Project Acronym:	**BIMERR**
Project Full Title:	**BIM-based holistic tools for Energy-driven Renovation of existing Residences**
Grant Agreement:	**820621**
Project Duration:	**45 months**

DELIVERABLE D5.10
AI-enabled tools (hardware & software) for in-situ digital building model annotation via smart-glasses 2

Deliverable Status:	**Final**
File Name:	**BIMERR_D5.10-v1.00.docx**
Due Date:	**30/06/2021 (M30)**
Submission Date:	**30/06/2021(M30)**
Task Leader:	**CERTH (T5.6)**

Dissemination level	
Public	X
Confidential, only for members of the Consortium (including the Commission Services)	

This project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°820621

The BIMERR project consortium is composed of:		
FIT	Fraunhofer Gesellschaft Zur Foerderung Der Angewandten Forschung E.V.	Germany
CERTH	Ethniko Kentro Erevnas Kai Technologikis Anaptyxis	Greece
UPM	Universidad Politecnica De Madrid	Spain
UBITECH	Ubitech Limited	Cyprus
SUITE5	Suite5 Data Intelligence Solutions Limited	Cyprus
HYPERTECH	Hypertech (Chaipertek) Anonymos Viomichaniki Emporiki Etaireia Pliroforikis Kai Neon Technologion	Greece
MERIT	Merit Consulting House Sprl	Belgium
XYLEM	Xylem Science And Technology Management Gmbh	Austria
CONKAT	Anonymos Etaireia Kataskevon Technikon Ergon, Emporikon Viomichanikonkai Nautiliakon Epicheiriseon Kon'kat	Greece
BOC	Boc Asset Management Gmbh	Austria
BX	Budimex Sa	Poland
UOP	University Of Peloponnese	Greece
UEDIN	University Of Edinburgh	United Kingdom
UCL	University College London	United Kingdom
NT	Novitech As	Slovakia
FER	Ferrovial Agroman S.A	Spain

Disclaimer

BIMERR project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°820621. The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the European Commission (EC). EC is not liable for any use that may be made of the information contained therein.

AUTHORS LIST

Leading Author (Editor)			
Surname	**First Name**	**Beneficiary**	**Contact email**
Tsakiris	Athanasios	CERTH	atsakir@iti.gr

Co-authors (in alphabetic order)				
#	**Surname**	**First Name**	**Beneficiary**	**Contact email**
1	Andriopoulos	Athanasios	CERTH	tandrio@iti.gr
2	Tsita	Anastasia	CERTH	a.tsita@iti.gr
3	Pantraki	Evangelia	CERTH	epantrak@iti.gr
4	Papadopoulos	Vasilios	CERTH	vpapadop@iti.gr

REVIEWERS LIST

List of Reviewers (in alphabetic order)				
#	**Surname**	**First Name**	**Beneficiary**	**Contact email**
1	Falcioni	Damiano	BOC	damiano.falcioni@boc-eu.com
2	Varga	Jan	NT	varga@novitechgroup.sk

TABLE OF CONTENTS

LIST OF FIGURES

ACRONYMS

Acronym	Meaning
AI	Artificial Intelligence
AR	Augmented Reality
AEC	Architecture Engineering Construction
API	Application Programming Interface
ARIBFA	Augmented Reality enabled In-situ Building Feature Annotation
BICA	Building Information Collection Application
BIF	BIMERR Interoperability Framework
BIM	Building Information Management
BIMERR	BIM-based holistic tools for Energy-driven Renovation of existing Residences
BISP	Building Information Secure Provisioning
FAQ	Frequently Asked Question
GUI	Graphical User Interface
H&S	Health and Safety
HMD	Head Mounted Display
HTTPS	HyperText Transfer Protocol Secure
HVAC	Heating, Ventilation, and Air Conditioning
IFC	Industry Foundation Class
IoT	Internet of things
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
MEP	Mechanical, Electrical and Plumbing
PWMA	Process & Workflow Modelling & Automation
RAD	Rapid Application Development
REST	Representational State Transfer
UC	Use Case
WCAG	Web Content Accessibility Guidelines

EXECUTIVE SUMMARY

This deliverable focuses on the updated design, implementation, and integration of the BIMERR ARIBFA toolset which caters for the in-situ Augmented Reality (AR) Building Information Management (BIM) Visualisation and Annotation during a renovation project.

The ARIBFA toolset is part of the BIMERR Digital Building Modelling Tools system, which aims to support the role of workers and building surveyors during the renovation process by providing recognition and registration of features in a building, while displaying information to the user through the Augmented Reality Head Mounted Display (AR HMD).

The ARIBFA toolset accepts as input the BIM model in Industry Foundation Class (IFC) format, which is a standardized, digital description of the building geometry and components, and reports user-added annotations and detected building components to BIMERR Interoperability Framework (BIF). The following five objectives cover the activities and submodules that comprise the solution as a whole and are the scope of this deliverable.

1. **BIM 3D Model Visualisation submodule**: This submodule acts as a BIM viewer on the AR HMD. Updates were done on the previous version of the submodule, which was presented in Deliverable D5.9, to enable the editing of the IFC File.

2. **BIM 3D Model Registration and Tracking submodule**: This component registers (i.e., aligns) the 3D BIM model to the real world. Similar to the work presented in D5.9, registration is performed by means of image target detection. Although the registration pipeline remains the same, the implementation of this submodule was updated regarding the scene setup, the image targe positioning on scene, and by targeting the latest Unity's SDK.

3. **Indoor Localization submodule**: This component keeps track of the registered 3D BIM model and localizes the user constantly into it during a renovation process. As presented in Deliverable D5.9, after the initial registration, the aligned 3D BIM model is set as a persistent spatial anchor that the application should track during runtime and should load on start-up. In the updated version presented in this deliverable, the user can be actively informed for their current position in the BIM model and be notified for the IfcSpace they are currently in.

4. **Marker-less Feature Recognition submodule**: This component provides the capability of real-time MEP component detection on the application running on the AR HMD (Hololens). As mentioned in Deliverable D5.9, a key process for this submodule is to pair the Hololens to a local computer, which a) receives the camera stream from Hololens, b) performs MEP component detection on the received frames, and c) promotes detection results to Hololens to be visualised in the 3D space. In this deliverable, a desktop application, namely DesktopARIBFA, was implemented to accompany ARIBFA and facilitate the communication of the local desktop with Hololens. In this version of ARIBFA, the detected MEP components are visualised using 3D bounding boxes, whose position, scale, and rotation can be optimized by the user using appropriate handlers. In addition, the detected components can be added to the IFC file, along with the properties that the user enters using the newly created menus responsible for adding properties to the detected objects.

5. **AR Annotation & Context Aware-Visualisation submodule**: This module is responsible to support the attachment of AR annotations to building components and the visualisation of the current renovation process tasks. In this deliverable, improvements were made on the AR annotation capabilities by allowing the attachment of image/video files to the annotation, as well as updates to the menus to reflect the finalized data models leading to a more intuitive user experience for visualising annotations and renovation tasks. Finally, the appropriate menus for handling error reports regarding the IFC validation process were introduced.

Moreover, this deliverable serves as the final update to ARIBFA's overall architecture and workflow. The connections to and from the other applications and tools in the BIMERR project are outlined. The enhanced technical specifications of each submodule, the updated API, and hardware that were used are clearly defined. Every component's structure and usage are analyzed along with how the component relates to the BIMERR's framework challenges and end goal. Finally, a user manual is included containing all essential information to help the user get the most out of the ARIBFA application.

1 INTRODUCTION

ARIBFA is one of the crucial parts of the BIMERR project as it is highly associated with the end-user. The main ability of ARIBFA is to facilitate the surveying process by providing AR-powered capabilities, such as detection and annotation of building components, as well as issue reporting on objects of interest (MEP plans, building zones) during the planning and renovation stages of construction.

In this deliverable, the main objectives and the relevant use cases ARIBFA is going to encounter are described in detail. In addition, a complete and thorough overview of the updated design and architecture of the components and submodules of the application is provided. From design to implementation, every development stage is presented in the next chapters. All subsystems that facilitate the handling of building data, object detection, and user data input are described in the following chapters:

- **Connection to Other Deliverables** (Section 1.2): Relevant tools and deliverables from the rest of the BIMERR ecosystem.
- **Relevant User Requirements and Use Cases** (Section 2)***:*** Collection of use cases for the ARIBFA application.
- **System Design and Architecture** (Section 3): Brief overview of the submodules described in this deliverable.
- **Software Design and Development** (Section 4): Thorough analysis of each submodule's updated design and revised implementation stages.
- **API Documentation** (Section 5): Outline of all the processes regarding data exchange with 3rd parties.
- **Deployment Plan** (Section 6): Overview of the applications installation and initialization process.
- **User Manual** (Section 7): Analysis of the various elements of the application from the user's perspective.
- **Conclusion and Outlook** (Section 8): Closing remarks about the impact of the deliverable.

1.1 CHANGES TO THE FIRST VERSION OF THE ARIBFA TOOL

In the Deliverable 5.9, submitted in M23, the first version of the ARIBFA tool was described. In this section, the functionalities added (or modifications made) to the initial version of the tool are summarized:

1. **BIM 3D Model Visualisation submodule**: Updates were done on the previous version of the submodule to enable the editing of the IFC File.

2. **BIM 3D Model Registration and Tracking submodule**: Similar to the work presented in D5.9, registration is performed by means of image target detection. Although the registration pipeline remains the same, the implementation of this submodule was updated regarding the scene setup, the image targe positioning on scene, and by targeting the latest Unity's SDK.

3. **Indoor Localization submodule**: As presented in Deliverable D5.9, after the initial registration, the aligned 3D BIM model is set as a persistent spatial anchor that the application should track during runtime and should load on start-up. In the updated version presented in this deliverable, the user can be actively informed for their current position in the BIM model and be notified for the IfcSpace they are currently in.

4. **Marker-less Feature Recognition submodule**: As mentioned in Deliverable D5.9, a key process for this submodule is to pair the Hololens to a local computer, which a) receives the camera stream from Hololens, b) performs MEP component detection on the received frames, and c) promotes detection results to Hololens to be visualised in the 3D space. In this deliverable, a desktop application, namely DesktopARIBFA, was implemented to accompany ARIBFA and facilitate the communication of the local desktop with Hololens. In this version of ARIBFA, the detected MEP components are visualised using 3D bounding boxes, whose position, scale, and rotation can be optimized by the user using appropriate handlers. In addition, the detected components can be added to the IFC file, along with the properties that the user enters using the newly created menus responsible for adding properties to the detected objects.

5. **AR Annotation & Context Aware-Visualisation submodule**: In this deliverable, improvements were made on the AR annotation capabilities by allowing the attachment of

image/video files to the annotation, as well as updates to the menus to reflect the finalized data models leading to a more intuitive user experience for visualising annotations and renovation tasks. Finally, the appropriate menus for handling error reports regarding the IFC validation process were introduced.

Since this deliverable serves as the final update to ARIBFA's overall architecture and workflow, the connections to and from the other applications and tools in the BIMERR project are clearly and explicitly outlined. Compared to the D5.9, this deliverable includes enhanced technical specifications for each submodule, detailed information regarding the updated API, and a complete guide for the hardware that is used. Finally, this deliverable includes a user manual, containing essential information to help the user get the most out of the ARIBFA application.

1.2 CONNECTION TO OTHER DELIVERABLES

This deliverable reports the second version (v2) of the AI-enabled tools (hardware & software) for in-situ digital building model annotation via smart-glasses. The final version of the ARIBFA tool is a part of this deliverable. The first version (v1) of ARIBFA was delivered at M23 and documented in Deliverable D5.9.

Relation to other BIMERR Tools and Deliverables

The ARIBFA tool is connected to other tools and deliverables as follows:

- the functionalities and uses cases of the ARIBFA tool are described in the BIMERR system Architecture (WP3; Deliverables D3.5 and D3.6).
- ARIBFA provides the capability to complete/correct the BIM model, which is provided by the Scan-to-BIM tool (WP5; Deliverables D5.3 and D5.4), on site.
- output annotation data of the ARIBFA tool will be stored in the BIMERR Data Stores and pulled/pushed using the BIF (WP4; Deliverables D4.8 and D4.9).

- the ARIBFA tool, similarly to all components connected to BIF, relies on a shared ontology and a common data model (WP4; Deliverables D4.2 and D4.3).

2 RELEVANT USER REQUIREMENTS AND USE CASES

The User Requirements are derived from the Deliverables D3.5 and D3.6, specify what users expect from the system in terms of its functionalities, and set the guidelines for the design and development of the ARIBFA application. The functional requirements define the functionalities of ARIBFA application and all its submodules. These functionalities define the basic system behavior, as expected by its users. The non-functional requirements enhance the ARIBFA application and provide a user-friendly environment.

Functional Requirements

- Building components recognition.
- Mapping of the 3D BIM model to the site.
- Tracking every component that has been registered.
- Determination of user's position and orientation.
- Ability to locate every recognized characteristic of the building.
- Information provision for every mapped component of the building.
- Ability to update information data of every component.
- Registration and annotation of unrecognized or unregistered building components.
- Visualisation of tasks assigned to building components.
- Ability to notify the user for building components with missing IFC properties and to accept values for missing properties

Non-Functional Requirements

- Friendly user interface.
- Wi-Fi connectivity.
- Information extraction from 3D BIM models.
- Ability to support multiple languages.

- A menu to discern its different functionalities.
- Ability to receive data from sensors network, cameras, AR HMD's sensors, and BIF.

The ARIBFA application recognizes, and registers features present in a building, using AR HMD, i.e., AR glasses, to enable the user wearing the glasses to walk through a building and (semi-)automatically annotate building characteristics on a digital building model (enhanced BIM), report issues, e.g., on structural damage, and enrich the BIM model by updating it with the registration of unrecognized or unregistered building components. The user of the ARIBFA application is a site manager, a building surveyor, or an experienced worker/foreman. The ARIBFA use cases are outlined in the Table 1, in which the associate user is involved, within the scope of the BIMERR project.

Table 1: ARIBFA use cases.

BIMERR Use Case 2	
UC-02	Accelerate the collection of data about the building systems through BIM-based internal audit support tools and interaction with building managers and occupants.
Surveyor	The building surveyor, before the renovation phase, walks through the building and annotates the BIM model with energy related equipment and other related building materials via the ARIBFA app.
BIMERR Use Case 8	
UC-08	Daily renovation activity schedules are automatically generated (based on accurate project scheduling) and individual guidelines are provided to the workforce responsible through ambient interfaces and apps.
Worker/ Foreman	Working crews are automatically notified about tasks assigned to them while new tasks are accompanied with all needed information (instructions, drawings, etc.) that can be superimposed on real world in AR.
BIMERR Use Case 9	
UC-09	Continuous monitoring and updates of renovation activity schedules (based on reporting from the workforce and monitoring of process execution) towards effective devising and avoidance of delays (bi-directional communication through ambient interfaces).

Site Manager	A site manager who uses their AR equipment and follows the construction works progress that is displayed on BIM model as they walk through the site.
BIMERR Use Case 10	
UC-10	Continuous reporting from workforce and occupants for changes performed over the initial renovation design (location-based on a BIM representation) and automated update of the BIM model (as-built documentation).
Surveyor	During the building's renovation phase, when the workers notice small deviations from the as-designed renovation actions, they can report them, along with the proposed changes that they need to perform, back to the workflow manager via the ARIBFA application.
BIMERR Use Case 10	
UC-10	Continuous reporting from workforce and occupants for changes performed over the initial renovation design (location-based on a BIM representation) and automated update of the BIM model (as-built documentation).
Surveyor	During the building's renovation phase, when the workers notice small deviations from the as-designed renovation actions, they can report them, along with the proposed changes that they need to perform, back to the workflow manager via the ARIBFA application.
BIMERR Use Case 11	
UC-11	Identification of threats and dangers and provision of alerts to workforce and occupants through BIM-based apps and UIs.
Worker	Workers, through the ARIBFA app, receive Health & Safety (H&S) notifications that are: • attached to the workflow of the task that is assigned to them, • displayed when the worker is repositioned inside the building (e.g., changes floor) and before their day-shift.
BIMERR Use Case 12	
UC-12	Continuous reporting from workforce and occupants for dangers and threats (location-based on a BIM representation) and automated update of the BIM model.
Worker	Through the ARIBFA app, workers create H&S issues that are automatically highlighted in the BIM model. The site manager, who can also create H&S issues, automatically creates periodically H&S reports using the history log of registered issues and their status. The status of H&S issues is changed by the H&S manager of the construction site.

3 SYSTEM DESIGN AND ARCHITECTURE

The ARIBFA application is organized into five submodules:

- **BIM 3D Model Visualisation submodule** (in Section 4.2): This component acts as a BIM viewer. It takes as input a BIM model of the IFC4 file format generated from the BIF platform and outputs the 3D BIM model to be used with the AR glasses.
- **BIM 3D Model Registration and Tracking submodule** (in Section 4.3): This component registers (i.e., aligns) the 3D BIM model to the real world. Registration is achieved by means of image target detection and feature matching between the images streamed from the AR HMD's camera and the extracted features of the image target. The aligned 3D BIM model starts to be tracked immediately after registration and is continuously tracked by the Indoor Localization submodule.
- **Indoor Localization submodule** (in Section 4.4): This component keeps track of the registered 3D BIM model using spatial anchors and localizes the user constantly into it.
- **Marker-less Feature Recognition submodule** (in Section 4.5): This component provides the capability of real-time object detection on the application running on the AR HMD. Images from the AR HMD's camera are streamed to a paired computer and are evaluated for detection of building MEP components, such as plugs, switches, and HVAC components. Detected objects can be added to the IFC, along with their IFC properties provided by the user using appropriate GUI.
- **AR Annotation & Context Aware-Visualisation submodule** (in Section 4.6): This component provides the user wearing the AR HMD with the ability to gaze and select a building component of interest in their viewpoint as well as visualise data downloaded from BIF. In addition, it provides interactive menus that contain the selected building component's properties. The user has the capability to edit existing properties or add text annotations on selected 3D building components. The user is also notified for the tasks assigned to building components and for building components with missing IFC properties. All new and edited data is bundled and sent to the BIF platform.

The architecture of the submodules contributing to the ARIBFA application is presented in Figure 1.

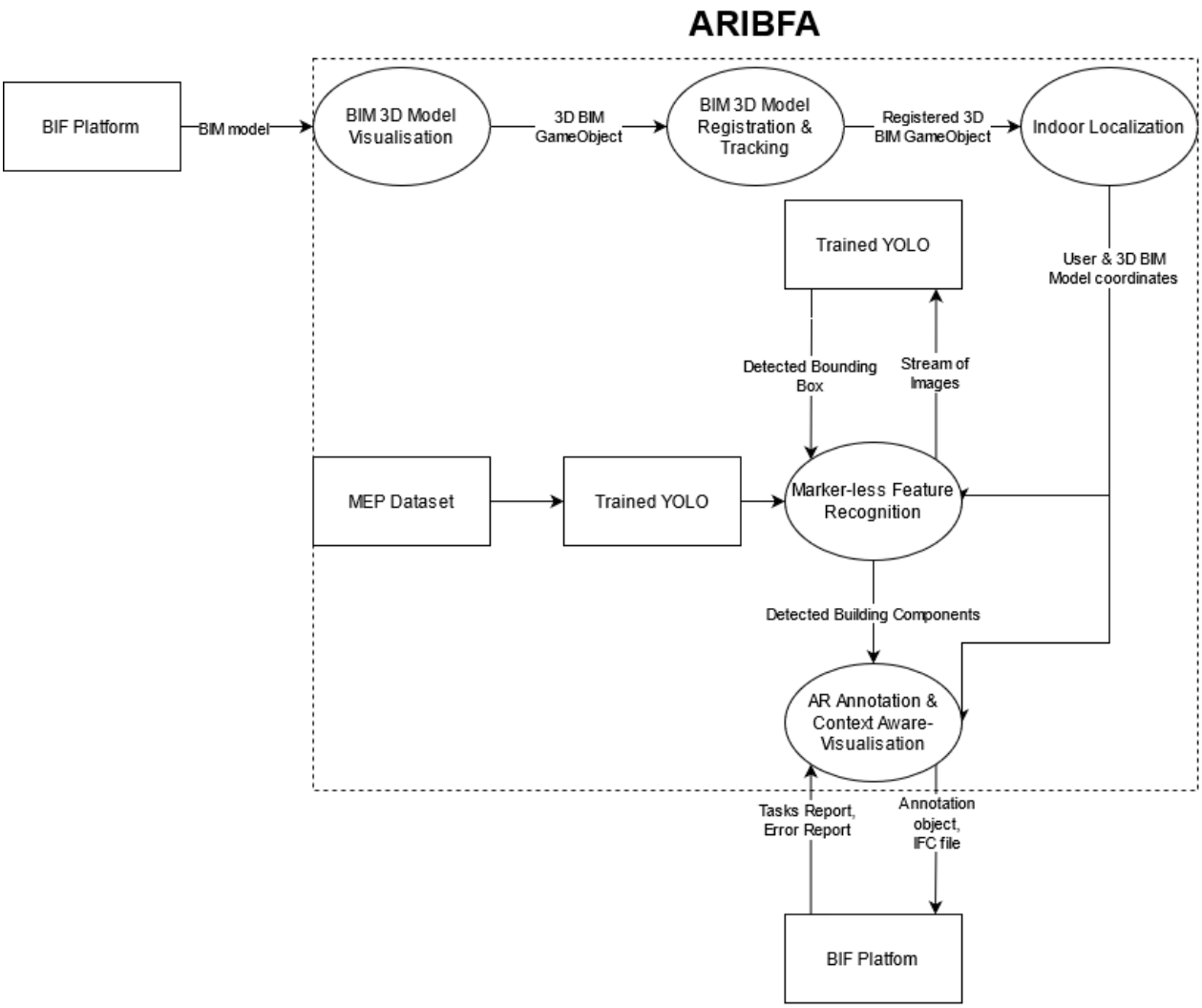


Figure 1: Overview of the submodules in the ARIBFA application.

4 SOFTWARE DESIGN AND DEVELOPMENT

4.1 DEVELOPMENT METHODOLOGY

The main characteristic of ARIBFA is the fact that its design and development must follow the user's needs for an intuitive, informative, and pleasant experience while navigating through the augmented 3D space using the HMD. Thus, fast iterations and continuous testing of newly implemented features are necessary.

Based on the above criteria, the development team decided to follow closely the principles of the Rapid Application Development (RAD) model [1], which puts rapid prototyping and rigorous feedback cycles at the center stage of the development process. With these two key methods, developers have the ability to design and develop multiple iterations of the same module without needing to start the development schedule from scratch each time a new feature is needed.

A brief overview of the key benefits of RAD [2]:

- Flexibility and adaptability as developers can make adjustments on the fly.
- Quick iterations that reduce development time and speed up delivery.
- Emphasis on code reuse leading to shorter testing times.
- Increased stakeholder satisfaction due to the continuous collaboration of developers, users, clients.

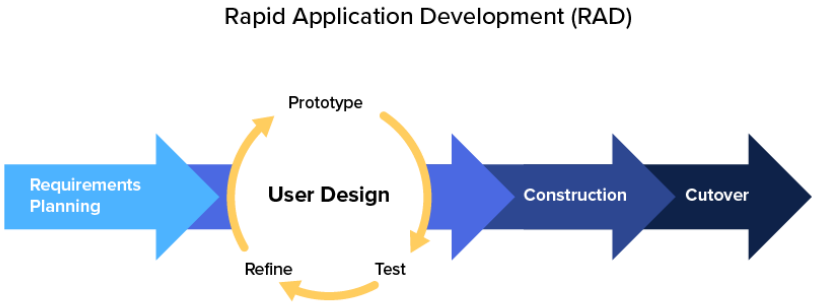


Figure 2: Rapid Development model in detail.

The development process using the RAD method was split according to the following phases:

1. **Requirements planning:** During this phase, developers, software users and the development team communicated and determined the goals and expectations for the final product as well as potential pitfalls that can halt the developing process.
2. **User Design:** During this phase, developers work hand in hand with the users to ensure the overall experience's needs are met at every step of the development process, with extensive testing done on each functional prototype of the module.
3. **Rapid Construction:** In this phase, the prototypes and the systems that were implemented as alpha or beta versions are converted into fully working processes. Because of the fast iterations in the previous stages the construction happens rapidly.
4. **Cutover:** This is the implementation and finalization stage where the finished module becomes ready to be consumed by the production branch of development and it includes the final testing iterations, as well as user training on the newly added functionalities in the application.

Lastly, the emphasis on the user experience and user feedback over strict planning and requirement cataloguing that the RAD principles dictate proved it the most flexible method.

4.1.1 Development Toolset

The AR HMD used in the ARIBFA application is the Microsoft Hololens AR glasses. The Hololens' application ecosystem is based around Microsoft's Mixed Reality Toolkit (MRTK), which enables access to the device's capabilities and sensors. Using the Unity3D engine and the C# programming language, developers can use the interfaces and abstractions which the MRTK provides to create AR-powered applications. Testing can be performed in the official Hololens emulator, while building and deploying the application to the physical Hololens device can be done from within Unity and Visual Studio.

Unity3D. Unity [3] is a Game Engine created by Unity Technologies. The engine is widely used to build 3D and 2D applications, simulations, and games as well as Virtual Reality (VR) and AR applications, such as the ARIBFA app. It is worthwhile to mention that Unity was also used to

develop the PWMA [4] (Process & Workflow Modelling & Automation app for residents) application, which was also led by CERTH, because of the AR SDKs it supports. It should be noted that the two applications are targeted to different devices, i.e., the ARIBFA app is being deployed on Hololens while the PWMA app is targeted on smartphones.

Programming Language C#. Programs in Unity can be written in the object-oriented programming language C#. C# enables the development of a variety of secure and robust applications that run in the .NET ecosystem. The scripts to implement the ARIBFA submodules in Sections 4.2, 4.3, 4.4, 4.5, and 4.6 are developed in C#.

Microsoft Hololens. The necessary hardware for the ARIBFA application is the smart glasses with all the necessary sensors and AR capabilities. The AR-HMD device specifications are described in Task T5.5 "Refinement of Augmented-Reality Smart Glasses for the ARIBFA app". Microsoft Hololens, depicted in Figure 3, are a pair of mixed reality smart glasses developed and manufactured by Microsoft [5]. Microsoft Hololens is the first fully self-contained holographic computer; the users can move freely, with no wires or external packs needed.

Figure 3: Microsoft Hololens smart glasses [5].

Mixed Reality Toolkit. Microsoft provides substantial support for both users and developers for Hololens. The Mixed Reality Toolkit (MRTK) [6] is a part of the Hololens device ecosystem and provides useful interfaces and implementations for creating AR-powered applications. Full documentation and support are provided for developing Mixed Reality applications in Windows [7].

Hololens Emulator. To facilitate the development of Hololens applications, Windows Mixed Reality SDK provides the Hololens Emulator [8]. The Hololens Emulator enables the testing of holographic applications on a computer without a physical Hololens. The human and environmental inputs that are usually read by Hololens sensors are simulated from keyboard, mouse, or Xbox controller [9]. Notably, applications running on the emulator do not need to be built from scratch to be deployed on the Hololens device. The Hololens emulator that is compatible to the Windows SDK of the Hololens used in the ARIBFA application (i.e., Windows SDK 10.0.17763) was used for testing the application during development. This way, the Hololens is not an integral part of the development procedure, but many developers can concurrently contribute to a project without unnecessary demands for oversupply of hardware devices. Nevertheless, the Hololens device was mandatory for testing the BIM 3D Model Registration and Tracking submodule of the ARIBFA app. It is necessary to wear the Hololens and wander to the physical location in the real world to test the registration capabilities of the submodule. On the other hand, development of other modules, such as the AR Annotation & Context Aware-Visualisation submodule, was greatly accelerated by the incorporation of the Hololens Emulator into the development procedure. The main window of Hololens Emulator is depicted in Figure 4.

Figure 4: Main window of Hololens Emulator running Hololens OS [9].

4.2 BIM 3D MODEL VISUALISATION SUBMODULE

4.2.1 Overview

The BIM 3D Model Visualisation submodule functions as a Unity BIM viewer and provides basic editing functionalities to the BIM file in IFC format. The Unity game engine does not provide the capability to generate the 3D geometry directly from the IFC file. To solve this problem, an OBJ file is generated from the IFC and is imported, along with the IFC, as input to the BIM 3D Model Visualisation subcomponent. For the viewer to be complete it needs the geometry representation of the BIM and the parsing of the IFC data.

Geometry Representation: The Geometry representation of the BIM model is achieved through the transformation of the IFC file to a file format that is supported by the Unity game engine. Such a format is the OBJ file format. Although the BIM 3D Model Visualisation subcomponent supports the IFC to OBJ transformation with the use of the IfcConvert tool (of the IfcOpenShell library), it was decided that the ARIBFA app will receive as input the OBJ along with the IFC file, due to the incapability of the IfcConvert tool to create an error free OBJ from the enhanced IFC produced by the BIM-MP platform.

Ifc Parser: Importing the IFC file into Unity, extracting the IFC data, and mapping those data to the 3D model are handled by custom C# classes based on the Xbim library [10].

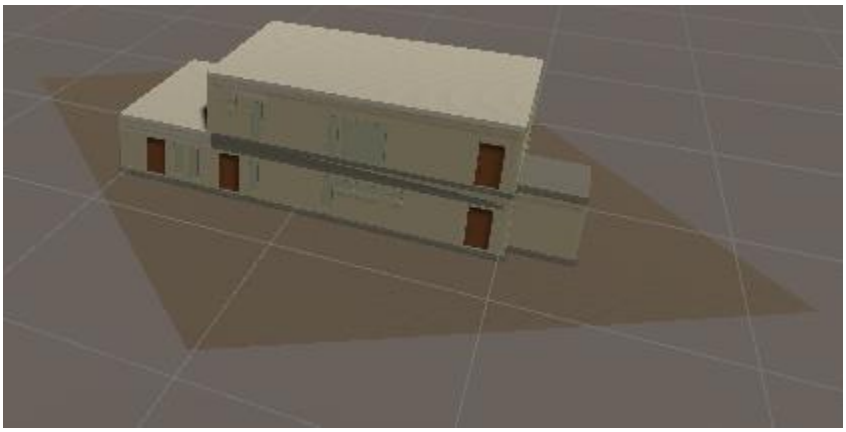


Figure 5: Visual representation of BIM model in Unity.

4.2.2 Submodule Architecture and Workflow

The parsing process of the IFC file takes place in the IfcParser submodule. This process is recursive and can be divided in three tasks, as shown in Figure 6.

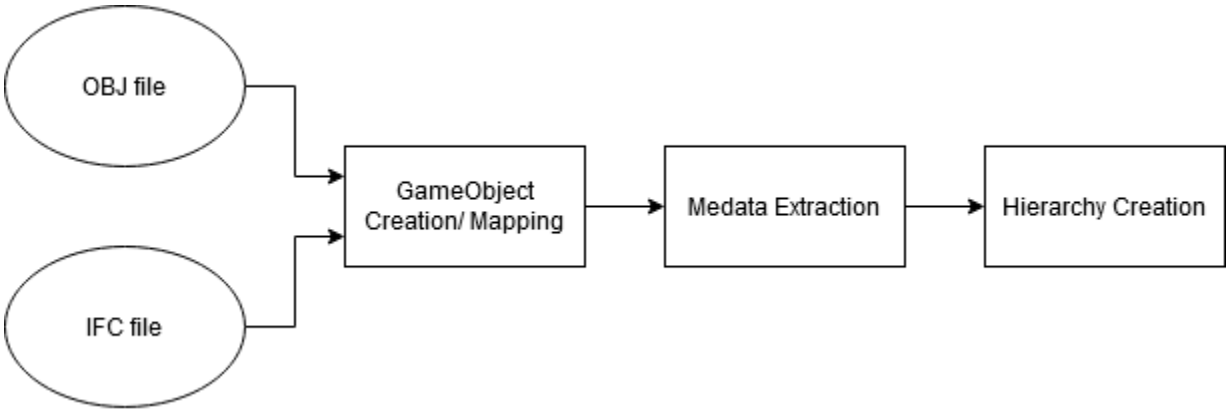


Figure 6: Process of getting IFC data into Unity.

- **GameObject Creation/Mapping:** In this task, the IfcParser reads IFC files, searches for the imported OBJ GameObjects based on the IfcGlobalId, and renames them with the name of the individual IfcElement. In the case of IFC elements as IfcProject or IfcSite that cannot generate geometry, the IfcParser creates empty GameObjects with the corresponding name. For example, the OBJ has no component for the IfcZone element, so an empty GameObject is created according to the imported IFC file, but an IfcDoor element exists in the OBJ file with the name of its Global Id and so the OBJ component is renamed accordingly.
- **Metadata Extraction:** In this task, the extraction of properties, materials quantities, and information of the IfcElement takes place. All the metadata of an individual IfcElement retrieved by the queries are stored in a custom C# class and are added as scripting components to the corresponding GameObject.
- **Hierarchy Creation:** In this stage, the IfcParser creates the hierarchy of the BIM model by setting parent-children relationships between the GameObjects. The following relationships are used:

- IfcRelAggregates that defines the Spatial Hierarchy between the IFC products,
- IfcRelAssignsToGroup that associates the IfcSpaces to the IfcZones.

Figure 7 shows the space and zone composition.

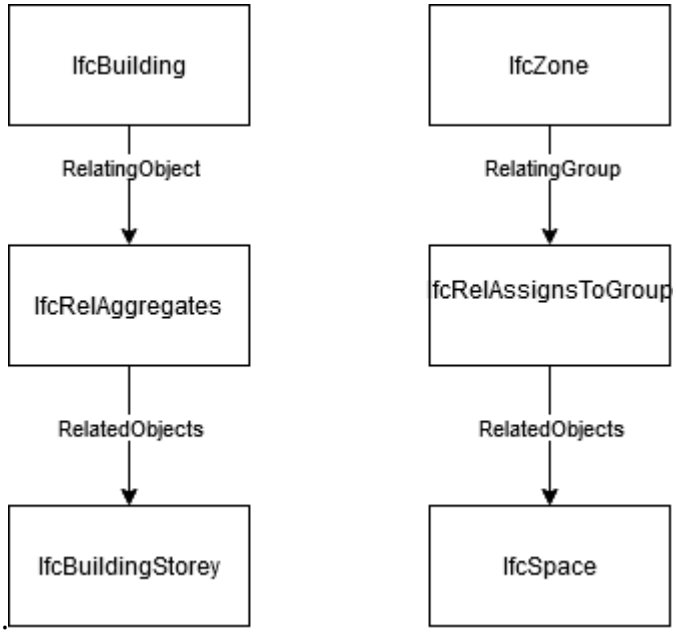


Figure 7: Space and Zone composition.

4.2.3 Development Tools

For the development of the BIM 3D Model Visualisation submodule, a set of tools and libraries were used to allow the functionality of the following tasks:

- Runtime importation of the 3D Object
- IFC Data Extraction, Hierarchy creation, IFC editing.

The first task is implemented by the Unity Asset OBJ runtime Importer, a free tool written in C# [11]. Then, the IFcParser is responsible for the second task with the use of the Xbim.Essentials Library. Table 2 summarizes the tools and libraries used for the implementation of this module. In Figure 8, the block diagram of the software components utilized in the development of the BIM 3D Visualisation submodule is presented.

Table 2: Tools and libraries used in the 3D BIM Model Visualisation subcomponent of the ARIBFA Tool.

Library	Version	Licence
OBJ Runtime Importer	2.02	*Extension Asset*
Xbim.Essentials	5.1.289	*CDDL*

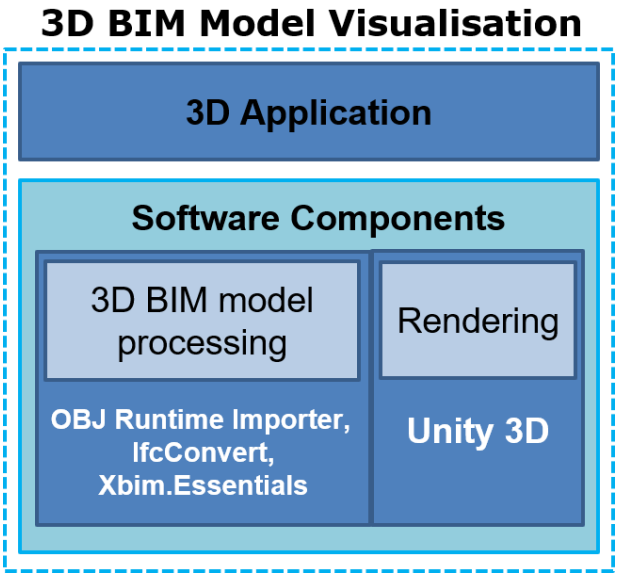


Figure 8: Block diagram of the software components used for the development of the BIM 3D Visualisation submodule.

4.2.4 Implementation Features

The fundamental objects that the Unity Game Engine uses are the GameObjects, thus the OBJ file containing the 3D representations are imported as GameObjects. The name of each GameObject contains a GlobalID according to the Ifc Global Id of the IFC file. The free Asset OBJ Runtime Importer is used to import the OBJ model into the Unity Scene as GameObject.

The connection between the geometry and the IFC information of the BIM model takes place in the IfcParser submodule. The parsing of the IFC and all the necessary data by the IfcParser, is achieved by using LINQ for Optimal Performance. LINQ [12] stands for a Language-Integrated Query and is a set of technologies based on the integration of query capabilities directly into the C# language. It implements deferred execution, which means it can chain

the query statements. Figure 9 shows how the visualisation submodule interacts with an IFC file through LINQ.

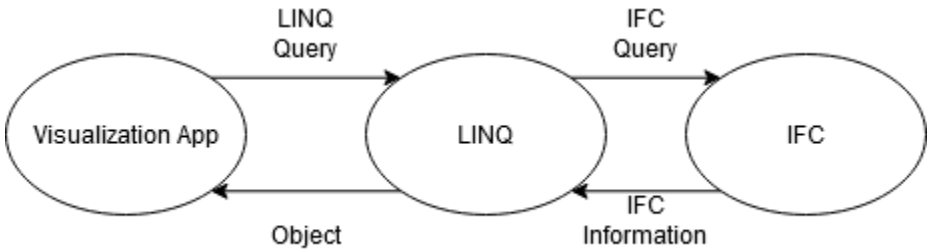


Figure 9: LINQ to IFC.

A query is asked by the IfcParser and the retrieval of data from the IFC file is accomplished through LINQ. For this process to be successful, the correct query must be asked, so knowledge of the IFC Schema is essential.

The purpose of the first query to be asked by the IfcParser is to find the IfcProject. Then, a recursive process begins:

- An empty GameObject is created with the name of the IfcProject, its IFC information gets added as an IFC Data Scripting Component to the GameObject. In Figure 10, the IFC information as an IFC data component is shown.
- Then, through the relationship with other IFC elements the process starts again with the new IfcElement as child GameObject to the parent GameObject as defined in the IFC hierarchy.
- If a GameObject with the same name as the Global Id of an IfcElement exists, then no new GameObject is created from this process, but instead the renaming of the GameObject to the name of the IfcElement occurs, along with the attachment of its corresponding metadata and then new child-parent relationships are set. Thus, the mapping of the IfcElement to the OBJ GameObject is implemented.

When this process is completed, the result is a hierarchical structure, where each GameObject has its own IFC Data component. Also, during this process, for its component a mesh collider is added, to be used in the Indoor Localization Submodule. Figure 11a shows

the structure of the BIM model in the Unity Hierarchy tab as imported from the OBJ Runtime Importer asset, while Figure 11b, a tree structure as a result of the IFC parser is constructed.

