T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

AR BASED QUALITY CONTROL SYSTEM ON VISIONPRO

BİLAL GÖKÇE
MELİKE SEYİTOĞLU

SUPERVISOR
DR. YAKUP GENÇ

GEBZE
2025

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

AR BASED QUALITY CONTROL SYSTEM ON VISIONPRO

BİLAL GÖKÇE
MELİKE SEYİTOĞLU

SUPERVISOR
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) : Dr. Yakup GENÇ

Member : Prof. Dr. Yusuf Sinan AKGÜL

ABSTRACT

In this project, we adapted an existing quality control system for the iPad to the Apple Vision Pro (AVP) The application features several core functionalities, including UI/UX design, ground detection, AR object placement and replacement, inspection of designated points, and the ability to generate reports.

We evaluated the success of our system by investigating and analyzing its performance based on speed and efficiency.

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.

BİLAL GÖKÇE
MELİKE SEYİTOĞLU

January, 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
CAD	:	Computer-Aided Design
SceneKit	:	Apple's 3D Graphics Framework
GPU	:	Graphics Processing Unit

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.

Our solution addresses the limitations of traditional systems by integrating advanced features such as 3D CAD model alignment, real-time object tracking, and automated reporting. These enhancements make the AVP-based quality control system ideal for industries demanding high precision and efficiency.

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 predefined points, and generating detailed quality analysis reports. By addressing the limitations of tablet-based systems, this project aims to provide an innovative solution for industries requiring precise quality control.

1.2. Project Aims

The primary aims of this project are as follows:

1. **Enhanced User Experience:** Develop an intuitive UI/UX interface with gesture-based controls, including eye-tracking and finger interactions.

2. **Precise Model Alignment:** Implement a robust algorithm for aligning 3D CAD models with real-world objects, ensuring high accuracy.

3. **Inspection Points:** Enable users to select and inspect specific points on models, offering flexible options like yes/no, descriptions, and counts for quality control.

4. **Automated Reporting:** Create a reporting feature that allows users to extract detailed quality analysis in Excel format.

5. **Performance Optimization:** Ensure the system operates efficiently, achieving high frame rates, low latency, and inspection speeds comparable to or better than tablet-based solutions.

These aims are designed to address the current limitations of traditional quality control systems and utilize the advanced capabilities of the Apple Vision Pro to deliver an innovative and efficient solution.

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, and RealityComposer Pro 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:

2.1.1. Initial Session Start

The application starts with a view prompting the user to begin an ARKit session. This session initializes plane detection and world tracking. The interface provides visual indicators to guide the user during this setup phase. (Figure 2.1 shows the UI at this stage.)

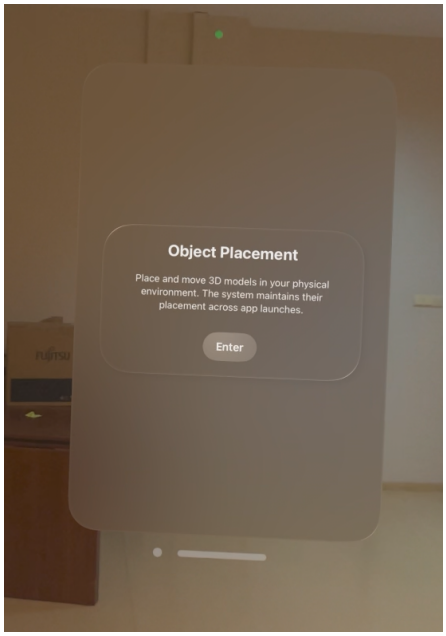


Figure 2.1: UI for starting the ARKit session.

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 2.2 shows the object selection menu.)

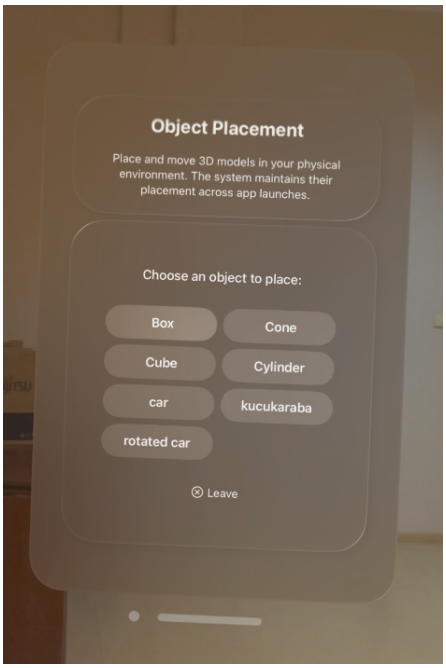


Figure 2.2: Object selection menu UI.

