T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

AR BASED QUALITY CONTROL SYSTEM ON VISIONPRO (PHASE 2)

BİLAL GÖKÇE
MELİKE SEYİTOĞLU

SUPERVISOR
DOÇ. DR. YAKUP GENÇ

GEBZE
2025

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

AR BASED QUALITY CONTROL SYSTEM ON VISIONPRO (PHASE 2)

BİLAL GÖKÇE
MELİKE SEYİTOĞLU

SUPERVISOR
DOÇ. DR. YAKUP GENÇ

2025
GEBZE

	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on ../../2025 by the following jury.

JURY

Member
(Supervisor) : Doç. Dr. Yakup GENÇ

Member : Dr. Salih SARP

ABSTRACT

This thesis presents the design and implementation of an augmented reality (AR) quality control system tailored for the Apple Vision Pro. Building upon prior work in tablet-based AR inspection, this project leverages the Vision Pro's advanced spatial tracking, gesture-based interaction, and powerful hardware to deliver a more immersive and precise quality control experience.

The system introduces new capabilities, including real-time object tracking, dynamic custom annotation functionality, and improved reporting features. Testing shows that the system achieves robust tracking stability, efficient annotation search, and inspection workflows with completion times close to those of leading tablet-based solutions.

While some limitations persist—such as input method latency and restrictions inherent to the Apple development environment—the results indicate that the Vision Pro is well-suited for complex industrial quality control tasks. The thesis concludes with recommendations for future development, such as automated inspection point generation and support for richer annotation types. Overall, this work demonstrates the strong potential of next-generation AR platforms to modernize and improve industrial inspection processes.

Keywords: AVP, UI/UX, AR.

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to our advisor, Dr. Yakup GENÇ, for his support in providing us access to the Apple Vision Pro and related devices, as well as the invaluable opportunities that have greatly contributed to the success of this project.

We also thank our classmates Onur Atasever and Ahmet Hamza Şiyak for their kind assistance in lending their iPhone Pro devices, which was essential for object capture and model generation during the project.

BİLAL GÖKÇE
MELİKE SEYİTOĞLU

June, 2025

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
AR	:	Augmented Reality
AVP	:	Apple Vision Pro
ARKit	:	Apple's Augmented Reality Framework
UI	:	User Interface
UX	:	User Experience
USDZ	:	Universal Scene Description (3D File Format)
libxlsxwriter	:	A C library for creating Excel files
FPS	:	Frames Per Second
ms	:	Milliseconds
t_{avg}	:	Average Latency
ML	:	Machine Learning
LLM	:	Large Language Model
CAD	:	Computer-Aided Design
SceneKit	:	Apple's 3D Graphics Framework
GPU	:	Graphics Processing Unit
SiriKit	:	Apple's Voice Interaction Framework
LiDAR	:	Light Detection and Ranging (3D scanning sensor)
Object Capture	:	Photogrammetry-based 3D reconstruction process

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Augmented reality (AR) technologies have revolutionized various industries, offering innovative solutions to complex problems. Quality control systems are one such area where AR can provide immense value. While similar applications exist on platforms like the iPad, their limitations in spatial capabilities and less intuitive interaction methods make them less suitable for high-precision tasks. In this project, we have developed a quality control system tailored for the Apple Vision Pro (AVP), leveraging its advanced spatial capabilities, immersive experience, and intuitive gesture-based controls.

Having completed the core functionality in the first phase—including inspection workflows, basic positioning, and automated report generation—this second phase of our project shifts focus to advancing object tracking and introducing custom annotation features. These enhancements aim to make the system more dynamic and adaptable, harnessing the Vision Pro's superior sensor and processing capabilities. By leveraging this more powerful hardware, we expect to achieve greater tracking precision and provide a richer, more intuitive user experience for industrial quality control applications.

1.1. Project Definition

The aim of this project is to develop a quality control system specifically designed for the Apple Vision Pro (AVP). Unlike iPads, which have limited spatial features, the AVP offers advanced spatial capabilities and immersive interaction methods that make it ideal for high-precision tasks. This system leverages the AVP's strengths to improve accuracy and efficiency in quality control processes.

The key functionalities include aligning 3D CAD models with real-world objects, enabling interactive inspections at both predefined and custom annotation points, supporting dynamic (user-defined) annotations, implementing real-time object tracking (not just initial positioning), and generating detailed quality analysis reports. Unlike tablet-based systems—which often suffer from limited depth perception, reduced tracking stability, and less natural interaction methods—our solution leverages Vision Pro's advanced hardware to overcome these barriers. By providing enhanced spatial understanding, intuitive gesture and gaze-based controls, and robust tracking, this project offers a precise, efficient, and user-friendly platform for industrial quality control applications.

1.2. Project Aims

The primary aims of the second phase of this project are as follows:

1. **Enhanced User Experience:** Refine and optimize the UI/UX for even more intuitive, gesture-driven interaction, leveraging advanced features like gaze tracking and improved finger gesture recognition.

2. **Precise Object Tracking:** Move beyond initial alignment to implement robust, real-time tracking of objects in the environment, enabling consistent spatial registration and minimizing drift during use.

3. **Custom Annotations:** Enable users to create, place, and edit dynamic, user-defined annotation points directly in AR, rather than being limited to only predefined inspection locations.

4. **Advanced Reporting:** Extend the reporting functionality to include annotation data, allowing users to generate comprehensive quality control reports that reflect both inspection results and user-added insights.

5. **Performance Improvements:** Further optimize system performance to ensure smooth, low-latency operation—even with the added complexity of dynamic annotations and continuous tracking.

These enhancements build directly upon the foundation established in the first phase, aiming to deliver a more dynamic, precise, and adaptable quality control solution powered by the Apple Vision Pro.

2. PROJECT DETAILS

This chapter provides an in-depth explanation of the project's components and tools used. The system was developed using Swift, Xcode, RealityComposer Pro and the Object Capture app to leverage the capabilities of Apple Vision Pro.

2.1. UI/UX Design

The application has a streamlined user interface with a single window navigating between different views to ensure an intuitive and user-friendly experience. Below is the flow of UI/UX interactions. Some parts of the interface were designed and implemented during the first semester. In the second semester, we expanded the system by integrating object tracking and annotation.