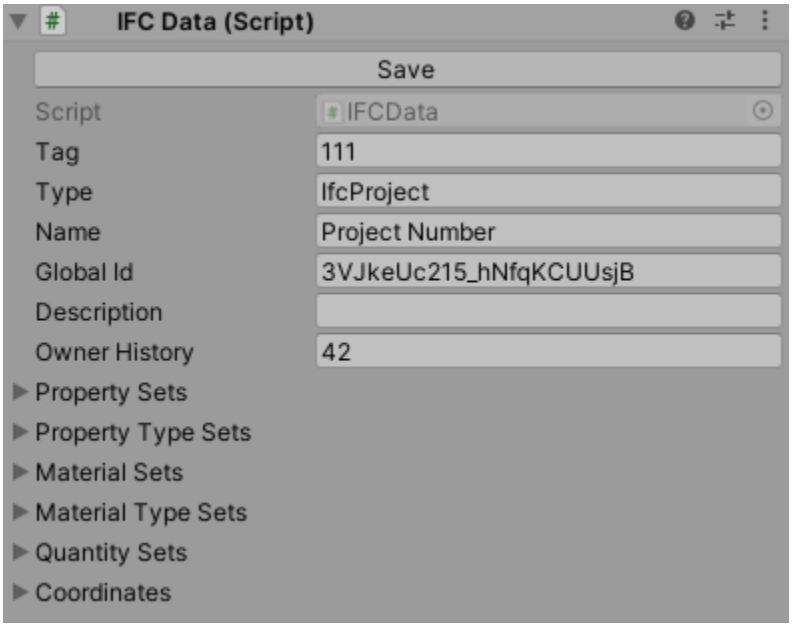


Figure 10: IFC Data Component.

Figure 11a: Non-hierarchical structure of imported OBJ in Unity.

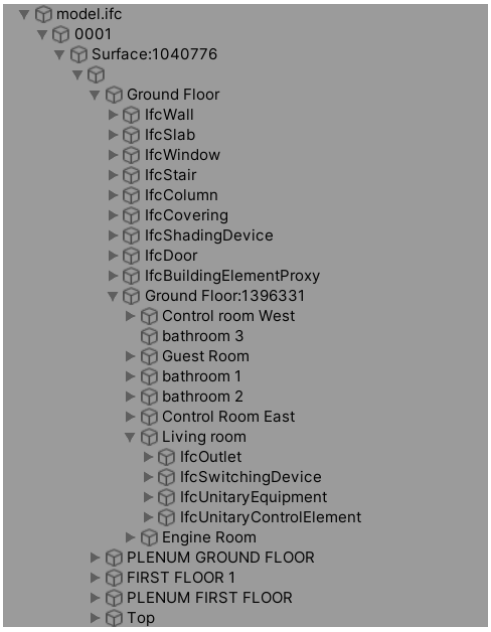


Figure 11b: Hierarchical Structure created according to the IFC structure.

The hierarchy is based on the structural relations between the IFC elements and is of the following form IfcProject/ IfcSite/ IfcBuilding/ IfcBuildingStorey/ IfcType/ IfcElement. In the case an IfcElement belongs to an IfcSpace, the hierarchy changes to IfcProject/ IfcSite/ IfcBuilding/ IfcBuildingStorey/ IfcZone/ IfcSpace/ IfcType/ IfcElement. In the first case, the IfcElement "Floor:My first floor with AIR:993332" is categorized by its IfcType, it belongs to IfcBuildingStorey "Ground Floor" of the IfcBuilding with no name, of the IfcSurface "Surface"1040776" that belongs to the IfcProject "001". In the second case, all IfcElements with Type IfcOutlet under the IfcSpace "Living room" are under the IfcZone "Ground Floor:1396331" that follows the hierarchy as described above.

The IFC Data component stores the following fields:

- Tag refers to the tag identifier at the particular instance of an IFC product.
- IfcType refers to the type of an IfcObject (IfcSlab, IfcWall, etc).
- Name refers to the name of an IfcObject.
- GlobalID refers to global unique identifier attribute of an IfcObject.
- Description refers to the description of an IfcObject.
- Owner History refers to the IfcOwnerHistory that defines all history and identification related information.
- Property Sets refer to the Property Sets for Objects concept that describes how an object occurrence can be related to a single or multiple property sets. A property set contains a single or multiple properties. The data types of an individual property are single value, enumerated value, bounded value, table value, reference value, list value, and combination of property values.
- Property Type Sets refer to the concept template Property Sets for Objects that describes how an object type can be related to a single or multiple property sets. A property type set contains a single or multiple properties. The data types of an individual property are the same as the data types of the Property Sets. The property values assigned to an object type apply equally to all occurrences of this object type.

- Material Sets refer to the Material Association concept that describes how a product relates to its associated materials that indicates the physical composition of an object. The material association can be of the Association type Material Single, Material Layer Set, Material Layer Set Usage, Material Profile Set, Material Profile Set Usage, and Material Constituent Set.
- Material Type Sets refer to the Material Association concept that describes how a product type relates to its associated materials that indicates the physical composition of an object type. The material type association is the same as the material association in the Material Sets.
- Quantity Sets contain all multiple quantity occurrences that relate to an object. The data type of quantity occurrence values is count, length, area, volume, weight, time, or a combination of quantities.
- Coordinates refer to the world coordinates of an IfcObject calculated from the relative coordinates of the Project.

Additionally, the IFCData component has the functionality to save changes from the user to the IFC file. The changes that are supported are the editing of the Ifc property values existing in the Property Sets and the addition of new Property Sets. Figure 12a shows an example of the editing. The selected IfcWindow in the PropertySet "Pset_WindowCommon" has the property Thermal Transmittance with the value 1.88. In Figure 12b, the property value is changed to 3 and is saved. Figure12c shows the changed IFC property in the BimVision viewer. This functionality is used in subsections 4.6.4.2 and 4.6.4.5 for the editing of the IFC properties.

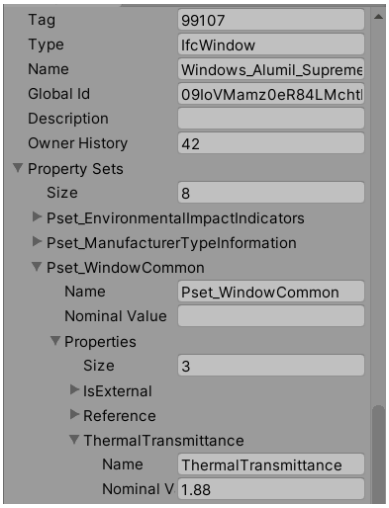


Figure 12a: Thermal Transmittance with Nominal Value 1.88.

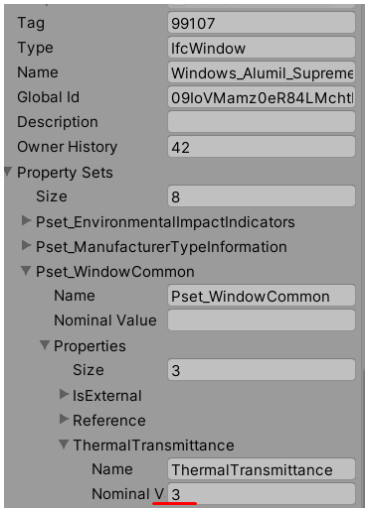


Figure 12b: Thermal Transmittance's Nominal Value changed to 3.

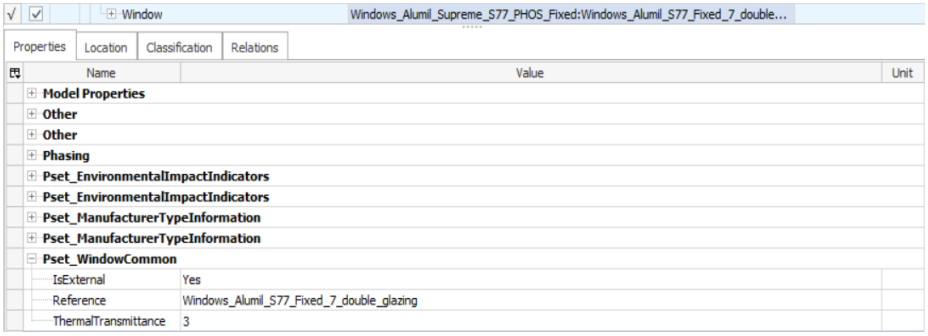


Figure 12c: The Thermal Transmittance value successfully changed as viewed in BimVision.

Figure 13 and Figure 14 show two selected components of a BIM model as seen in the 3D BIM Model Visualisation submodule in the Unity Editor Scene.

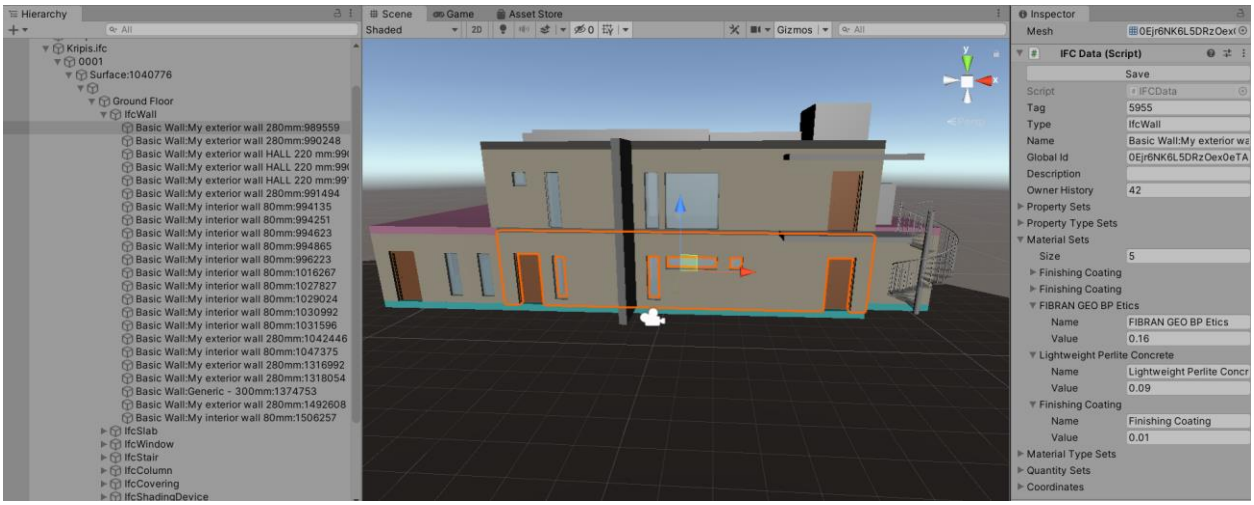


Figure 13: Wall selection.

First, an external wall is selected. In the IFC data component, further information of the selected object is displayed. In this example, the material layers used for the selected wall are of the Type Finishing Coating, Finishing Coating, FIBRAN GEO BP Etics, Lightweight Perlite Concrete, and Finishing Coating again.

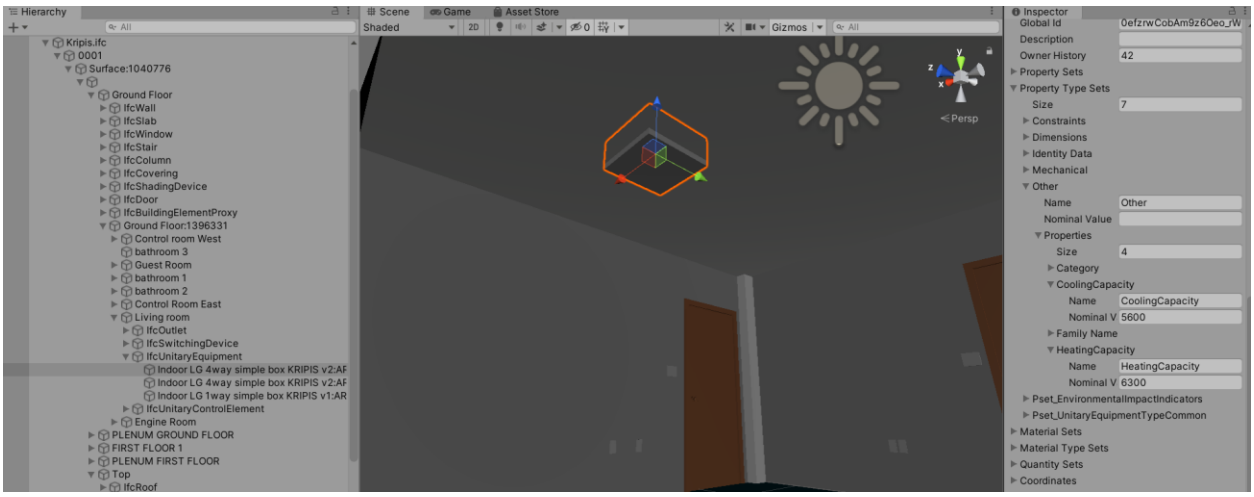


Figure 14: HVAC selection.

Aside from the materials of an Object, further information is provided. In the case of the HVAC properties and type properties such as Pset_UnitaryEquipmentTypeCommon, Pset_EnvironmentalImpactIndicators, Constraints, Dimensions, Identity Data, mechanical,

and Other are displayed, that contains crucial information, like the Cooling and Heating capacity.

Finally, a functionality for adding a new component to the IFC file was developed in order for the ARIBFA tool to enrich the IFC file with newly detected components by the AR glasses. The detected object is represented by a box GameObject in the Unity environment and has attached the IFCData component. This procedure is described in detail in Section 4.5 and merely in subsection 4.5.4.

4.3 BIM 3D MODEL REGISTRATION AND TRACKING SUBMODULE

4.3.1 Overview

The BIM 3D Model Visualisation submodule, which is described in Section 4.2, implements the visualisation of the 3D BIM model on a Unity viewer tool. The **BIM 3D Model Registration and Tracking submodule** implements the real-time visualisation of the 3D BIM model on the AR HMD, i.e., the Hololens. To have a reliable visualisation of the 3D BIM model, the model should be aligned to the real world. The 3D building components included in the BIM model should overlay the real-world building components. The location of the person wearing the Hololens in the real world should be aligned with the corresponding location in the 3D BIM model, which is loaded in the application running on the Hololens. This is the task of the 3D BIM Model Registration and Tracking submodule. In Section 4.3.2, the submodule's architecture and workflow are described and graphically illustrated. In Section 4.3.3, the development tools used for the implementation of this submodule are listed. Finally, Section 4.3.4 presents the implemented features of this submodule and demonstrates its use.

4.3.1.1 3D MODEL REGISTRATION BACKGROUND

3D model registration is a challenging task that has attracted great research interest over the years. 3D point cloud registration aims at aligning a template point cloud to a given reference point cloud [13]. To facilitate the alignment, some approaches utilize the color information of the point cloud [14], while others rely on the shape of the point cloud [13]. Many research approaches employ deep learning to train neural networks on point clouds for a variety of applications, such as point cloud registration, object classification, and part segmentation [15], [16], [17]. An emerging challenging research problem is to achieve registration using cross-source 3D point clouds from different sensors [18]. In the Architectural, Engineering, and Construction (AEC) sector, the registration of 3D point clouds to CAD models [19] or BIM models [20] is a demanding task, since scanned data can be incomplete and/or contain data from non-model objects. Moreover, symmetries and self-similarities in many construction buildings cast the 3D registration task even more challenging [20]. Recently, research approaches have emerged that study the registration of 3D BIM models using AR devices

[21], [22]. Using AR devices for 3D BIM model registration is a dynamic process due to the dynamic spatial understanding of the AR glasses.

4.3.1.2 3D MODEL REGISTRATION AND TRACKING SUBMODULE METHODOLOGY

Our approach for the registration of the 3D BIM model relies on **image targets**. An image target is an image that the application running on the Hololens will detect and track. This image will be the link between the 3D static world (BIM model) and the real world. The image is printed and positioned on a location in the real world, so that it is accessible to the person wearing the Hololens. At the same time, the image is placed on the exact same position in the 3D BIM model. While the person wearing the Hololens walks around the building, Hololens is constantly capturing data with its sensors and cameras. Features are extracted from the camera stream and are compared to the reference features extracted from the image target. In the context of pattern recognition, the features that are extracted in advance from the image target constitute the pattern that the algorithm searches across the continuous flow of data streams. When the person wearing the Hololens looks at the image target, the features extracted from the data streaming on the Hololens is matched to the known pattern of features belonging to the image target. Therefore, the image target is detected. An image target used in the ARIBFA application, as well as an illustration of the extracted features, are depicted in Figure 15.

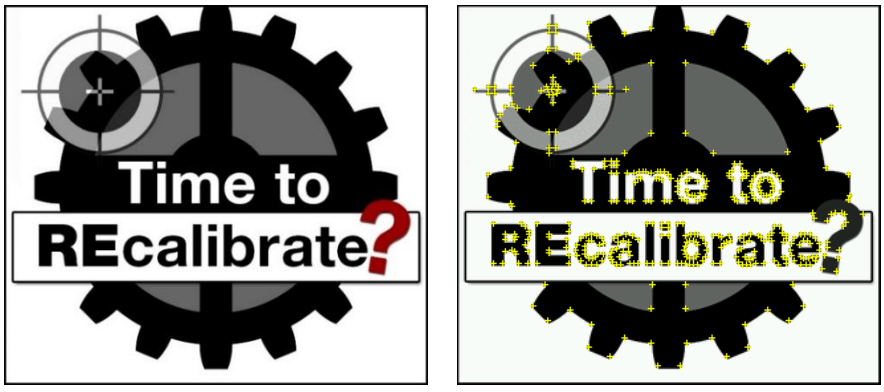


Figure 15: An image target used in the ARIBFA application and an illustration of the extracted features.

The registration process can be viewed as a three-step procedure, as illustrated in Figure 16:

1. **Static scene capture**. The 3D BIM model reflects the static scene since it provides a static model of the building. The image target is also positioned in the static scene.
    a. **Feature extraction from image target.** Features are extracted in advance, i.e., before the deployment of the application on the Hololens, from the image target. These features correspond to the pattern that the ARIBFA application will search to achieve registration.
2. **Dynamic scene capture.** At runtime, the Hololens constantly captures data with its sensors and cameras.
    a. **Feature extraction from camera images.** Features are extracted from the stream of images from the Hololens camera and are constantly compared to the features of the image target.
3. **Dynamic and static scene alignment.** When there is a match between the extracted features from the Hololens camera images with the reference features extracted from the image target, the image target is successfully detected. Therefore, the static scene is aligned with the dynamic scene and the registration of the 3D BIM model is achieved.

The procedure to align the static model with the real world is illustrated in Figure 16. The image target is placed on a specific location in the real world. The image target is also located in the exact same location in the static model, i.e., the 3D BIM model. Upon image target detection, the coordinate system of the static model is aligned to the real-world coordinate system. Therefore, 3D BIM model registration is achieved. To maintain the initial registration of the 3D BIM model, the registered 3D BIM model should be continuously tracked. In the ARIBFA application, the registration of the 3D BIM model is tracked using **spatial anchors**. Spatial anchors represent important points in the world that the system should keep track of over time. If the registered 3D BIM model is set as a spatial anchor, the 3D BIM model is not only correctly aligned with the real world but is also constantly tracked by the Hololens. This tracking procedure merely belongs to the Indoor Localization submodule and is described in detail in Section 4.4.

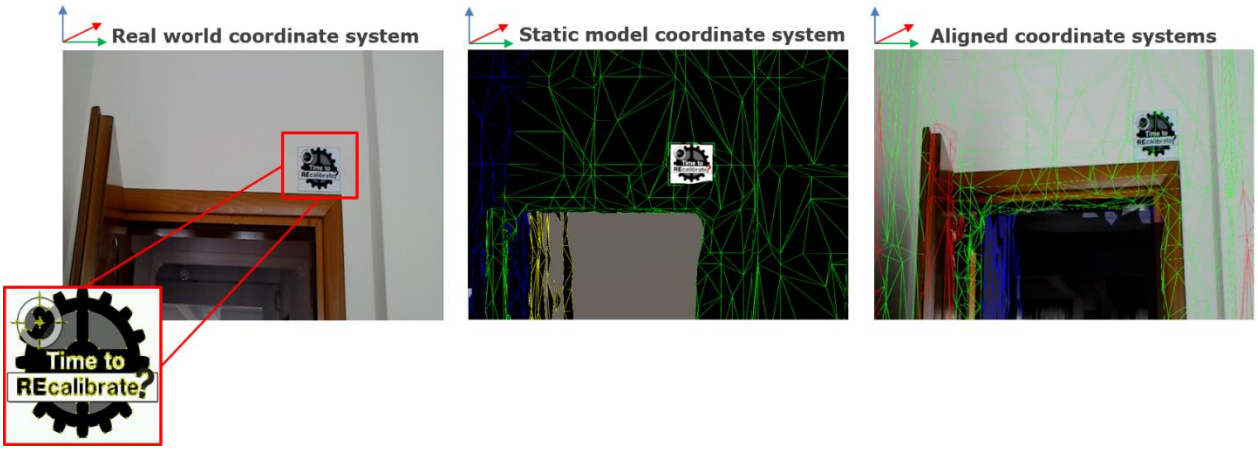


Figure 16: The image target is used for registration, i.e., align the static 3D BIM model with the real world.

4.3.1.3 DEVELOPMENT PLAN

The development of the BIM 3D Model Registration and Tracking submodule of the ARIBFA application followed an iterative, accumulative procedure. The initial design was based on mock-ups that depicted the broader idea of what the functionalities of the submodule should be and how they should look. Since the development of the BIM 3D Registration submodule of the ARIBFA application began concurrently with the development of the BIM 3D Model Visualisation submodule (described in Section 4.2), the mock-ups depicted individual building components and not the entire 3D BIM model. Instead of a BIM model of the building, the static 3D model was a 3D textured mesh of the building, scanned by the Hololens camera. To speed the procedure, the 3D scanned model reflected only a segment of the building. Therefore, to initially implement the registration module, only a part of the building was scanned by the Hololens camera. It should be noted that the Hololens is constantly trying to localize the user in the environment. Even without loading any type of architectural model of the building, the Hololens is constantly trying to understand the environment surrounding the user. The unique spatial understanding capability of Hololens facilitated the registration procedure and enabled the initial implementation of the registration module using a scanned 3D model of a building space. The 3D textured mesh of the building scanned by the Hololens camera that was used in our first registration efforts is depicted in Figure 17.

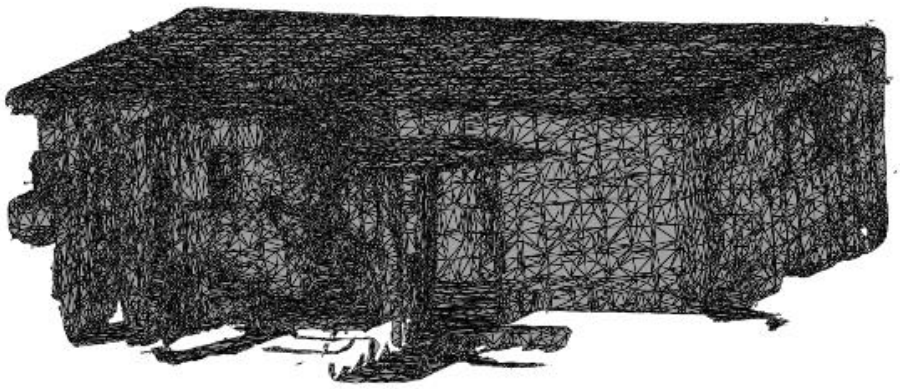


Figure 17: Illustration of the 3D textured mesh of the building scanned by the Hololens camera.

After the successful completion of the 3D BIM model visualisation (described in Section 4.2), the registration could be based on the 3D BIM model. Therefore, instead of aligning the scanned 3D mesh to the real world, the 3D BIM model was registered using the registration procedure based on image targets. Figure 18 demonstrates how the image target is placed on a specific location in the 3D BIM model (left image) and in the exact same position in the real world (right image). Figure 19 depicts a broader area of the 3D BIM model and more specifically, the living room, where the image target was placed in the Kripis building.

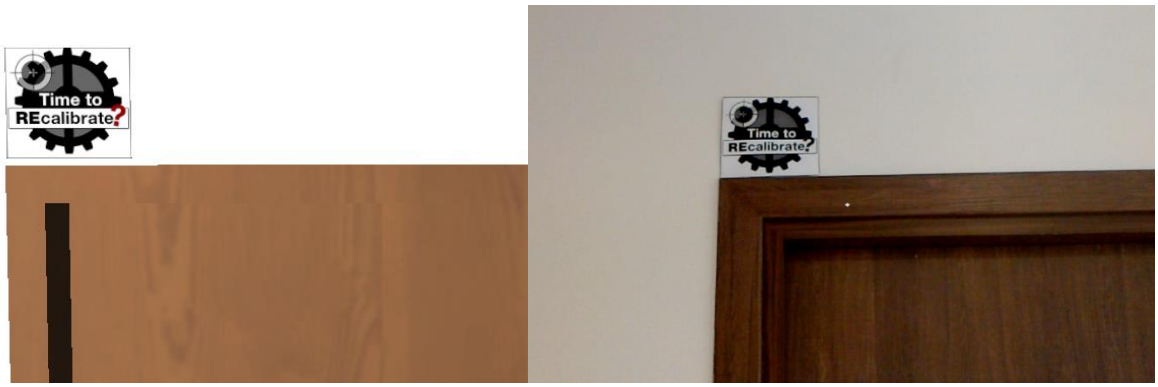


Figure 18: The image target is placed on a specific location in the 3D BIM model (left). The printed image target is placed in the exact same position in the real world (right).

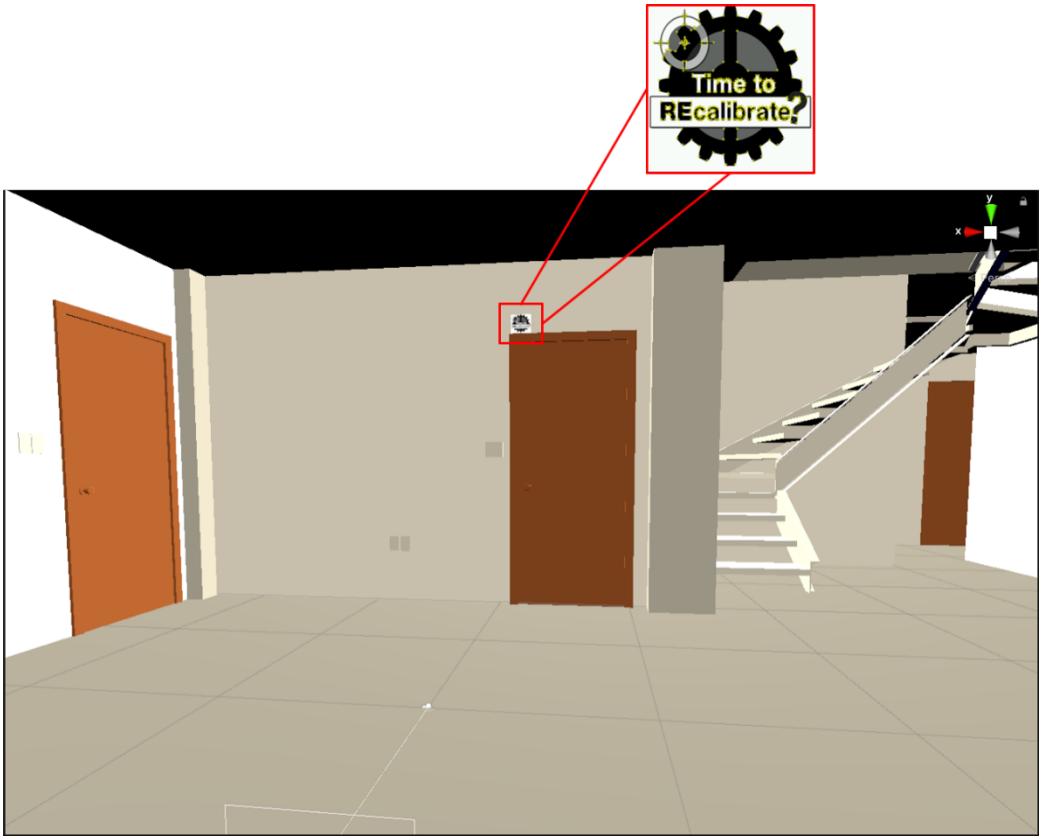


Figure 19: Broader area of the living room in Kripis building where the image target was placed for the registration of the 3D BIM model by the ARIBFA application.

4.3.2 Submodule Architecture and Workflow

The architecture of the BIM 3D Model Registration submodule is illustrated in Figure 20. As can be seen, the static scene is aligned to the dynamic scene based on a feature matching process. The static scene corresponds to the 3D scene of the ARIBFA application as visualised and processed on a 3D Editor (Unity) before the execution of the application on the Hololens. The static scene comprises the static 3D BIM model and the image target. The image target is positioned on a specific location in the static scene according to the actual location of the printed image target in the real world. The dynamic scene corresponds to the scene captured by the Hololens sensors during the execution of the ARIBFA application on the device. The dynamic scene comprises the 3D spatial understanding of the Hololens, i.e., the device's capability to constantly scan its surroundings to create a 3D map of the environment, and

the constant stream of images captured by the Hololens camera. The dynamic scene also communicates with the Marker-less Feature Recognition submodule to facilitate the online detection of objects of interest (MEP components) via the ARIBFA application running on the Hololens. The Marker-less Feature Recognition submodule is described in detail in Section 4.5.

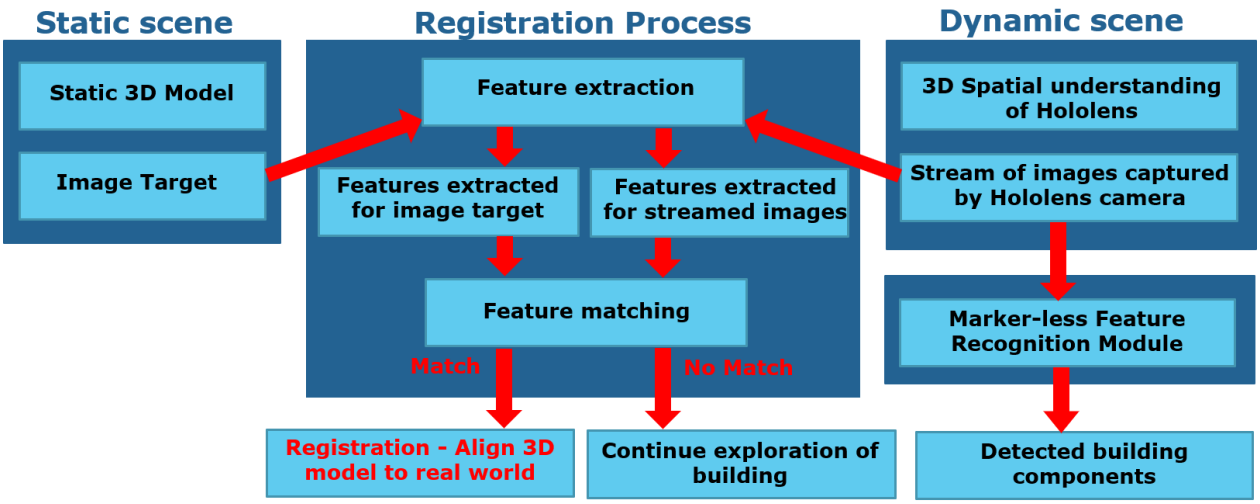


Figure 20: Architecture of the BIM 3D Model Registration submodule, as implemented in the ARIBFA application.

As can be seen in Figure 20, the registration process consists of two basic steps: feature extraction and feature matching. Feature extraction of the image target is performed in advance, i.e., before the execution of the application on the Hololens. While the ARIBFA application is running on the Hololens, feature extraction is dynamically applied to the stream of images captured by the Hololens camera. The detection of image target relies on a feature matching procedure. If the person wearing the Hololens looks at the image target placed as a poster somewhere on the building, the extracted features of the Hololens camera image are matched to the pattern of features that is known in advance that belongs to the image target. If there is a match, the application knows that the target has been detected. This way, the application localizes the image target position in the dynamic 3D scene and registration is achieved.

The first step of the ARIBFA application is the registration of the 3D BIM model. Upon successful registration, the application stops the feature extraction process and the scanning for the image target is terminated. This feature is implemented in the ARIBFA application using a voice command. More specifically, the person wearing the Hololens can start/stop/cancel the scanning for image target using the corresponding voice commands. The implementation features of the BIM 3D Model Registration and Tracking submodule that correspond to each functionality of the 3D BIM model registration performed by the module are presented in detail in Section 4.3.4.

4.3.2.1 INFORMATION ARCHITECTURE

Subsequently, a usage scenario of the application is provided that corresponds to the workflow and methodologies used to cater for this scenario and explain the decisions made by the design team regarding Information Architecture.

Usage Scenario: A building surveyor wants to visualise the 3D BIM model of a building in-situ. The building surveyor wears the Hololens and starts the ARIBFA application on the device. When the application starts, the BIM model is loaded in the application, but it is not aligned with the real world. To perform the 3D BIM model registration, the user uses the voice command “Scan for Marker”. This command enables the scanning for marker process of the application. The scanning is stopped by the application when the image target is detected. Additionally, the user can cancel the scan before the detection of image target using the voice command “Cancel scan”. When scanning for image target, the building surveyor approaches the printed image target on site. When the image target is detected, the 3D BIM model is registered and aligned to the actual building. To make the registered 3D BIM model a spatial anchor, the user uses the voice command “Anchor”. This usage scenario reflects the initial 3D BIM model registration that is performed only the first time the ARIBFA application is deployed on the Hololens. Subsequently, the registered 3D BIM model is tracked using spatial anchors using a procedure that is explained in detail in Section 4.4. The usage scenario is illustrated in Figure 21.

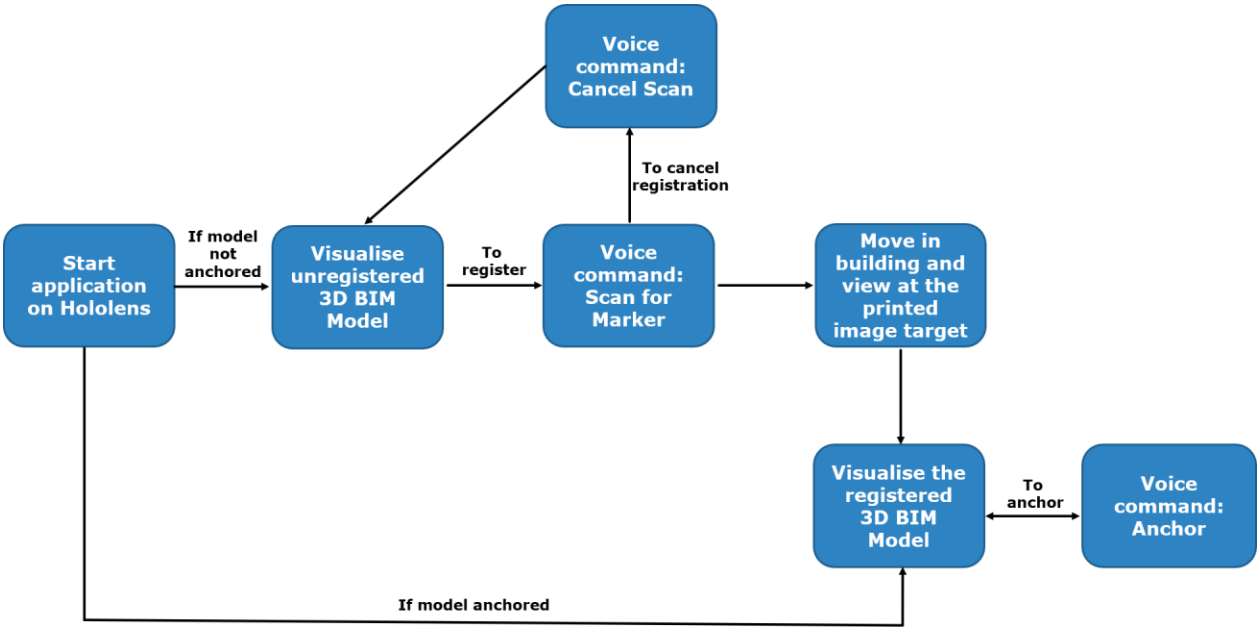


Figure 21: Usage scenario for the registration process.

4.3.3 Development Tools

In this section, the development toolset that was used for the development of the BIM 3D Model Registration and Tracking submodule of the ARIBFA application is described. Table 3 summarizes the tools and libraries used for the implementation of this submodule. In Figure 22, the Unity development environment used for the BIM 3D Registration and Tracking submodule is presented.

Table 3: Tools and libraries used in the 3D BIM Model Registration and Tracking submodule of ARIBFA application.

Library/Tool	Version	Licence
Microsoft Hololens	1st (gen), SDK 10.0.17763.0	-
Windows Mixed Reality	SDK 10.0.19041.2034	*Software licensing*
Unity 3D Editor	2020.5	*Software licensing*
Mixed Reality Toolkit (MRTK)	2.5.3	*Software licensing*
Vuforia Engine	9.6.4	*Proprietary*

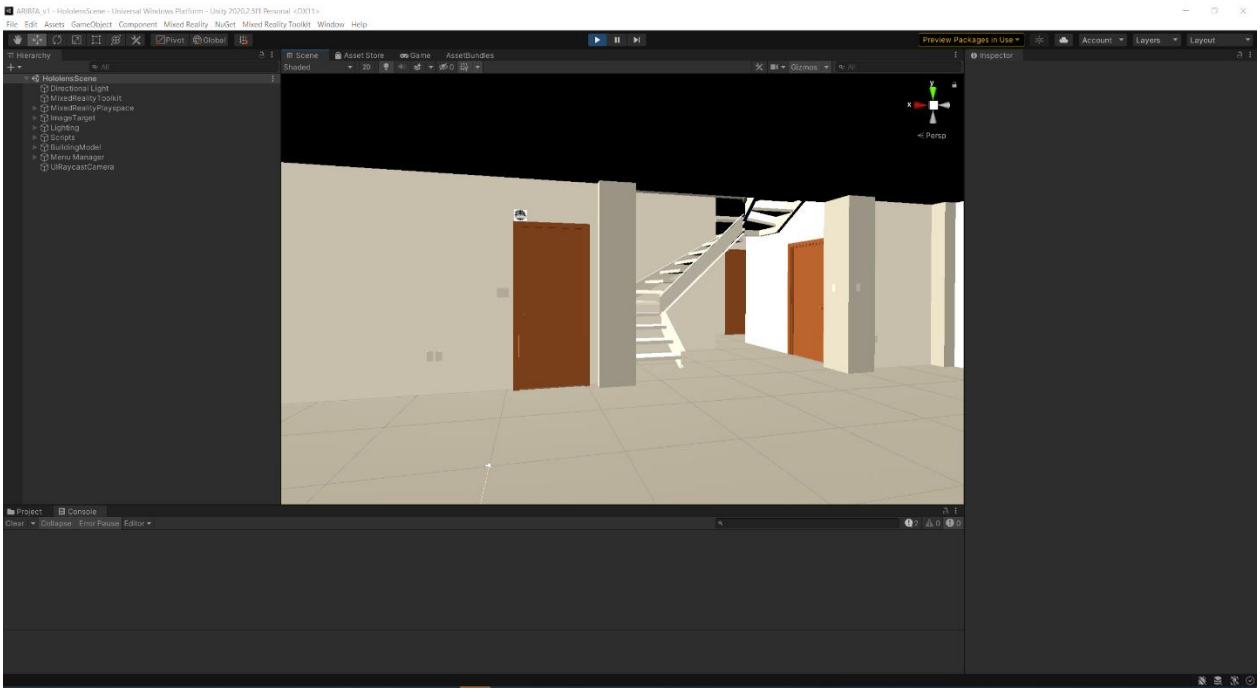


Figure 22: Unity development environment for the BIM 3D Registration and Tracking submodule.

4.3.3.1 DEVELOPMENT TARGETED TO UNITY'S XR SDK

Development on Hololens utilizes the Windows Mixed Reality package in Unity, as summarized in Table 3. The first version of ARIBFA, which was described in Deliverable D5.9, was using the Legacy built-in XR[1] support for Hololens in Unity and the Windows Mixed Reality package was the employed Virtual Reality SDK. The built-in XR support has now been deprecated in Unity, that has transitioned to a new plug-in framework to support XR platform integrations, including the Windows Mixed Reality package. From Unity 2020, the new XR SDK will be the only available XR pipeline in Unity. To this end, the final version of ARIBFA, which is described in detail in this deliverable, is implemented using Unity's new XR SDK.

[1] The term XR is used to cover Virtual Reality (VR), Mixed Reality (MR), and Augmented Reality (AR) applications.