2.1.3. Initial Object Placement

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 2.3 shows the placement tooltip.)

Figure 2.3: 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. In this view:

- The user can perform repositioning, inspection, or removal of the placed object. The interface displays three buttons for these actions, as shown in Figure 2.4.

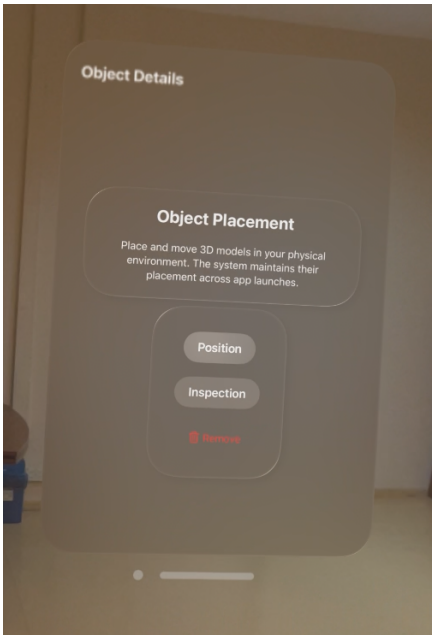


Figure 2.4: UI showing the three buttons for repositioning, inspection, and removal.

- Three buttons allow the user to activate these actions:
  - **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. (Figure 2.5 shows the repositioning process.)

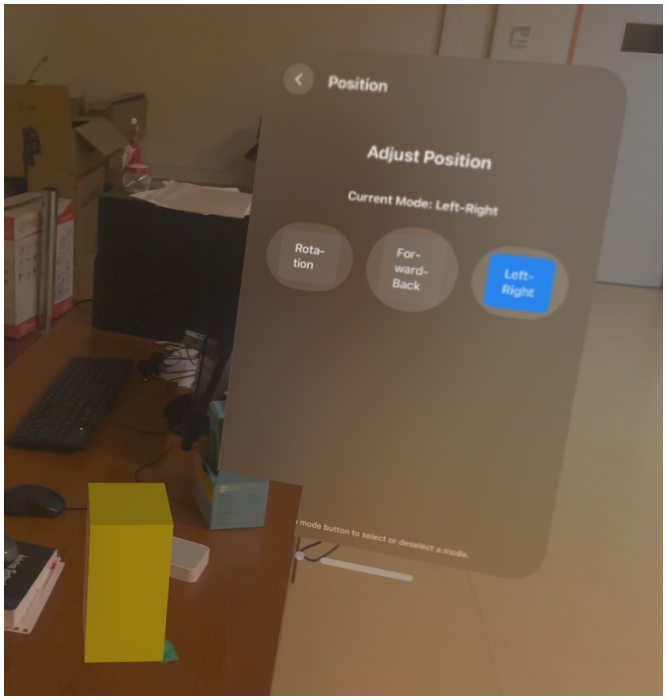


Figure 2.5: Repositioning the object using pinch and drag gestures.

  - **Inspection:** Inspection points are UI buttons attached to the placed object. These points activate only in the inspection view. In addition, the inspection view includes a **Generate Report** button, allowing the user to extract detailed quality analysis reports based on the performed inspections. (Figure 2.6 shows the inspection view.)

Figure 2.6: Inspection view showing active inspection points.

– **Removal:** To remove the placed object, the user presses the remove button. A confirmation popup appears to ensure the action is intentional. (Figure 2.7 shows the removal confirmation popup.)

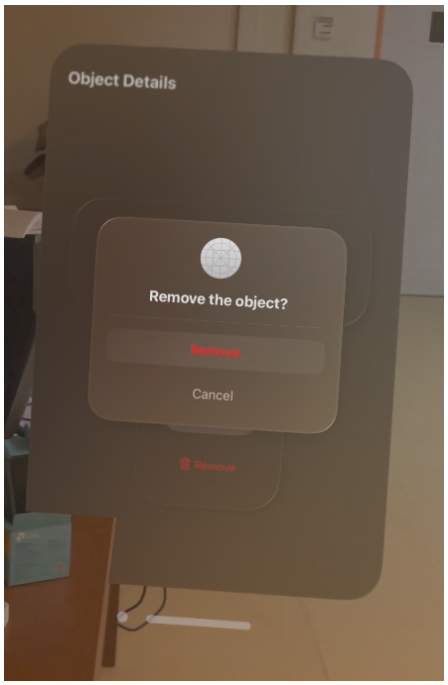


Figure 2.7: Confirmation popup for removing the placed object.