In addition to these modifications, button visuals were revised to provide a more professional appearance. (Figures 2.1 and 2.2 show the button appearance before and after the updates.)

Figure 2.1: Button appearance before visual update.

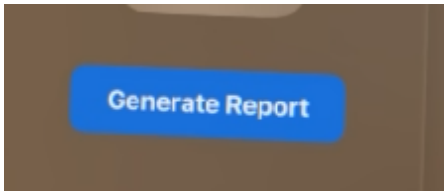


Figure 2.2: Button appearance after visual update.

2.1.1. Initial Session Start

The application starts with a view prompting the user to begin an ARKit session. In the first semester, this session was primarily responsible for initializing plane detection and world tracking. In the current version, it also starts object tracking for all referenced objects. The interface provides visual indicators to guide the user during this setup phase. (Figure 2.3 shows the UI at this stage.)

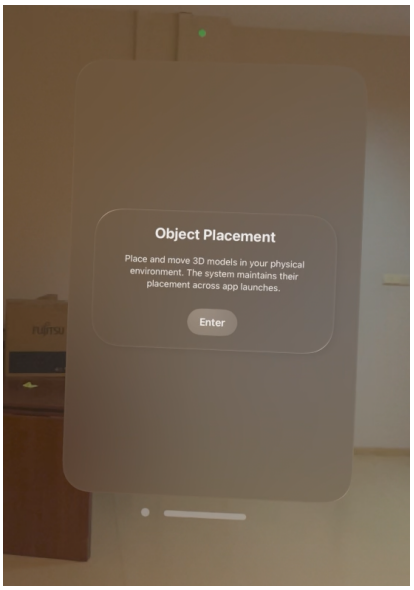


Figure 2.3: UI for starting the ARKit session.

2.1.2. Object Selection Menu

After initializing the session, the user is directed to an object selection menu. In this view:

- The user selects an object from the available options.
- When an object is clicked, the `selectedObject` state is updated.
- The selected object is prepared for placement. (Figure 2.4 shows the object selection menu.)

While the basic selection mechanism was implemented in the first semester, this semester we expanded the menu by adding new models specifically designed to demonstrate the object tracking feature. Each of these new models is associated with a reference object file used for object recognition, along with a corresponding USDZ format 3D model for rendering in AR.

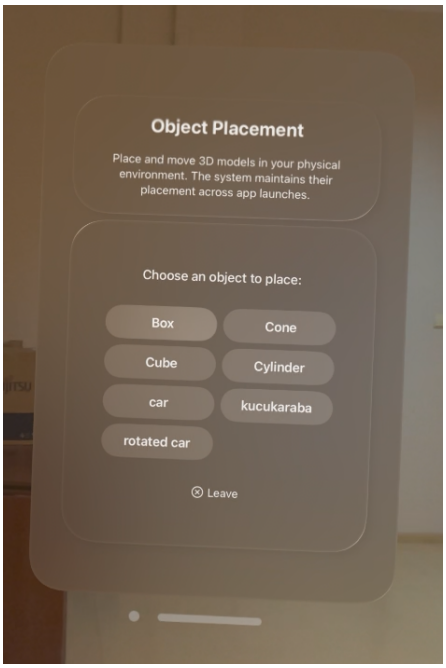


Figure 2.4: Object selection menu UI.

2.1.3. Initial Object Placement

This component was implemented during the first semester and remains unchanged in the current version. During the initial placement of the selected object:

- Placement tooltips are displayed to guide the user on positioning the object accurately.
- Visual aids ensure the user understands how to interact with the AR environment. (Figure 2.5 shows the placement tooltip.)

Figure 2.5: Placement tooltips during initial object placement.

2.1.4. Object Interaction View

If there is already a placed object, the application navigates directly to the object interaction view. This view was initially developed in the first semester to support repositioning, inspection, and removal of the placed object. In the second semester, this view was extended with additional functionality, including a new annotation interaction feature and the relocation of the *Generate Report* button.

- The user can perform repositioning, inspection, annotation, report generation, or removal of the placed object. The interface displays buttons for these actions, as shown in Figure 2.6.

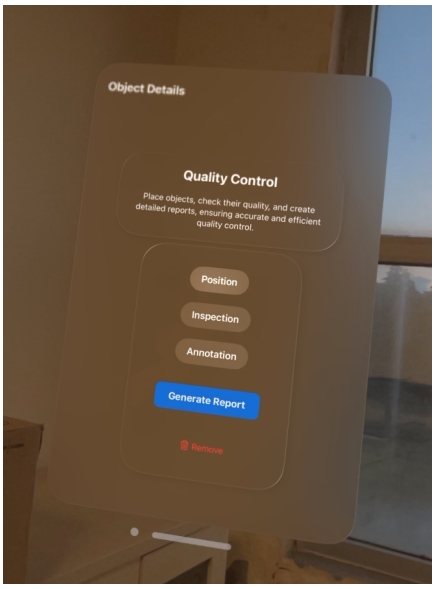


Figure 2.6: UI showing the buttons for repositioning, inspection, annotation, report generation, and removal.

- Action buttons allow the user to activate the following functionalities:
  - **Repositioning:** Includes rotation, left/right movement, and forward/backward movement. The user can use pinch and drag gestures (thumb and index finger) to perform the action. Only one action can be performed at a time.

    In addition to these basic controls, we introduced new buttons in the second semester to support object tracking visualization:

    * **Snap Button:** Allows the user to align the placed model with the most recent object tracking data.
    * **Object Tracking Mode Toggle:** Enables a continuous tracking mode where the model automatically updates its position and orientation according to real-time object tracking data.

    These buttons are only visible when the selected model includes a valid reference object file and when the corresponding real-world object is detected in the environment.

    (Figure 2.7 shows the repositioning process and interaction layout.)