4.3.3.2 MIXED REALITY TOOLKIT (MRTK)

As mentioned in Section 4.1.1, MRTK [23] is a cross-platform toolkit for building Mixed Reality experiences for Virtual Reality (VR) and Augmented Reality (AR). The development path for the ARIBFA application combined the MRTK with the Unity development platform. When the development of the BIM 3D Model Registration and Tracking submodule began, the recommendation was to combine MRTK with Unity 2018. At the time that the first version of ARIBFA was delivered, the latest MRTK version (2.4.0) was used that worked with Unity 2019.4. Since the second version of ARIBFA utilizes the new XR SDK in Unity, it is now targeted for Unity 2020.2 and utilizes the 2.5.3 version of MRTK. This mitigation process is important to deliver the most cutting-edge software capabilities in the ARIBFA app and the most dynamic content in the application.

4.3.3.3 VUFORIA ENGINE

To use image targets in Unity, the Vuforia Software Development Kit (SDK) [24] is employed. Vuforia Engine is the most widely used software platform for creating Augmented Reality apps, with support for smartphones, tablets, and eyewear (such as Hololens). Vuforia provides Application Programming Interfaces (API) in C++, Java, Objective-C++, and the .NET languages through an extension to the Unity game engine. Vuforia Engine enables the detection and tracking on a variety of images and objects, such as image targets (printed 2D images), model targets (pre-existing 3D models), area targets (3D model of an area scanned by a 3D scanner), and object targets (3D models created by scanning real objects) [25]. In the BIM 3D Model Registration and Tracking submodule, Vuforia is installed in Unity to activate the image target, in a process that is described in detail in Section 4.3.1. To create the image target presented in Figure 15, the image is firstly converted to a Vuforia database using the Vuforia Target Manager. Afterwards, the Vuforia database is loaded in the Unity project.

In Figure 23, the block diagram of how the hardware and software components are utilized in the development of the BIM 3D Model Registration and Tracking submodule is presented. The toolset comprises both hardware and software components. The required hardware

comprises the AR HMD, i.e., the Hololens. The required software components are the Unity 3D Game Engine, the Windows Mixed Reality SDK, and the Vuforia Engine. Unity was used for rendering and editing the scene, Windows Mixed Reality provided support for Hololens (including the MRTK and the Hololens Emulator). Vuforia SDK was used for the image target detection.

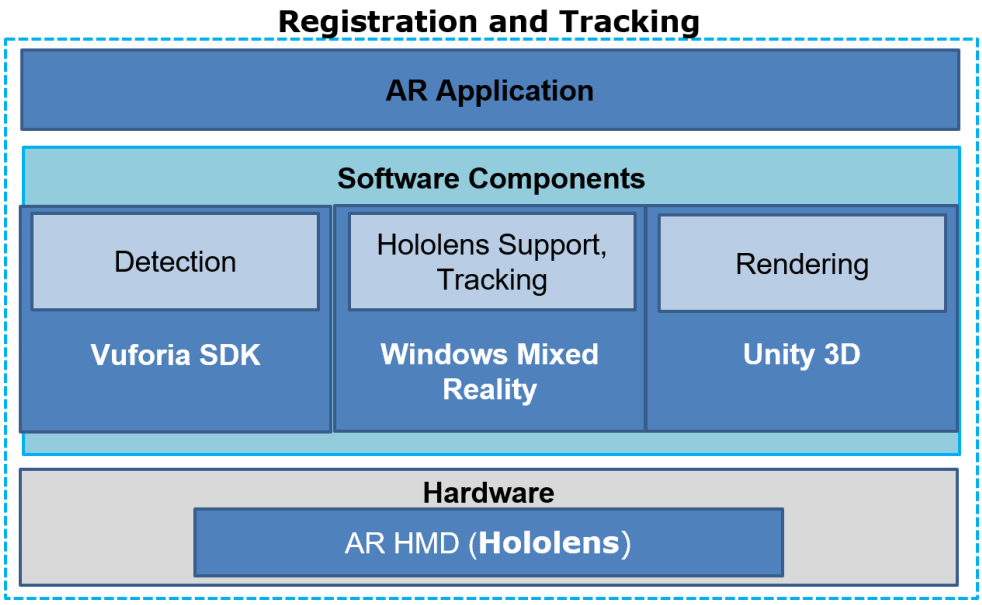


Figure 23: Block diagram of the software and hardware components used for the development of the BIM 3D Model Registration and Tracking submodule.

4.3.4 Implementation Features

In this section, the implementation process of the BIM 3D Model Registration and Tracking submodule is analyzed. Additionally, the features of the submodule are presented. The steps for this submodule are:

- Scene setup
- Image target positioning and scene hierarchy
- Registration pipeline
- Tracking with spatial anchors

These steps are analyzed below.

4.3.4.1 SCENE SETUP

The Graphical User Interface (GUI) of the application is developed using the Unity UI toolkit. The Mixed Reality Toolkit (MRTK) and the Vuforia SDK are included in the project. The scene is organized following the recommended settings for Hololens, according to the MRTK [26]. Following MRTK recommended configuration settings, a camera is loaded in the scene using the recommended clipping planes for Hololens applications. Appropriate scripts are attached to the camera to control a gaze pointer based on the gaze of the person wearing the Hololens. An EventSystem is also attached to the camera that is responsible for processing and handling events in the Unity scene. It handles input, ray casting, and sending events and coordinates between modules. Vuforia is needed in the scene only for the registration process. Vuforia is activated in the scene by enabling the "Vuforia Behaviour" script that is attached to the camera. When the "Vuforia Behaviour" script attached to the camera is disabled, Vuforia is deactivated.

4.3.4.2 IMAGE TARGET POSITIONING AND SCENE HIERARCHY

An image target object (Image Target GameObject) is inserted in the scene. In case of a building with many storeys, multiple image targets may be needed, one at each storey, to ensure the reliability of the registration process. The image target is placed in a location in the 3D BIM model that corresponds to the actual location of the printed image target in the real world. An example of an image target used for registration is depicted in Figure 15.

An empty GameObject (no render input) is created in the scene that replicates the Transform of the Image Target GameObject. Hereafter, this GameObject is denoted by Image Target Replica GameObject. The GameObject that holds the 3D BIM model hierarchy is created at start time when the BIM model is downloaded from BIF. This GameObject, denoted hereafter as BIM Model GameObject, follows the hierarchy of the IFC file of the building. For example, the GameObject is organized to Floors (e.g., "Ground Floor", "First Floor") according to the actual floors of the building to which it corresponds. The BIM Model GameObject is set to be

a child of the Image Target Replica GameObject. Therefore, it inherits any transformation applied to the parent GameObject. In Unity, each GameObject has an attached Transform, which is used to store the GameObject's position, rotation, scale, and parenting state. When a GameObject is a parent of another GameObject, the child GameObject will move, rotate, and scale exactly as its parent does.

After the image target has been detected, the Transform of the Image Target Replica GameObject is updated based on the actual Transform of the Image Target GameObject at the time of detection. Therefore, the BIM Model GameObject which is a child of the Image Target Replica GameObject, inherits its parent's Transform and is aligned to the real world based on the image target position. Since registration has been performed, Vuforia is no longer needed and is automatically deactivated upon image target detection.

4.3.4.3 REGISTRATION PIPELINE

At start time, the 3D BIM model is downloaded from BIF using the API described in Section 5.2. Subsequently, the BIM 3D Model Visualisation submodule is used to visualise the 3D BIM model on the AR device. If no registration has been performed, the rendered 3D BIM model is visualised, but it is not aligned to the real world. The BIM 3D Model Registration and Tracking submodule performs the registration process, as described in detail in Section 4.3.2 and illustrated in Figure 21. After successful image target detection, 3D registration is achieved, and the 3D BIM Model GameObject is visualised aligned to the real world. Sample images of the registered 3D BIM model are presented in Figure 24.

Figure 24: Sample images depicting the registered 3D BIM model.

4.3.4.4 TRACKING WITH SPATIAL ANCHORS

After the image target detection and the alignment of the 3D BIM model to the real world, it is crucial that this alignment remains intact while the application runs and the user wanders in the building making annotations and viewing building components. The ARIBFA application resorts to spatial anchors to achieve this. A script is developed that is attached to the Image Target Replica GameObject and controls the spatial anchors of the application. As mentioned,

at the time of detection, the Transform of the Image Target Replica GameObject is updated based on the Transform of the Image Target GameObject. Since the BIM Model GameObject is a child of the Image Target Replica GameObject, it inherits its parent's Transform and is aligned to the real world based on the image target position. To continue tracking the registered 3D BIM model, the Image Target Replica GameObject is set to be a spatial anchor using speech command "Anchor", as can be seen in Figure 21. Each spatial anchor has a coordinate system that adjusts as needed, relative to other anchors or frames of reference, to ensure that anchored holograms stay precisely in place. If the aligned 3D BIM model is anchored, the next time the application is deployed, the 3D BIM model is visualised aligned to the real world. Although the BIM model and the IFC file are loaded from BIF each time the application starts, if the specific BIM model has been anchored to the real world, it maintains this position at subsequent sessions.

4.4 INDOOR LOCALIZATION SUBMODULE

4.4.1 Overview

The key objective of the **Indoor Localization submodule** is to accurately determine the user's position and orientation within the building. The initial registration of the 3D BIM model is accomplished by the BIM 3D Registration and Tracking submodule, which is described in Section 4.3. The Indoor Localization submodule is responsible to keep track of the registered 3D BIM model and localize the user constantly into it. Successful completion of the BIM 3D Model Visualisation submodule (in Section 4.2) and the BIM 3D Model Registration and Tracking submodule (in Section 4.3) are prerequisites for the Indoor Localization submodule. After the 3D BIM model is effectively loaded and visualised and is successfully aligned to the real world, the user's position and orientation within the building should be constantly and accurately determined. The Indoor Localization submodule should provide reliable information regarding the user's position to the other modules that depend on it. In particular, the AR Annotation & Context Aware-Visualisation submodule depends greatly on the outcome of the Indoor Localization submodule. The correct localization of the user is crucial to the annotation process since the visualised building components that are annotated should be reliably matched to the real-world building components.

In this section, an overview of 3D Indoor Localization approaches is presented, along with the methodology that was followed for the development of this submodule in the ARIBFA application. In Section 4.4.2, the submodule's architecture is described and graphically illustrated. In Section 4.4.3, the development tools used for the implementation of this submodule are listed. Finally, Section 4.4.4 presents the implemented features of this submodule.

4.4.1.1 INDOOR LOCALIZATION BACKGROUND

Indoor localization is a research topic that receives great attention due to the tremendous expansion of the smart devices' use. Indoor location information is the key enabler of a range of applications including navigation guidance, location-based services, emergency response, and augmented reality [27]. Indoor localization is always challenging since it may deal with

unstable environment in the building [28]. Simultaneous Localization and Mapping (SLAM) techniques build a map of an unknown environment and localize the sensor in the map focusing on real-time operation [29]. Among the different sensor modalities, cameras are cheap and provide rich information of the environment that allows for robust and accurate place recognition. Therefore, Visual SLAM solutions, where the main sensor is a camera, are of significant interest nowadays [29]. In [29], ORB-SLAM2 is proposed which is a complete SLAM system for monocular, stereo, and RGB-D cameras, including map reuse, loop closing, and re-localization capabilities. The BIM model of the building is utilized for SLAM in the model-based visual tracking framework proposed in [27], where localization is performed by matching image sequences captured by a camera with a 3D model of the building. A visual localization approach based on deep learning is proposed in [30]. More specifically, a deep convolutional neural network trained on pose regression, namely Pose-Net, is finetuned using synthetic images obtained from the 3D indoor model (BIM model) to regress the camera pose. An indoor positioning system to locate users in large indoor scenes that relies on static objects, e.g., doors and windows, as references to locate users and utilizes images obtained by smartphone cameras is proposed in [31].

The built-in indoor localization capability of the Hololens is evaluated in [32]. More specifically, the Hololens' localization system, which is based on the device's sensors along with the device's four tracking cameras and the Time-of-Flight (ToF) range camera, is evaluated for the task of mapping indoor environments yielding impressive results. In [33], a geometric feature-based localization and registration approach of an as-built point cloud in an as-planned point cloud through an augmented reality device (e.g., Hololens), is presented. The indoor localization approach in [34] utilizes the coarse triangle meshes provided by the Hololens' ToF range camera. The unstructured 3D data are voxelized, a voxel grid is obtained, and rooms are detected by segmenting connected voxel components of ceiling candidates and extruding them downwards to find floor candidates. In [35], a marker-based localization method for the Hololens is proposed. A marker is placed in the building environment that is augmented with model data and the registration process achieves a remarkable spatial accuracy of few centimetres. Indoor localization using Hololens is also addressed in [36], where an image-based indoor localization method is proposed for the purpose of facility

management. This method estimates the user's indoor position and orientation by comparing the user's perspective to BIM model based on a deep learning computation.

4.4.1.2 INDOOR LOCALIZATION SUBMODULE METHODOLOGY

After the initial registration of the 3D BIM model by the 3D BIM Model Registration submodule of the ARIBFA application, the localization of the user in the BIM model should be continuous. To address this in the ARIBFA application, the initial registration of the 3D BIM model is continuingly tracked using **spatial anchors**. Spatial anchors represent important points in the world that the system keeps track of over time. By setting the 3D BIM Model as a spatial anchor, the 3D BIM model is constantly tracked by the Hololens. This tracking procedure is the main task of the Indoor Localization submodule. More specifically, the objectives of the Indoor Localization submodule are:

- Determination of user's position and orientation within a building.
- Translation of user's position into relative coordinates within a building's space.
- Registration of building characteristics to digital 3D models.

A key objective of the Indoor Localization submodule is the determination of the user's position and orientation within a building. To determine the user's position, the spatial mapping capability of Hololens is leveraged. **Spatial mapping** (also called 3D reconstruction) is the ability to create a 3D map of the environment. Augmented reality devices, such as Hololens, must create an understanding of the user's physical space and additionally, use this understanding to be able to interact with virtual objects. During runtime, the MRTK provides the capability to visualise the Hololens' spatial understanding, i.e., the Hololens' mapping of the surrounding environment, as a spatial mesh that overlays the environment. This mesh can also be obtained offline via the device portal and comprises a set of triangles (in three dimensions) that are connected by their common edges or corners. The triangled mesh is produced and placed on the device's environment and helps the device understand its surroundings. Sample images depicting the spatial mesh obtained by Hololens are presented in Figure 25. As the Hololens gathers new data about the environment, and as changes to the environment occur, spatial surfaces will appear, disappear, and change in the spatial mesh.

Spatial mapping makes it possible to place objects on real surfaces. This way, it is possible to anchor objects in the user's world, using real-world depth information.

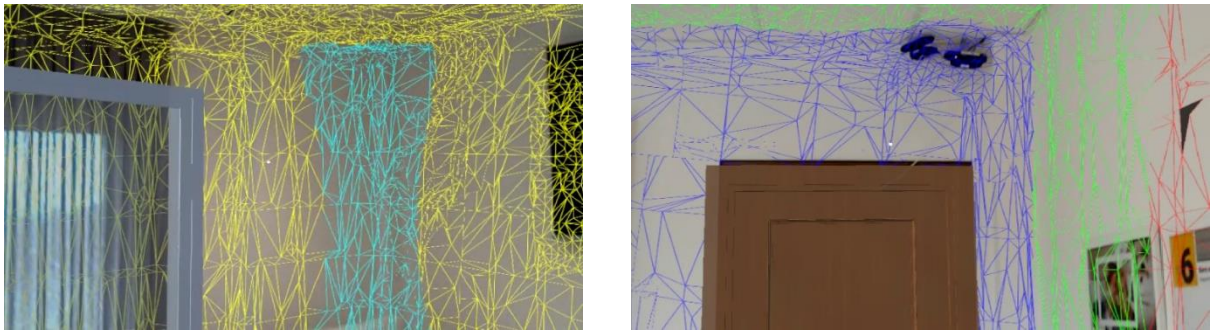


Figure 25: Sample images depicting the spatial mesh obtained by the Hololens for its surrounding environment.

The BIM 3D Model Registration and Tracking submodule and the Indoor Localization submodule perform several transformations between different coordinate systems to align the 3D BIM model to the real world and keep localizing the person and the BIM model in the real world. Following the notation in [35], let us denote by T_A^B the pose of an object A in a coordinate system B. T_A^B consists of a 4×4 matrix in the form:

$$\boldsymbol{T_A^B} = \begin{pmatrix} \boldsymbol{R_A^B} & \boldsymbol{t_A^B} \\ \boldsymbol{0^T} & \boldsymbol{1} \end{pmatrix} \in \mathbb{R}^{\boldsymbol{4x4}} \quad (1)$$

Here, R_A^B is an orthogonal 3×3 matrix, which describes the orientation of the respective object A in relation to the coordinate frame B while the vector t_A^B describes its position. The physical scale of the printed marker is also defined. Therefore, scaling aims to preserve the exact scale between the marker and the 3D model. In Figure 26, an illustration of the transformations that are needed to achieve marker-based registration and indoor localization using Hololens are depicted. There are four coordinate frames: the App coordinate frame, the Hololens camera coordinate frame, the Marker coordinate frame, and the Model coordinate frame. The Hololens application frame, namely App frame, denotes the coordinate system that is defined when the application starts, and it has as origin the location of the camera. The device' pose when the application starts defines the App coordinate frame. Therefore, the App frame is different each time the application starts and depends on the position of the Hololens when the application starts. The current pose of the Hololens device in the App frame can be queried by the ARIBFA application via the Hololens SDK, which accesses the

built-in tracking system of the Hololens. By Hololens SDK, the Windows Mixed Reality and the MRTK are denoted. The Hololens camera coordinate frame is the local coordinate system of the Hololens camera. The Model coordinate frame corresponds to the coordinate frame of the 3D BIM model. The Marker coordinate frame is the coordinate frame defined by the image target.

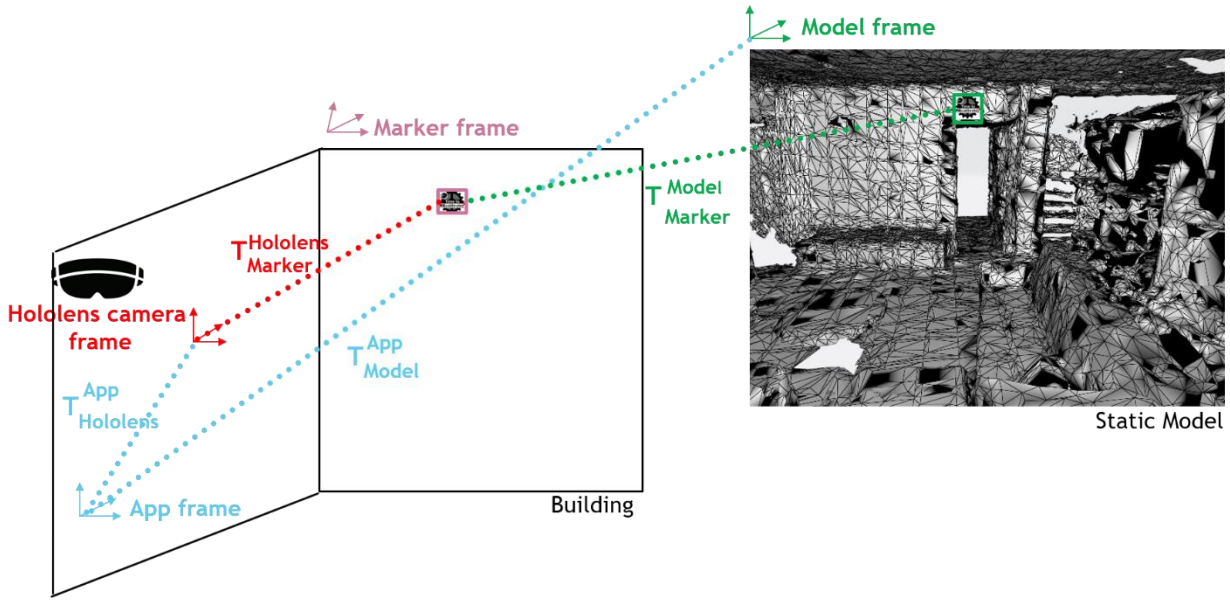


Figure 26: Illustration of the transformations between coordinate systems to achieve 3D Registration and Indoor Localization.

To achieve indoor localization, the pose of the Hololens device with respect to the BIM model, i.e., $T_{Hololens}^{Model}$, must be determined constantly while the application is running. This equates to determining the pose T_{Model}^{App} of the BIM model in the Hololens application coordinate frame that leads to a correct alignment between the virtual and physical building geometry. Leveraging image targets facilitates the registration procedure. The image target is placed in a specific location in the BIM model. The pose of the image target in the BIM model, i.e., T_{Marker}^{Model}, is determined manually, therefore is known to the application. The pose T_{Model}^{App} can be determined by

$$\boldsymbol{T_{Model}^{App} = T_{Marker}^{App}\, T_{Model}^{Marker}} \quad (2)$$

In Eq. (2), the pose T_{Model}^{Marker} can be determined by the known T_{Marker}^{Model}, since $T_{Model}^{Marker} = T_{Marker}^{Model\ -1}$. If we also knew the pose T_{Marker}^{App}, we could determine the pose T_{Model}^{App}. To determine the marker's pose in the App coordinate frame, i.e., T_{Marker}^{App}, we first determine the pose $T_{Marker}^{Hololens}$ by observing the marker with the Hololens camera. The pose of the Hololens camera in the App coordinate frame at exactly the moment of capturing the image of the marker can be queried via the Hololens SDK. This pose is denoted by $T_{Hololens}^{App}$. Therefore, the pose T_{Marker}^{App} can be determined by

$$\boldsymbol{T_{Marker}^{App} = T_{Hololens}^{App}\ T_{Marker}^{Hololens}} \quad (3)$$

The image target is utilized only during the initial registration of the 3D BIM model. After this procedure, we create a replica of the 3D BIM model as a hologram in the App coordinate frame. The pose of the 3D BIM model-hologram, i.e., $T_{HologramBIM}^{App}$, is set equal to the pose T_{Marker}^{App}. To make this BIM model-hologram maintain its position across different initializations of the application, we make it a **spatial anchor**. This way, the pose $T_{HologramBIM}^{App}$ is known to the application by querying the Hololens SDK.

4.4.2 Submodule Architecture and Workflow

The architecture of the Indoor Localization submodule is illustrated in Figure 27. In Figure 27, the SDKs and tools used in the Indoor Localization submodule are depicted. To better clarify the tasks of registration and indoor localization, the tools utilized for both tasks are illustrated comparatively. The ARIBFA application is implemented in Unity Editor. Unity is the platform to load and use individual SDKS, such as the Vuforia SDK and the Hololens SDK. As mentioned, Hololens SDK denotes the Windows Mixed Reality and the MRTK. All arrows in Figure 27 are two-way to indicate the constant queries asked regarding each pose during the execution of the application. The dotted arrows indicate that this type of pose information is not queried constantly in the application, but only after successful registration.

As can be seen in Figure 27, the Vuforia SDK is needed for registration to provide the image target capability. The application queries the Vuforia SDK regarding the Image Target Pose,

the 3D BIM Model Pose, and the Camera Pose (the Vuforia's AR Camera). After image target detection, the registration process is complete. The outcome is the Registered 3D BIM Model, whose pose can be obtained by the Vuforia SDK. Since Unity is the platform to use both Vuforia and Hololens SDK, it has access to every type of pose needed for registration. After registration, the Vuforia Engine is deactivated. The aligned 3D BIM Model is anchored using the spatial anchor capability of the Hololens SDK. Information regarding the Camera Pose, the User Pose, and the Anchored 3D BIM Model Pose can now be obtained by the Hololens SDK. The Indoor Localization submodule only queries the Hololens SDK regarding the three poses needed for tracking: The Camera Pose, the User Pose, and the Anchored 3D BIM Model Pose. It should be noted that the Hololens SDK is also active during registration, providing information regarding the User Pose. It is shown as inactive for registration in Figure 27 for demonstration purposes to highlight that the Vuforia SDK is the main SDK to perform the registration process.

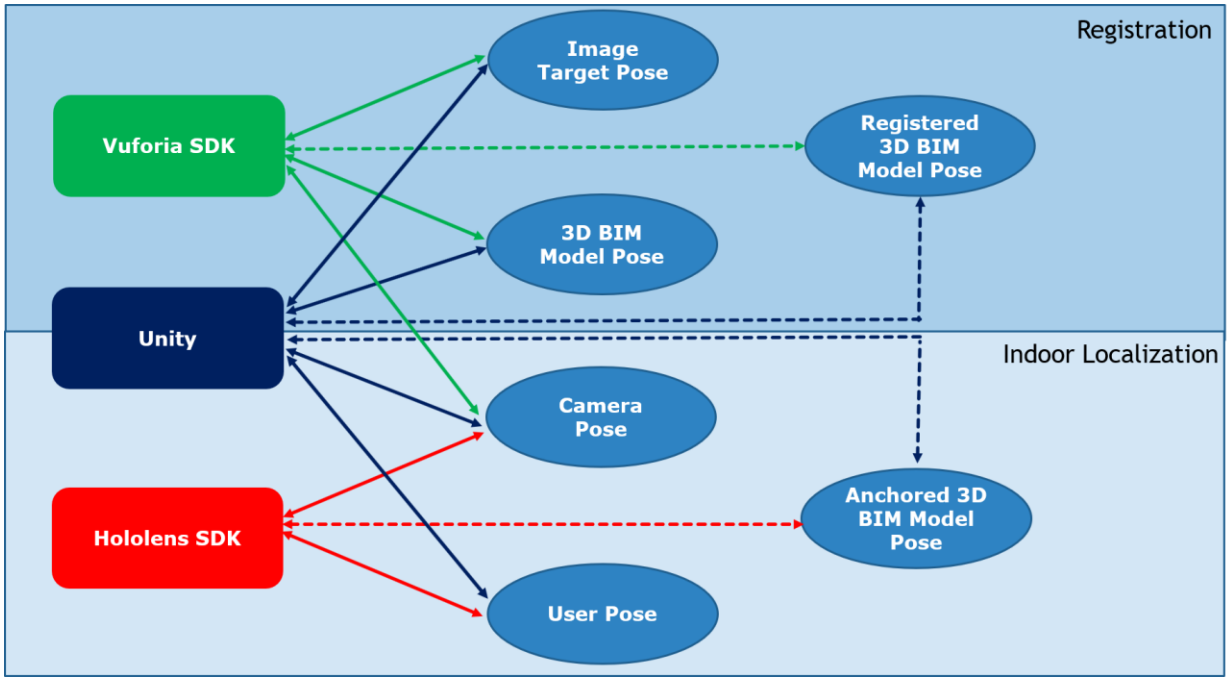


Figure 27: Architecture of the SDKs and tools that contribute to the 3D BIM Model Registration and Indoor Localization submodules.

4.4.3 Development Tools

In this section, the development toolset that was used for the development of the Indoor Localization submodule of the ARIBFA application is described. Table 4 summarizes the tools and libraries used for the implementation of the Indoor Localization submodule. The block diagram of the software and hardware components utilized in the development of the Indoor Localization submodule is presented in Figure 28.

Table 4: Tools and libraries used in the Indoor Localization submodule of ARIBFA application.

Library/Tool	Version	Licence
Microsoft Hololens	1st (gen), SDK 10.0.17763.0	-
Windows Mixed Reality	SDK 10.0.19041.2034	*Software licensing*
Unity 3D Editor	2020.5	*Software licensing*
Mixed Reality Toolkit (MRTK)	2.5.3	*Software licensing*
Vuforia Engine	9.6.4	*Proprietary*

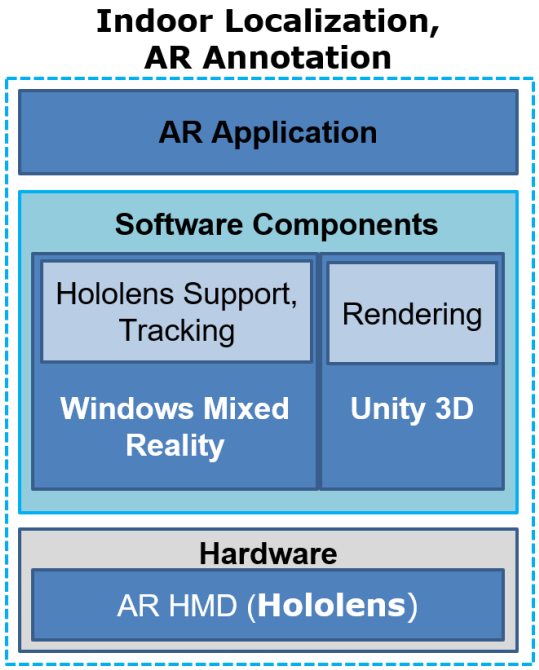


Figure 28: Block diagram of the software and hardware components used for the development of the Indoor Localization submodule and the AR Annotation & Context Aware-Visualisation submodule of the ARIBFA application.

4.4.4 Implementation Features

The implementation process of the Indoor Localization submodule cannot be explicitly disentangled from the implementation process followed for the BIM 3D Model Registration and Tracking submodule. Since the BIM 3D Model Registration and Tracking submodule and the Indoor Localization submodule have many common implementation features, the implementation process described in Section 4.3.3 also contains information regarding the implementation of spatial anchors in the Indoor Localization submodule. The alignment of the 3D BIM Model with the real world is maintained during the application via the Indoor Localization submodule. Therefore, Figure 24 that depicts images of the aligned 3D BIM model is also attributed to the Indoor Localization submodule.

4.4.4.1 LOCALIZATION INFORMATION

The user's position in 3D BIM model is important for many use cases of ARIBFA. For example, the task notifications are visualised in ARIBFA depending on the user's current position in the building. The user receives information for the tasks that involve the IfcSpace, they currently are in the 3D BIM model. Therefore, the user location in terms of IfcSpace is necessary at runtime. This feature utilizes the colliders, which were added to each IfcSpace, as described in Section 4.2. At runtime, the user can see location information using speech command "Show Location", as depicted in Figure 29. Displaying localization information is terminated with speech command "Hide Location".

Figure 29: Localization information in terms of IfcSpace is displayed upon user's request.

4.5 MARKER-LESS FEATURE RECOGNITION SUBMODULE

4.5.1 Overview

The main functionality of the Marker-less Feature Recognition submodule is to perform object detection for specific building components via the application running on the AR glasses, i.e., the Hololens. More specifically, the functionalities of this submodule are:

- Automatically detect MEP components from the captured images of the AR glasses.
- Detect structural building components and components of value to energy analysis, such as plugs, switches, heating radiators, light fixtures, interior air conditioning units, and vents.

A necessary task for this submodule is the collection and annotation of images related to MEP building components. As a result, an MEP Object Detection Dataset is collected that comprises images depicting plugs, switches, and HVAC components. Sample images of the collected MEP Object Detection Dataset is depicted in Figure 30.

Figure 30: Sample images of the collected MEP dataset depicting plugs, switches, indoor and outdoor HVAC components.

4.5.1.1 OBJECT DETECTION BACKGROUND

"You Only Look Once (YOLO)" is a state-of-the-art, real-time object detection algorithm applied on images [37]. In YOLO, object detection is addressed as a regression problem to spatially separated bounding boxes and associated class probabilities. The network predicts bounding boxes and class probabilities from full images in testing. Many extensions of YOLO have been proposed. YOLOv2 [38] is faster than YOLO in real-time object detection and due to its multi-scale training method, it can be run at a variety of image sizes. YOLOv3 [39] and YOLOv4 [40] also present some modifications to the object detection algorithm to make it

faster and more efficient. R-CNNs, which is a state-of-the-art approach for object detection, combine region proposals with Convolutional Neural Networks (CNNs) [41]. Fast R-CNN [42] is an extension of R-CNN to provide a faster and more accurate object detection solution. Faster R-CNN [43] extends Fast R-CNN by sharing full-image convolutional features with the detection network to enable nearly cost-free region proposals. Finally, Mask R-CNN [44] extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

Real-time 3D object recognition is an important task in computer graphics and computer vision for 3D environment understanding. In [45], volumetric CNNs that can work directly on 3D data are trained for object recognition. The proposed network learns discriminative 3D features from multitask learning, i.e., training on both class labels and orientation information from both global and partial shapes. An object detection network based on a synergy of deep point set networks and Hough voting, namely VoteNet, that is applied directly on point clouds is proposed in [46]. This network is extended to utilize color information [47]. The proposed framework, namely ImVoteNet, performs 3D object detection by fusing 2D votes in RGB-D images and 3D votes in point clouds. In [48], a novel two-stage 3D object detection framework, namely PointRCNN, that operates on 3D point clouds is presented. During the first stage, 3D bounding box proposals are generated. During the second stage, a point cloud region pooling operation is applied to pool the learned point representations from the first stage. In [49], Complex-YOLO, a real-time 3D object detection network on point clouds is proposed. Complex-YOLO expands YOLOv2, which is applied on RGB images, by a specific complex regression strategy to estimate multi-class 3D boxes in Cartesian space.

4.5.1.2 MARKER-LESS FEATURE RECOGNITION SUBMODULE METHODOLOGY

The YOLOv2 object detection algorithm [38] is employed to detect building components in the Marker-less Feature Recognition submodule. More specifically, **tiny YOLOv2** is employed. Tiny YOLOv2 is a condensed version of the original YOLOv2 model. It is much faster than the original YOLO model, but less accurate. Since our goal is to achieve real-time

object detection on the Hololens, the evaluation speed of the algorithm is crucial. Therefore, a trade-off between speed and accuracy is necessary for our application. Tiny YOLOv2 network comprises 15 layers and is trained on the Pascal VOC dataset (C=20 object classes). The architecture of tiny YOLOv2 is illustrated in Figure 31. It divides the input image into a 13 x 13 grid, where each cell is responsible for predicting B=5 bounding boxes. One bounding box has 5 properties (coordinates, width, height, probability that an object exists in this box) + C properties (conditional probabilities that the object belongs to a specific class). The number of output filters is computed by (C + 5) x B. At test time, the model designates each box with its class-specific confidence score by multiplying the confidence score of each box with its respective class-conditional probabilities. These scores evaluate the probability of the class appearing in the box and how precise the box coordinates are.

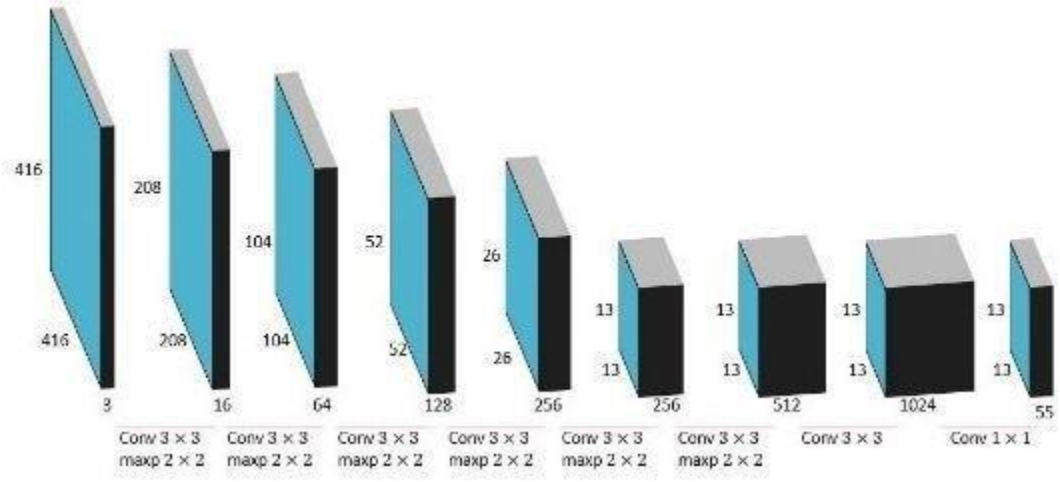


Figure 31: Tiny YOLO v2 network structure *[50]*.

The pre-trained tiny YOLOv2 is finetuned on the collected MEP Object Detection Dataset to detect MEP components. To train the algorithm, the images of the dataset are annotated with bounding boxes. Subsequently, the finetuned model is deployed on the Hololens. For real-time object detection, Hololens is paired to a computer. Images from the Hololens camera are streamed to the paired computer. Detection is performed on the local device by feeding the images received from Hololens to the trained object detection model. Subsequently,

detection coordinates are promoted to Hololens, and corresponding bounding boxes are drawn to overlay physical objects.

4.5.1.3 DESKTOPARIBFA APPLICATION

To facilitate the connection of Hololens to a computer, **DesktopARIBFA** was developed. DesktopARIBFA is a Windows application that, when installed on the computer to be paired with Hololens, handles the communication between the two devices (Hololens and computer), and performs object detection on the image stream coming from Hololens. The detected objects are visualised as 3D bounding boxes in the final version of the ARIBFA application. Appropriate handlers on the 3D bounding box give the user the capability to adjust and optimize the position, the scale, and the rotation of the bounding box. Furthermore, appropriate menus have been developed so that the user can insert IFC properties to the detected MEP components. The user can add the detected components to the BIM model at runtime and upload the modified IFC file to BIF using the corresponding API. The architecture and workflow of the aforementioned object detection procedure is discussed in detail in Section 4.5.2. In Section 4.5.3, the development tools utilized for Marker-less Feature Recognition Submodule are summarized. DesktopARIBFA and the menus for adding IFC properties to the detected MEP components are described in detail in Section 4.5.4. The API used to send modified BIM model to BIF is discussed in detail in Section 5.2.

4.5.2 Submodule Architecture and Workflow

The architecture of employing Tiny YOLOv2 object detection algorithm in the application running on Hololens is illustrated in Figure 32. As can be seen in Figure 32, the finetuning of the Tiny YOLOv2 model on the MEP datasets takes place on a computer. The finetuned model is firstly converted to Open Neural Network Exchange (ONNX) [51] file format to promote interoperability. ONNX is an open standard format for representing machine learning models that enables the deployment of models within a variety of frameworks, tools, runtimes, and compilers. At runtime, Hololens is paired to a computer. DesktopARIBFA that accompanies ARIBFA for object detection is installed to a local computer to facilitate the connection to

Hololens. This way, the computer is given access to the stream of images from Hololens camera, which are fed to the object detection algorithm for testing. Upon detection, detection coordinates are promoted to Hololens and ARIBFA is responsible to draw 3D bounding boxes to visualise the detected objects.

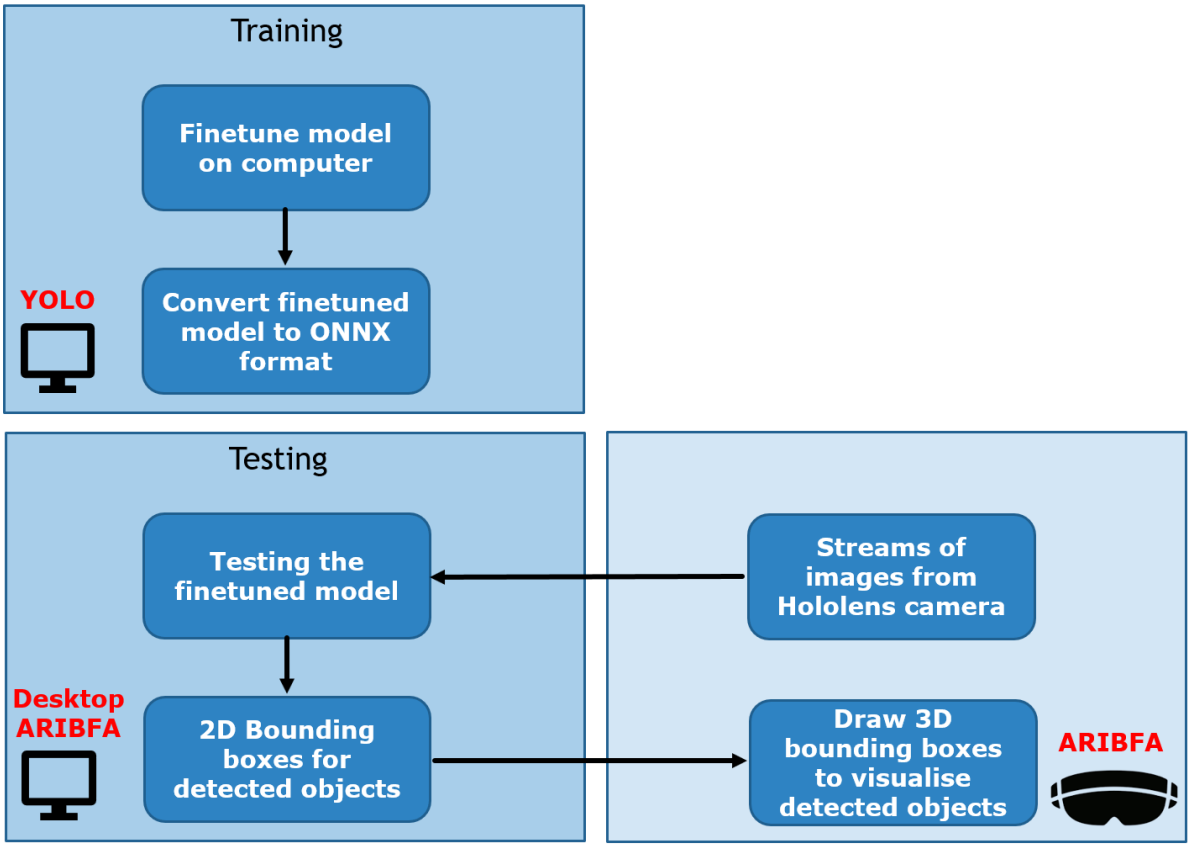


Figure 32: The architecture of employing Tiny YOLOv2 object detection algorithm in Marker-less Feature Recognition submodule. For testing, Hololens are paired to a local computer. DesktopARIBFA is installed on a local computer to facilitate the connection with Hololens, receive camera stream from Hololens, and perform object detection locally. Detection coordinates are promoted to Hololens and the ARIBFA application visualises detected objects with 3D bounding boxes.

