated bounding box information, is sent back to the headset for real-time AR annotation overlays, enhancing the user's interactive experience.

4.3 Challenges with Annotating in AR

Multiple challenges were encountered to get the annotations to populate correctly. Initially, we tried to populate a new text input and keyboard for each annotation. However, handling interactions and populating multiple instances of prefabs poses tracking and linking challenges. We improvised by creating a stand still text input and keyboard and extracting user controller poses to add text annotations. Overall, navigating Quest controller and hand interactions combined with the challenge of correctly scripting annotation behaviour with Unity was challenging.

4.4 Making it Seamless

One of the challenges we came across was that even when running a heavily quantized version of `llava:34b`, we still encountered blocking inference calls that would halt the screen recording loop. Initially, the synchronous LLaVA inference would freeze the YOLO detection pipeline until a response was received, interrupting the real-time video stream.

To address this, we decoupled the inference process from the main detection loop by integrating an asynchronous processing framework using Python's `asyncio`. Specifically, we wrapped the blocking inference function in an asynchronous wrapper using `asyncio.to_thread`. This allowed us to offload the computationally intensive inference task to a background thread, ensuring that the YOLO detection pipeline continued to capture and process frames in real time.

In addition, we implemented a throttling mechanism to prevent excessive inference calls. A global timestamp variable records the last time an inference was triggered, and we enforce a cooldown period (e.g., two seconds) before a new inference task can be spawned. By doing so, only the detection with the highest confidence within the given time frame initiates an inference call, balancing computational load with responsiveness.

The inference results are then sent to a persistent JSON viewer (implemented in a separate process) via a multiprocessing connection. This decoupled architecture ensures that the viewer is updated with the latest annotation data as soon as it becomes available, while the screen recording and object detection continue to run seamlessly. Overall, by offloading the LLaVA inference asynchronously and regulating its execution frequency, we maintained a smooth, real-time detection experience without noticeable interruptions in the video stream.

5 EVALUATION OF RESULTS

Initial evaluations demonstrate reliable real-time detection and annotation performance, averaging inference latencies below two seconds per object annotation. However, accuracy and latency trade-offs remain an active area for optimization. On the Quest, the manual annotations are implemented such that there is a red dot on the object which is being annotated and the text is right above it. The position tracking from the controller and the position setting for the annotation is highly accurate, and the annotation does not move around with the user. However, the angle of the annotation relative to the angle of the user does not change dynamically as well as expected.

6 DISCUSSION

Our method effectively enables rapid interaction and provides contextual information in real-time AR. However, latency in object annotation on the vLLM remains a key limitation, especially for rapid movements and cluttered environments.

7 FUTURE WORK

Future efforts will focus on:

- **Enhanced Object Highlighting Using Advanced Depth Sensing:** With the imminent release of Meta's video feed API—which will grant developers direct access to both the raw camera feed and high-resolution depth data—future work will leverage this new capability to improve object boundary creation. Instead of relying solely on rectangular bounding boxes from YOLO, we aim to integrate depth maps to extract more precise, three-dimensional object boundaries. This would enable natural AR overlays that accurately align with an object's true shape and location in 3D space. In our current setup, lacking the raw video feed meant that we could not project detections back into a 3D environment using depth information.

- **On-Device Processing and Latency Reduction:** Currently, our system uses an auxiliary device (a MacBook) for running the heavy YOLO and LLaVA models. With Meta's upcoming API providing a direct camera feed, we plan to port the entire pipeline to run on the Meta Quest device itself. By deploying a leaner version of our models (potentially using frameworks like `executorch`) and further optimizing via model quantization, we expect to dramatically reduce latency and simplify the overall system architecture.
- **Expanded Gesture Recognition for Targeted Inference:** Our current approach processes the entire video frame with YOLO and then selects the highest-confidence detection for LLaVA inference. In future iterations, we intend to integrate advanced hand tracking so that users can point directly to objects of interest. This targeted interaction would allow us to run inference specifically on the object being pointed at, rather than on the whole scene. Such a feature would not only enhance the user experience but also optimize computational resources by focusing on the area of interest.
- **Video Annotations and Immersion:** Our current XR application only supports text annotations. However, in the future we intend to add video and voice annotations for more immersive educational environments. We would also like to include live immersion with audio feedback depending on user interactions in the XR space.

8 CONCLUSION

Point, Detect, and Discover demonstrates a practical and immersive approach to AR-based object interaction and annotation, opening numerous potential applications in education, accessibility, and everyday convenience.

ACKNOWLEDGMENTS

We thank the CSE 493 staff for their guidance and support throughout this project.

REFERENCES

https://developers.meta.com/horizon/develop/unity
https://docs.unity3d.com/6000.0/Documentation/Manual/AROverview.html
https://developers.meta.com/horizon/downloads/package/meta-xr-sdk-all-in-one-upm/
https://huggingface.co/liuhaotian/llava-v1.6-34b
https://www.v7labs.com/blog/yolo-object-detection
https://doi.org/10.48550/arXiv.1506.02640