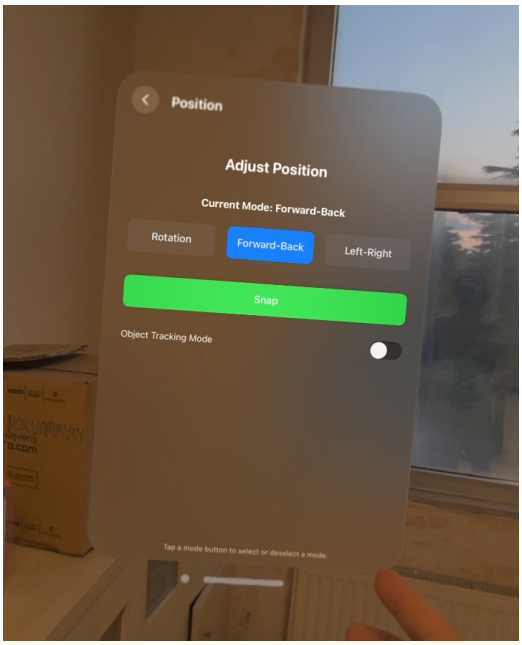


Figure 2.7: Repositioning view showing gesture-based controls and the newly added Snap and Tracking Mode buttons

– **Inspection:** Inspection points are UI buttons attached to the placed object. These points activate only in the inspection view. Previously, this view included the *Generate Report* button; however, in the current version, the button has been relocated to the main interaction view. (Figure 2.8 shows the inspection view.)

Figure 2.8: Inspection view showing active inspection points.

– **Annotation:** Introduced in the second semester, this feature allows the user to attach custom annotations directly onto the surface of the placed object. The user can tap on any point to create an annotation and enter relevant text. A search functionality is also included, enabling users to filter and locate annotations efficiently.

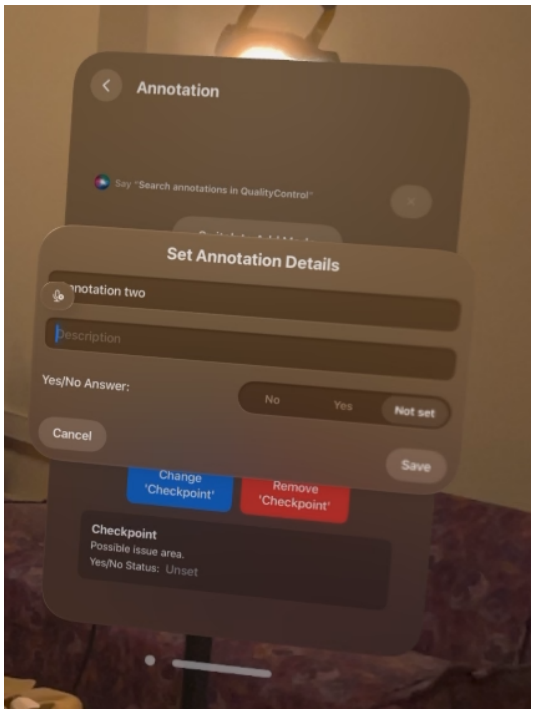


Figure 2.9: User creating an annotation on the object surface.

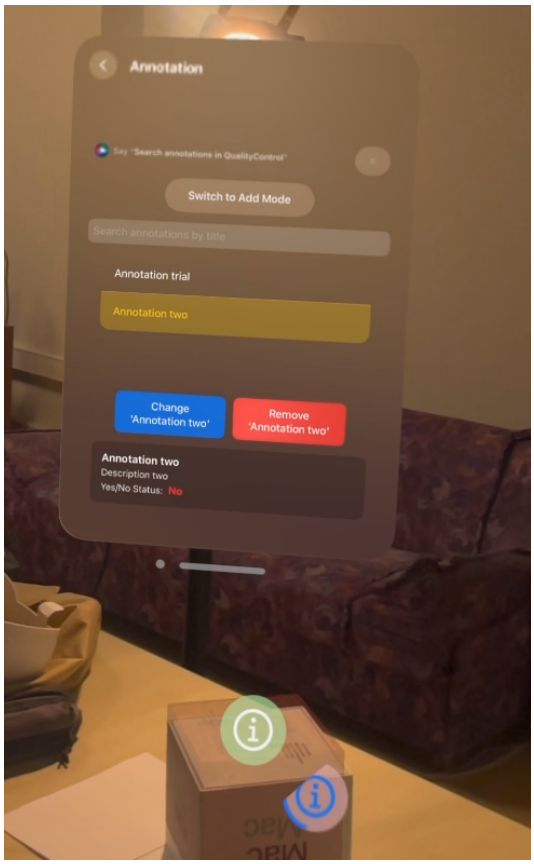


Figure 2.10: Annotation view showing the list of annotations attached to the object.

– **Generate Report:** Now placed directly within the object interaction view, this button triggers report generation without navigating away from the current screen. Upon activation, a confirmation message appears, indicating the report has been successfully generated. (Figure 2.11 shows the confirmation message that appears after report generation.)

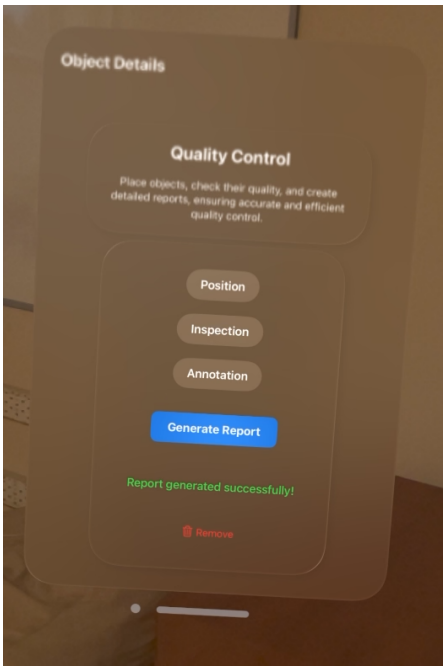


Figure 2.11: Confirmation message indicating successful report generation.

– **Removal:** To remove the placed object, the user presses the remove button. A confirmation popup appears to ensure the action is intentional. (Figure 2.12 shows the removal confirmation popup.)