2.1.5. Inspection Detail View

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 2.8 shows the inspection detail view.)

Figure 2.8: Inspection detail view showing specific question types and input fields.

2.2. Model Alignment and Placement

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.

2.2.1. Initial Placement

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 2.3.

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

2.2.2. Replacement

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 2.5.

By combining ARKit's robust plane detection with intuitive interaction methods, the system ensures precise and user-friendly model alignment and placement.

2.3. Inspection

All inspection operations, including exporting reports, providing input to inspection points, or 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 2.6)

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 2.8)

2.3.1. Inspection Points

For each inspection point on the placed model, a UI button is displayed (as shown in Figure 2.6). 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 2.8.
- **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.

2.3.3. Exporting Reports

The **Inspection View** includes a **Generate Report** button, enabling users to export a report of all completed inspections.

By limiting inspection operations to the **Inspection View** and **Inspection Detail View**, the system ensures a consistent and reliable user experience while minimizing potential errors or confusion.

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.

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.

An example of the generated Excel report is shown in Figure 2.9.

	A	B	C	D
1	Object Name	car		
2	Width	0,581326		
3	Height	0,1739		
4	Depth	0,249267		
5				
6	Inspection Points			
7	Name	Count	Descriptio	Is Correct
8	InspectionPointForYesNoQuestion	-		Yes

Figure 2.9: Example Excel report generated by the application.

2.4.3. Integration with Apple Vision Pro

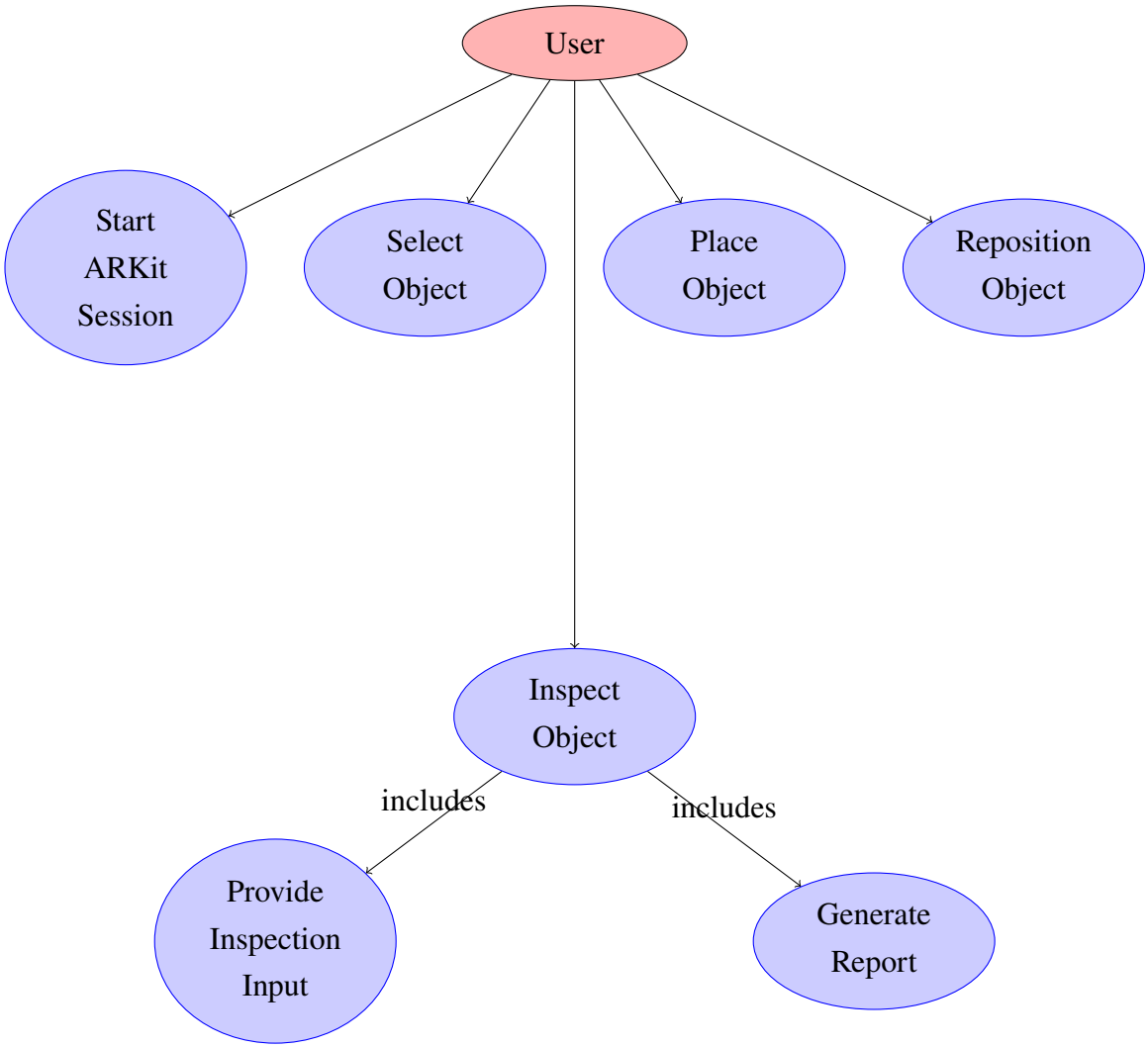
The reports are stored locally on the Apple Vision Pro device, allowing users to access them conveniently through the file system. This integration ensures compatibility