4.5.3 Development Tools

Table 5 summarizes the tools and libraries used for the implementation of the Marker-less Feature Recognition submodule. To promote the use of Hololens as a tool for Computer Vision and Robotics research, Microsoft provides samples and support for accessing the

device's streams in HololensForCV repository [52]. OpenCV is also employed in HololensForCV for image processing. The block of the software and hardware components utilized in the development of the Marker-less Feature Recognition submodule is the same with the one of the Indoor Localization submodule depicted in Figure 28, with the addition of HololensForCV tool in the Hololens Support Software.

Table 5: Tools and libraries used in the Marker-less Feature Recognition Module of ARIBFA application.

Library/Tool	Version	Licence
Microsoft Hololens	1st (gen), SDK 10.0.17763.0	-
Windows Mixed Reality	SDK 10.0.19041.2034	*Software licensing*
Unity 3D Editor	2020.5	*Software licensing*
Mixed Reality Toolkit (MRTK)	2.5.3	*Software licensing*
HololensForCV	-	*MIT License*

4.5.4 Implementation Features

In this section, the implementation features of the Marker-less Feature Recognition submodule are presented. The key implementation features of this submodule are:

- DesktopARIBFA application
- MEP components detection examples
- Object detection pipeline
- Addition of IFC properties to detected components
- Addition of detected objects to the BIM model

These features are analyzed below.

4.5.4.1 DESKTOPARIBFA APPLICATION

DesktopARIBFA is a desktop application developed to accompany the ARIBFA application and facilitate the communication between the Hololens and a local computer. It is a Windows application targeted to Windows 10 (Windows 10 are also running on the Hololens). Complete

installation guide is presented in Section 7.1.2.2. The main window of DesktopARIBFA is depicted in Figure 33. The user can insert Hololens' IP in field "Host name" and select "Connect" to connect to Hololens. Details on how to acquire Hololens IP are enclosed in Section 7.1. Upon successful connection, images of Hololens camera are streamed at the main window of DesktopARIBFA. Images are fed to the finetuned object detection model to detect MEP components. In DesktopARIBFA, user can visualise the live video from Hololens camera, along with the bounding boxes for the detected components. Subsequently, detection coordinates are promoted to Hololens and corresponding 3D bounding boxes are drawn to visualise the detected objects.

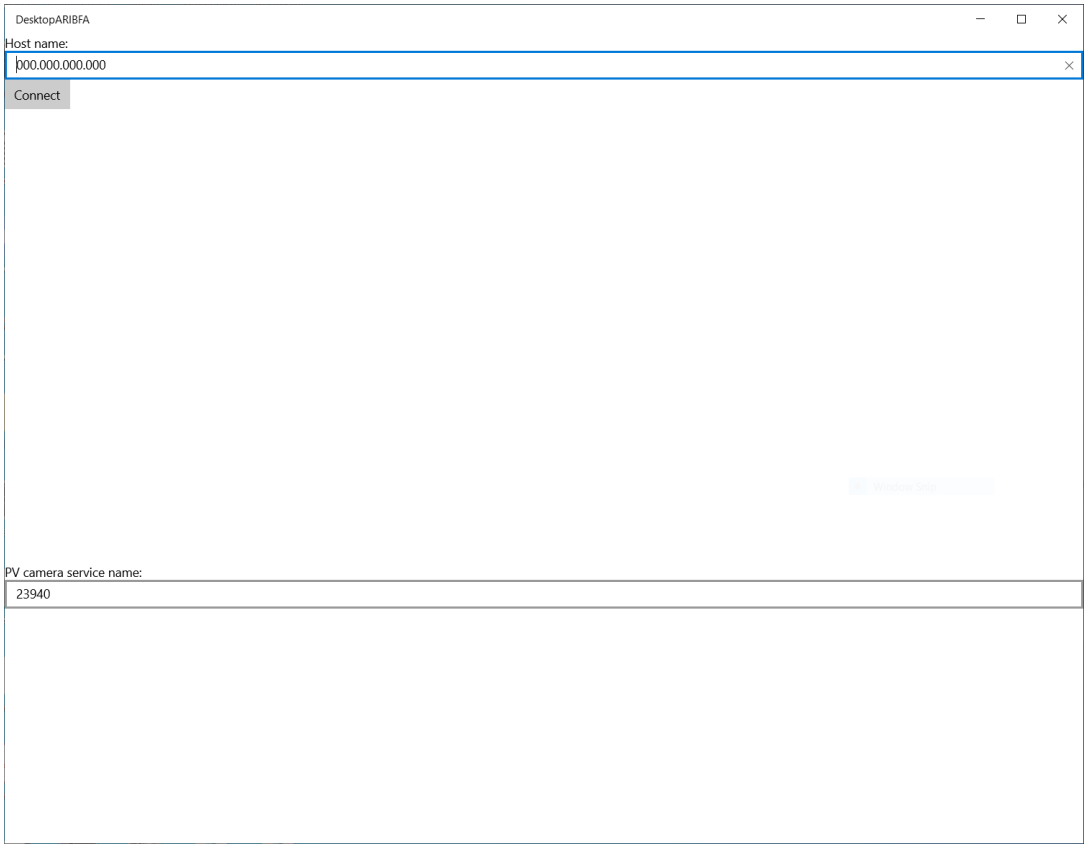


Figure 33: Main window of DesktopARIBFA.

4.5.4.2 MEP COMPONENTS DETECTION EXAMPLES

In Figure 34, sample object detection results using 2D bounding boxes are presented for object detection on MEP components via the Marker-less Feature Recognition submodule of

the ARIBFA application. The probability of the predicted class is also depicted in the bounding box. This representation that utilizes 2D bounding boxes demonstrates how the detection is visualised in the video streaming in DesktopARIBFA application. The actual representation of detected 3D objects in the ARIBFA application is presented in Figure 37. The detected objects are visualised using 3D bounding boxes that have hand manipulation handlers. Using hand manipulation, the users can adjust and optimize the scale, the position, and the rotation of the bounding box, as demonstrated in Figure 35 and Figure 36.

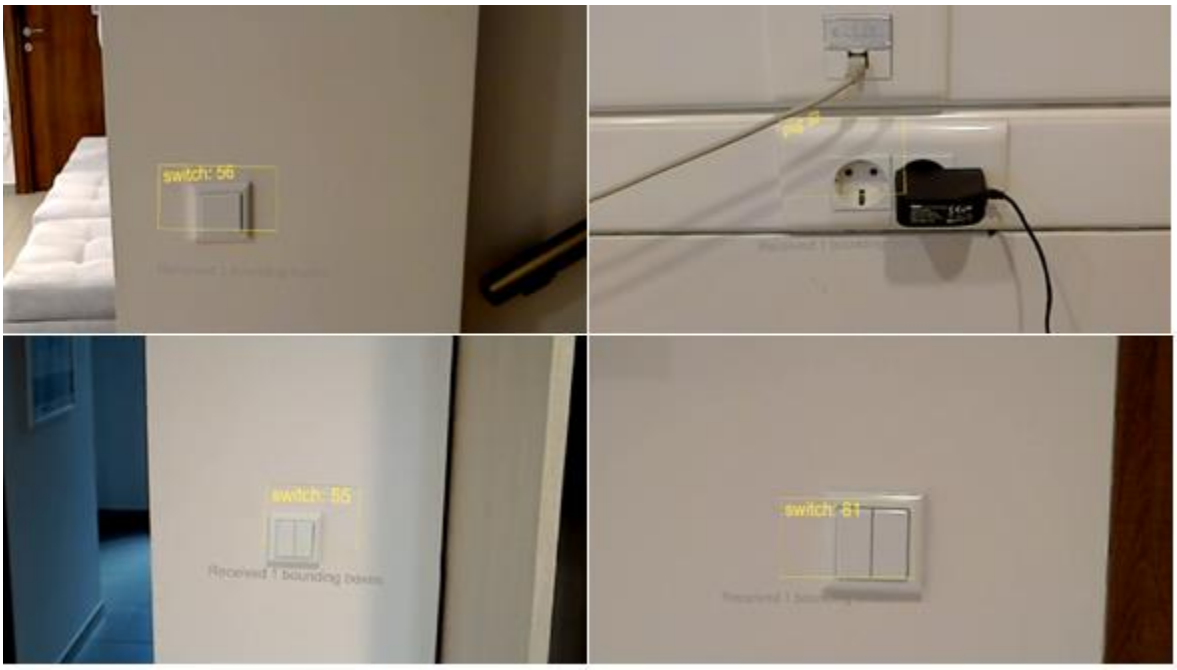


Figure 34: Images depicting detected MEP components using 2D bounding boxes in DesktopARIBFA application.

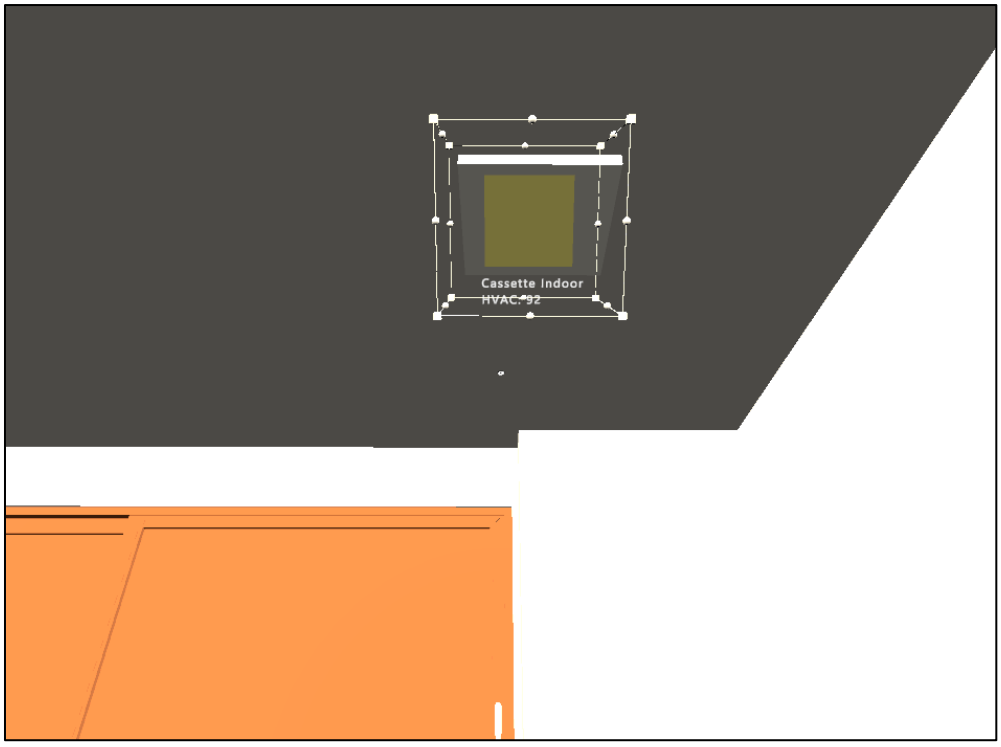


Figure 37: Demonstration of how a detected component is visualised using a 3D bounding box in the ARIBFA application.

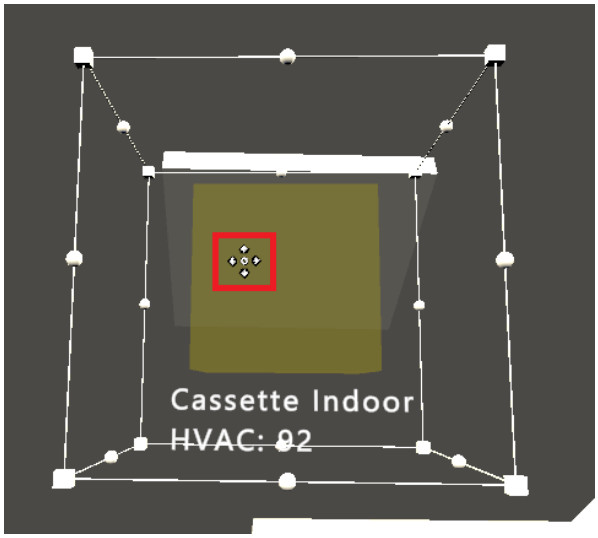


Figure 35: When the user moves their hand on the bounding box, this icon means that they can change the position of the bounding box in space in 4 directions.

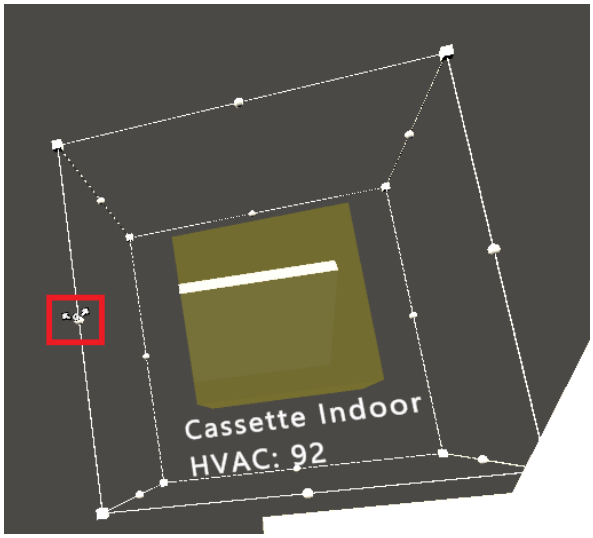


Figure 36: When the user moves their hand on the side handles of the bounding box, this icon means that they can change the rotation of the bounding box.

4.5.4.3 OBJECT DETECTION PIPELINE

Object detection pipeline is initiated in the ARIBFA application at runtime with speech command “Detect”. In order not to interrupt the user’s AR experience, the paired computer’s IP is requested via ARIBFA at start time. The GUI for requesting paired computer’s IP is depicted in Figure 38. When the user enters IP and selects the “Submit” button, the IP is stored locally on Hololens so that it can be loaded at subsequent sessions, without requesting input from the user. More specifically, at start time ARIBFA checks if there is a stored IP for a paired computer. If there is no IP stored, the menu in Figure 38 is presented to the user to request one. If there is a stored IP, ARIBFA uses this IP to pair for object detection. At any time, the user can change the paired computer’s IP using speech command “IP” that activates the menu in Figure 38.

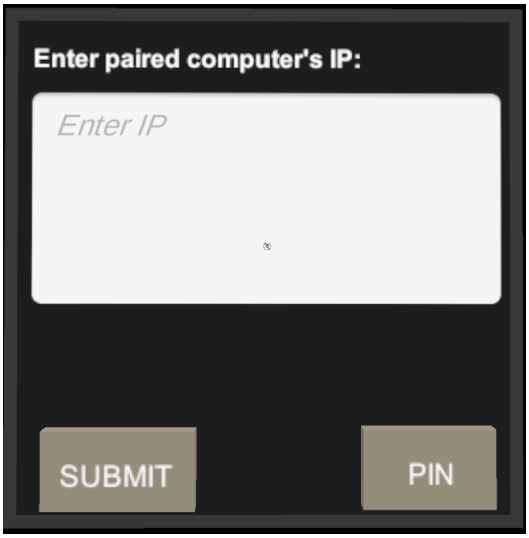


Figure 38: The ARIBFA application‘s GUI to request paired computer’s IP for object detection.

The ARIBFA application uses the paired computer’s IP, given at start time by the user, to establish a connection between Hololens and the local computer. Therefore, at any time while ARIBFA runs, the user can run DeskopARIBFA to receive camera stream from Hololens. To perform object detection in the ARIBFA application and receive bounding boxes from the local desktop, the speech command “Detect” is used. This command initializes the reception of bounding boxes in Hololens and their visualisation in 3D space.

As mentioned, the object detection process is initialized with the speech command “Detect”. Subsequently, 2D bounding boxes are received in Hololens for detected MEP components, as depicted in Figure 34. These bounding boxes are visualised as 3D bounding boxes in 3D space by ARIBFA, as demonstrated in Figure 37. Upon detection, Hololens stops receiving new bounding boxes so that the user can process the detected 3D bounding box using manipulation handlers, as depicted in Figure 35 and Figure 36. When the user has finished the manipulation of the 3D bounding box, they can open the GUI (presented in Figure 40) to insert IFC properties to the detected object with the speech command “Add properties”. When the user has finished inserting the IFC properties for the detected object, they can add the detected object to the BIM model using the speech command “Add to model”. This can also be accomplished by tapping on the “Add” button in the GUI, as depicted in Figure 40. The object detection pipeline is graphically illustrated in Figure 39.

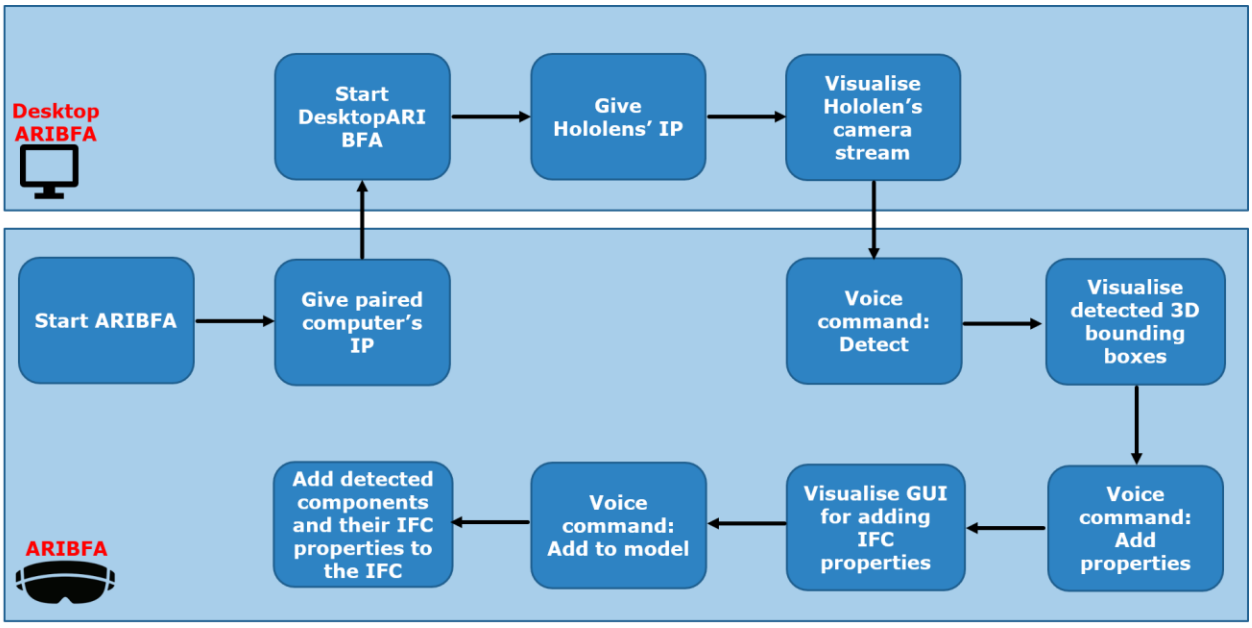


Figure 39: Object detection pipeline.

4.5.4.4 ADDITION OF IFC PROPERTIES TO DETECTED COMPONENTS

When the user has completed hand manipulation on the 3D bounding box (i.e., the bounding box is optimized in terms of scale, position, and rotation), they can add IFC properties to the detected objects using appropriate GUIs. The speech command "Add properties" opens up suitable menus so that the user can insert IFC properties to the detected objects. Different menus are attached to indoor HVAC components, outdoor HVAC components, plugs, and switches, depending on the detection label. It should be noted that the IfcType is automatically given to the detected object based on the detection label. More specifically, the detected HVAC components are given the "IfcUnitaryEquipment" type, the detected plugs are given the "IfcOutlet" type, and the detected switches are given the "IfcSwitchingDevice" type. Menus to add IFC properties to detected indoor and outdoor HVAC components are depicted in Figure 40 and Figure 41, respectively. Each menu depicts the IfcType of the detected component and the detected label. Moreover, each menu requires input from the user for IFC properties that are relevant to the detection label.

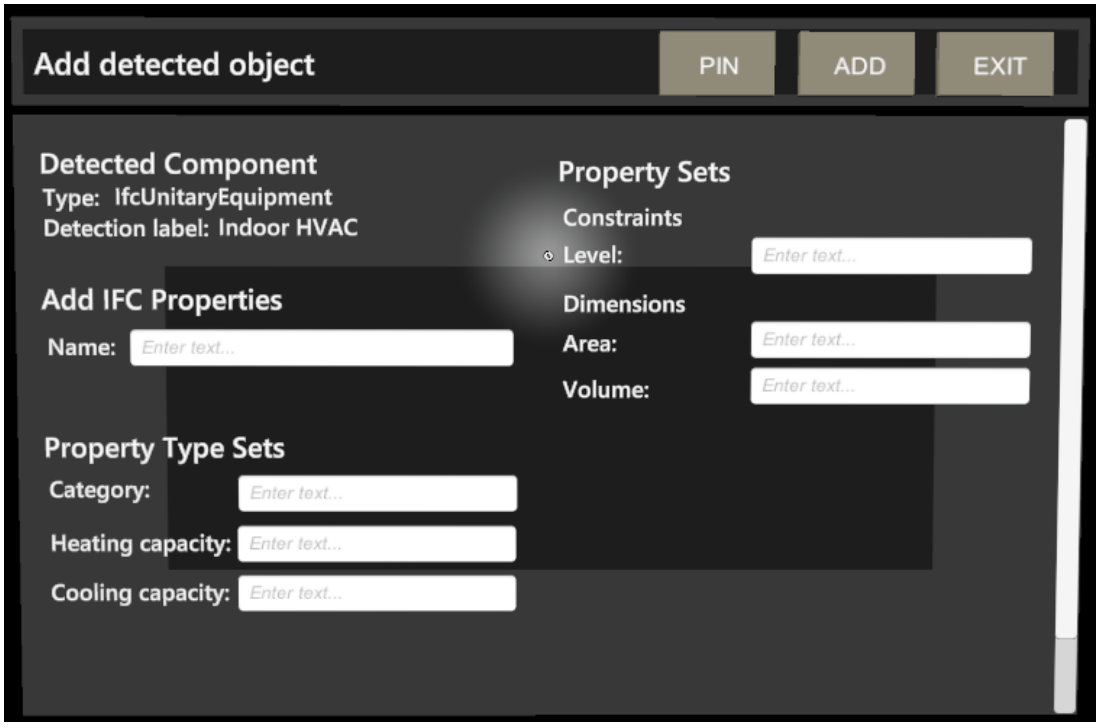


Figure 40: Menu to add detected indoor HVAC components to the BIM model.

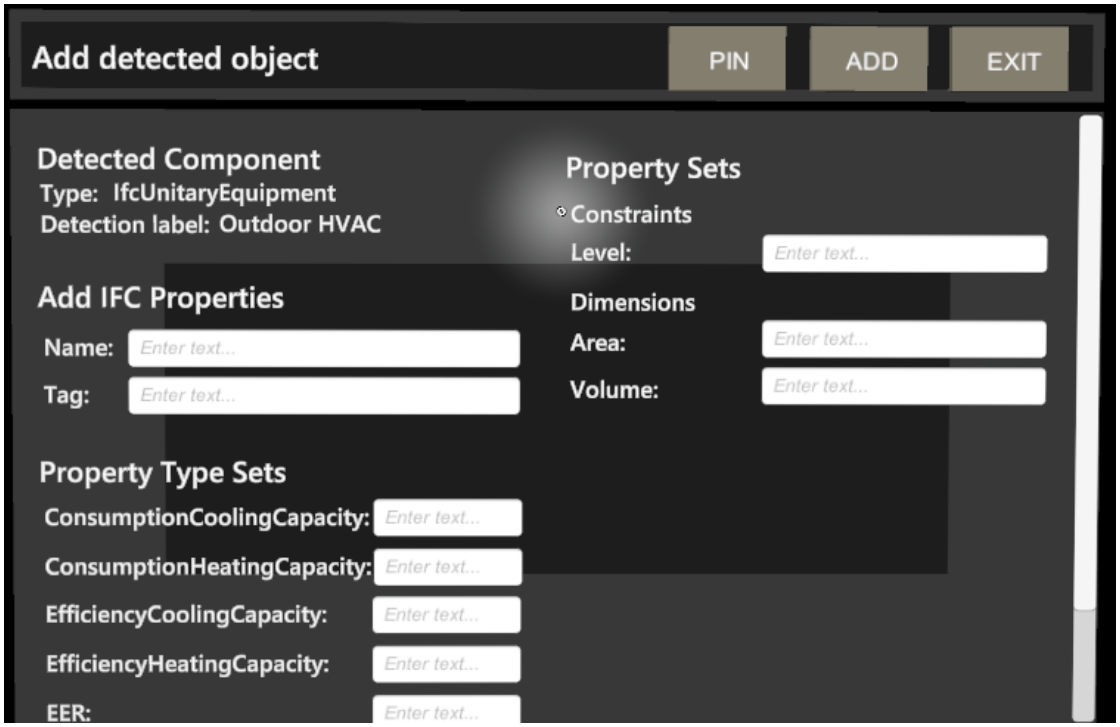


Figure 41: Menu to add detected outdoor HVAC components to the BIM model.

4.5.4.5 ADDITION OF DETECTED OBJECTS TO BIM MODEL

As can be seen in Figure 40 and Figure 41, in the top right corner of each menu for adding IFC properties to detected objects, the "Add" button adds the detected object along with the inserted IFC properties to the BIM model. As mentioned, the type of the component is set according to the detection label, i.e., the detected HVAC components are given the "IfcUnitaryEquipment" type, the detected plugs are given the "IfcOutlet" type, and the detected switches are given the "IfcSwitchingDevice" type. When the "Add" button is selected, the IFCdata component is added to the detected object and the detected object is added to the IFC file, belonging to the IfcSpace it was found, in location relative to it. In addition, the Name, the Property sets, and the Property Type Sets are added.

The detected object is represented by a box GameObject in the Unity environment. The geometry of the box is extracted by the collision box attached to it. Then, in the IFC file the geometry representation is created as "Extrude Area Solid" based to the extracted Box

Geometry. The updated version of the BIM model is created in Hololens storage and is forwarded to BIF, as discussed in detail in Section 5.2.

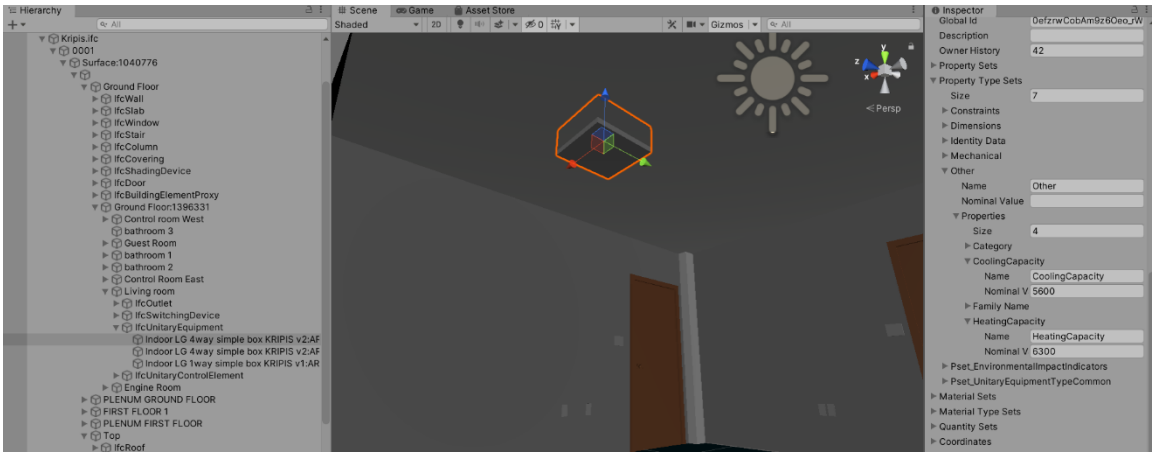


Figure 42: Upon detection of unregistered element, a simple box geometry is created to represent the object.

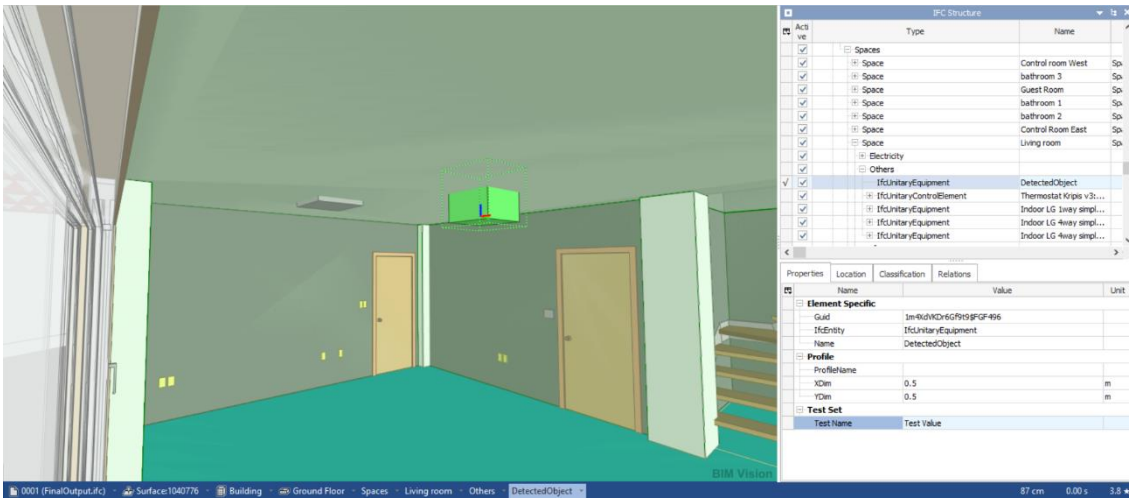


Figure 43: After updating the IFC file, the detected object is now a part of the IFC model, as visualised in BimVision viewer.

Figure 42 shows the interior of the living room of the Kripis BIM model in the Unity Game Scene. The yellow box represents a detected object of type IfcUnitaryEquipment that does not belong in the original IFC file. The box has the IFCData component attached, where the Type is defined based on the detection label. Also, the Name and one sample property set (Test set in the example) are also defined. In Figure 43, the BimVision viewer shows the detected object added in the IFC file as IfcUnitaryEquipement with the name DetectedObject

and a property set with some dummy value, belonging in the IFC hierarchy as a child of the IfcSpace where the detection took place (in this case the “Living room”). It is worth noticing that the coordinate system of the Unity environment is different to the coordinate system of the IFC, so in order for the object to be in the correct location, the proper translation of the Unity coordinate system to that of the IFC is necessary.

4.6 AR ANNOTATION & CONTEXT AWARE-VISUALISATION SUBMODULE

4.6.1 Overview

The main goal of the augmented reality annotation and context-aware visualisation sub-module is to use data extracted from the first few stages of development and provide it so that it can be viewed in a discreet interface that interacts with the user's environment. The real-time adaptability of this intuitive augmented reality user interface not only provides the ability to check BIM data, but also provides the ability to update it and/or create and store new data locally. The user can perform actions in a conventional way, such as touching a button in the floating augmented reality menu that appears in the HMD viewpoint with hand gestures or using the voice recognition function of the HMD modular user interface.

This sub-module simplifies user operations in the building area in the following ways, as shown in the following, updated from the previous version outlined in D5.9, diagrams by:

- Allowing users to look at and select building components of interest.
- Providing an interactive menu that displays the IFC properties of the selected building component.
- Giving the opportunity to view renovation tasks related to the user’s current location in the building space.
- Detailing an assigned task’s specifics related to the selected building component.
- Offering the ability to edit existing IFC attributes or add new AR text annotations on components of interest through the appropriate menus and panels.

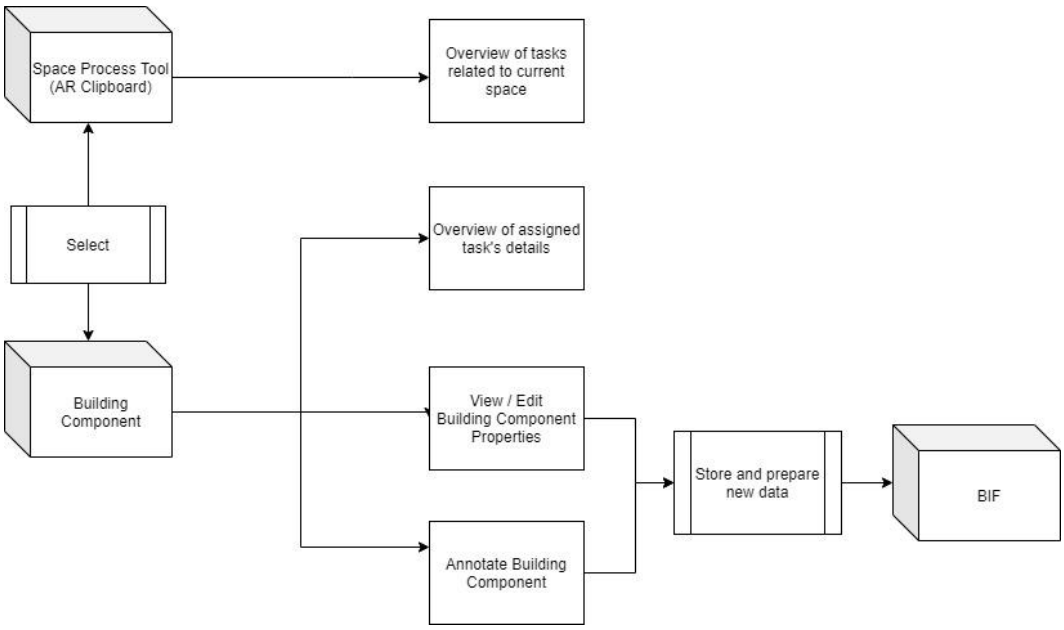


Figure 44: Overview of the AR Annotation & Context Aware Visualisation submodule's updated design.

Finally, after performing all user operations and creating and storing any new data, the sub-module provides it to the BIMERR Interoperability Framework (BIF) which uses the created data files and groups it for distribution to other components and modules of the BIMERR project.

4.6.2 Submodule Architecture and Workflow

The AR Annotation and Context Aware-Visualisation submodule offers an intuitive user-friendly experience in the form of a minimal and self-explanatory workflow. All the required functions and User Interface elements have been created and enhanced for use in an AR context so that they support the basic actions of Opening, Exiting, and Pining in the 3D space. Every menu transition and action are directly visualised in an appropriate to the user's HMD viewpoint position and orientation.

In this section a high-level outline of the various menus and subsystems that have been updated or newly created since the previous version of the tool, will be presented.

4.6.2.1 SPACE PROCESS TASK LIST TOOL

First, as a user walks in a building the Space Process Task List head's up display element becomes available according to the current space id. This tool, floats in the user's viewpoint at eye level and tracks their movements during the building space traversal. Users have the ability to air tap on the visual's tool hologram presented in the HMD's screen to bring up the Space Process overview panel which is filled automatically based on the current occupied space, with the relevant list of renovation tasks.

The architecture of the Space Process's Task list overview workflow is defined by the following diagram:

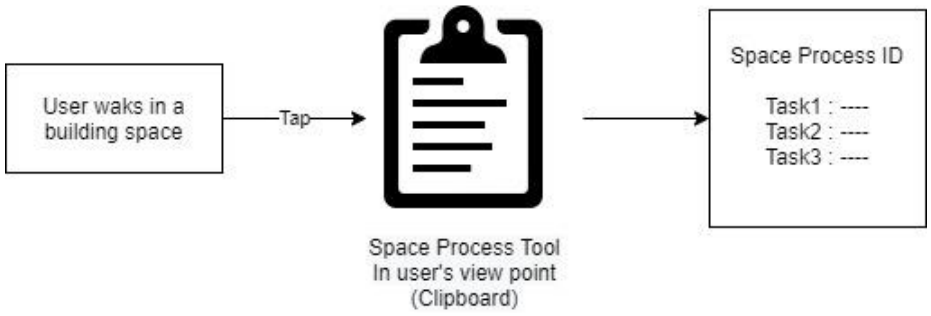


Figure 45: In depth architecture of the Space Process tool.

4.6.2.2 BUILDING COMPONENT

Every building component in the 3D space becomes selectable using the MRTK's IMixedRealityPointerHandler interface. In combination with the custom C# script "OpenPropertyMenu", a menu is brought up every time a user air taps on a building component. This menu is selected appropriately to match the type of the selected building component and includes the following categories:

- Structural
- Electrical
- HVAC Indoors

- HVAC Outdoors

The above categories dictate the form of the menu as well as the values displayed on it, which reflect the appropriate IFC Properties of the currently selected building component. The following figure outlines the architecture of what a building component means in the AR Annotation & Context Aware Visualisation tool's high-level context.

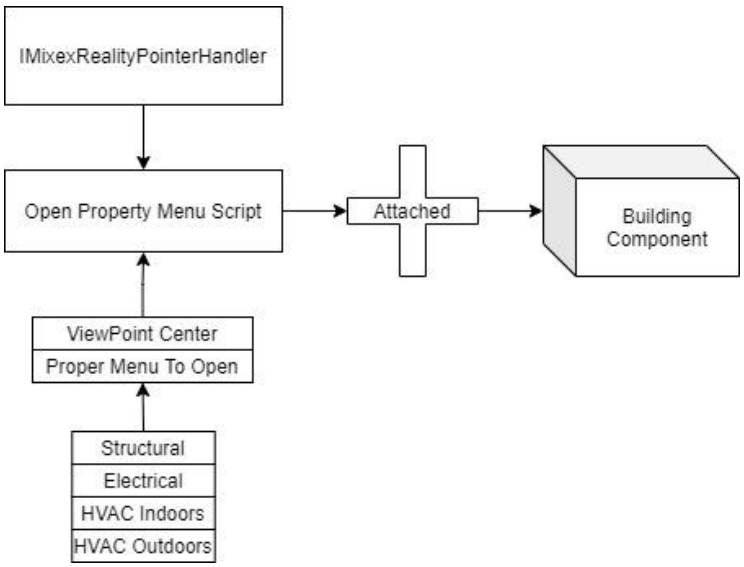


Figure 46: Building Component's architecture.

4.6.2.3 GENERAL USER INTERFACE BUTTONS AND MENU SYSTEMS

The designed and implemented menu systems and subsystems are resolved around a shared set of functions. These take the form of User Interface buttons including the following:

- **Exit / Home:** Enables the action of closing down a menu hiding it from a user's viewpoint.
- **Pin / Unpin:** Enables or Disables the BillBoard and RadialView (see Description in 5.6.2.4) mode of the AR floating menus that are responsible for following the orientation of the user's viewpoint while traversing the Building Space.