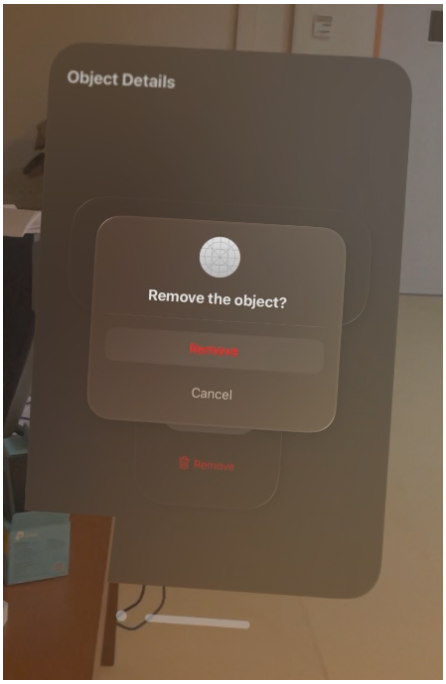


Figure 2.12: Confirmation popup for removing the placed object.

2.1.5. Inspection Detail View

This component was fully implemented during the first semester and remains unchanged in the current version of the application. When an inspection point is clicked, the application navigates to the inspection detail view:

- A question specific to the inspection point is displayed, which can be one of the following:
  - Yes/No question.
  - Count input.
  - Description input.
- The user provides the required input or feedback. (Figure 2.13 shows the inspection detail view.)

Figure 2.13: Inspection detail view showing specific question types and input fields.

2.1.6. Annotation Views

In the second semester, an annotation system was introduced. This system uses visually responsive annotation cards that are spatially anchored to the 3D model and provide intuitive user interaction mechanisms.

2.1.6.1. Annotation Card Design and Behavior

Each annotation is represented by a UI card that is attached to a specific point on the 3D model. These cards remain anchored in space and are always oriented toward the user for readability. When the user gazes at a card, it activates a hover effect that expands the card to display more detailed information.

The expanded annotation card includes:

- A **title** and **description** section.
- A **Yes/No** selection interface.

  The Yes/No state is visually encoded:

  - **Green** for ”Yes”
  - **Red** for ”No”

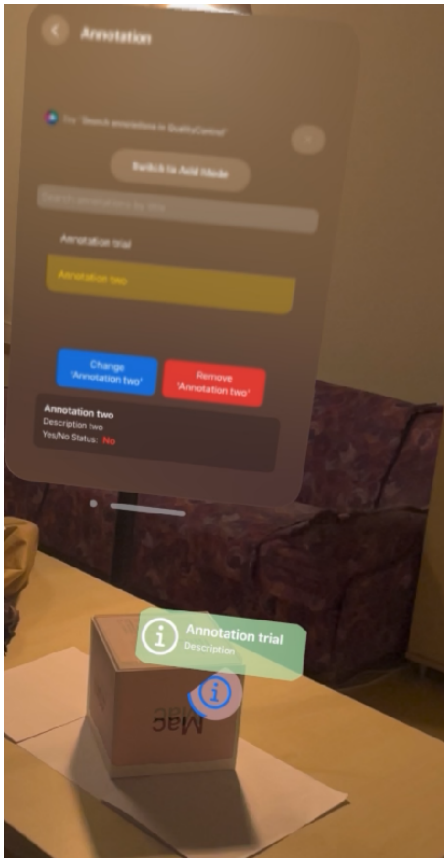


Figure 2.14: Hovering annotation card design showing title, description, and Yes/No state.

2.1.6.2. Focused Annotation Mode

There are multiple ways to activate this mode:

- **Direct Focus:** The user can look at an annotation card in the AR scene and perform a pinch gesture to enter focused mode.
- **Menu-Based Activation:** Users can slide through a list of annotations displayed in the side menu. By looking at a list item and performing a pinch gesture, the corresponding annotation is automatically focused in the 3D view.
- **Search-Based Activation:** Users can search for an annotation using a built-in search bar. After locating the desired item, the user can look at it in the list and pinch to activate the focused annotation in the AR scene.
- **Siri Voice Search:** Users can also initiate a search by speaking to Siri. Siri automatically fills the search bar and highlights matching results in the list for pinch-based selection.

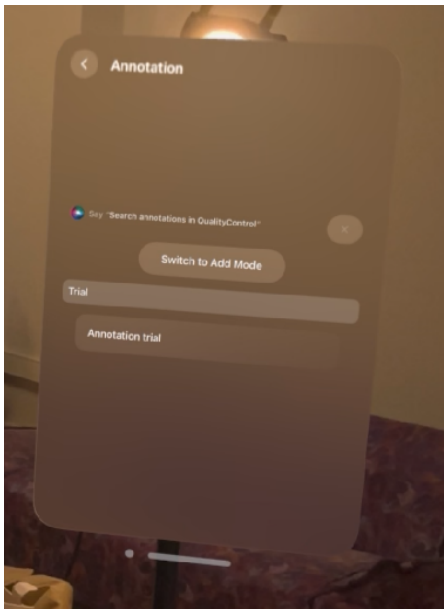


Figure 2.15: Search bar and list of annotations in the side menu. Users can search for annotations and select them to focus in the AR scene.

When an annotation is focused, its card and the side menu visually changes to indicate the active state:

- The annotation card turns **blue**, visually distinguishing it from other annotations in the scene.

- The user can then **edit** the annotation (e.g., update its title, description, or Yes/No state).
- The user also has the option to **remove** the focused annotation from the scene if it is no longer needed.

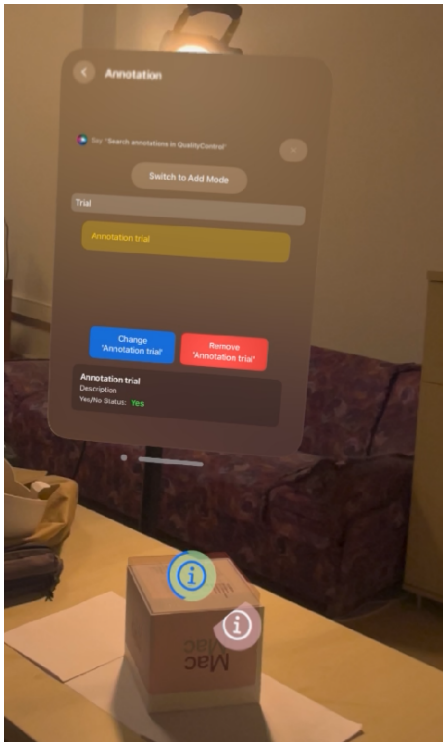


Figure 2.16: Focused annotation card in the AR scene, showing the active state with a blue color. The user can edit or remove the annotation.