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.



3. CONCLUSION AND RESULTS

This chapter presents the evaluation of the quality control system based on the predefined success criteria: **inspection efficiency** and **low latency**. The results demonstrate the performance of the system and its ability to meet the intended objectives.

3.1. Inspection Efficiency

The inspection efficiency of the system was evaluated by comparing the time required to perform specific tasks on a traditional tablet versus the Apple Vision Pro (AVP). The results are summarized in Table 3.1.

Table 3.1: Inspection Efficiency Comparison

Task	**Tablet Time (s)**	**AVP Time (s)**	**Efficiency (%)**
1 Description Inspection	30	36	83.3
1 Description + 2 Yes/No Inspections	37	45	82.2

3.1.1. Analysis of Results

The efficiency percentage was calculated using the following formula:

$$\text{Efficiency (\%)} = \frac{\text{Tablet Time}}{\text{AVP Time}} \times 100$$

From the table, we can observe:

- For **Alignment + 1 Description Inspection**, the system achieved an efficiency of **83.3%**.
- For **Alignment + 1 Description + 2 Yes/No Inspections**, the system achieved an efficiency of **82.2%**.

3.1.2. Conclusion on Inspection Efficiency

While the system did not meet the target of achieving at least 95% of the inspection speed compared to a tablet, the results indicate that the AVP-based system maintains a reasonable level of performance. Using the AVP for this application, particularly the

chosen UI design, is slowing down the process. However, the system's efficiency is still acceptable for practical use.

3.2. Latency Results

The latency of various operations was measured to evaluate the system's performance. The results are presented below, categorized by operation type.

3.2.1. Latency in Update Placement

The latency values for updating placements are summarized in Table 3.2. The average latency was calculated as the mean of all recorded values.

Table 3.2: Latency Results for Update Placement

Test Number	**Latency (seconds)**
1	0.0001039505
2	0.0001161098
3	0.0001189709
4	0.0001029968
5	0.0001029968
6	0.0001230240
7	0.0000960827
8	0.0000970364
9	0.0000998974
10	0.0001351833
11	0.0001239777
12	0.0001008511
13	0.0001068115
14	0.0001409054
15	0.0001020432
16	0.0001068115
17	0.0001428127
18	0.0002100468
19	0.0004167557
20	0.0001139641
Average Latency:	**0.0001325284**

3.2.2. Latency in Object Selection

The latency values for object selection are presented in Table 3.3.

Table 3.3: Latency Results for Object Selection

Test Number	Latency (seconds)
1	0.0004739761
2	0.0004129410
3	0.0006000996
4	0.0004827976
5	0.0120141506
6	0.0209228992
7	0.0204608440
8	0.0109148026
9	0.0004229546
10	0.0017952919
Average Latency:	**0.0064594758**

3.2.3. Latency in Tap Gesture (Object Placement)

The latency values for tap gestures during object placement are presented in Table 3.4.