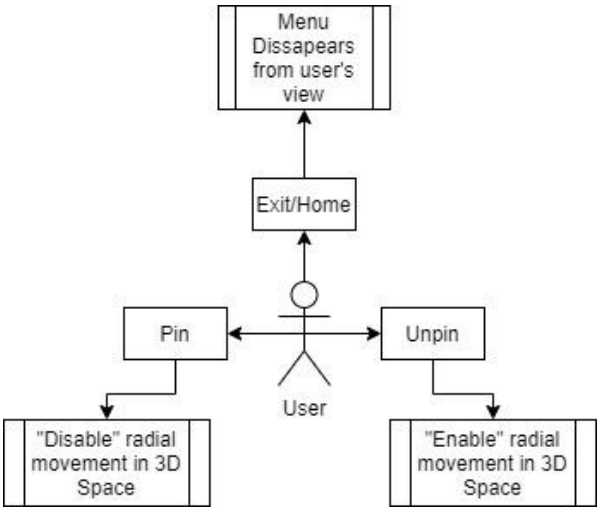


Figure 47: Shared Menu functions.

4.6.2.4 IFC PROPERTIES MENU

The IFC Properties Menu acts as the "Main Menu" of the AR Annotation and Context Aware-Visualisation submodule procedures. The appropriate menu type is defined in the "OpenPropertyMenu" Script attached to each Building Component and brought up once it is air tapped. The positioning and orientation of this menu is handled by the BillBoard and RadialView scripts provided by the MRTK's standard framework of tools and interfaces. These scripts ensure that the menu is always oriented properly according to the user's position and gaze, thus making it viewable at all times in the user's heads up display.

In addition, each type of Properties Menu owns a different set of displayed parameters uniquely assigned to the four different types of building elements the user might encounter during a visit to a building space, including:

- **PropertiesMenuStatic:**
  - **IFC Properties:** Label, Type, Global ID, Name
  - **PropertySets:** Area, Volume
  - **Constraints:** Level

    - **Material Sets:** Material1, Material2…MaterialN etc
- **PropertiesMenuElectric:**
    - **IFC Properties:** Label, Type, Global ID, Name
    - **PropertySets:** Load, Load Classification, Switch Voltage
    - **Constraints:** Level
- **PropertiesMenuIndoorHVAC:**
    - **IFC Properties:** Label, Type, Global ID, Name
    - **PropertySets:** Cooling Capacity, Heating Capacity
    - **Constraints:** Level
- **PropertiesMenuOutdoorHVAC:**
    - **IFC Properties:** Label, Type, Global ID, Name
    - **PropertySets:** Consumption Cooling/Heating Capacity, Efficiency Cooling/Heating Capacity, COP, EER
    - **Constraints:** Level

All the above mentioned IFC Fields are filled using the appropriate PropertyMenuTypeFields(eg PropertyMenuElectricFields etc) based on the type of selected building component. This script loads the IFC field values from the IFC Data which results from the BIM 3D Model Visualisation submodule and places them in the appropriate User Interface Entries in the AR Menu panels.

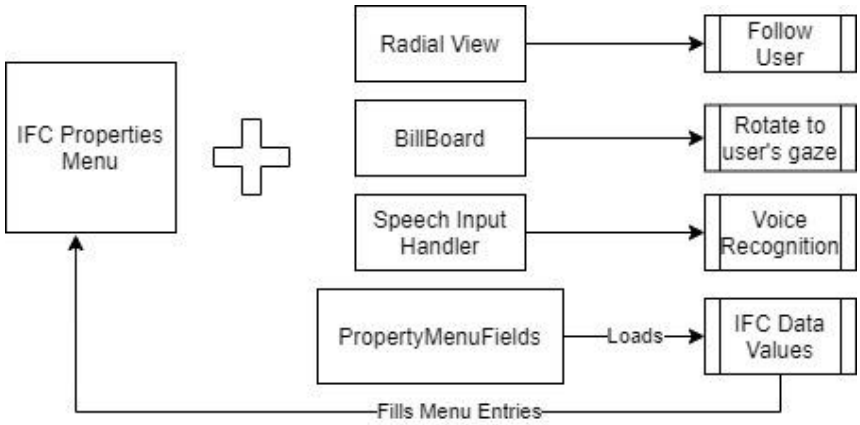


Figure 48: Properties Menu Architecture.

4.6.2.5 VIEW TASK DETAILS MENU

The View Task button becomes enabled in the appropriate Properties Menu if there is a relation between a member of the current renovation process task list and a building component. Once tapped the user can view the respective related task's details in an organized view.

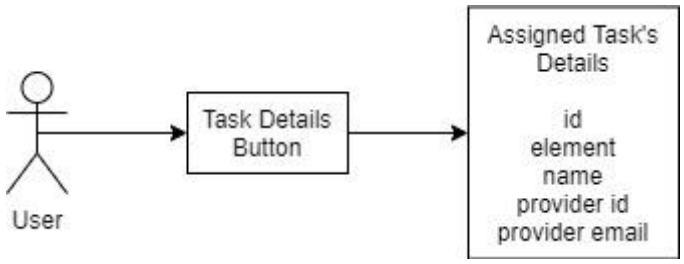


Figure 49: Task Details Button Design.

4.6.2.6 ADD ANNOTATION MENU

The Add Annotation Menu mirrors the basic IFC Properties field from the previous, main menu with the addition of an Input Field Canvas element. This element acts as the main data entry point in the user interface and provides the capability to type text annotations through a popped-up AR Keyboard. Each text annotation is stored in, an instantiated in the 3D Space, red floating annotation mark near the building component of interest.

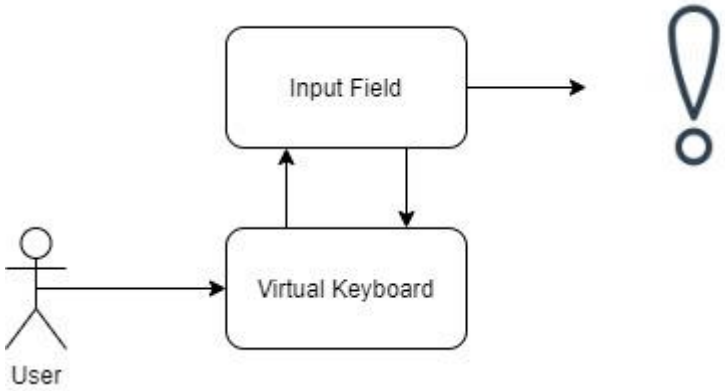


Figure 50: Add Annotation Menu Design.

4.6.2.7 VIEW ANNOTATION MENU

The ability to view and edit already created annotations is provided by the module using the View Annotation Menu. Once a floating red exclamation mark is encountered in the user's viewpoint and is selected/tapped, the View Annotation Menu is brought up. This menu owns an input field which displays the currently saved text annotation string and once clicked it can be edited by the user. Additionally, a Delete button is offered which once selected disposes the currently saved text annotation and the 3D exclamation mark from the user's surroundings.

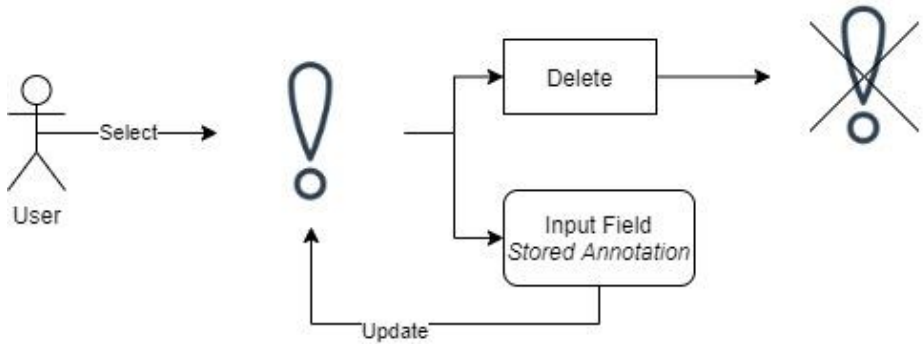


Figure 51: View Annotation Menu Workflow Design.

4.6.2.8 ANNOTATION CREATION SYSTEM

The Annotation Creation System is governed by the Annotation Manager script in the form of the Annotation Manager static class. This class features a function called "CreateAnnotation" which takes as input the current position of the user, the currently selected building component, and the text to be stored inside the floating red exclamation mark. Next the same function performs Linear Interpolation, which is a mathematical function that returns a value between two others at a point on a linear scale, on the Vectors of the user's and the position the user tapped on the building component. Thus, the floating exclamation mark is placed in an intuitive manner near the annotated building component while also staying in the user's viewpoint. This update to the previous workings of the

annotation creation system presented in the last deliverable, opens up the possibility to place multiples of annotation-exclamation marks in a more visually appealing way.

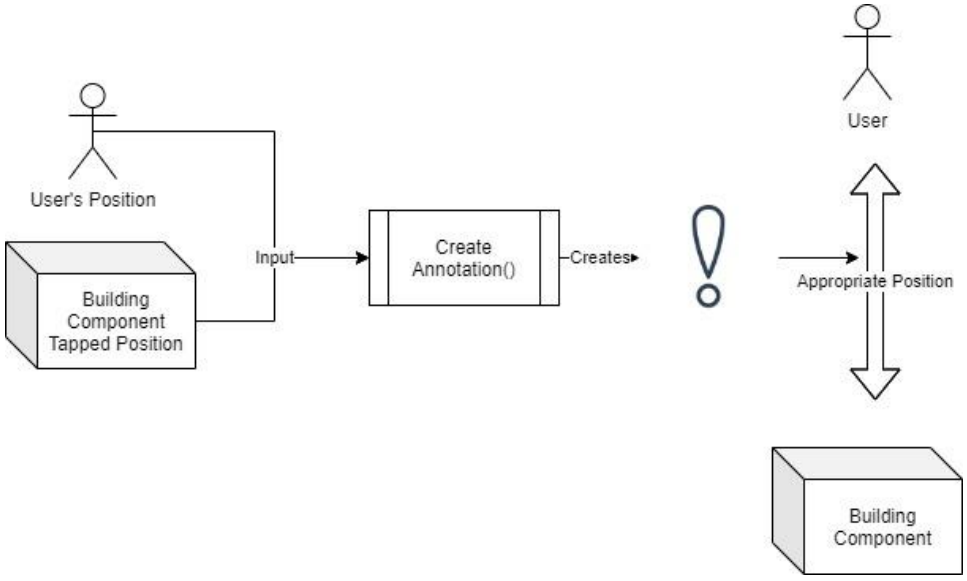


Figure 52: Annotation Creation System.

4.6.3 Development Tools

In this section, the development toolset that was used for the development of the AR Annotation & Context Aware-Visualisation submodule of the ARIBFA application is described. Table 6 summarizes the tools and libraries used for the implementation of the submodule. The block diagram of the software and hardware components utilized in the development of the submodule is presented in Figure 28.

Table 6: Tools and libraries used in AR Annotation & Context Aware-Visualisation submodule.

Library/Tool	Version	Licence
Microsoft Hololens	1st (gen), SDK 10.0.17763.0	-
Windows Mixed Reality	SDK 10.0.19041.2034	*Software licensing*
Unity 3D Editor	2020.5	*Software licensing*
Mixed Reality Toolkit (MRTK)	2.5.3	*Software licensing*

4.6.4 Implementation Features

4.6.4.1 OVERVIEW

After the initial scope and version of the module was defined, refactoring and updates begun on the components developed during the procedures of D5.9. Specifically, the central menu management system was refactored from the ground up using development patterns to ensure an even more rapid, flexible, and scalable iteration process between the application's versions. These changes empowered the addition of the new menu buttons, tools, and the underlining implementations of them.

4.6.4.2 MENU HANDLING

Unity3D prefabs have been utilized to create re-useable menus and buttons under a central Menu Manager object in the application's scene. The Menu Manager serves as the central hub which facilitates the Menu opening, closing and stores details such as the currently selected building component and tapped position on a building component. These stored data are used for proper menu positioning as well as starting points for the initialization and filling of the various menu entries and fields.

The Menu Manager takes the form of an empty Unity3D GameObject with the Menu Manager utility script attached to it. In addition, the Menu Manager serves as the main parent of the various menus and tools which act as children of the central manager game object, in the Unity Hierarchy Graph. The list of items of game objects included as children of the Menu Manager central parent includes the following:

- **PropertiesMenuStatic/Electric/IndoorHVAC/OutdoorHVAC:** Visualises IFC properties.
- **AddAnnotationMenu:** Handles the AR Text Annotation Creation of building components.
- **ViewAnnotationMenu:** Displays the stored text annotation info.
- **ShowAvailableTasksButton:** HUD Clipboard tool that enables the ViewAvailableTaskMenu.

- **ViewAvailableTaskMenu:** Provides a detailed list of tasks in the current Space Renovation Process.
- **TaskDetailsMenu:** Offers an analytical view on the specifics of an assigned task.

There are four different types of basic Properties Menu responsible for visualising the IFC Data of the building components as outlined in 5.6.2.4. The various type of Properties Menu become visible once the desired building component is selected via an air tap from the HMD's user. Proper positioning inside the user's field of view is ensured via the Billboard and RadialView components attached to the Menu's GameObject.

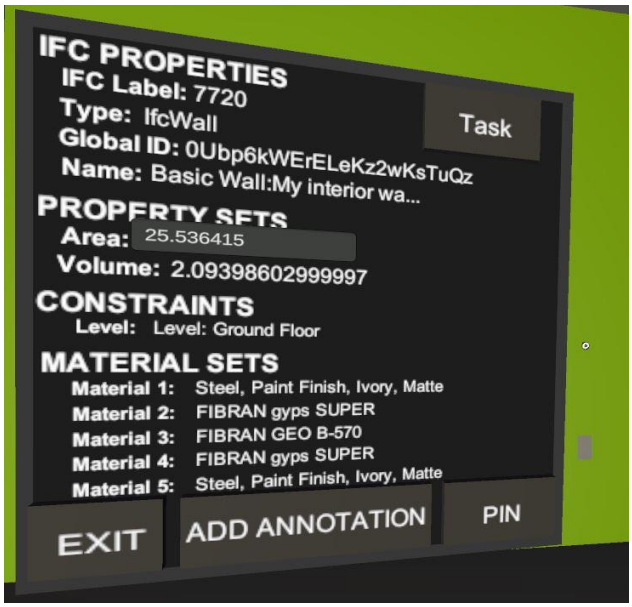


Figure 53: Static Properties.

Figure 54: Electrical Properties Menu.

Figure 56: Indoor HVAC Properties Menu.

Figure 55: Outdoors HVAC Properties Menu.

A highlighting function has been implemented. Its role is to facilitate the altering of the materials of the 3D Model Representation of a selected building component, so it appears colored in the user's viewpoint. Thus, making it easily disguisable while a user is navigating the building's internal spaces and viewing a building component's IFC properties.

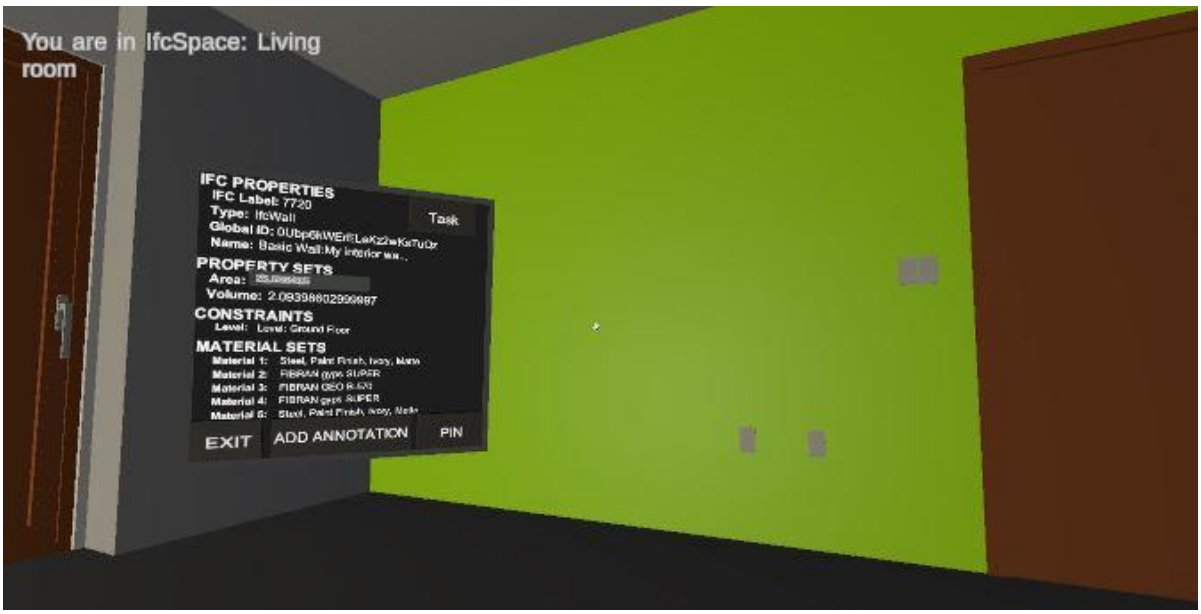


Figure 57: Building Component Highlighting.

4.6.4.3 TASK VISUALISATION

The ShowAvailableTasks Button is a floating and interactable 3D Model that exists in the user's viewpoint and follows them during a renovation process inside a building zone, by taking advantage of the features the MRTK RadialView and SolverHandler components offer.

Figure 58: ShowAvailableTasksButton as seen in the Hololens device.

The ViewAvailableTasksMenu is enabled in the application's scene once the ShowAvailableTasksButton is tapped as mentioned in the previous section.

Figure 59: ViewAvailableTasksMenu.

The specifics of a particular assigned task are available through the TaskDetailsMenu which becomes visible once the **Task** button (as seen in 5.6.4.2) is tapped in the Properties Menu of a building component.

Figure 60: Populated Task Details Menu.

4.6.4.4 **ADD ANNOTATION**

Compared to the first version of ARIBFA described in Deliverable D5.9, the final version of ARIBFA application presented here employs annotation menus that are based on the annotation data model. The annotation data model is based on the Annotation Objects ontology that aims to represent the annotations produced during the development of a building renovation project. The annotation objects match a JSON file structure and are firstly stored locally on Hololens internal storage in JSON format. Subsequently, they are promoted to BIF, where they can be accessed by other BIMERR applications. In the second and final version of ARIBFA, the menu for adding annotations to building components is adapted to the annotation model, as depicted in Figure 61. Input from menu in Figure 61 is suitably stored to a JSON file in Hololens storage and is forwarded to BIF using the API described in Section 5.3.

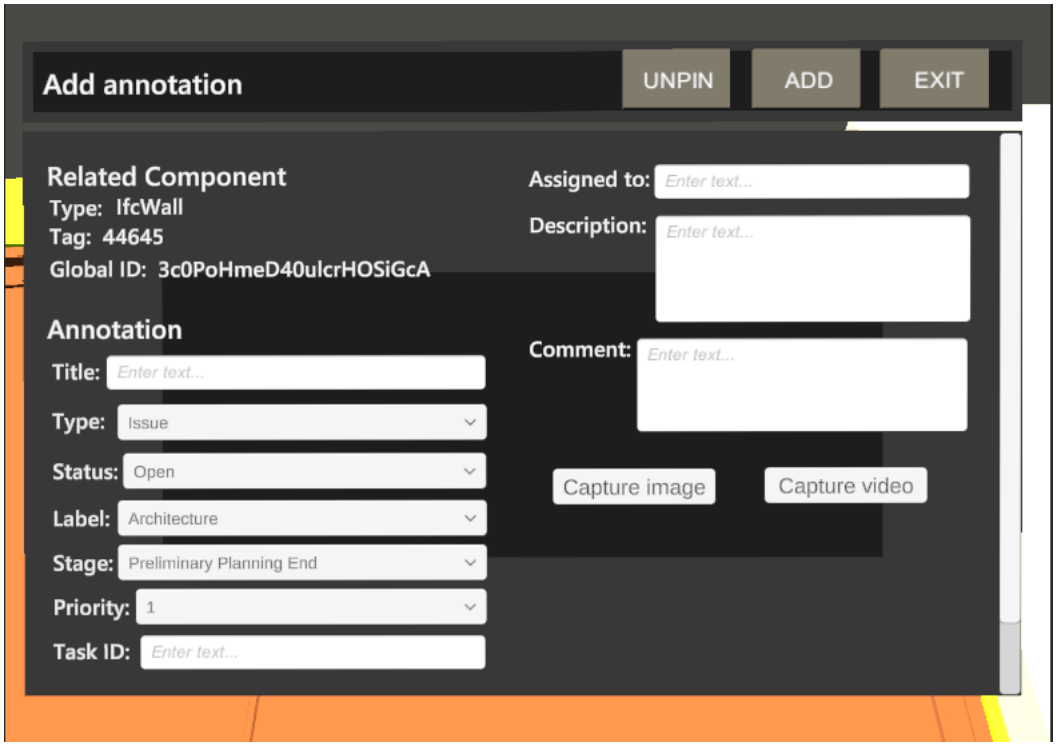


Figure 61: Menu for adding annotations which is adapted to the annotation data model.

Drop-down list for menu fields. The user can add annotations on building components by performing the air tap gesture on the 3D objects. The add annotation menu opens-up so that the user can fill in the appropriate annotation fields. More specifically, in the top left corner of the new annotation menu, basic IFC properties such as the IfcTtype, the tag, and the global id of the building components are displayed. As can be seen in Figure **61**, the new annotation menu requires keyboard inputs for fields, such as the "Title", the "Assigned to", and the "Description" fields. For fields with predefined set of values, such as the "Type" and the "Status" fields, drop down lists are implemented to assist the user to select a value. As demonstrated in Figure 62, if the user clicks on "Type" field, a drop-down list opens up with the possible values for this field, i.e., "Comment", "Issue", "Request", and "Solution". Predefined drop-down lists are also implemented for field "Status" (possible values: "Open", "In Progress", "Closed", and "ReOpened"), for field "Labels" (possible values: "Architecture", "Structure", "Mechanical", "Electrical", "Specifications", "Technology"), for field "Stage" (possible values: "Preliminary Planning End", "Construction Start", "Construction End"), and

for field “Priority” where the priority of the topic is indicated by an integer number in range 1-4. Since the annotation menu is large to include all annotation data model properties, scroll bars are implemented, as depicted in the Figure 63, to assist the user to efficiently focus on the desired menu items.

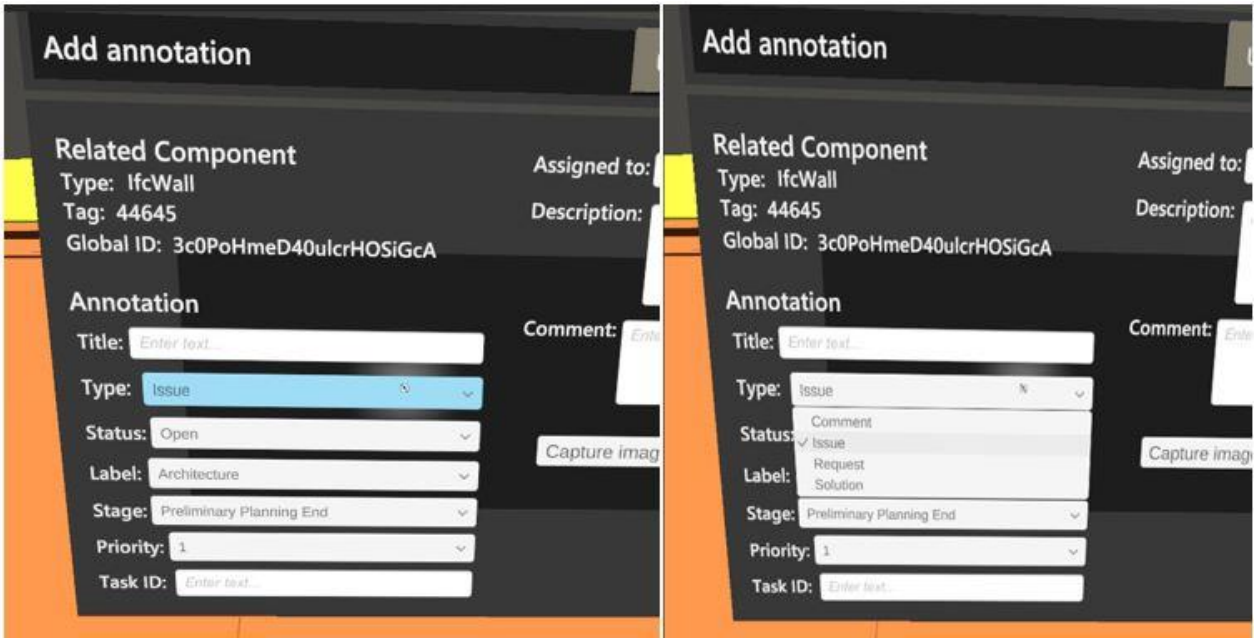


Figure 62: Drop-down menus are implemented for annotation fields with predefined values to assist the user complete the annotation.

Attach images/videos to an annotation

Moreover, the user can capture images and videos from Hololens and attach them to the annotation object by clicking on the “Capture image” and the “Capture video” buttons, respectively. A close-up of these buttons in the add annotation menu is depicted in Figure 64. When the user clicks on the “Capture image” button, ARIBFA anticipates the user to take a screenshot. In Hololens, the user can take a photo of their current view by pressing the Volume up and Volume down buttons on their HoloLens device at the same time. This is actually the Hololens version of taking a Mixed Reality screenshot, since AR content is visualised in the captured screenshot. If the user clicks on “Capture image” button and subsequently, takes a screenshot, ARIBFA correlates this screenshot to the annotation. The same applies for the “Capture video” button that attaches a video file to the annotation. In

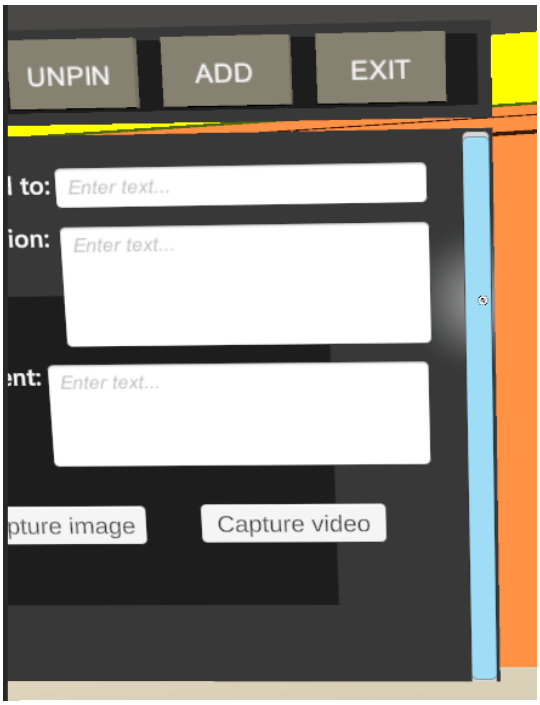


Figure 63: Scroll bar to assist viewing.

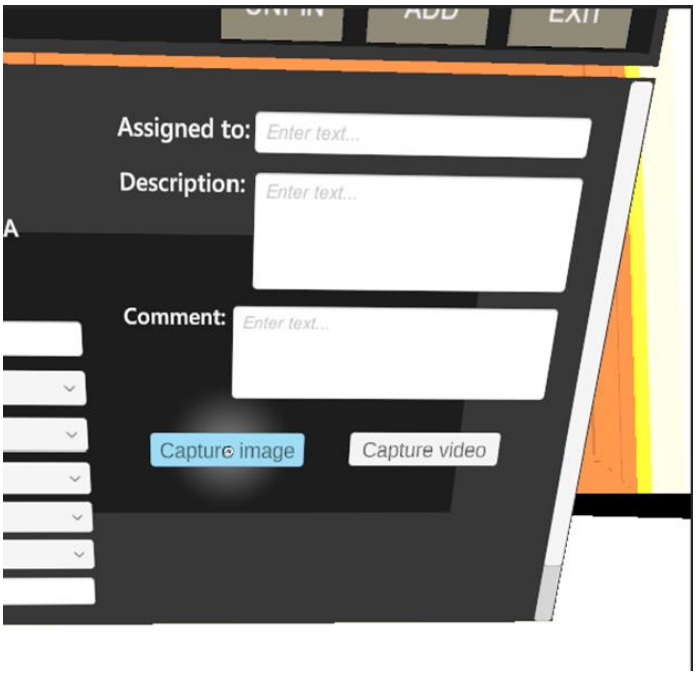


Figure 64: Buttons to attach captured images and video files to the annotation.

Hololens, the user can record a video by pressing and holding the volume buttons simultaneously until a countdown of three seconds begins. To stop recording, the user must tap both buttons simultaneously. These capturing capabilities of Hololens, along with other important device functionalities, are described in detail in Section 7.1.1.2. The captured images and videos are stored in Hololens internal storage. Subsequently, the binary image and video files are sent to BIF using the API described in Section 5.3.

The created Text Annotation is transformed and placed appropriately in the scene's hierarchy in the form of a red exclamation mark instanced prefab, and it stores information such as the text and the building component it is attached to.

Figure 65: Annotation mark in the 3D BIM Model.

The main stored text annotation info becomes available once the exclamation mark is selected via an air tap or the appropriate voice command.

Figure 66: ViewAnnotationMenu.

4.6.4.5 IFC VALIDATION

An important ARIBFA use case is to inform the user regarding invalid or missing IFC properties in the BIM model and provide the user the capability to correct or enter the missing IFC properties. For this purpose, ARIBFA receives error reports from BIF. The error reports are in JSON format and include IFC entities that are invalid or missing for specific types of building components. More specifically, ARIBFA receives error reports for missing/invalid properties in HVAC components, for missing/invalid thermal properties, and for missing/invalid properties in IfcSpaces. ARIBFA receives and processes the error reports in JSON format from BIF at runtime. The building components with missing properties are visualised in different color to notify the user. Relevant valid properties are also included in the error report and are displayed to assist the user to decide values for the invalid/missing properties. An error report menu for an HVAC component is depicted in Figure 67. In the left column of the menu, relevant valid IFC properties are displayed along with their values. In the right column of the menu, the missing property is displayed, and a value is requested from the user. By selecting the "Add" button at the top of the menu, the missing IFC property is added to the IFC. Subsequently, the updated/corrected IFC file is forwarded to BIF via the API described in Section 5.2.

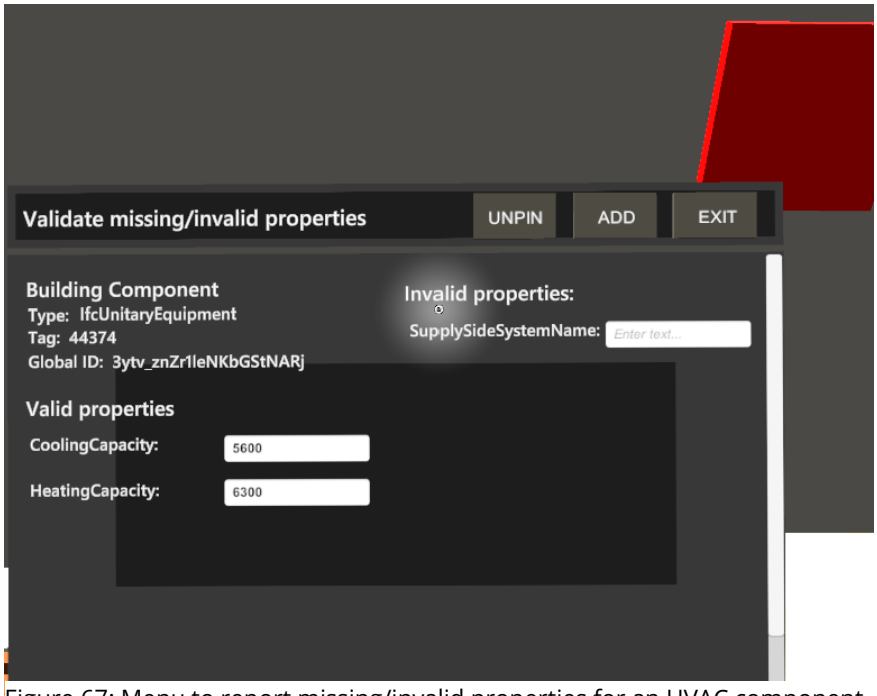


Figure 67: Menu to report missing/invalid properties for an HVAC component.

5 API DOCUMENTATION

5.1 OVERVIEW

The BIMERR Interoperability Framework is used to facilitate the various data collection and exchanges from and between the set of the tools and services in the BIMERR ecosystem. These data exchanges are based on the concept of requests which is related to the concept of the Representational State Transfer. This concept is a software architectural style which uses a subset of the HTTP protocol. It is mainly used to create interactive applications that use Web Services.

In the context of BIMERR, after a data collection job is created using the BIF's user interface, the information provided, related to the endpoints the request will be send, is visible.

Figure 68: BIF Upload to API endpoint.

Using Unity3D's **UnityWebRequest** and the endpoint URL provided by BIF's tools, a request can be constructed to upload data to the BIF's API. This process is the same for any other service/url we want to establish a connection to.

```
using UnityEngine;
using UnityEngine.Networking;
using System.Collections;

Unity Script | 0 references | 0 changes | 0 authors, 0 changes
public class MyBehavior : MonoBehaviour
{
    Unity Message | 0 references | 0 changes | 0 authors, 0 changes
    void Start()
    {
        StartCoroutine(Upload());
    }

    1 reference | 0 changes | 0 authors, 0 changes
    IEnumerator Upload()
    {
        WWWForm form = new WWWForm();
        form.AddField("myField", "myData");

        using (UnityWebRequest www = UnityWebRequest.Post("http://www.my-server.com/myform", form))
        {
            yield return www.SendWebRequest();

            if (www.result != UnityWebRequest.Result.Success)
            {
                Debug.Log(www.error);
            }
            else
            {
                Debug.Log("Form upload complete!");
            }
        }
    }
}
```

Figure 69: UnityWebRequest Example Code.

5.2 **BIM MODEL**

A collection to the BIF's platform needs to be setup so the BIM Model can be uploaded. The data type must be set to text and binary.

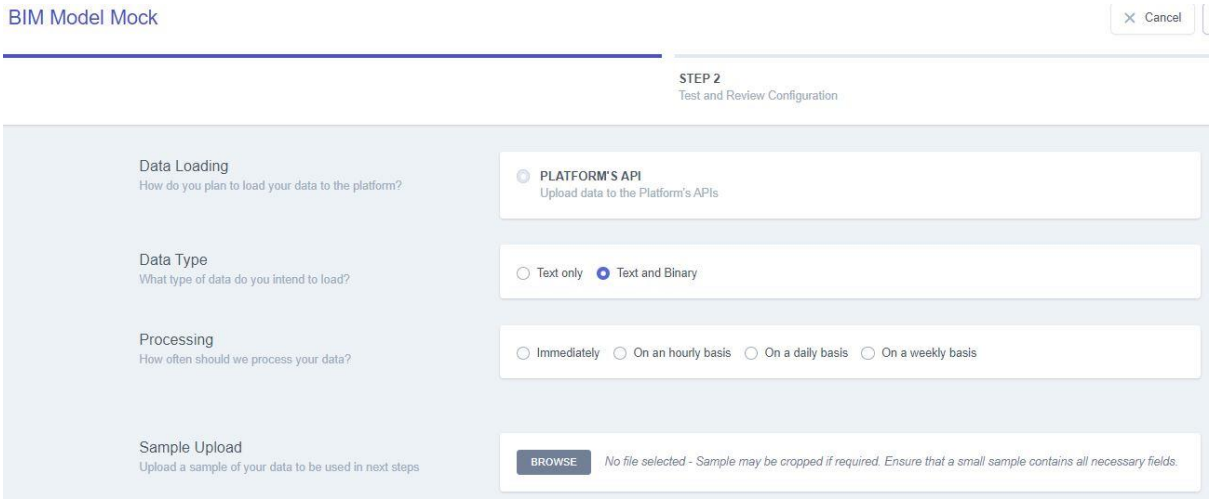


Figure 70**:** BIM Model collection job setup.

The REST request is set up according to the info about the endpoint that is provided in the final stage of the collection job setup.

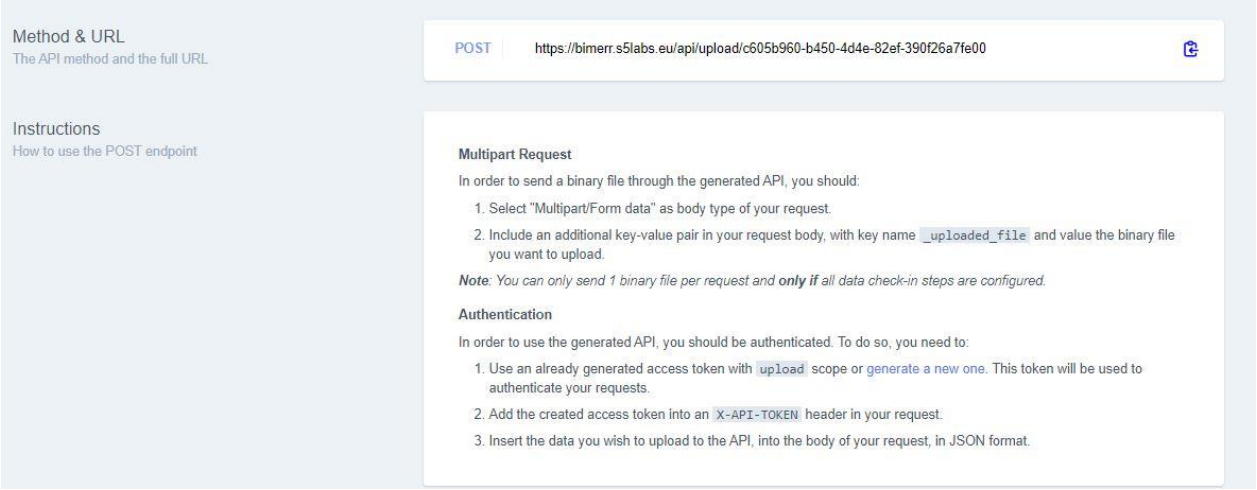


Figure 71**:** Endpoint and form info for binary upload.

5.3 ANNOTATION DATA

A collection to the BIF's platform needs to be setup so Annotation Data can be uploaded. In the first stage of the collection setup, the name and the pre-processing attributes of the job are defined.

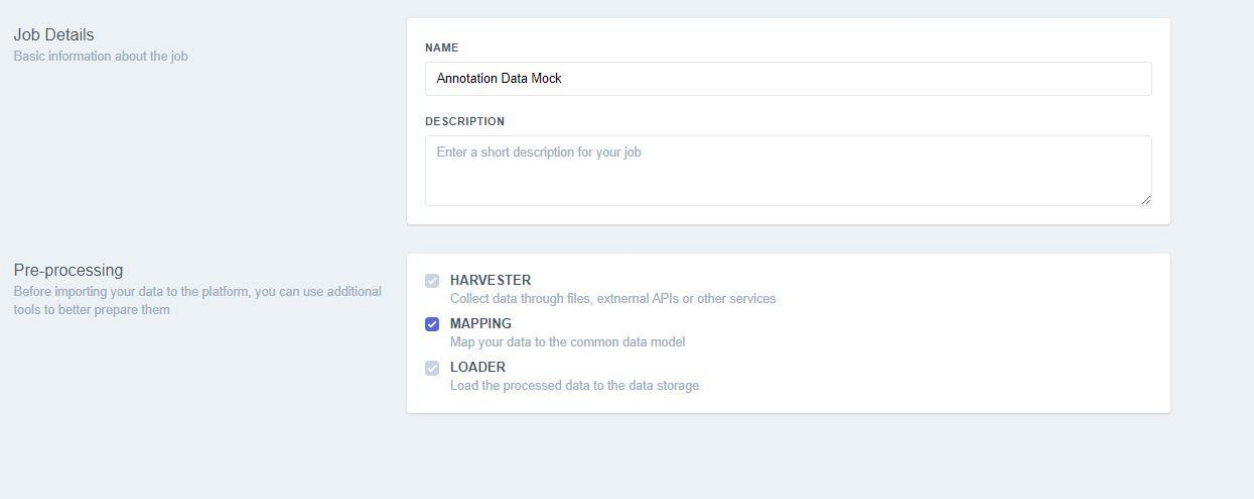


Figure 72**:** Annotation Data collection setup 1.