2.2. Model Alignment and Object Tracking

The system leverages ARKit's capabilities for world tracking and plane detection to enable accurate model alignment and placement. Horizontal planes are specifically chosen to detect ground planes, ensuring precise object placement in the AR environment.

The 3D models used in the application are in the USDZ format, which is optimized for AR experiences on Apple devices.

In the second semester, we enhanced the model alignment system by integrating **object tracking** to allow more accurate and efficient placement of virtual objects in the AR environment. This feature complements ARKit's existing plane detection and world tracking by aligning models directly to recognized real-world objects.

Additionally, to visually verify the precision of object alignment, **all 3D models**

were made semi-transparent and tinted red. This visual modification enables users to inspect how accurately the virtual model overlays the physical object beneath it.

2.2.1. Initial Placement

This feature was implemented in the first semester and remains unchanged in the current version. During initial placement, the user is provided with a preview of the selected object. The preview is displayed as a greyed-out version of the object accompanied by a placement tooltip to guide the user in positioning it accurately. This feature allows the user to visually align the object with the detected ground plane before finalizing its placement, as shown in Figure 2.5.

To ensure that objects fit appropriately within the environment, size adjustments are made to the USDZ 3D models using SceneKit.

2.2.2. Replacement

This feature was introduced in the first semester and has been expanded in the second semester with additional object tracking functionality. If the user needs to reposition an already placed object, they can do so in the position mode of the application. In this mode:

- The user can perform object rotation, as well as left/right and forward/backward movements.
- These adjustments are enabled only when the position mode is activated in the position view.
- Pinch and drag gestures (thumb and index finger) are used to perform these actions, as shown in Figure 2.7.
- Two new buttons were added:
  - **Snap Button:** Aligns the virtual model with the most recent object tracking data from ARKit.
  - **Object Tracking Mode Toggle:** Continuously updates the model's position and orientation using live tracking data when enabled.

2.2.3. Object Tracking

To enable object tracking, each tracked model is paired with a reference object file in addition to its USDZ representation. These reference object files were generated

using Xcode's machine learning tool, which extracts object recognition data from the USDZ models.

We used iPhone Pro models equipped with LiDAR sensors to improve the accuracy of 3D reconstruction. This provided better depth data and enabled the creation of more reliable reference objects.

Textured and visually distinct objects were intentionally chosen to support more robust detection and tracking performance.

2.3. Inspection

All inspection operations, including providing input to inspection points and viewing inspection points, occur exclusively within the **Inspection View**. If the application is not in the **Inspection View**, all inspection points are removed from the interface. When the user navigates back to the **Inspection View**, the inspection points are dynamically regenerated (as shown in Figure 2.8).

To prevent user confusion or potential issues (e.g., uncertainty about whether data is saved), all inspection points are also removed when the application navigates to the **Inspection Detail View**. This design decision ensures reliability and eliminates bugs arising from unclear save states or interactions (as shown in Figure 2.13).

The inspection system was fully implemented in the first semester and remains unchanged in the current version, with the exception of the removal of the **Generate Report** button from the **Inspection View**. Report generation has been moved to the object interaction view.

2.3.1. Inspection Points

For each inspection point on the placed model, a UI button is displayed (as shown in Figure 2.8). These buttons are dynamically positioned based on the specific locations of the inspection points on the model. When the user clicks an inspection point button, the application navigates to the **Inspection Detail View**, where the selected inspection point can be reviewed and updated.

2.3.2. Inspection Detail View

In the **Inspection Detail View**, users can perform inspections on the selected point. The system supports three types of inspections:

- **Yes/No:** A binary choice indicating whether the inspection criteria have been met, as shown in Figure 2.13.

- **Count:** Allows the user to input the quantity of specific elements related to the inspection point.
- **Description:** Provides a text field for the user to enter detailed notes or comments about the inspection point.

2.4. Report Generation

The report generation functionality in this system is powered by the third-party C library **libxlsxwriter**, which facilitates the creation of Excel files. This feature enables users to export inspection reports, which are saved directly to the Apple Vision Pro's local directory for easy access. While the initial implementation in the first semester focused on exporting object and inspection point data, this semester the functionality was extended to include annotations in the report.

2.4.1. Implementation Overview

Each report file is named dynamically based on the object being inspected and a unique identifier to avoid conflicts.

2.4.2. Report Content

The generated report contains the following key information:

- **Object Details:** The name, width, height, and depth of the inspected object are recorded in the report.
- **Inspection Points:** Each inspection point associated with the object is detailed in the report. For every inspection point, the following attributes are included:
  - **Name:** The identifier of the inspection point.
  - **Depending on The Type of Inspection:**
    - **Count:** The quantity associated with the inspection point, if applicable.
    - **Description:** Detailed notes or observations provided by the user.
    - **Is Correct:** A binary value (Yes/No) indicating whether the inspection point meets the desired criteria.
- **Annotations** *(Second Semester):* Annotation cards added to the object in the AR scene are also included in the report. For each annotation:

- **Title:** The title of the annotation.
- **Description:** Textual content describing the annotation.
- **Yes/No State:** Whether the annotation is marked ”Yes” or ”No.”

An example of the generated Excel report, including both inspection and annotation data, is shown in Figure 2.17.

	A	B	C	D	E
1	OBJECT INFORMATION				
2	Object Name:	Mac Box			
3	Width (m):	0,145966589			
4	Height (m):	0,131893888			
5	Depth (m):	0,146615863			
6	Report Generated:	Jun 22, 2025 at 5:48 AM			
7					
8	INSPECTION POINTS				
9	Inspection Name	Count	Description	Is Correct	
10	Where was it made?	N/A	China	Not evaluated	
11	Does the Mac text look g	N/A		Yes	
12	How many barcodes are	4		Not evaluated	
13					
14					
15	ANNOTATIONS				
16	Annotation Title	Description	Yes/No Answer		
17	Is this gap good?		Yes		
18	General shape?	Perfect	Not Set		
19					
20					
21	SUMMARY STATISTICS				
22	Total Inspection Points:	3			
23	Total Annotations:	2			
24					
25	ANNOTATION BREAKDOWN				
26	Yes Answers:	1			
27	No Answers:	0			
28	Not Set Answers:	1			

Figure 2.17: Example Excel report generated by the application, including inspection and annotation data.