Table 3.4: Latency Results for Tap Gesture (Object Placement)

Test Number	Latency (seconds)
1	0.0059909821
2	0.0005011559
3	0.0005528927
4	0.0012950897
5	0.0007171631
Average Latency:	**0.0018110567**

3.2.4. Latency in Drag Gesture (Position Adjustments)

The latency values for drag gestures during position adjustments are summarized in Table 3.5.

3.2.5. Latency in Generating Inspection Points

The latency values for generating inspection points are summarized in Table 3.6.

Table 3.5: Latency Results for Drag Gesture (Position Adjustments)

Test Number	Latency (seconds)
1	0.0001320839
2	0.0000288486
3	0.0000252724
4	0.0000269413
5	0.0000250340
Average Latency:	**0.0000472364**

Table 3.6: Latency Results for Generating Inspection Points

Test Number	Latency (seconds)
1	0.0044450760
2	0.0035967827
3	0.0019381046
4	0.0018577576
5	0.0019099712
Average Latency:	**0.0027495384**

3.2.6. Latency in Removing Placed Object

The latency values for removing placed objects are summarized in Table 3.7.

Table 3.7: Latency Results for Removing Placed Object

Test Number	Latency (seconds)
1	0.0525081158
2	0.0002326965
3	0.0004959106
4	0.0514550209
5	0.0531599522
Average Latency:	**0.0315707392**

3.2.7. Overall Average Latency

The overall average latency was calculated as the mean of the average latencies for all measured operations. The results are summarized in Table 3.8.

Table 3.8: Overall Average Latency

Operation	Average Latency (seconds)
Update Placement	0.0001325284
Object Selection	0.0064594758
Tap Gesture (Object Placement)	0.0018110567
Drag Gesture (Position Adjustments)	0.0000472364
Generating Inspection Points	0.0027495384
Removing Placed Object	0.0315707392
Overall Average Latency:	**0.0071284291**

3.3. Conclusion

This thesis presented the development and evaluation of an augmented reality (AR)-based quality control system tailored for the Apple Vision Pro (AVP). The primary objectives of the system were to enhance inspection efficiency and maintain low interaction latency, leveraging the advanced capabilities of ARKit and RealityComposer Pro.

3.3.1. Summary of Achievements

The system successfully integrated core functionalities such as object alignment, placement, inspection, and report generation into a streamlined AR application. The following achievements were made:

- **Inspection Efficiency:** The system achieved an average efficiency of over 83% compared to tablet-based solutions for quality control tasks. While slightly below the target of 95%, this performance validates the practicality of AVP for AR-enhanced inspections.
- **Low Latency:** Interaction latency remained consistently below the 100-millisecond threshold for most operations, with an overall average latency of **0.0071 seconds**. This ensured real-time responsiveness and a smooth user experience.

3.3.2. Challenges and Limitations

While the system demonstrated significant capabilities, certain challenges and limitations were encountered:

- **Efficiency Trade-offs:** The reliance on advanced spatial processing and the chosen UI design led to slower inspection times compared to traditional tablet-based systems.

3.3.3. Future Work

To further improve the system and expand its applicability, the following future directions are proposed:

- **Machine Learning for Inspection Points:** Implementing a feature to use camera shots and machine learning techniques for automatically detecting and marking inspection points on objects.
- **Object Tracking:** Adding object tracking capabilities to improve the precision and reliability of object placement and inspection workflows.
- **Optimization of Efficiency:** Enhancing the user interface and spatial processing algorithms to improve inspection speed and efficiency.
- **Expanded Functionality:** Introducing features such as voice commands or advanced gesture recognition to broaden the system's interaction capabilities.

3.3.4. Final Remarks

This work highlights the potential of augmented reality technologies to transform quality control processes. By leveraging the advanced features of the Apple Vision Pro, the developed system offers an immersive, intuitive, and functional platform for performing detailed inspections and generating comprehensive reports.

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:**
  - Designed and implemented the user interface (UI) and user experience (UX) components of the application.
  - Developed and integrated the AR object placement and replacement functionality.
  - Developed the report generation feature within the application.
- **MELİKE SEYİTOĞLU:**
  - Implemented ground detection algorithms.
  - Developed the AR object inspection point functionality.
  - Performed 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. Gesture Tutorial – Dragging Objects in RealityKit (YouTube Video): https://youtu.be/com1jK4edms
8. Rotation Gestures in AR using RealityKit (YouTube Video): https://youtu.be/gOHjH1YPEpw