In the second stage of the setup, the data loading needs to use the **Platform's API** option so the direct uploading to BIF's internals is possible.

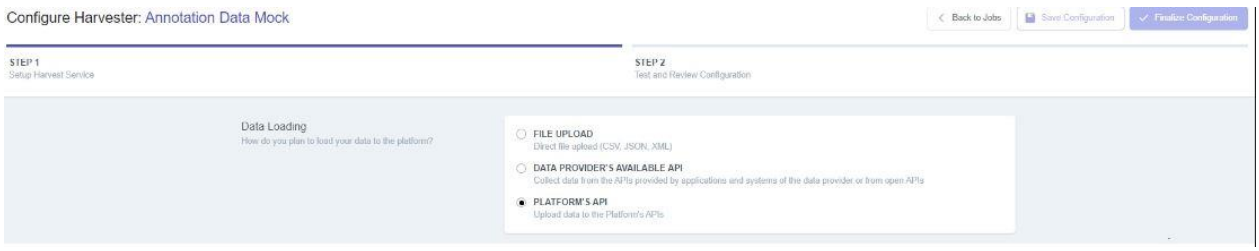


Figure 73**:** Annotation Data collection setup 2.

In the third stage of the setup, the data type and processing time are selected while a sample of the data to be uploaded is provided.

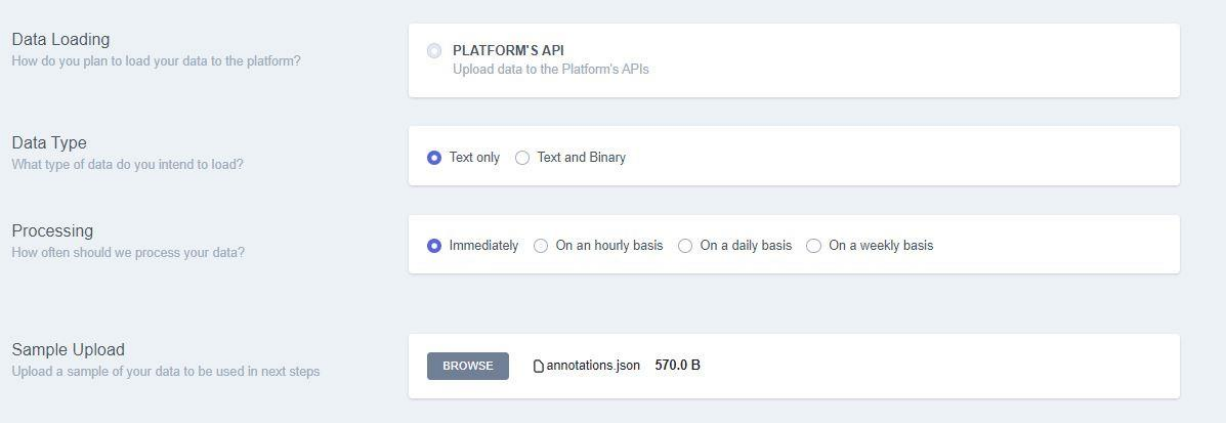


Figure 74**:** Annotation Data collection setup 3.

Finally, in the last stage of the collection setup the endpoint that will be used in the Annotation Data send request is created. In addition, an overview of the uploaded data is provided.

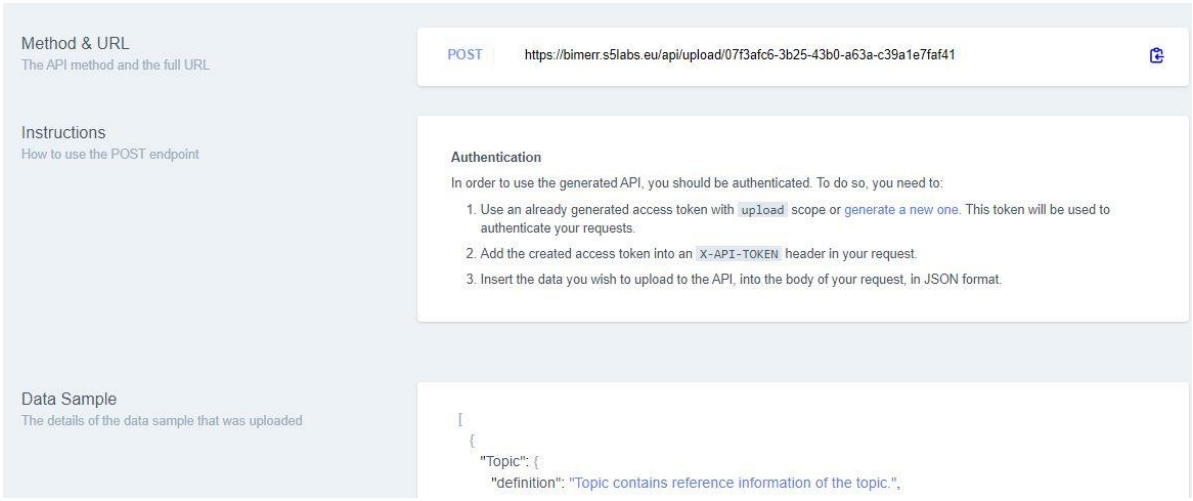


Figure 75**:** Annotation Data collection setup 4.

Once an Annotation is created based on user input from the appropriate menu, the locally parsed and agreed Annotation Data model is serialized and send as a REST POST request to the endpoint URL that was provided by BIF previously.

```
[Serializable]
7 references
public class AnnotationObject
{
    public string identifier;
    public Hastopic hasTopic;
}

[Serializable]
1 reference
public class Hastopic
{
    public string identifier;
    public string topicType;
    public string topicStatus;
    public string title;
    public string label;
    public string creationDate;
    public string creationAuthor;
    public string assignedTo;
    public string description;
    public string stage;
    public string taskId;
    public Relatedimagefile relatedImageFile;
    public Relatedvideofile relatedVideoFile;
    public Relatedaudiofile relatedAudioFile;
    public Relatedcomponent relatedComponent;
    public Relatedspace relatedSpace;
    public Relatedifcfile relatedIFCFile;
    public Hascomment hasComment;
```

Figure 76**:** Annotation Data model.

5.4 **USER LOCATION**

Once a user enters a building space the **IFCSpaceCollisionDetection** script calls the Create() function of the **SendUserLocation** script.

```
Transform[] collisionChildren = collision.gameObject.GetComponentsInChildren<Transform>();
foreach (Transform collisionChild in collisionChildren)
{
    // If user is in a Space and it's different than the previous one change location label
    if (collisionChild.gameObject.tag == "IfcSpace" && !collision.gameObject.name.Equals(spaceName))
    {
        spaceName = collision.gameObject.name;
        //print(spaceName);
        // Update current space GUI Label
        debugLabel.GetComponent<TMPro.TextMeshProUGUI>().text = "You are in IfcSpace: " + spaceName;
        // Get Space ID
        IfcDatas.IFCData roomData = GameObject.Find(spaceName).GetComponent<IfcDatas.IFCData>();
        // Create and send user location data
        GetComponent<SendUserLocation>().CreateRequest("Test-User", spaceName, roomData.GlobalId, Camera.main.transform);
```

Figure 77**:** IFCSpaceCollisionDetection Create Request.

The necessary info is provided in the CreateRequestMethod such as the spaceName, the room's id and the position info of the user. The data in this stage corresponds to the User Location Data Model which declares the class that will be instantiated and used as part of the UnityWebRequest to the chosen endpoint.

```
public class UserLocation
{
    public string id;
    public string space;
    public string spaceID;
    public float[] position = new float[3];
    public float[] rotation = new float[4];

    1 reference | Thanos Restas, 20 days ago | 1 author, 1 change
    public UserLocation(string id, string space, string spaceID, Transform transform)
    {
        this.id = id;
        this.space = space;
        this.spaceID = spaceID;

        position[0] = transform.position.x;
        position[1] = transform.position.y;
        position[2] = transform.position.z;

        rotation[0] = transform.rotation.x;
        rotation[1] = transform.rotation.y;
        rotation[2] = transform.rotation.z;
        rotation[3] = transform.rotation.w;
    }
}
```

Figure 78**:** UserLocation data model.

The **SendUserLocation** script which contains the CreateRequest method, takes the user's data and instantiates the necessary object based on the UserLocation Data Model. Next the SendRequest method is called. This method initiates the creation of the request to the chosen endpoint using a Coroutine, which asynchronously and independent of the rest of the

application creates the appropriate WWW Form. The URL of the endpoint as well as the serialization of the UserLocation object to a string, are provided in this form. Finally the request is send and the message “ok” is received as part of the **req.downloadHandler.text** response. In the case of wrongly formed request that is not accepted by the endpoint’s services, an appropriate error message is received through the **req.error** response.

```
1 reference | Thanos Restas, 20 days ago | 1 author, 1 change
public void CreateRequest(string id, string space, string spaceID, Transform transform)
{
    userLocationRequest = new UserLocation(id, space, spaceID, transform);
    SendRequest("WorkerLocationRequest");
}

1 reference | Thanos Restas, 20 days ago | 1 author, 1 change
public void SendRequest(string request)
{
    StartCoroutine(request);
}

0 references | Thanos Restas, 19 days ago | 1 author, 2 changes
IEnumerator WorkerLocationRequest()
{
    WWWForm userCredentialsForm = new WWWForm();
    // NOVITECH worker position endpoint
    string novitechEndpoint = "https://i3d.econtent.lu/i3d2/webservices-core/demo_new/worker_position";
    // Fill web request form
    string json = JsonUtility.ToJson(userLocationRequest);
    var req = new UnityWebRequest(novitechEndpoint, "POST");
    byte[] jsonToSend = new System.Text.UTF8Encoding().GetBytes(json);
    req.uploadHandler = (UploadHandler)new UploadHandlerRaw(jsonToSend);
    req.downloadHandler = (DownloadHandler)new DownloadHandlerBuffer();
    req.SetRequestHeader("Content-Type", "application/json");
    //Send the request then wait here until it returns
    yield return req.SendWebRequest();

    if (req.isNetworkError)
    {
        Debug.Log("Error While Sending: " + req.error);
    }
    else
    {
        Debug.Log("Received: " + req.downloadHandler.text);
    }
}
```

Figure 79**:** UserLocation form and send request example code.

5.5 TASK DATA

The Process JSON file is created once the user location is sent to the endpoint. Moreover, the contents of the Process file are based on the response of the POST request that was sent. The contents are parsed based on the agreed data model using NewtonSoft's JSON.Convert and visualised in Unity3D.

```
[Serializable]
2 references | Thanos Restas, 20 days ago | 1 author, 1 change
public class Task
{
    public string i3d_result_id;
    public string id;
    public string element;
    public string name;
    public string result;
    public string bimerr_provider_id;
    public string bimerr_provider_email;
}

[Serializable]
2 references | Thanos Restas, 20 days ago | 1 author, 2 changes
public class Process
{
    public string process_id;
    public string i3d_id;
    public string execution_status;
    public string process_name;
    public string name;
    public string planned_start;
    public string planned_finish;
    public string actual_start;
    public string actual_finish;
    public string space_id;
    public string space_name;
    public string storey;
    public string bimerr_provider_id;
    public string bimerr_provider_email;
    public string bimerr_manager_id;
    public string bimerr_manager_email;
    public Dictionary<string, DataModels.Task> tasks;
```

Figure 80**:** Process Data Model.

The Process part of the class is the one that is visualised in the ViewAvailableTasks Menu after the user taps the clipboard button. In addition, the same class encompasses the individual Task Details in the tasks field. This field uses the type of the Dictionary data structure so each key corresponds to a task object.

```
"tasks": {
  "2871": {
    "i3d_result_id": "i3d-workorder-task-2871",
    "id": "_089af3d2-4ff2-41a1-bb2b-2c2aed70d3fb",
    "element": "StartEvent",
    "name": "Start Event",
    "result": "Empty",
    "bimerr_provider_id": "9f053f7e-4b99-43b3-b28d-32f31a6fc6e6",
    "bimerr_provider_email": "demo@bimerr.eu"
  },
```


Figure 81**:** Task Key and Object.

The contents of each task are visualised on the appropriate Building Component's TaskDetailsMenu.

5.6 **ERROR REPORT**

The Error Report is received from using a GET request from set up collection job. This report contains the necessary info based on the agreed data model to indicate which building components present errors in their properties and marks them appropriately.


```
"entities": [
    {
        "valid": false,
        "name": "Outdoor  LG simple box KRIPIS v1:ARUN080LTE4:1442645",
        "global": "0WfKv63y9AXxoSruVWpscy",
        "express": 53367,
        "rules": [
            {
                "valid": false,
                "description": "Name",
                "parameters": [
                    {
                        "valid": false,
                        "name": "Property",
                        "value": "Name"
                    }
                ]
            },
            {
                "valid": true,
                "description": "EER"
            },
            {
                "valid": true,
                "description": "COP"
            },
            {
                "valid": true,
                "description": "ConsumptionCoolingCapacity"
            },
            {
                "valid": true,
                "description": "ConsumptionHeatingCapacity"
            },
            {
                "valid": true,
                "description": "EfficiencyCoolingCapacity"
            },
```


Figure 82**:** Error report data model outline.

5.7 KEYCLOAK

The Keycloak API is used for identity and access management in the BIMERR ecosystem of tools and applications. A login system is implemented through the creation of a POST Request. The form that is sent via this request is filled with fields such as the grant_type, client_id, username, password and scope based on the user's credentials and roles in a renovation project.

```
IEnumerator GetAccessToken()
{
    // Succesful Real-User authentication
    // Create and use account details from https://bimerr.s5labs.eu/
    WWWForm userCredentialsForm = new WWWForm();
    userCredentialsForm.AddField("grant_type", "password");
    userCredentialsForm.AddField("client_id", "bimerr-client");
    userCredentialsForm.AddField("username", usernameInputField.text);
    userCredentialsForm.AddField("password", passwordInputField.text);
    userCredentialsForm.AddField("scope", "roles");

    // The appropriate URL for authentication from Confulence documentation
    string getTokenURL = "https://auth.fit.fraunhofer.de/kc/realms/bimerr/protocol/openid-connect/token";

    using (UnityWebRequest www = UnityWebRequest.Post(getTokenURL, userCredentialsForm))
    {
        www.downloadHandler = new DownloadHandlerBuffer();

        yield return www.SendWebRequest();

        if (www.isNetworkError || www.isHttpError)
        {
            Debug.Log(www.error);
        }
        else
        {
            Debug.Log("Token Granted");
            var response = www.downloadHandler.text;
            // Remove hyphens from variable name
            response = response.Replace("not-before-policy", "not_before_policy");
            // Save Token Response into a Security Token object
            myObject = JsonUtility.FromJson<SecurityToken>(response);
            // Preview the JSON we just created
            print(myObject.PrintJson());
        }
    }
```

Figure 83: Keycloak Integration Example Code.

In the case of a successful request, a security Token is returned as a response based on the following data model.

```
public class SecurityToken
{
    public string access_token;
    public int expires_in;
    public int refresh_expires_in;
    public string refresh_token;
    public string token_type;
    public string id_token;
    public int not_before_policy;
    public string session_state;
    public string scope;
```

Figure 84**:** Keycloak Token Data Model.

6 DEPLOYMENT PLAN

This section presents a detailed deployment plan for the ARIBFA application.

Required permissions. To deploy the application, the user opens the application in Hololens. The first time that ARIBFA is deployed, the user's permission to let ARIBFA access the Hololens camera is asked. Moreover, ARIBFA queries for user's permission to access the device's microphone. Assess to these streams are necessary to capture videos, use speech commands, and receive camera stream from Hololens.

Identity check - Keycloak. The ARIBFA user must provide credentials to sign in. The Keycloak API is used for identity and access management in the BIMERR ecosystem of tools and applications. A login system is implemented through the creation of a POST Request. The form that is sent via this request is filled with fields such as the grant_type, client_id, username, password, and scope based on the user's credentials and roles in a renovation project.

Load 3D BIM model and IFC hierarchy on scene. When ARIBFA starts, the first task is to download the IFC file and the 3D model of building (.obj file) from BIF, using the API described in Section 5.2. The 3D BIM Model Visualisation Submodule (Section 4.2) is employed to build the hierarchy on runtime and attach IFC properties to the 3D building components. Based on the size of the building and the device's computing capabilities, this procedure might require some minutes to complete. In pilot sites, to speed up this procedure, an alternative option is also provided: to load the 3D model locally from the application's folder. The 3D BIM models (along with their IFC properties) are provided as AssetBundles. An AssetBundle includes content that is stored separately from the main application and is loaded at runtime. This helps minimizing the impact on network and system resources, since the core of the application remains light and heavy resources (such as the 3D BIM model) are added to content post-release.

Registration. The first time that a 3D BIM model is loaded in ARIBFA, it is visualised, but it is not aligned to the real world. The BIM 3D Model Registration and Tracking Submodule (Section 4.3) is used to perform the registration. As mentioned, the registration process relies on image targets. For each 3D building model that is loaded in ARIBFA, the position to place

the image target is selected in order to optimize the accuracy of the registration process. It is advised that the marker is placed on corners or wall edges, since they can be easily detected in the BIM model. For example, possible image target positions for the BIM model of the Spanish pilot site are depicted in Figure 85. These positions are chosen in ARIBFA based on the BIM model with the criterion to optimize the registration accuracy. Subsequently, detailed instructions are given to the users to place the printed image targets at the predefined positions on the actual building.

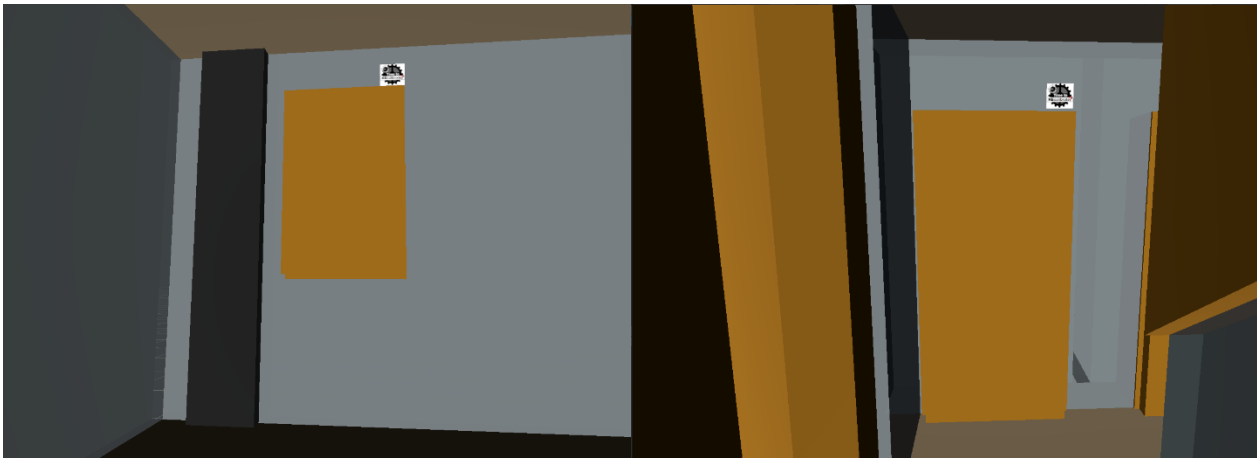


Figure 85: Possible positions for placing the image targets in the Spanish pilot site to optimize the registration accuracy.

In case of a building with many storeys, as the Spanish pilot site, which is depicted in Figure 86 as visualised in the Unity Editor, multiple image targets are needed to achieve the best registration accuracy. A different image target is placed on each storey and the users are guided to the image target corresponding to each storey and to where it should be placed. Therefore, the user should repeat the registration process at each storey.

Figure 86: The BIM model of the Spanish pilot site, as visualised in the Unity Editor.

To perform registration, the user uses the speech command “Scan for Marker” and approaches the image target in the building/storey. After the image target has been successfully detected, the 3D BIM model is visualised aligned to the real world, i.e., the 3D building components are overlaying the physical building components. To maintain this alignment to subsequent sessions, the aligned 3D model is anchored using speech command “Anchor”. The next time that ARIBFA is deployed, the 3D BIM model is visualised aligned to the real world at start time. Of course, since the IFC hierarchy is built dynamically at start time, some minutes of waiting are still needed. Registration is performed the first time the application loads a new 3D BIM model from the BIF or when the user thinks that the alignment should be corrected or when the user moves to a different storey in multi-storey buildings.

Localization. At any time during the application, the user can request localization information from ARIBFA. Using speech command “Show Location”, the user’s current position in terms of IfcSpace is displayed on Hololens’ camera (as depicted in Figure 29). To stop visualising localization information, the user can use the speech command “Hide Location”.

Visualisation and editing of IFC properties. The user can walk around the building and view the 3D BIM model overlaying the actual building. The user can perform air tap (selection gesture on Hololens) to a 3D building component of interest. A menu opens up that displays IFC properties of the building component. The depicted IFC properties vary based on the type of the selected building component, as described in detail in Section 4.6.4.2 and depicted in Figure 53, Figure 54, Figure 56, and Figure 55. The user can edit the value for the depicted IFC properties by tapping on them on the menu. Subsequently, the IFC file is updated with the new values and is promoted to BIF.

Task visualisation. At start time, ARIBFA receives from the BIF information regarding the tasks assigned to building components. At runtime, when the user enters an IfcSpace that includes building components with assigned tasks, a notification icon appears, as depicted in Figure 58. By tapping on this icon, the user can be informed for the workorder that concerns this IFC space, as depicted in Figure 59.

Error report visualisation. At start time, ARIBFA receives error reports from the BIF regarding missing/invalid IFC properties in HVAC components, missing/invalid thermal properties, and missing/invalid properties in IfcSpaces. At runtime, the building components with missing/invalid IFC properties are visualised in red color to notify the user. When the user taps on a red component (i.e., a component with missing/invalid IFC properties), an error report menu opens-up as depicted in Figure 67. In the left column of the menu, relevant valid IFC properties are displayed along with their values. In the right column of the menu, the missing property is displayed, and a value is requested from the user. By selecting the "Add" button at the top of the menu, the missing IFC property is added to the IFC file. Subsequently, the updated/corrected IFC file is forwarded to BIF via the API described in Section 5.2.

Addition of annotations. The user can add an annotation to a building component by tapping on it. A menu displaying the IFC properties of the specific building component is visualised, as depicted in Figure 54, Figure 53, Figure 56, and Figure 55. By selecting the "Add annotation" button, the menu is displayed. The user fills in the annotation menu fields either by keyboard (fields that require keyboard input are those with the "Enter text…" notification) or by using drop-down lists. Some annotation fields, such as the current user's id and the date, are automatically imported by ARIBFA in the JSON file representing the annotation

object. Moreover, using the corresponding buttons in Figure 64, the user can attach images and videos to the annotation. After clicking on the “Capture Image” button or the “Capture Video” button, the user is expected to take a screenshot or record a video, using the Volume up and Volume down buttons on the Hololens device as described in Section 7.1.1.2. Finally, the JSON with the completed annotation fields, along with the binary image or video files, are promoted to BIF when the user selects the “Add” button of the menu, as depicted in Figure 61.

MEP components detection. At start time, the user is asked to enter the IP of the paired computer. Pairing Hololens with a local computer is necessary for the Marker-less Feature Recognition Submodule, which performs object detection for missing MEP components, i.e., MEP components not included in the 3D BIM model. Before opening ARIBFA, the user is expected to install DesktopARIBFA application to a computer that is on the same network with Hololens. The installation process and the main functionalities of DesktopARIBFA are described in detail in Section 7.1.2.2 and Section 4.5.4, respectively. In summary, DesktopARIBFA handles the communication of a local computer with Hololens and facilitates the streaming of Hololens camera to the computer, so that detection is performed locally on the computer and detection results are promoted to Hololens for visualisation. When ARIBFA is deployed, the menu in Figure 38 is presented, where the user is asked for the paired computer’s IP. After inserting the IP, the user can open DesktopARIBFA on the local computer and insert the Hololens IP in the corresponding field of the application, as depicted in Figure 33. Instructions for finding the Hololens IP are included in Section 7.1.1.3. Following this procedure, the user can view the Hololens camera stream in the corresponding field in DesktopARIBFA.

At any time during runtime, the user can initialize object detection with speech command “Detect”. This enables Hololens to receive detection results from the paired desktop. Detected MEP components are visualised with 3D bounding boxes, as depicted in Figure 37. Upon detection, Hololens stops receiving new bounding boxes so that the user can process the detected 3D bounding box using hand manipulation to adjust the position, the scale, and the rotation of the bounding box, as depicted in Figure 35 and Figure 36. The user can also implicitly stop object detection at any time using speech command “Stop detection”. With speech command “Add properties”, a menu opens-up so that the user can insert IFC

properties to the detected object. Different menus are attached to indoor HVAC components, outdoor HVAC components, plugs, and switches, depending on the detection label. Menus to add IFC properties to detected indoor and outdoor HVAC components are depicted in Figure 40 and Figure 41, respectively. After inserting values to the IFC properties depicted in the menu, the detected object along with its IFC properties are added to the IFC, when the user taps on the “Add” button of the menu (top right corner in Figure 40 and Figure 41). The detected component is added to the IFC hierarchy as a child of the IfcSpace where the detection took place. Finally, the updated version of the BIM model is stored in Hololens storage and is forwarded to BIF using the API described in detail in Section 5.2.

7 USER MANUAL

The user manual contains all the necessary information for the user to deploy the ARIBFA application. In Section 7.1, a complete initial setup guide is provided. Section 7.2 encloses safety considerations during the use of the ARIBFA application and Section 7.3 describes the compliance of ARIBFA to the General Data Protection Regulation (GPDR). Finally, a complete and thorough demonstration of the user interface in ARIBFA and DesktopARIBFA is described in Section 7.4.

7.1 INITIAL SETUP

The initial setup includes the hardware setup, which is presented in subsection 7.1.1, and a complete installation guide for the ARIBFA and DesktopARIBFA applications, which is presented in subsection 7.1.2.

7.1.1 Hardware setup

This section provides important information for the Hololens device and instructions to get the Hololens ready to use (in subsection 7.1.1.1). Basic Hololens interactions are described in subsection 7.1.1.2. A network setup guide is provided in subsection 7.1.1.3.

7.1.1.1 GET HOLOLENS READY TO USE

The Hololens components are depicted in Figure 87 and summarized in Table 7. Hololens device specifications are summarized in Table 8.

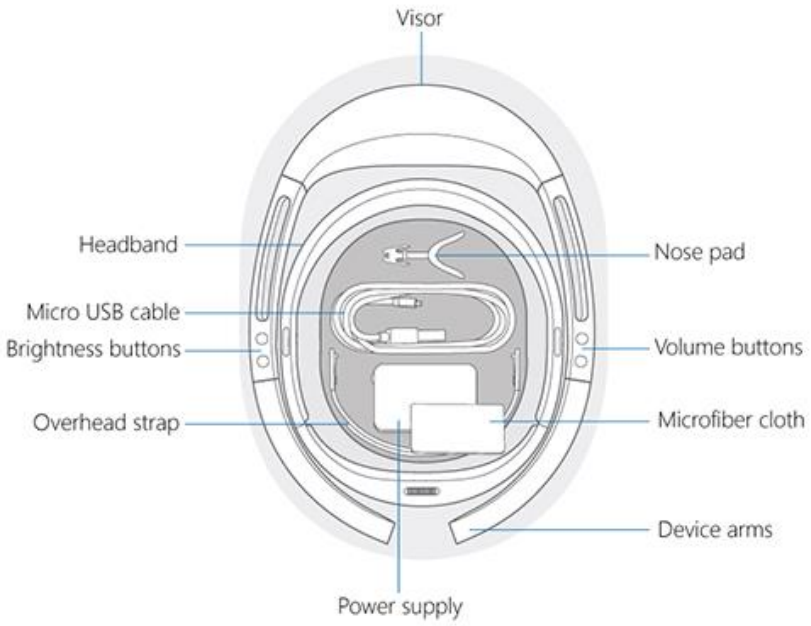


Figure 87: Hololens components[2].

Table 7: Hololens components.

Component	Description
Visor	Contains the HoloLens sensors and displays. The visor can be rotated up while wearing the Hololens.
Headband	To put the HoloLens on, use the adjustment wheel to expand the headband. With the HoloLens in place, tighten the adjustment wheel until the headband is comfortable.
Power button	Located on the back, right side; displays a LED battery level indicator.
Brightness buttons	Located on the left side when the user is wearing the Hololens.
Volume buttons	Located on the right side when the user is wearing the Hololens.
Overhead strap	Attach for comfort when wearing the Hololens for an extended period.
Device arms	When you pick up, put on, or take off your HoloLens, always grasp or hold it by the device arms.
USB-C Cable	Connect to power supply for charging or connect to a computer.
Power supply	18W charger, 9V at 2A
Microfiber cloth	Use to clean the visor.

[2] https://docs.microsoft.com/en-us/hololens/hololens1-hardware

Table 8: Hololens device specifications.

HoloLens Device Specifications

Software	Windows 10 Windows Mixed Reality
Weight	579g (1.28 lbs.)
Optics / Display	2.3 megapixel widescreen see-through holographic lenses (waveguides) 2 HD 16:9 light engines (screen aspect ratio) Holographic Density: >2.5k radiant (light points per radian) 1 2.4-megapixel photographic video camera Automatic pupillary distance calibration
Sensors	1 IMU (Accelerometer, gyroscope, and magnetometer) 4 environment sensors 1 energy-efficient depth camera with a 120°x120° angle of view Four-microphone array 1 ambient light sensor
Processors	Intel 32-bit (1GHz) with TPM 2.0 support Custom-built Microsoft Holographic Processing Unit (HPU 1.0)
Memory	2GB RAM
Storage	64GB (flash memory)
Power	Battery Life 2-3 hours of active use Up to 2 weeks on standby mode Fully functional when charging Passively cooled (no fans) Battery status LED nodes (battery level and power/standby mode settings)
Security	Windows 10 software updates Additional security and device management available for Commercial Suite

Wireless	Wi-Fi 802.11ac wireless networking Bluetooth 4.1 Low Energy (LE) wireless connectivity
Audio	3D audio speakers 3.5mm audio jack
Ports	Micro USB 2.0
Physical Buttons	Power Volume up/down Brightness up/down
What's in the box	HoloLens Development Edition Clicker Carrying case Charger and cable Microfiber cloth Nose pads Overhead strap
OS and Apps	Windows 10 Calibration Holograms Learn Gestures Settings Windows Feedback Windows Store Microsoft Edge Photos
Hardware / Software Requirements	Windows 10 PC Visual Studio 2015 Unity

Turn on/put on stand-by/shut down. Hololens has indicator lights which are depicted in Figure 88. To turn Hololens on and off or to put it in standby mode, the user uses the Power button, which is depicted in Figure 89. The battery indicators blink off. To wake it from standby, the user presses the Power button again. To shut down (turn off) Hololens, the user holds the Power button down for four seconds. The battery indicators turn off one by one and the device shuts down. Table 9 provides a detailed description of what Hololens indicator lights mean[3].

Figure 88: Hololens indicator lights.

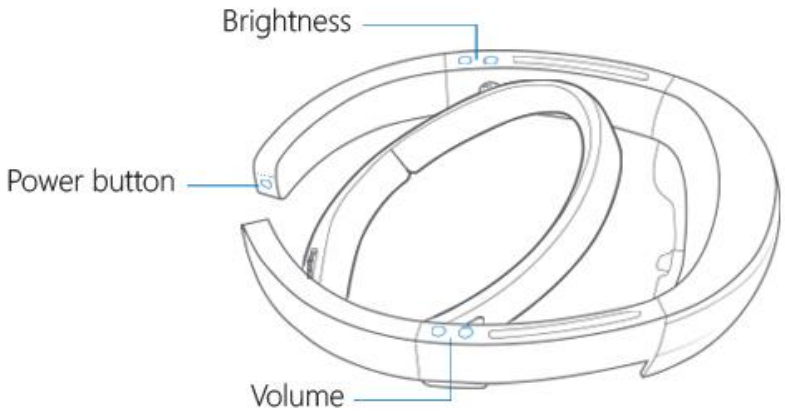


Figure 89: Hololens brightness, volume, and power buttons.

[3] https://docs.microsoft.com/en-us/hololens/hololens1-hardware

Table 9: Description of what Hololens indicator lights mean.

When the lights do this	It means
Scroll from the center outward.	HoloLens is starting up.
Stay lit (all or some).	HoloLens is on and ready to use. Battery life is shown in 20 percent increments.
Scroll, then light up, then scroll.	HoloLens is on and charging. Battery life is shown in 20 percent increments.
Turn off one by one.	HoloLens is shutting down.
Turn off all at once.	HoloLens is going into standby.
All light up, then one blinks briefly, then all turn off.	Battery is critically low. HoloLens needs to charge.
All scroll, then one blinks, then all scroll.	Battery is critically low. HoloLens is charging.

Adjust volume and brightness. The Brightness and Volume buttons are on top of the device arms—volume to your right and brightness to your left, as depicted in Figure 89.

Charge your HoloLens. To charge the Hololens, the user connects the power supply to the charging port by using the included Micro USB cable, as depicted in Figure 90. Then, the user plugs the power supply into a power outlet. When the device is charging, the battery indicator (depicted in Figure 88) will light up in a wave pattern.

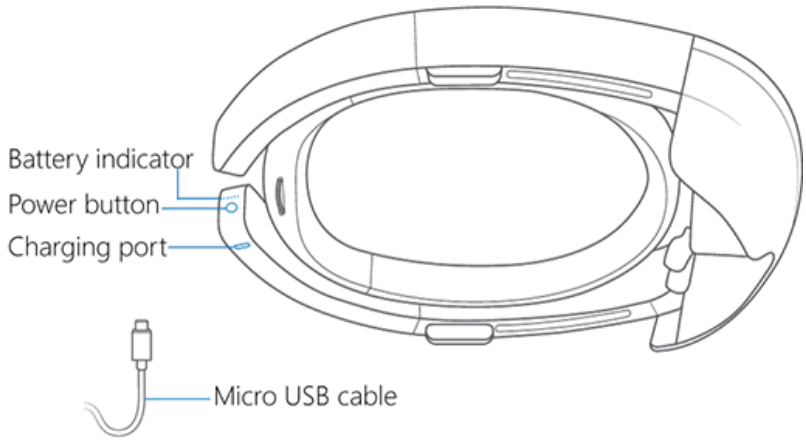
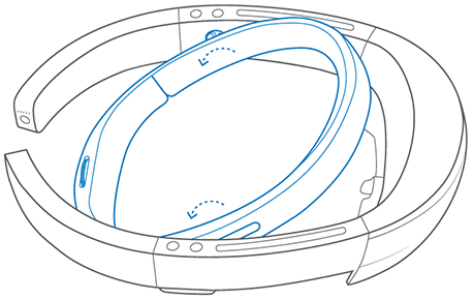


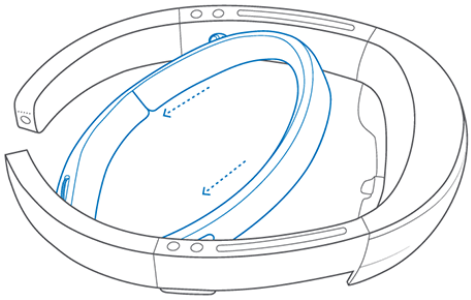
Figure 90: Charge the Hololens.

Adjust fit. The steps for adjusting Hololens fit are presented in Figure 91.

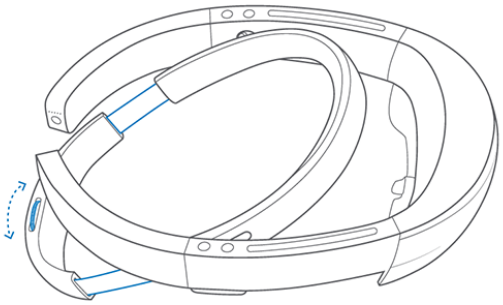
1. Rotate the headband up to about 20-30 degrees.



2. Push the headband back. Do not pull it back, or manipulate the band behind the hinge, because over time this can break the band.



3. Turn the adjustment wheel to extend the headband all the way out.



4. Hold the device by the device arms and place it on your head. Make sure that the headband sits at the top of your forehead, and then tighten the adjustment wheel.

5. Slide the visor back, and then check the fit of the device. The headband should sit at the top of the forehead, just below your hairline, with the speakers above your ears. The lenses should be centered over your eyes.

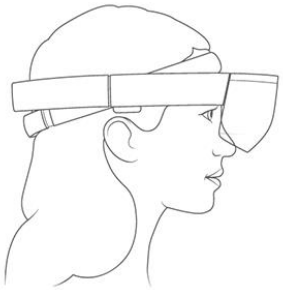


Figure 91: Instructions on how to adjust Hololens fit.

7.1.1.2 BASIC HOLOLENS INTERACTIONS

On Hololens, holograms blend with the physical environment to look and sound like they are part of the user's world. Even when holograms are placed in the 3D space, the user can still see their surroundings, move freely, and interact with other people and objects. Getting around Hololens is a lot like using a smart phone. The user can use their hands to manipulate holographic windows, menus, and buttons. The user can use their gaze, their voice, and gestures to select apps and holograms and to get around Hololens. This section presents a brief description on the Hololens basic interactions.

Gaze. Hololens uses the position and orientation of the user's head to determine their head direction vector. Gaze can be considered as a laser pointing straight ahead from directly between the user's eyes. This provides a fine-grained measurement of where the user is looking at. The head-gaze cursor is a dot that attaches to the end of an invisible head-gaze vector that uses the position and rotation of the head to point. The head-gaze cursor is depicted in Figure 92, as visualised in ARIBFA.

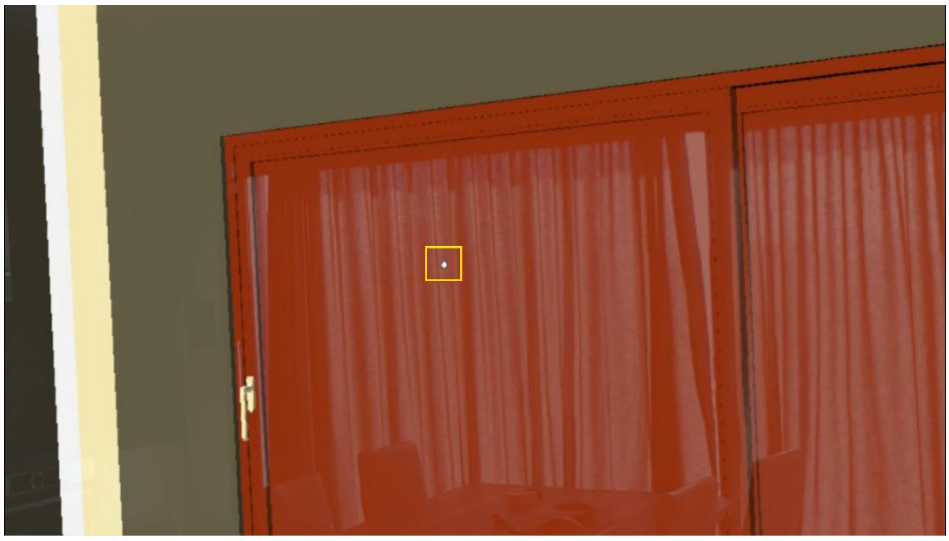


Figure 92: The head-gaze cursor is drawn at the location the user is looking at (to let the user know what they are targeting).

When using the application, the easiest way for the user to get around is to gaze at an item (menu or button, for example). They use gaze to target an item, and then act on that selection

by using a gesture (gestures are described in detail subsequently in this section). When the user gazes, they should turn their whole head—not just their eyes. The cursor will follow.

Voice. Hololens processes voice data to recognize commands and dictation. On Hololens, speech recognition will always function in the Windows display language configured in the Hololens device Settings. A guide to open the Hololens Settings is provided shortly below in this section. The voice commands in ARIBFA are in English. To assist the users, the GUI of ARIBA (which is in English) can be translated to other languages, e.g., to Spanish and Polish for the pilot sites, but speech commands will only be recognized in English.