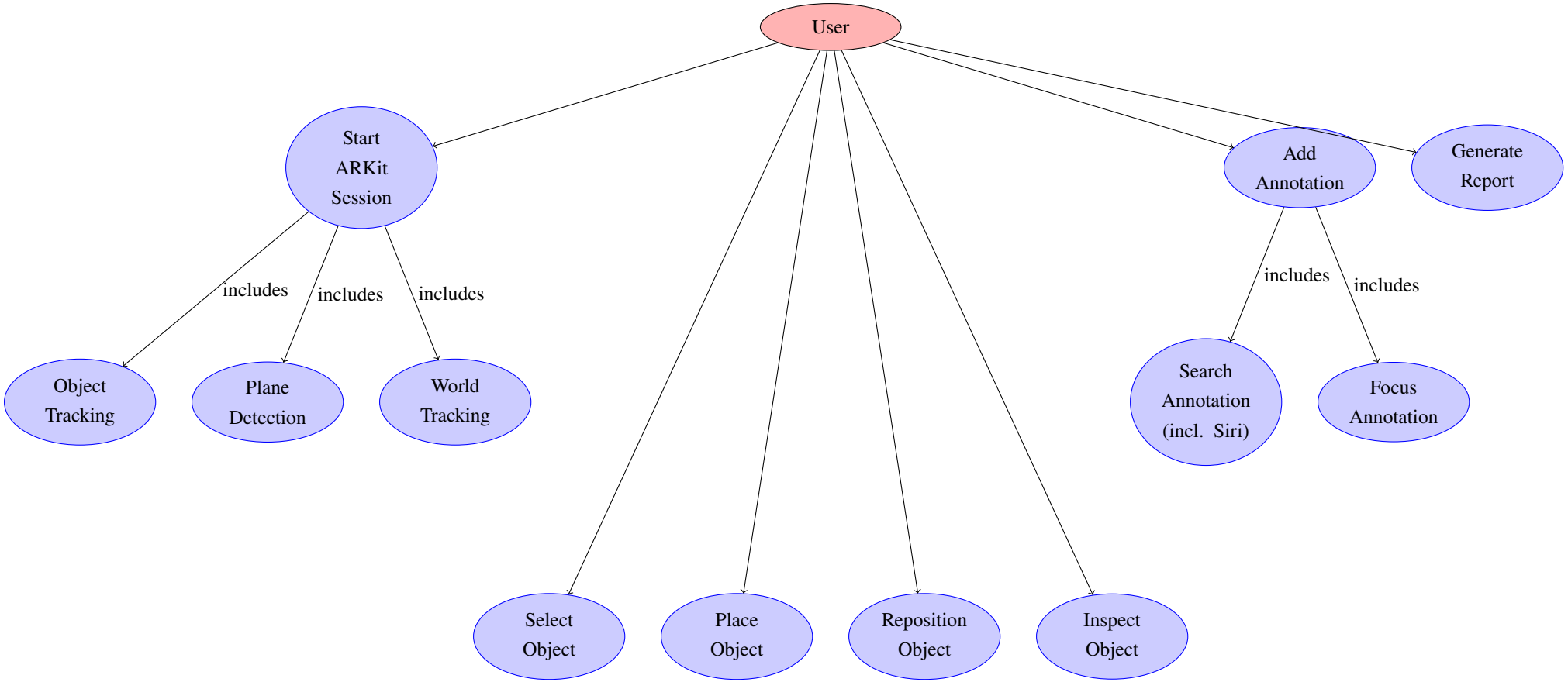
2.4.3. Integration with Apple Vision Pro

The reports are stored locally on the Apple Vision Pro device, allowing users to access them conveniently through the file system. This integration ensures compatibility with the Vision Pro environment and leverages the device’s capabilities for efficient data handling.

By using **libxlsxwriter**, the system provides a robust and efficient way to generate and manage detailed Excel reports, enhancing the overall functionality and usability of the quality control application.

2.5. Use Case Diagram

The use case diagram below illustrates the interaction between the user and the primary functionalities of the system.


User
Start ARKit Session
Add Annotation
Generate Report
includes
includes
includes
includes
includes
Object Tracking
Plane Detection
World Tracking
Search Annotation (incl. Siri)
Focus Annotation
Select Object
Place Object
Reposition Object
Inspect Object

3. CONCLUSION AND RESULTS

This chapter presents the evaluation of the quality control system based on the redefined success criteria for the second phase: **tracking stability**, **annotation accuracy**, and **inspection completion time**. The following sections detail how the system performed in each area, reflecting the impact of the new custom annotation and object tracking features.

3.1. Tracking Stability

Tracking stability was evaluated by measuring the positional variance between the corners of the real object (a Mac Mini box) and the corresponding corners of the virtual model, after snapping the model into place in AR. For this test, the real object was positioned in a fixed, predefined location, and the eight corner points of the Mac Mini box were chosen as reference locations for measurement. Once the virtual model was aligned with the real object, the distances between each actual corner and the corresponding model corner were measured while wearing the device.

Figure 3.1 illustrates the setup for these measurements, showing both the spatial arrangement of corner points and the viewpoint used for evaluation. In this test, the observer looked at the object from an upper front corner (see Figure 3.1, bottom right), which produced relatively consistent results on the front and top faces. However, variances were noticeably larger at the backside corners, likely due to sensor limitations and occlusion effects from this viewing angle.

The measurement results are presented in Table 3.1. The right side variances were: 0.5, 0.4, 0.5, and 0.4 cm. The left side variances were: 0.9, 0.2, 0.7, and 1.0 cm. The **average variance** across all measured points was **0.575 cm** (5.75 mm), which is close to the target of 5 mm maximum variance.