Gestures. This section lists the most important gestures to know when using the Hololens. A detailed guide can be assessed at [4]. At Hololens, a complete gesture tutorial can be found on the Start menu of the device, i.e., the Learn Gestures app. Details on how to open the Start menu, which is depicted in Figure 96, are provided subsequently in this section. Microsoft Hololens has sensors that can see a few feet to either side of the user. When the user uses gestures, they need to keep them inside that frame, or Hololens will not see them. As the user moves around, the frame moves with the user. When the user's hand is inside the frame, the cursor looks like a ring. When Hololens cannot see the user's hand, the cursor changes to a dot. Hololens gesture frame is depicted in Figure 93.

[4] https://docs.microsoft.com/en-us/dynamics365/mixed-reality/guides/authoring-gestures

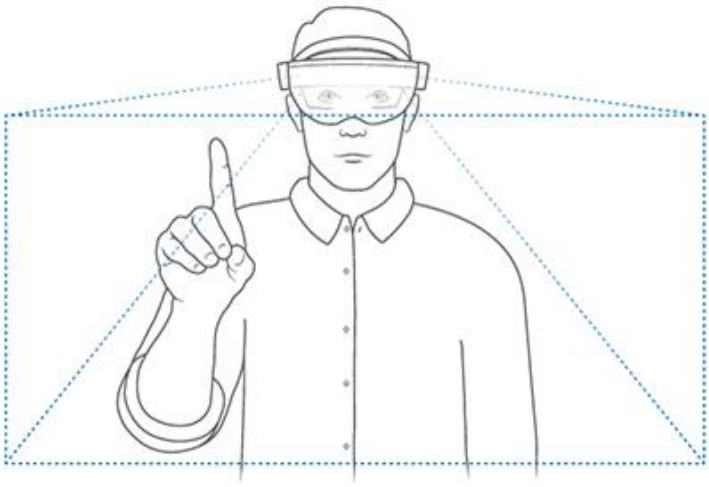


Figure 93: Hololens gesture frame.

The gestures on Hololens are:

1. **Bloom**. The Bloom gesture opens the Start menu. Most of the time, the user only needs to use the Bloom gesture once to get to Start menu. If the user is not sure what to do at any time, the Bloom gesture is a good way to get reoriented. To do the Bloom gesture:
    a. The user holds out their hand with their palm up and their fingertips together in front of them, so it is in the gesture frame.
    b. The user opens their hand.

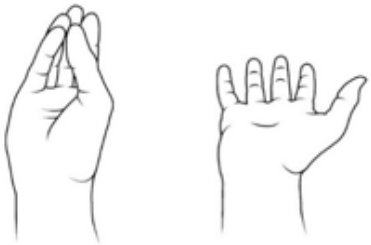


Figure 94: Bloom gesture on Hololens.

2. **Air tap**. The user uses air tap, along with gaze, to select holograms and any gaze/dwell buttons. To do an air tap:
    a. The user gazes at a hologram.

b. The user holds their hand straight out in front of them in a loose fist, then point their index finger straight up toward the ceiling. The user does not need to raise their whole arm—they are advised to keep their elbow low and comfortable.
c. The user taps their finger down, then quickly raises it back up again.

1. Finger in the ready position.

2. Press finger down to tap or click.

Figure 95: Air tap gesture on Hololens.

3. **Air tap and hold**. The user uses "air tap and hold" to move, rotate, or scale holograms when authoring. To air tap and hold, the user starts with an air tap, but they keep their finger down instead of raising it back up again.

Use the Start menu. The Start menu on Hololens is where the user will open apps, see important status info, and access tools like the camera. Wherever the user uses the Hololens, they can always open the Start menu by performing the Bloom gesture. The Start menu of Hololens is depicted in Figure 96.

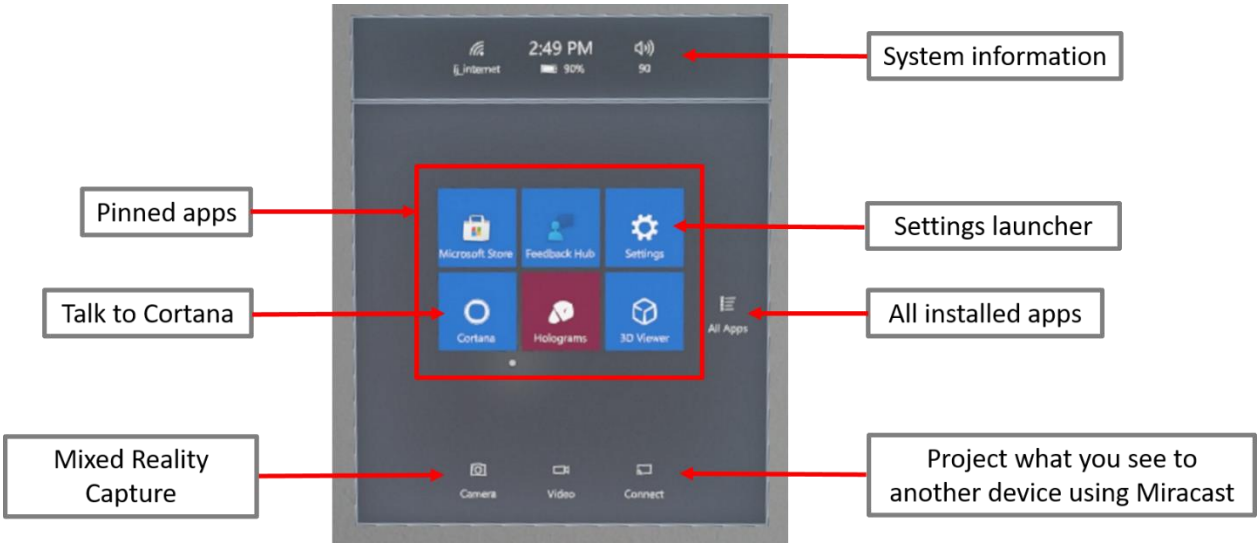


Figure 96: Start menu on Hololens.

The Start menu consists of:

- System information (network status, battery percentage, current time, and volume)
- Settings launcher
- Cortana
- Pinned apps. Here, the apps that are pinned to the Start menu are displayed.
- The “All Apps” button, which opens a list with all the installed applications on the Hololens.
- Photo and video buttons for Mixed Reality Capture
- Connect button to project to another device using Miracast.

The user can close the Start menu either with the Bloom gesture or by gazing at the menu and using the voice command “Close”.

Capture image or video. Hololens provides capabilities of taking screenshots of the user’s current view, that can be either images or videos. These capturing capabilities are important for ARIBFA during the annotation process in order for the user to attach images and videos

to the annotations, as mentioned in Section 4.6.4.4 and depicted in Figure 64. The user can take image screenshots of their current view, by pressing the Volume up and Volume down buttons on the Hololens device at the same time. These buttons are depicted in Figure 89. To capture a Mixed Reality video, the user presses and holds the Volume buttons simultaneously until a countdown of three seconds begins. To stop recording, both buttons should be simultaneously tapped.

7.1.1.3 NETWORK SETUP

To connect Hololens to a Wi-Fi, the user performs the Bloom gesture and selects "Settings" in the Hololens Start Menu. Subsequently, the user selects "Network & Internet" > "Wi-Fi" and checks that Wi-Fi is turned on. After selecting the desired network from the scroll-down list, the user selects "Connect". If prompted for a network password, the user types it and then selects "Next". The steps for connecting the Hololens to Wi-Fi are depicted in Figure 97.

To find Hololens IP, the user selects "Settings" in the Hololens Start menu. Subsequently, they select "Network & Internet" > "Wi-Fi" and choose the desired network. By tapping on "Advanced options", the properties of the network are displayed. The Hololens IP is the value of the depicted IPv4 address. This procedure is depicted in Figure 98. The Hololens IP is requested from DesktopARIBFA to connect to the Hololens and perform object detection for MEP components. More specifically, the user is prompted to enter the Hololens IP in the "Host name" field of DesktopARIBFA, as demonstrated in Figure 33.

For more detailed demonstration for network setup in Hololens, please refer to the Hololens user's manual at [5].

[5] https://docs.microsoft.com/en-us/hololens/hololens-network

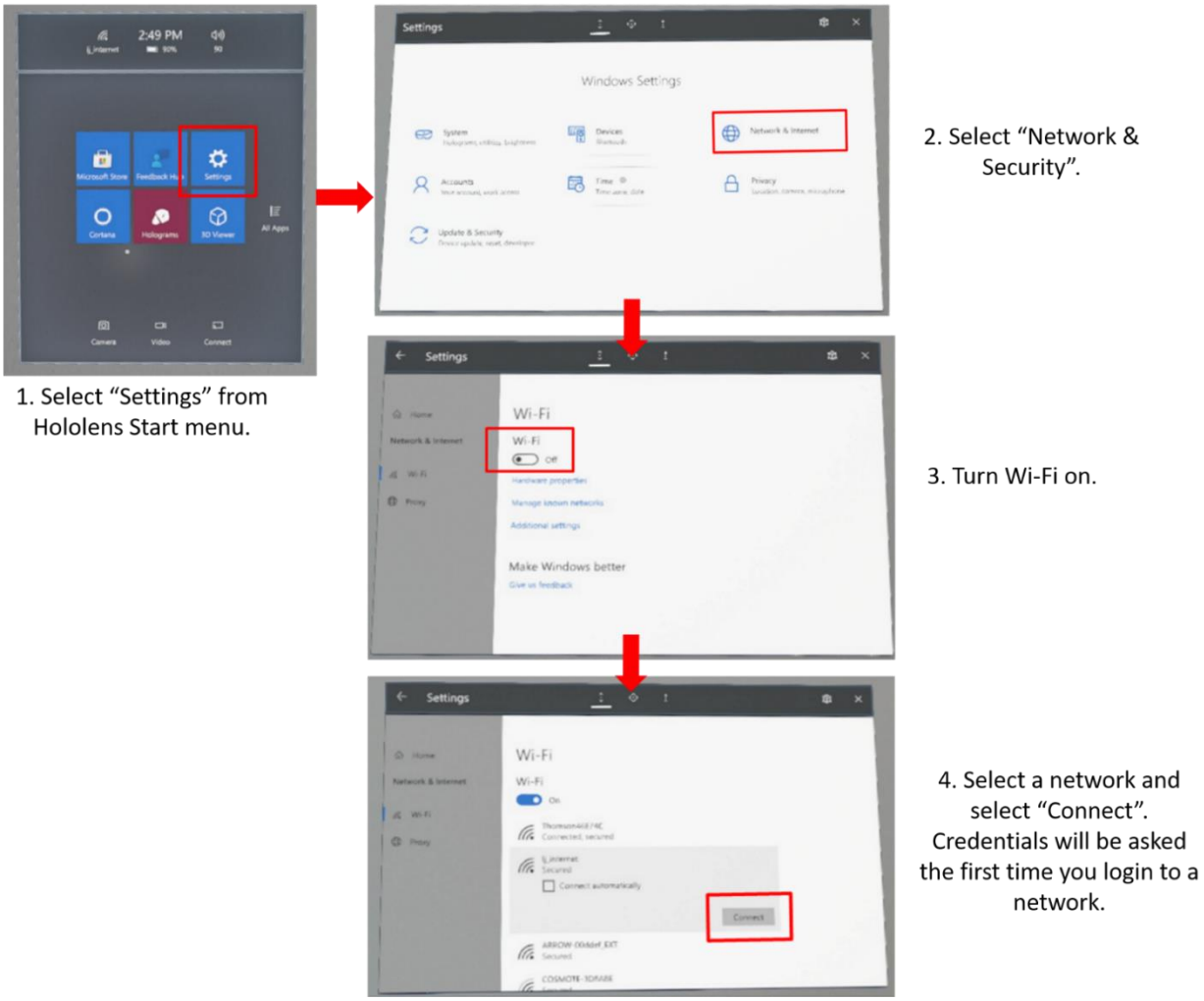


Figure 97: Steps to connect Hololens to a Wi-Fi.

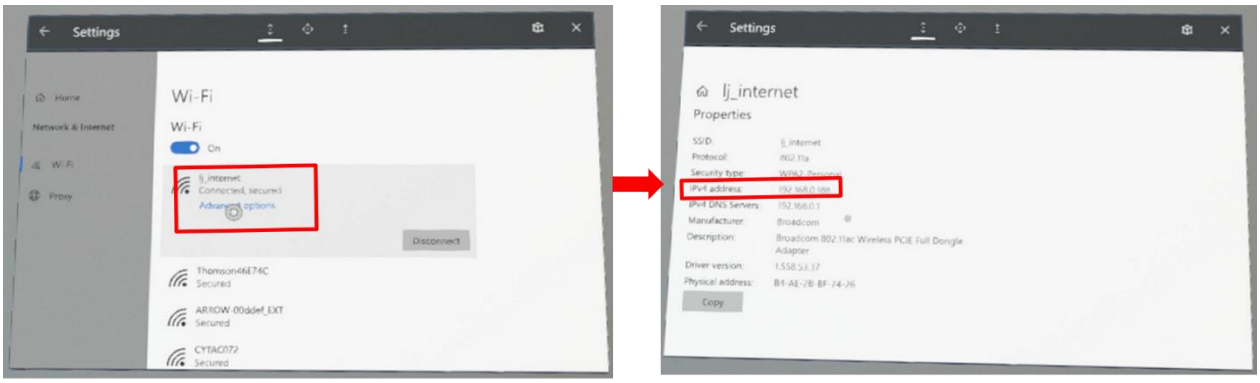


Figure 98: Steps to find Hololens IP.

7.1.2 Installation Guide

This section includes detailed instructions for installing the ARIBFA application (in subsection 7.1.2.1) and the DesktopARIBFA application (in subsection 7.1.2.2).

7.1.2.1 **ARIBFA INSTALLATION**

The ARIBFA application is provided to potential users upon request. ARIBFA is a Universal Windows Platform (UWP) application and is provided as an .appx app package for deployment and installation. With the .appx app package, the certificate of the application is also provided as an .cer file.

Using Microsoft Edge Internet Explorer (default and pre-installed internet explorer on Hololens), the user of Hololens can navigate to the download link and download the application files, i.e., the .appx and the .cer file. By default, files are saved to the "Downloads" folder of the device, but the user may select another folder. The application will be installed via the Hololens' Device Portal. The user can enable Device Portal by navigating to Hololens "Settings" and selecting "Update & Security" (Figure 99). Subsequently, the user selects "For developers" in the side panel and enables the "Use developer features" and "Enable device portal", as depicted in Figure 100. When enabling developer features, the user will be prompted to the confirmation window in Figure 101, which should be accepted.

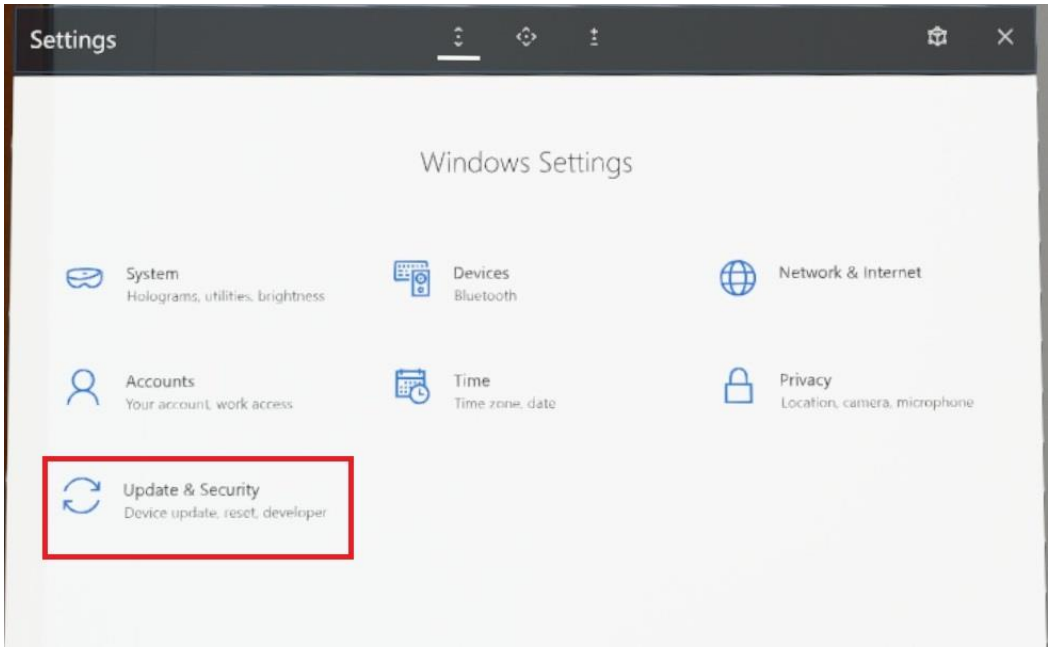


Figure 99: Navigate to “Update & Security” in Hololens Settings.

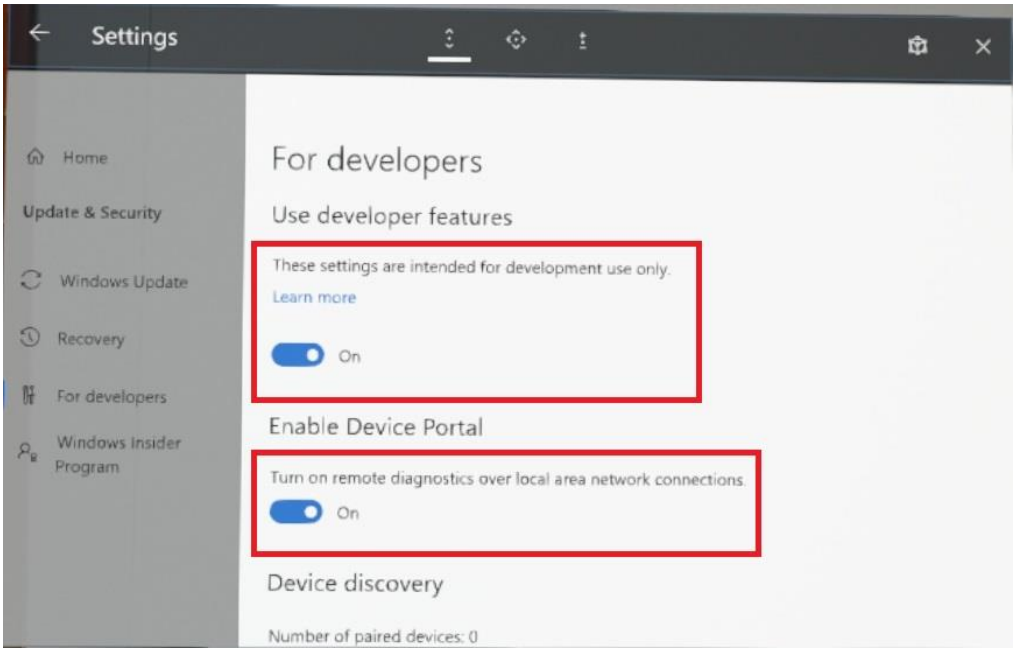


Figure 100: Enable “Use developer features” and “Device Portal” on Hololens.

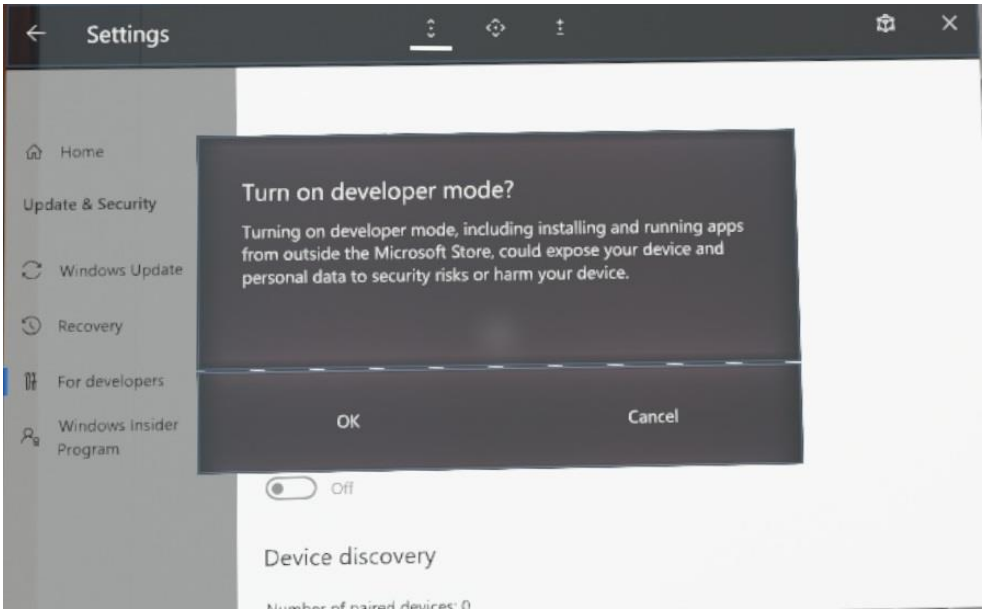


Figure 101:Confirmation for enabling developer features on Hololens**.**

After downloading the .appx and the .cer files of ARIBFA in Hololens storage, e.g., “Downloads”, the user can open the Device Portal of Hololens from a web browser on their PC by navigating to http://{HololensIP}, e.g., http://127.0.0.1. The procedure to get Hololens IP has been described in Section 7.1.1.3 and is illustrated in Figure 97. The Device Portal is depicted in Figure 102. To connect, the user may be asked to sign into their user account (if not signed in). Subsequently, the user can install the application by selecting “Views” > “Apps” in the Device Portal, as depicted in Figure 103. Subsequently, the user selects the application package from the Hololens storage and selects “Install”. The application’s certificate can be installed by selecting the “Install Certificate” button, as depicted in Figure 103.

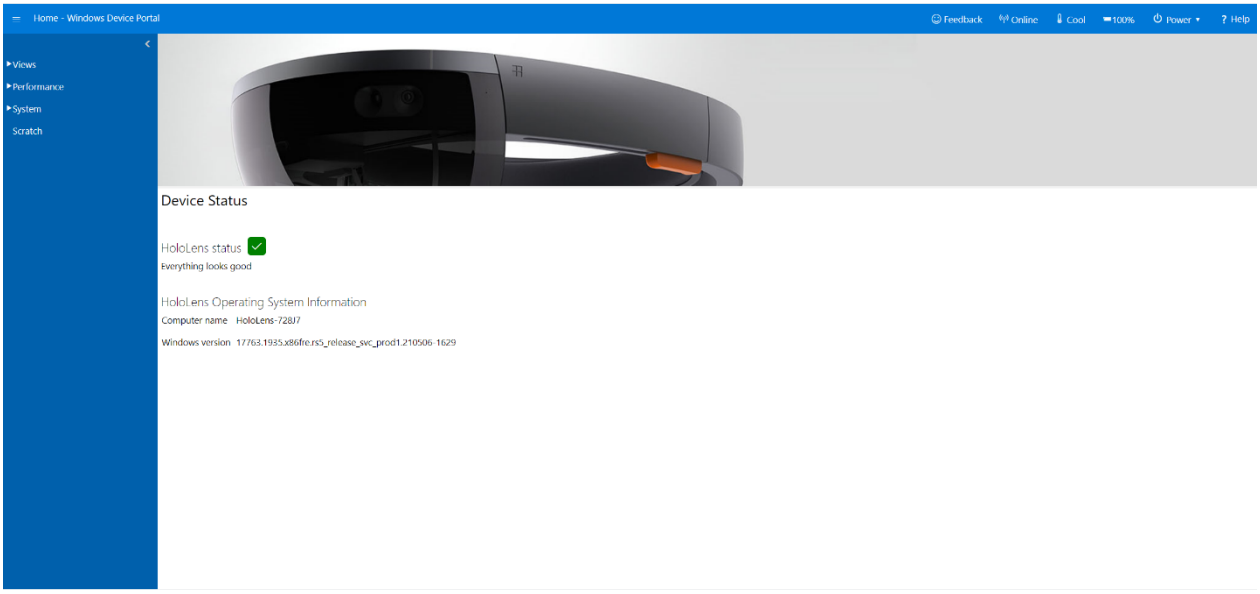


Figure 102: Hololens Device Portal.

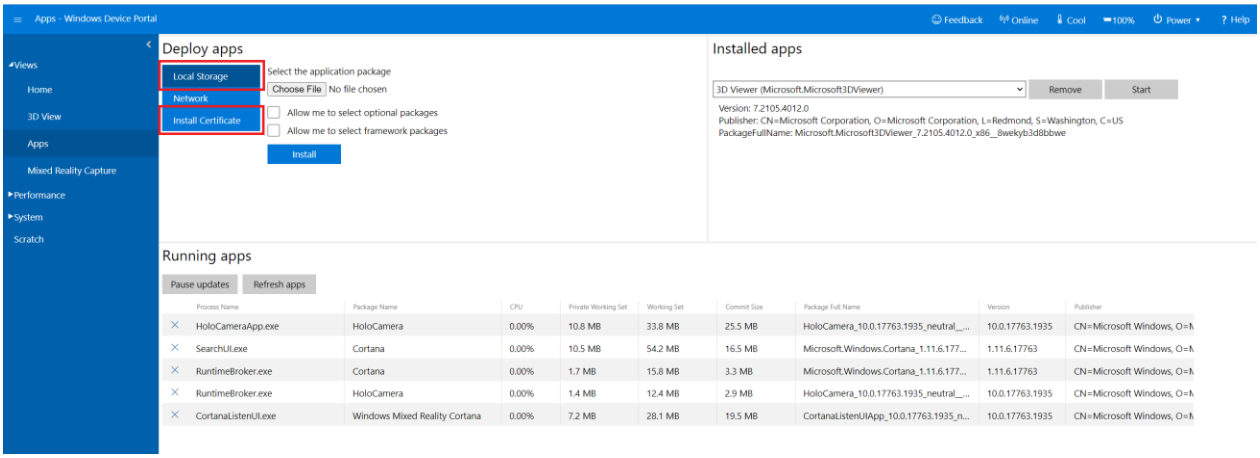


Figure 103: Installation of application package and certificate via the Hololens Device Portal.

7.1.2.2 DESKTOPARIBFA INSTALLATION

The DesktopARIBFA application is also provided upon user request. DesktopARIBFA is provided as an MSIX Bundle (MSIX is the latest Windows app package format), which is distributed with a corresponding certificate (.cer file). The application can run on both x86 and x64 architectures. To install the DesktopARIBFA application, the user must firstly accept the application's certificate (DesktopARIBFA.cer). By double-clicking on the certificate, the window in Figure 104 appears. By clicking on "InstallCertificate", the user can select to store certificate for current user (Figure 105). Subsequently, user can either select "Automatically select the certificate store based on the type of certificate" or choose "Place all certificates in the following store" (Figure 106). If the user chooses the second option, they will be promoted to choose the folder to save certificate. After successful certificate installation, the installation wizard is completed with the window in Figure 107. After successful installation of the application's certificate, the user can install the application by double-clicking on the provided MSIX package (DesktopARIBFA.msixbundle). The window in Figure 108 appears and the user selects "Install" to install the application.

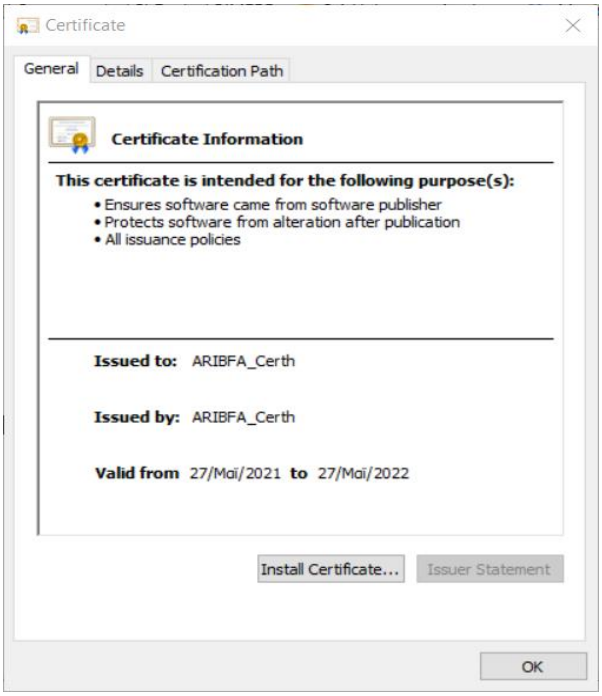


Figure 104: Install certificate window for DesktopARIBFA.

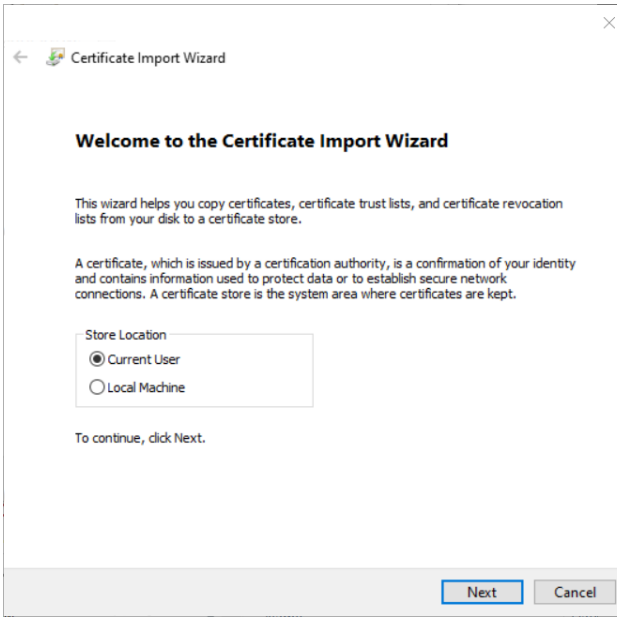


Figure 105: Install certificate as Current User.

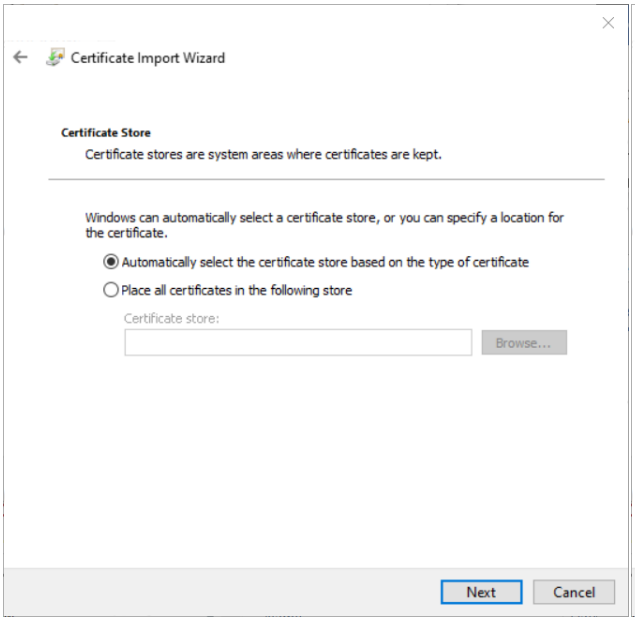


Figure 106: Choose where to store certificate.

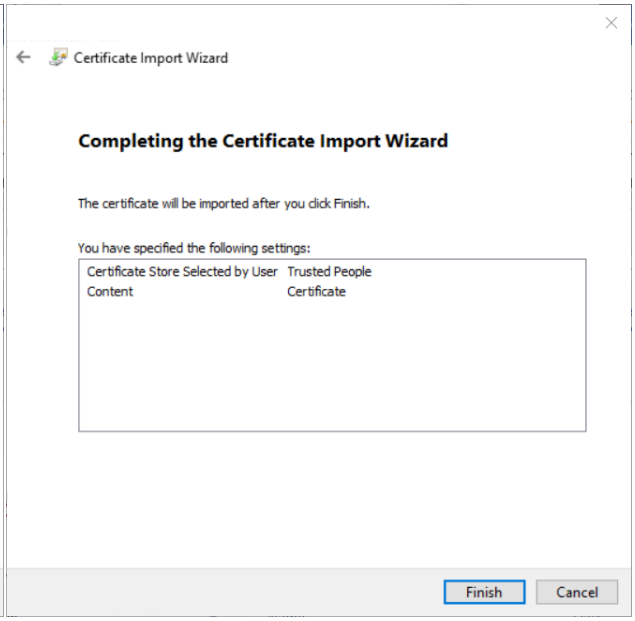


Figure 107: Window when certificate import is completed.

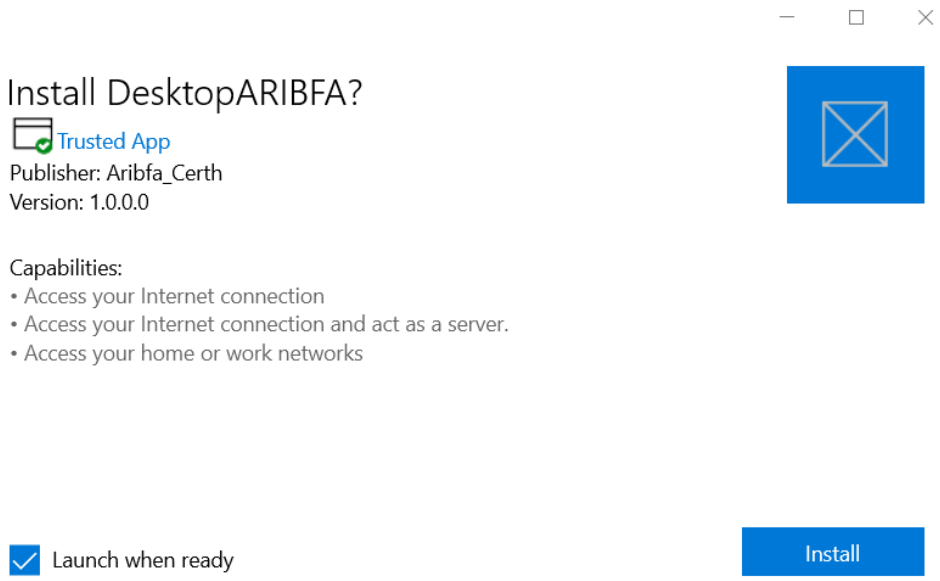


Figure 108: Installation wizard for DesktopARIBFA.

7.2 SAFETY CONSIDERATIONS DURING USE

When using the Hololens, the ARIBFA user walks around the building to examine the building components and inspect the renovation process. Attention is required so that ARIBFA is deployed in a place that is free of obstructions and tripping. It is advised that the area is cleared of any tripping hazards and enough clear space is provided to the user to move freely. Moreover, ARIBFA should be deployed to buildings/rooms with adequate light and plenty of space. Dark spaces may cause distraction and interfere with Hololens calibration. ARIBFA users, especially new users, are advised to take breaks periodically and rest if they experience any discomfort during the AR session. It may need some sessions for the user to get used to the AR experience. Correct calibration of the Hololens device is recommended since it can decrease any discomfort created during an AR session.

7.3 GPDR COMPLIANCE

The General Data Protection Regulation (GDPR) consent for all media uploaded by the users will be enforced and, in any case, the uploaded content will not be shared publicly but only with the building manager.

7.4 DESCRIPTION OF USER INTERFACE

In this section, a brief explanation of all interactable user interfaces of ARIBFA is provided.

7.4.1 How To Registrate

To perform registration, the user uses the speech command “Scan for Marker” and approaches the printed image target. The user gazes at the image target so that it can be detected. Upon image target detection, the 3D BIM model is visualised aligned to the real world. After successful registration, the user uses the speech command “Anchor” to anchor the aligned model, so that it can be loaded aligned to the real world at subsequent sessions.

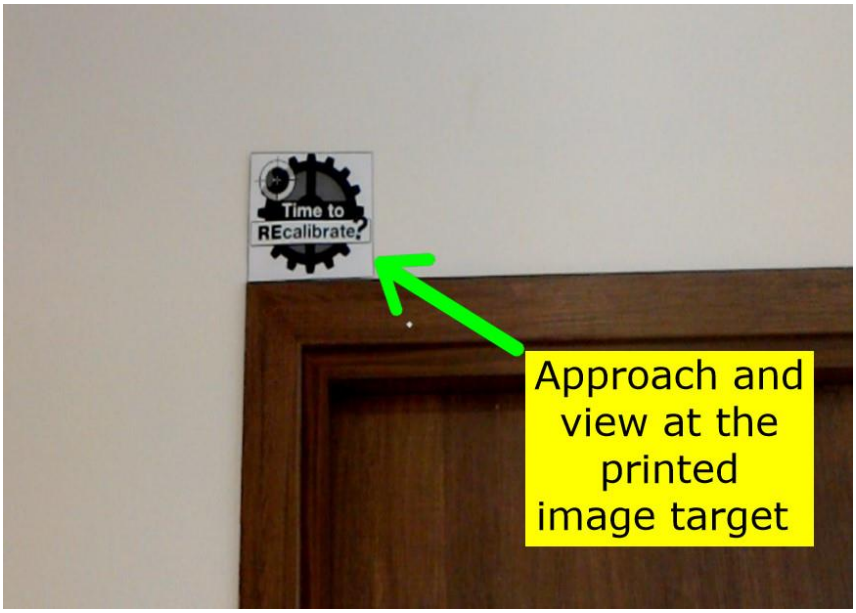


Figure 109: Registration is performed after gazing at the printed image target.

7.4.2 How To Visualise Tasks

The ShowAvailableTasks Button floats into the user's viewpoint. The users can distinguish it by the resemblance of a 3D Clipboard.

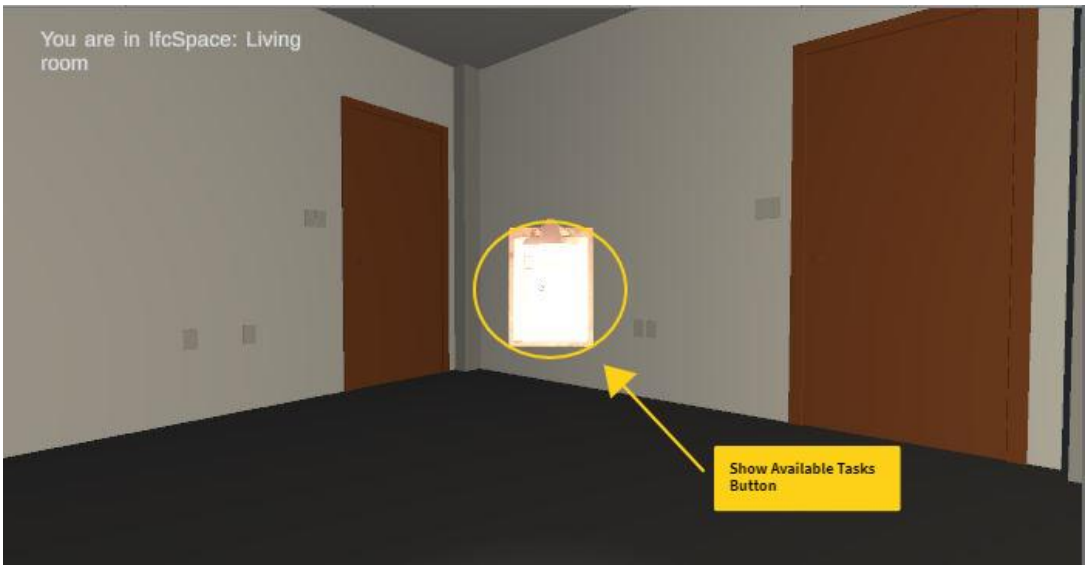


Figure 110: Clipboard floating in the user's viewpoint.

The user gazes on the ShowAvailableTasks Button and performs air tap on it to bring up the ViewAvailableTasksMenu.