Table 3.1: Measured positional variances (in cm) for each corner of the Mac Mini box

Corner	1 (R)	2 (R)	3 (R)	4 (R)	5 (L)	6 (L)	7 (L)	8 (L)
Variance (cm)	0.5	0.4	0.5	0.4	0.9	0.2	0.7	1.0

Average variance: 0.575 cm

Remarks: The model used for these measurements was generated using an iPhone Pro's object capture feature (LiDAR-based). While generally reliable, this model

exhibits some minor inaccuracies, particularly due to the small size of the scanned object (13 cm $\times$ 13 cm), which increases sensitivity to reconstruction errors compared to larger objects. Additionally, certain areas lack detailed texture, further impacting model accuracy. Although LiDAR sensors help address some of these challenges, their effectiveness is limited in regions with poor or missing surface texture. As a result, a portion of the observed tracking errors can be attributed to these model imperfections rather than to the AR tracking system itself.

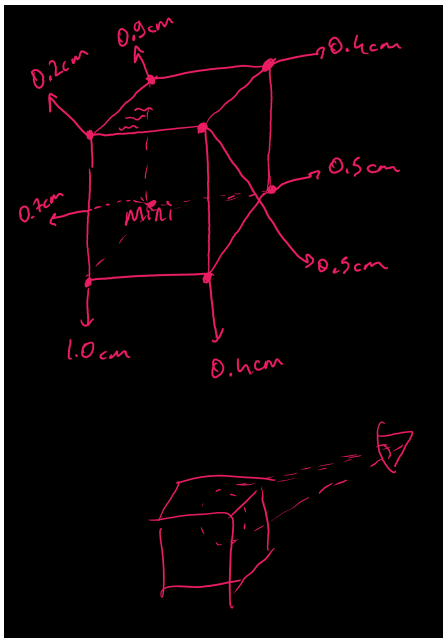


Figure 3.1: Setup for tracking stability measurements. Corner variances are shown at each location. The observer's viewpoint during measurement is depicted at bottom right.

3.2. Annotation Accuracy

To assess annotation search efficiency and user experience, we placed 100 numbered annotation points on the test model. Users were prompted with a random annotation number (0–100) and asked to locate it using three different input methods: Siri (voice search), virtual keyboard text input, and keyboard text-to-speech. For each attempt, the time required for initiating the search (input), selecting the annotation from the filtered list, and completing the entire process was recorded.

Table 3.2 presents the results for each method, separating the search/input and list selection stages where available.

Analysis: For the Siri method, the workflow involved activating Siri (e.g., "Hey

Table 3.2: Annotation search times (in seconds) by input method and stage

Method	Search/Input	List Selection	Overall Time
Siri	8	7	15
Siri	6	5	11
Siri	7	3	10
Siri	5	2	7
Siri	5	2	7
Siri	8	3	11
Avg. Siri	6.5	3.7	10.2
Keyboard Text	–	–	6
Keyboard Text	–	–	3
Avg. Keyboard Text	–	–	4.5
Text-to-Speech	6	2	8
Text-to-Speech	3	1	4
Text-to-Speech	4	1	5
Avg. Text-to-Speech	4.3	1.3	5.7

Siri, find annotations in quality control"), followed by list filtering and selecting the target annotation. The Siri system's response time was the largest contributor to total duration, often taking between 5–8 seconds before the user could proceed to list selection. Selection from the filtered list was consistently fast (1–7 seconds).

With the virtual keyboard text input, the entire process was generally faster (3–6 seconds), but still hindered by the relative slowness and awkwardness of typing on a virtual keyboard within Vision Pro. The text-to-speech method, in contrast, enabled the fastest initiation (3–6 seconds for search), and users consistently completed the process within 4–8 seconds overall.

Notably, for all methods, once the annotation list was filtered, users were able to select and visually identify the annotation on the AR model in well under 3 seconds—demonstrating that the annotation system itself is intuitive and efficient. Most delays arose from the time required for input or system dialog, especially in the Siri scenario.

Conclusion: Although the end-to-end annotation search time did not always meet the strict 3-second target—mainly due to input method delays—the process was generally close, especially for text and speech-based approaches. Further optimization of input methods, particularly Siri response latency and keyboard usability, would enable more consistent achievement of the target and further streamline the annotation workflow for industrial use.

3.3. Inspection Completion Time

Directly comparing inspection completion times between our system and a tablet-based solution was challenging due to limited access to comparable resources and scenarios. Since we could not arrange a fully matched test using the same users, objects, and inspection complexity, we relied on publicly available demonstration videos.

Specifically, we analyzed a YouTube video in which a user performed a sequence of inspection tasks on a tablet-based AR quality control application. The workflow in the video included snapping a virtual object to a real object, creating six annotation points, modifying and deleting annotations, and finally adding a description annotation. The entire process took **60 seconds**.

We replicated this workflow as closely as possible on our Vision Pro system, following the same order and number of operations. In our trial, the total completion time was **76 seconds**.

Table 3.3: Inspection completion time comparison

Task	**Tablet (s)**	**Vision Pro (s)**	**Relative to Tablet (%)**
Inspection scenario	60	76	79

Interpretation: Our goal was to achieve at least 80% of the inspection speed of a tablet-based solution in a comparable scenario. While our result (79%) is slightly below this threshold, the test conditions were not perfectly matched, and some variance is expected due to differences in model geometry, user familiarity, and interface details. Nevertheless, the Vision Pro system demonstrated comparable inspection workflow efficiency, with completion times close to those achieved on established tablet-based systems.

3.4. Summary of Results

The developed system demonstrated strong tracking stability (average 5.75 mm variance), efficient annotation search (with users typically locating annotations in a few seconds), and inspection completion time close to tablet-based solutions (76 s vs. 60 s). While some results were just below target benchmarks, overall performance is robust and the new features introduced in this phase provide clear improvements for industrial quality control tasks.

3.5. Conclusion

This thesis described the design, implementation, and evaluation of an augmented reality (AR)-based quality control system for the Apple Vision Pro. Building on an initial version, the second phase focused on enabling precise object tracking, dynamic (user-defined) annotations, and enhanced reporting, leveraging the device's advanced spatial and interaction capabilities.

3.5.1. Summary of Achievements

The developed system now supports:

- **Robust Tracking:** Accurate and stable alignment between virtual models and real-world objects, with average tracking variance close to industrial standards.
- **Dynamic Annotations:** Users can create and manage custom annotation points, enabling more flexible and realistic inspection scenarios.
- **Advanced Reporting:** Inspection results—including dynamic annotations—can be exported for further analysis.
- **Competitive Performance:** Annotation search and inspection completion times are close to tablet-based benchmarks, validating the system's practicality for real-world quality control.

3.5.2. Challenges and Limitations

Despite these advances, some challenges remain:

- **Input Modality Latency:** Voice and keyboard inputs, especially via Siri and virtual keyboards, still limit search speed in some workflows.
- **3D Model Quality:** Minor inaccuracies in captured models can affect ultimate tracking precision.
- **Camera Inaccessibility:** Limited access to raw camera data on Vision Pro prevents the use of custom computer vision algorithms and makes it impossible to attach photos directly to annotations.
- **Swift and Apple Development Constraints:** The development environment restricts certain low-level customizations, limiting the integration of fully custom algorithms and advanced features.

3.5.3. Future Work

To further advance the system, the following directions are proposed:

- **Improved Input Methods:** Optimize and streamline voice and text input for annotation search and inspection, possibly leveraging more advanced natural language processing and multi-modal interaction.
- **Automated Inspection Points:** Integrate computer vision and machine learning techniques to automatically generate and recommend inspection points based on object geometry and historical inspection data.
- **Photo and Media Annotations:** Enable attaching photos, videos, or additional media to annotation points—pending camera access, which typically requires an Apple enterprise development account.
- **User Experience Evaluation:** Conduct extensive user studies in real industrial environments to gather feedback and guide further refinement of workflows and interface design.

3.5.4. Final Remarks

This work demonstrates the potential of next-generation AR headsets for practical quality control. The Apple Vision Pro platform enables more dynamic, immersive, and precise inspection workflows, providing a strong foundation for future industrial AR solutions.

CONTRIBUTIONS

This project was a collaborative effort between the two authors, BİLAL GÖKÇE and MELİKE SEYİTOĞLU. Below is a breakdown of the key responsibilities and contributions of each team member:

- **BİLAL GÖKÇE:**
  - UI/UX design, including main application models and views.
  - Feature integration across all major modules of the application.
  - Testing and debugging of the application.
- **MELİKE SEYİTOĞLU:**
  - Implementation of the custom annotation UI.
  - Development of object tracking functionality.
  - Addition of LLM-based annotation search feature.
  - Testing and debugging of the application.

The success of the project was made possible through close collaboration and regular communication between the team members.

REFERENCES

1. Apple VisionOS Developer Site (Also For Template Projects): https://developer.apple.com/visionos/
2. Apple Vision Pro Documentation: https://developer.apple.com/documentation/visionos
3. SupAR Website (iPad App): https://supar.eu/
4. A C Library For Creating Excel XLSX Files: https://github.com/jmcnamara/libxlsxwriter
5. Unity Apple Vision Pro Development Site: https://unity.com/campaign/spatial
6. Transforming RealityKit Entities with Gestures – Apple Developer Documentation: https://developer.apple.com/documentation/realitykit/transforming-realitykit-entities-with-gestures
7. 3-Sweep: Extracting Editable Objects from a Single Photo, ACM Transactions on Graphics, 32(6), November 2013.
   DOI: 10.1145/2508363.2508378
8. Scanning Objects Using Object Capture – Apple Developer Documentation: https://developer.apple.com/documentation/realitykit/scanning-objects-us
9. SiriKit – Apple Developer Documentation: https://developer.apple.com/documentation/sirikit/

APPENDIX

Annotation Accuracy Test Setup

Figure A.1 shows the experimental setup for the annotation accuracy test. Numbered annotation points were placed on the model, and users attempted to locate specific annotations using different input methods (Siri, virtual keyboard, and text-to-speech).

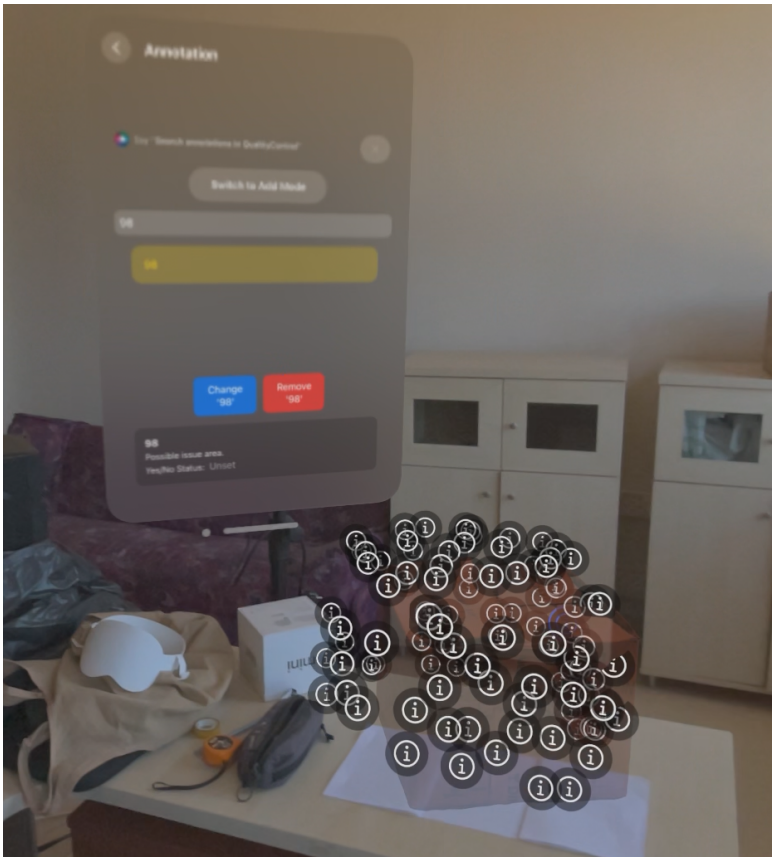


Figure A.1: Setup for annotation accuracy test: 100 annotations on a model with search performed using Siri, keyboard, and text-to-speech.