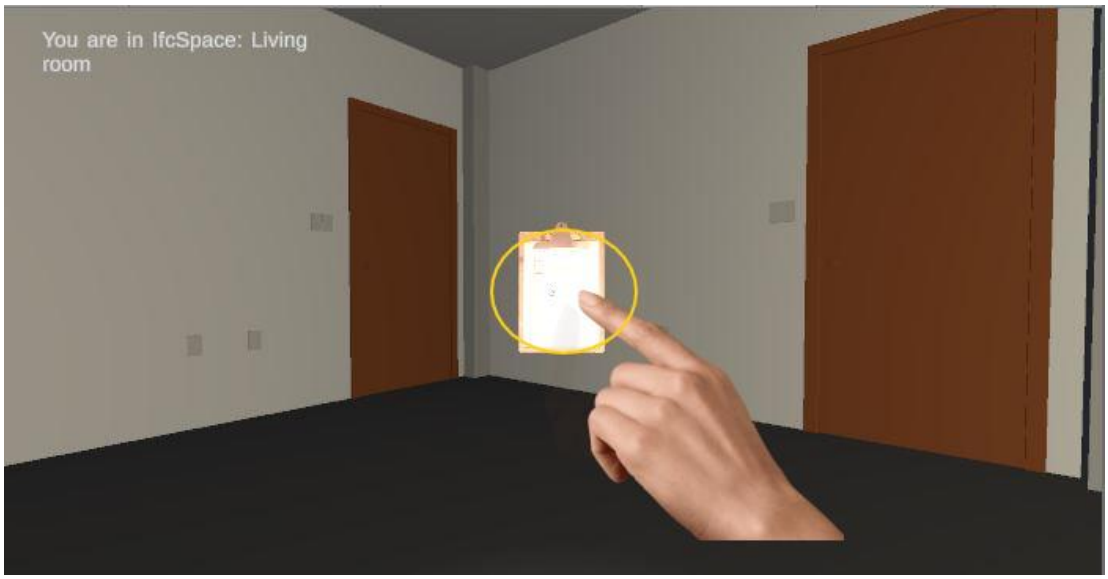


Figure 111: Air tap on the ShowAvailableTasks Button.

The ViewAvailableTasksMenu becomes visible in the user's field of view. The user can view the list of available tasks in the current building space's renovation process.

Figure 112: The ViewAvailableTasksMenu becomes visible.

The user can tap on the "PIN" button to pin the floating menu in its current position in the 3D space, preventing it from following the user's head movement.

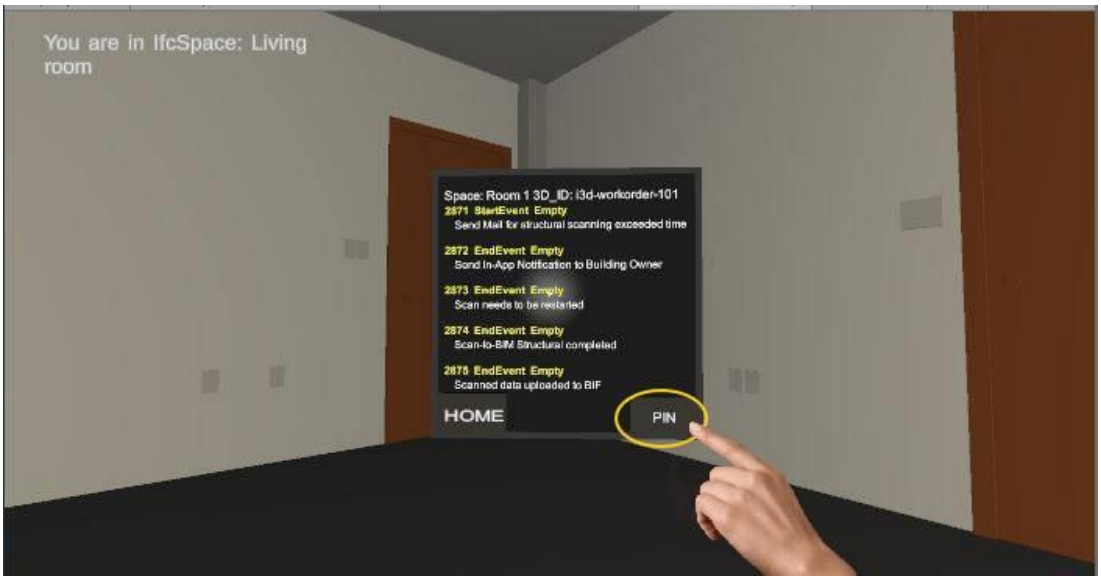


Figure 113: Air tap on the "PIN" button to pin the menu in its current location in 3D space.

The “UNPIN” button becomes visible and enables the free movement of the floating menu, when tapped.

Figure 114: The “UNPIN” button becomes visible.

The “HOME” button can be air tapped to close the menu.

Figure 115: Air tap on the “Home” button.

7.4.3 How To Visualise IFC Properties

Every building component in the user's surroundings is selectable.

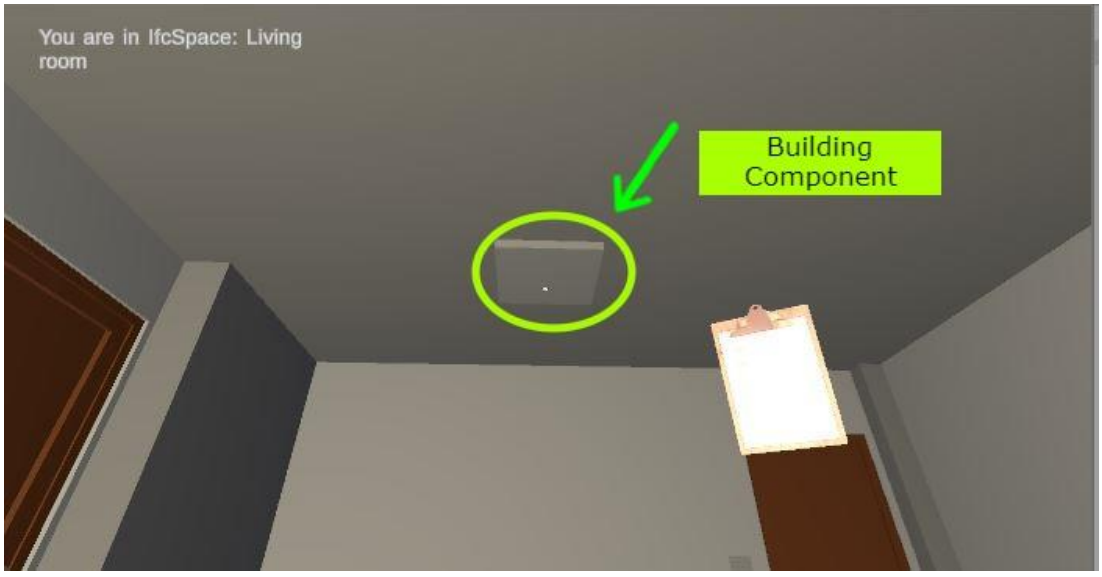


Figure 116: Selectable building component.

The user can air tap on a building component to select it.

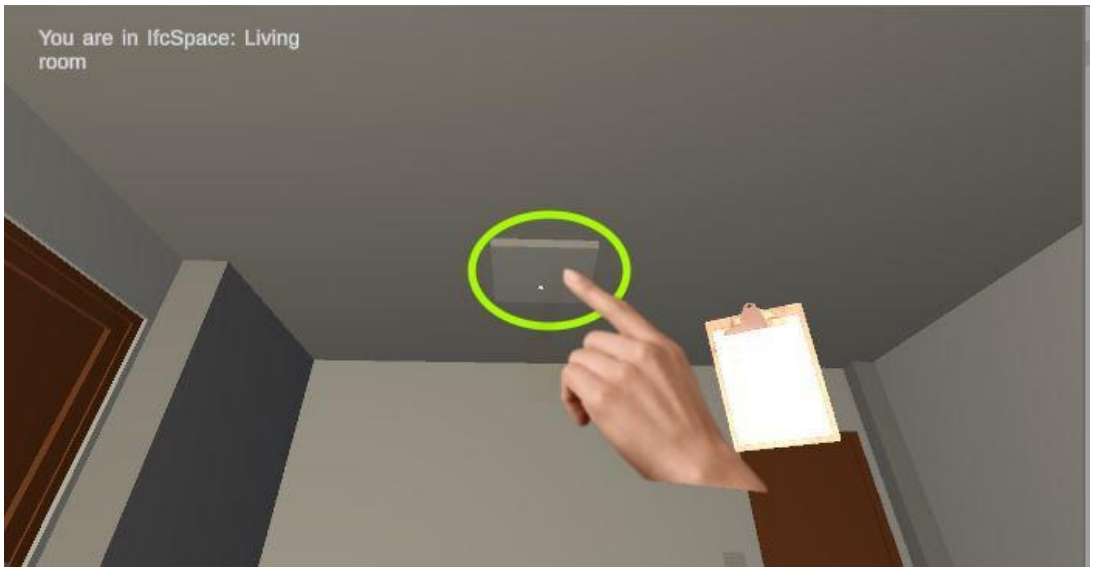


Figure 117: Building component can be air tapped.

Figure 119: Static Properties Menu and highlighted building component.

The appropriate Properties Menu becomes available. The selected building component becomes highlighted. An overview of the specific IFC properties of the building component is displayed.

Figure 118: Indoor HVAC Properties Menu and highlighted building component.

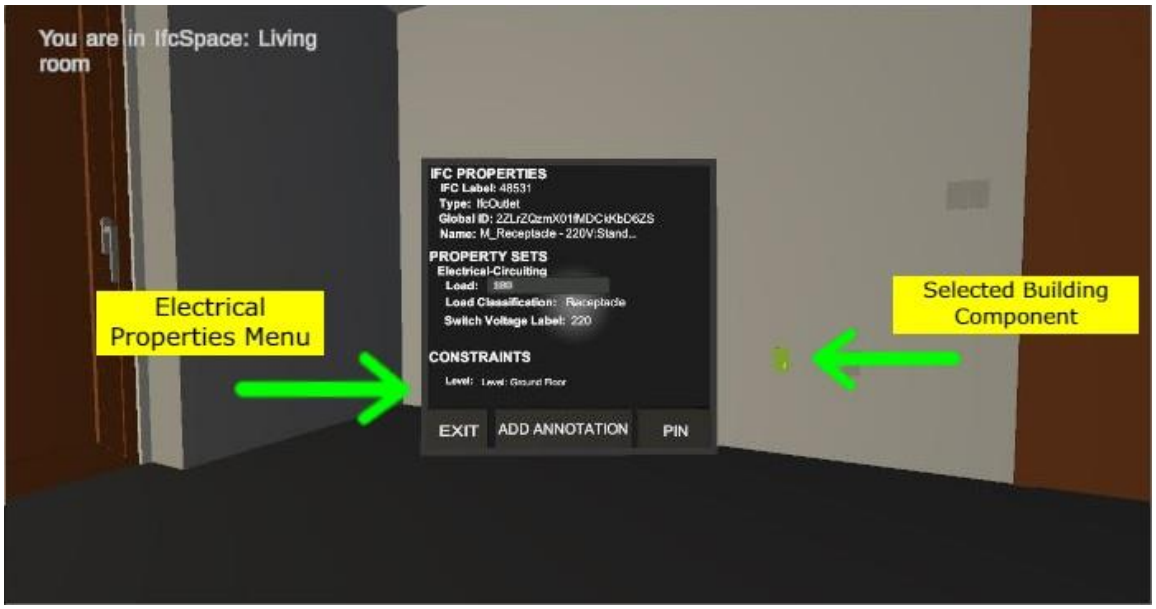


Figure 120: Electrical Properties Menu and highlighted building component.

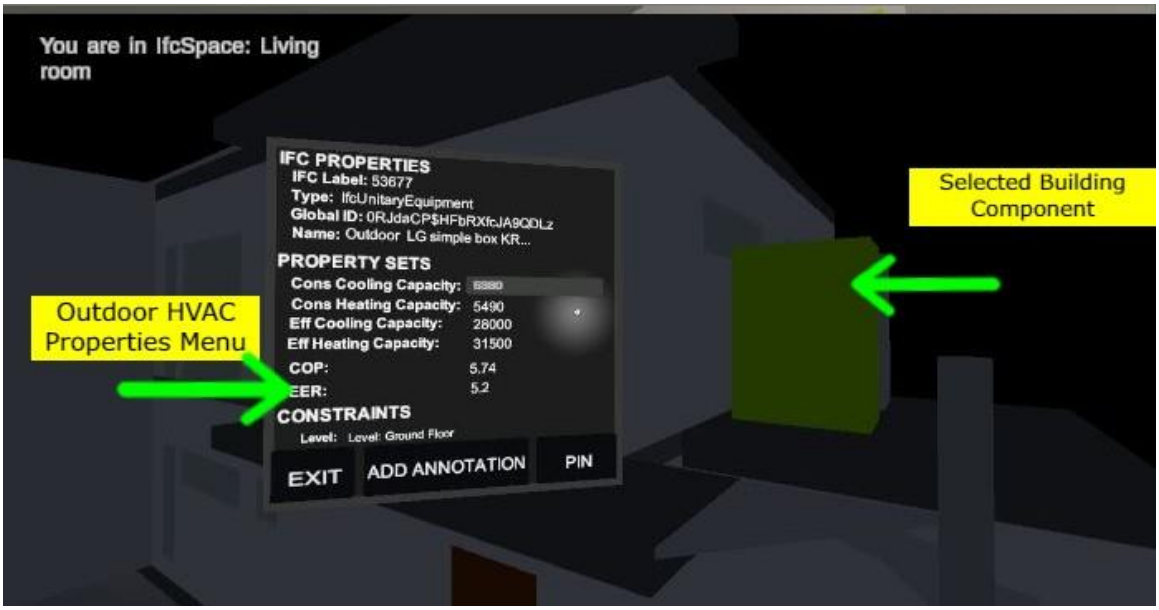


Figure 121: Outdoor HVAC Properties Menu and highlighted building component.

The user can gaze on the input field of a property, which is highlighted.

Figure 122: Highlighted Property Input Field.

The input field can be air tapped, the virtual keyboard is brought up, and the property value can be edited through the keyboard by typing using the air tap gesture.

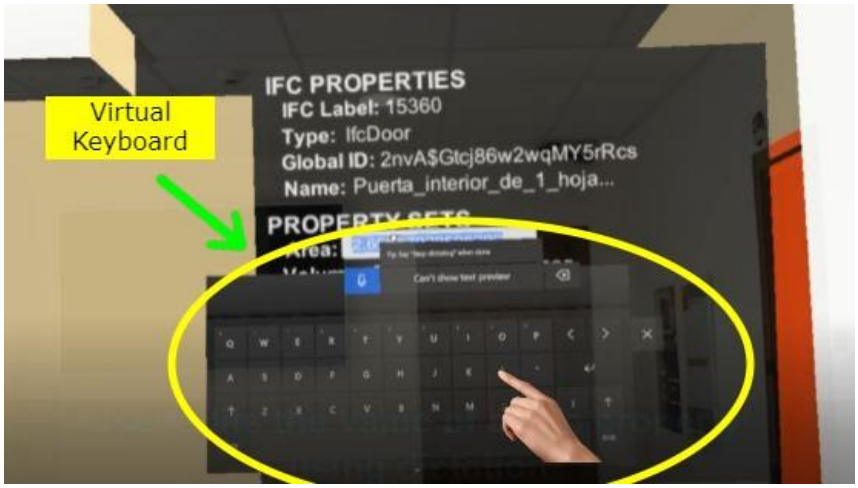


Figure 123: Edit IFC parameter through the virtual keyboard.

The “EXIT” button can be air tapped to close the menu. The “PIN” button can be air tapped to pin the floating menu in a specific 3D position. The “Add Annotation” button opens the building annotation menu.

Figure 124: Available buttons in the Properties Menu.

7.4.4 How To Visualise Task Details

The “Task” button appears on building components which are part of a renovation task. The user can air tap on it to bring up the Task Details Menu.

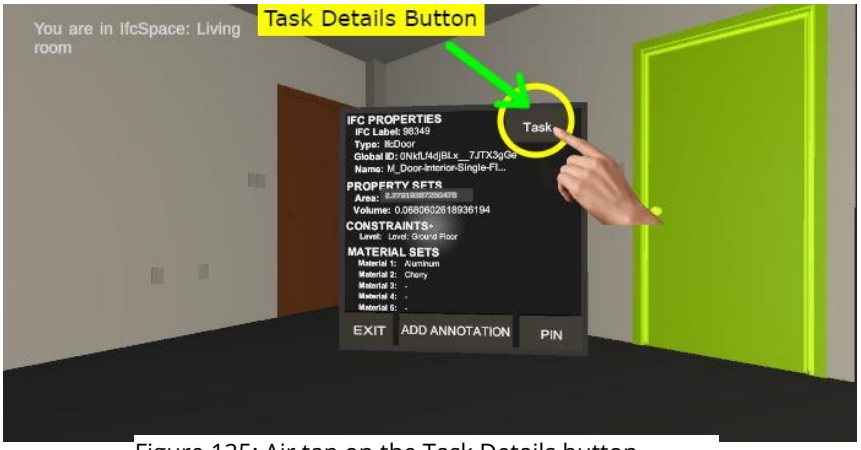


Figure 125: Air tap on the Task Details button.

The user can gaze over the list of a particular task's details on the Task Details Menu.

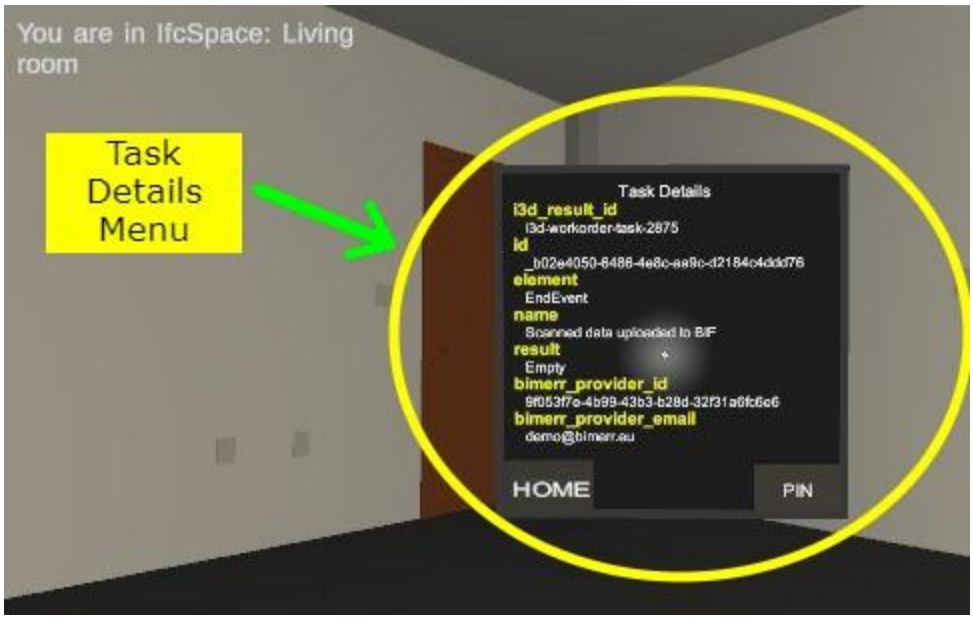


Figure 126: The Task Details Menu.

7.4.5 How To Add Annotation

The users can air tap on the "Add Annotation" button, which appears on the property menu.

Figure 127: Air tap on the "Add Annotation" button.

The Add Annotation menu becomes available.

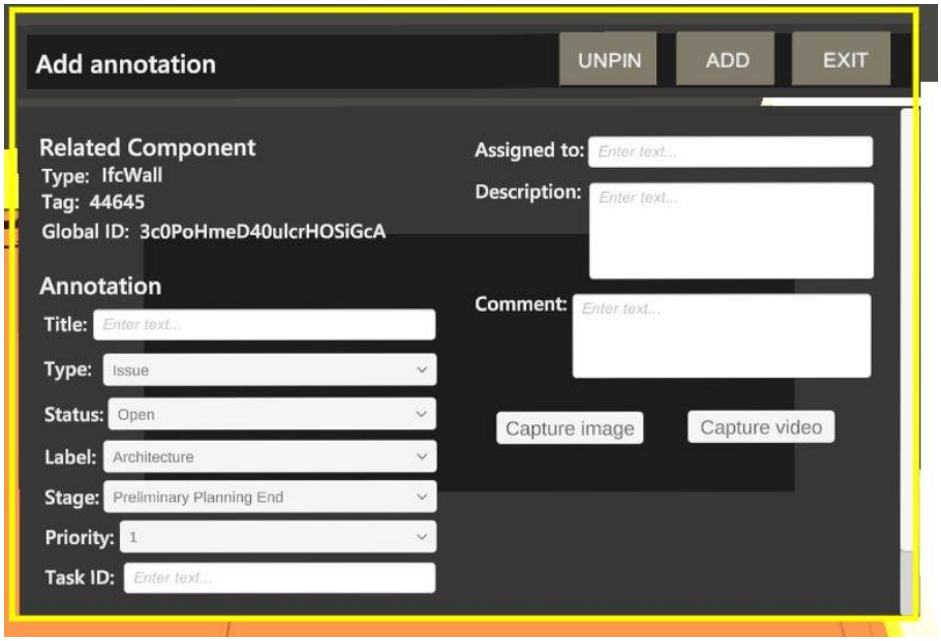


Figure 128: The Add Annotation Menu.

The various drop-down menus can be expanded via an air tap, to show a list of options.

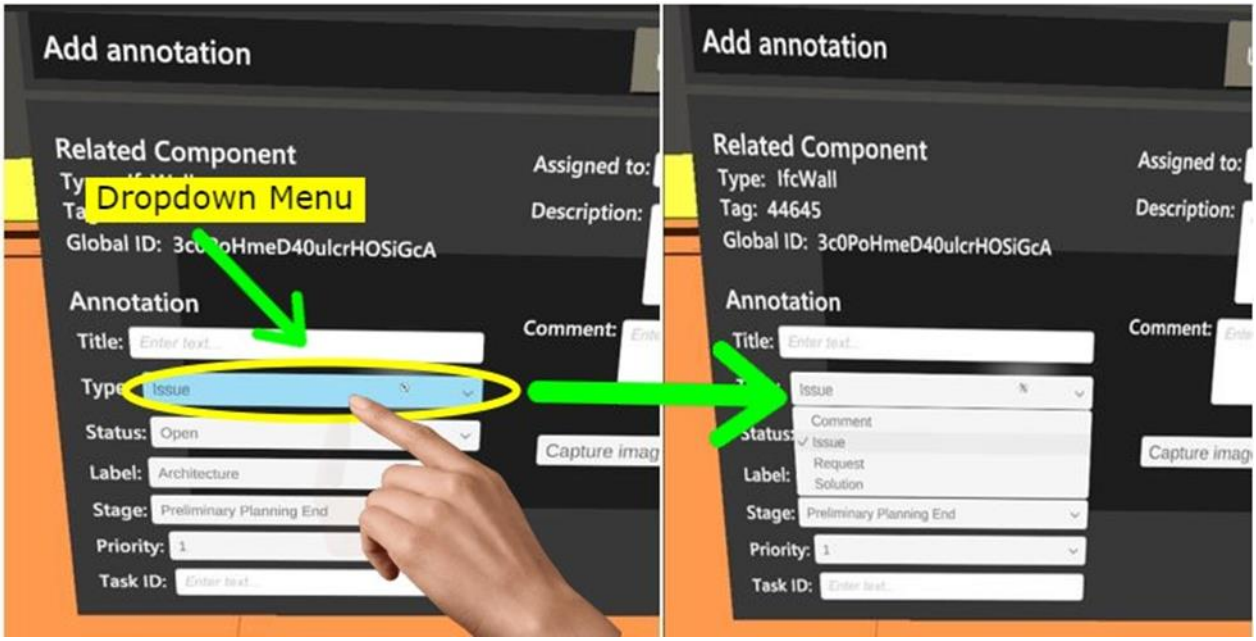


Figure 129: Type Dropdown in Add Annotation Menu.

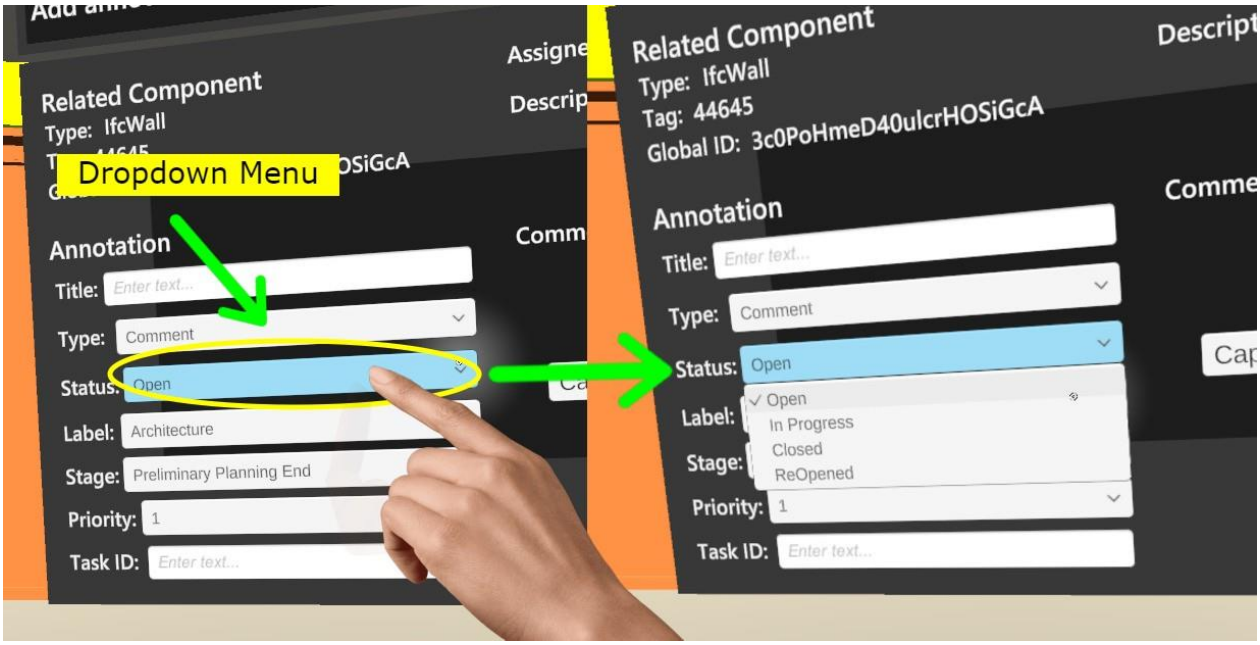


Figure 130: Status Dropdown in Add Annotation Menu.

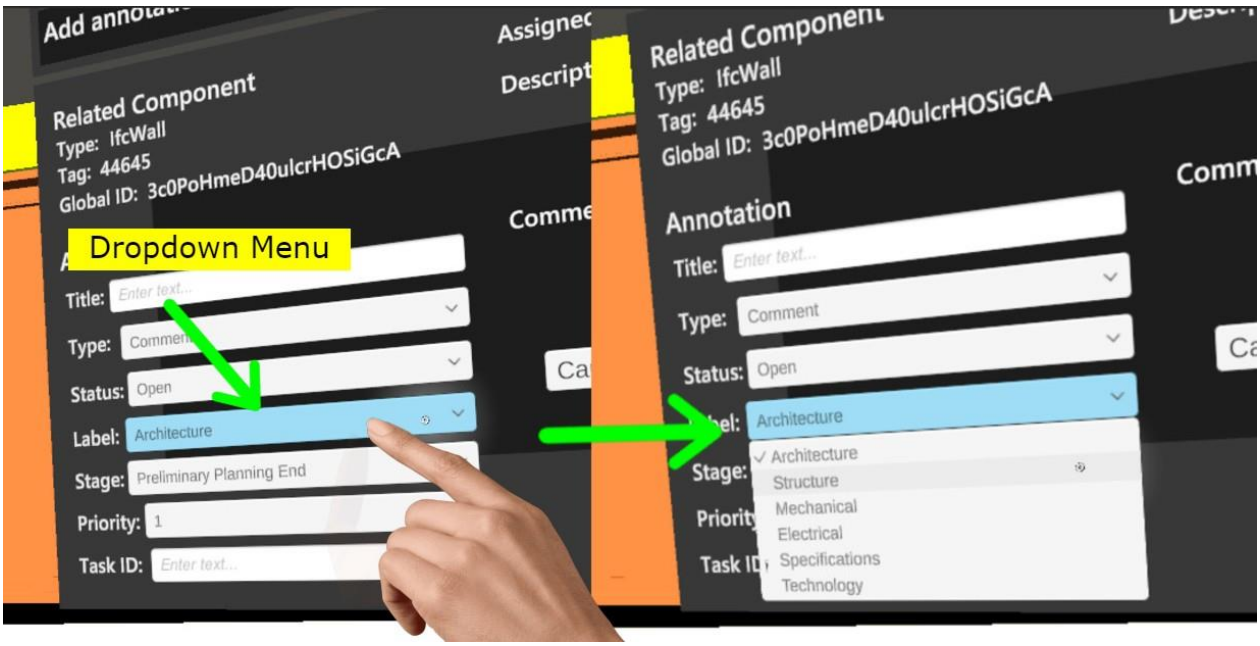


Figure 131: Label Dropdown in Add Annotation Menu.

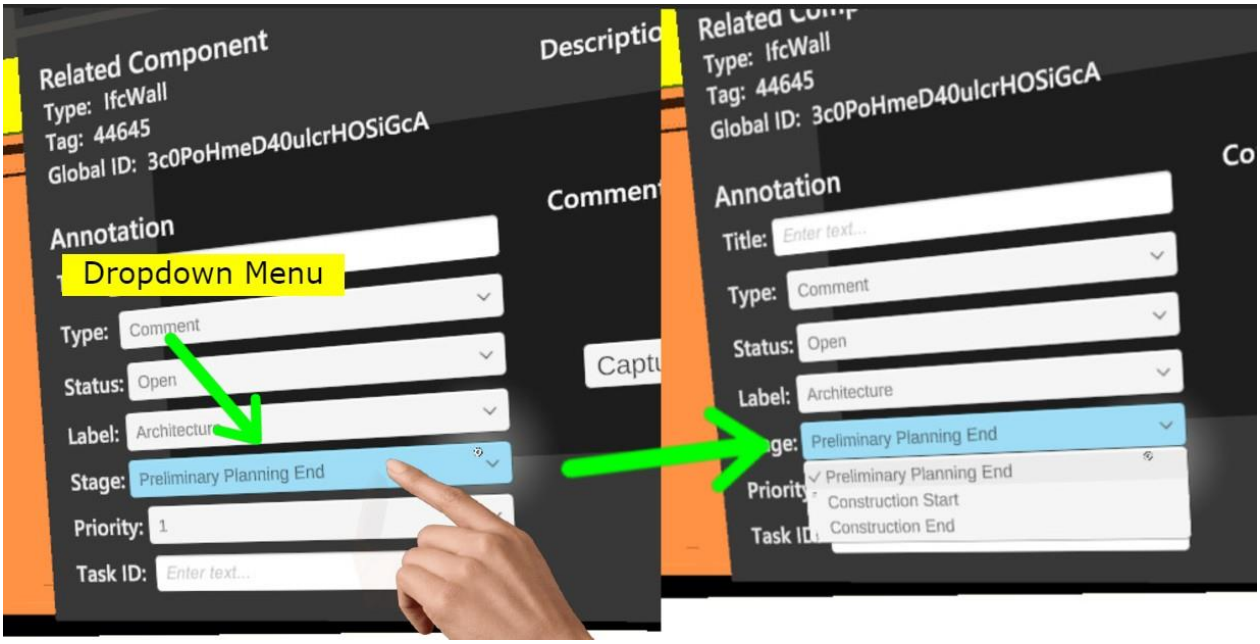


Figure 132: Stage Dropdown in Add Annotation Menu.

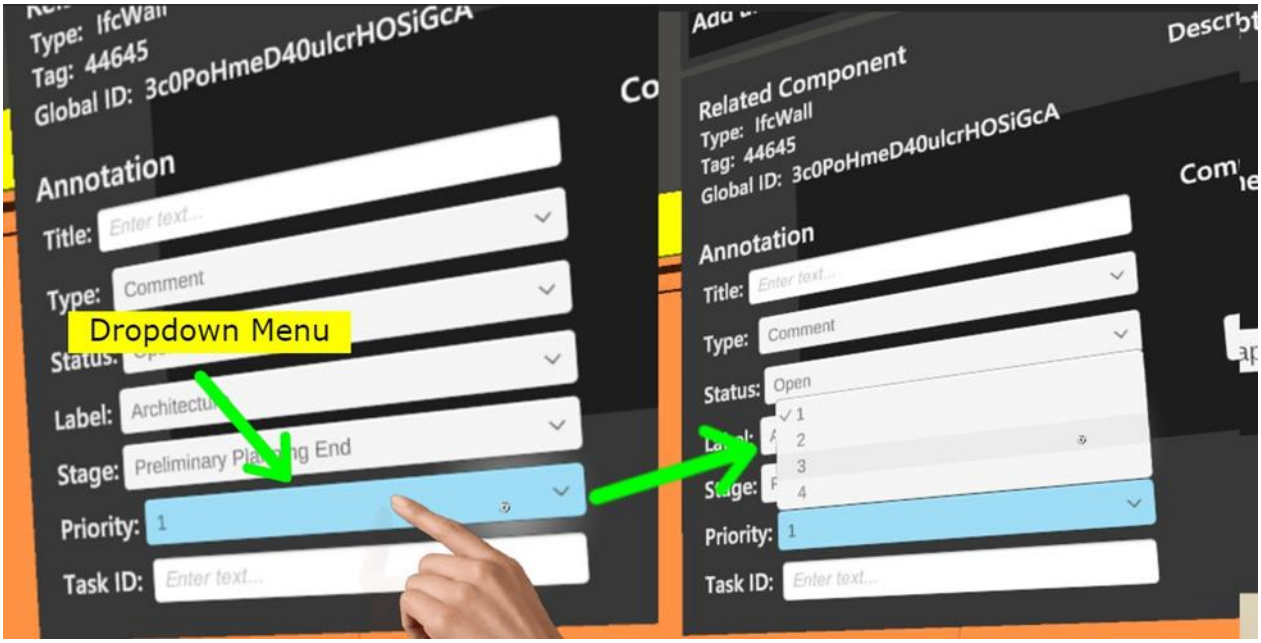


Figure 133: Priority Dropdown in Add Annotation Menu.

The various Input Fields that appear on the Add Annotation menu can be air tapped to be filled via the virtual keyboard.

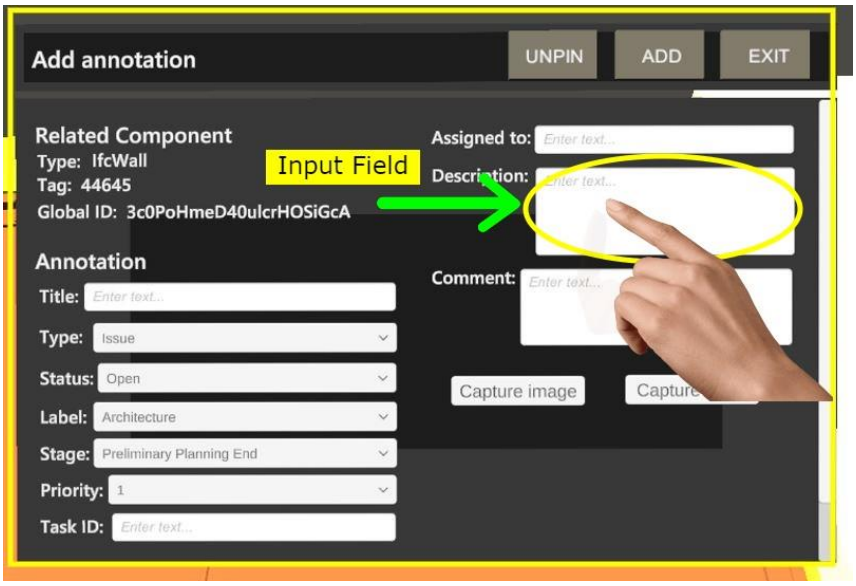


Figure 134: Air tap on the Input Field.

The user can also capture and attach image and/or video files to the annotation by air tapping on the “Capture image” and “Capture video” buttons, respectively. Once the “Capture image” or the “Capture video” button is tapped, the user is expected to capture a screenshot of their current view or to capture a video using the Hololens Volume up and Volume down buttons, as described in Section 7.1.1.2.

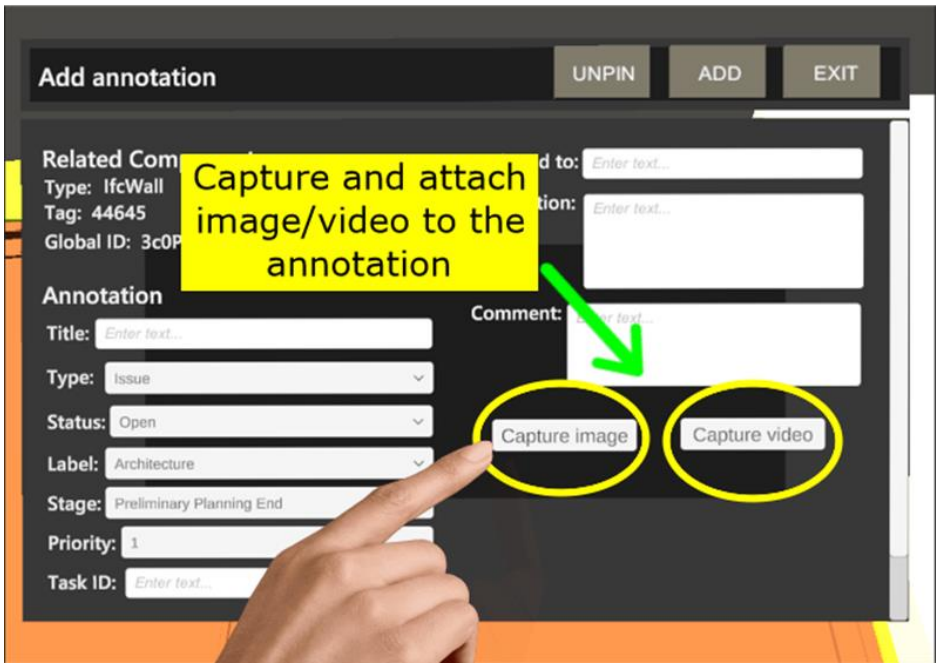


Figure 135: Air tap on the "Capture image" and/or "Capture video" buttons to attach an image/video file to the annotation.

Air tapping on the “Add” button finalizes the annotation process. Finally, the “Exit” button closes the menu.

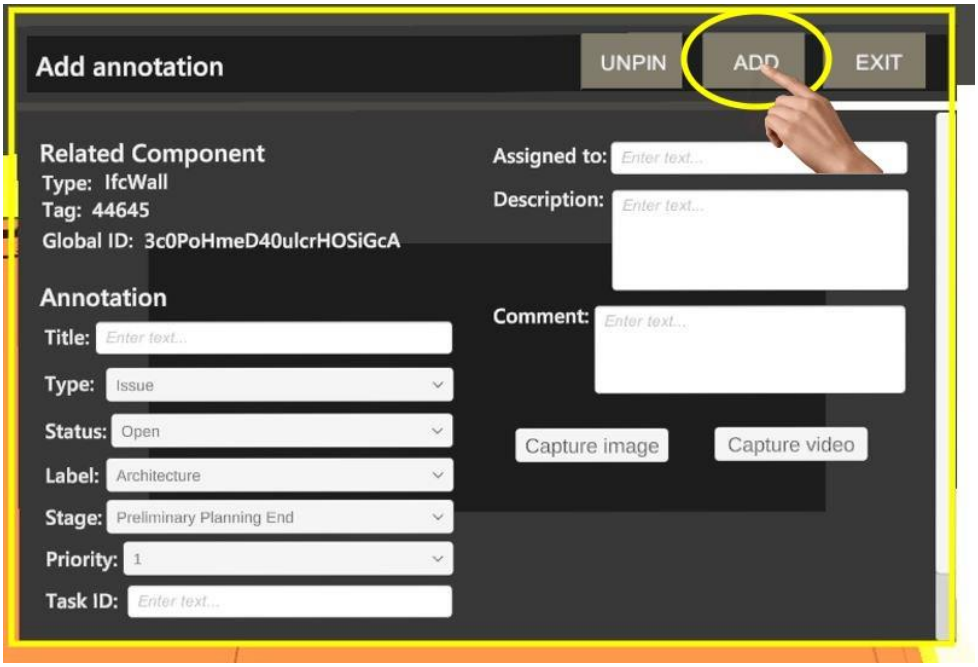


Figure 136: Air tap on the Add button to add the annotation.

Once the Add Annotation Menu is closed, the annotation is saved in a created floating red exclamation mark which is positioned near the newly annotated building component.

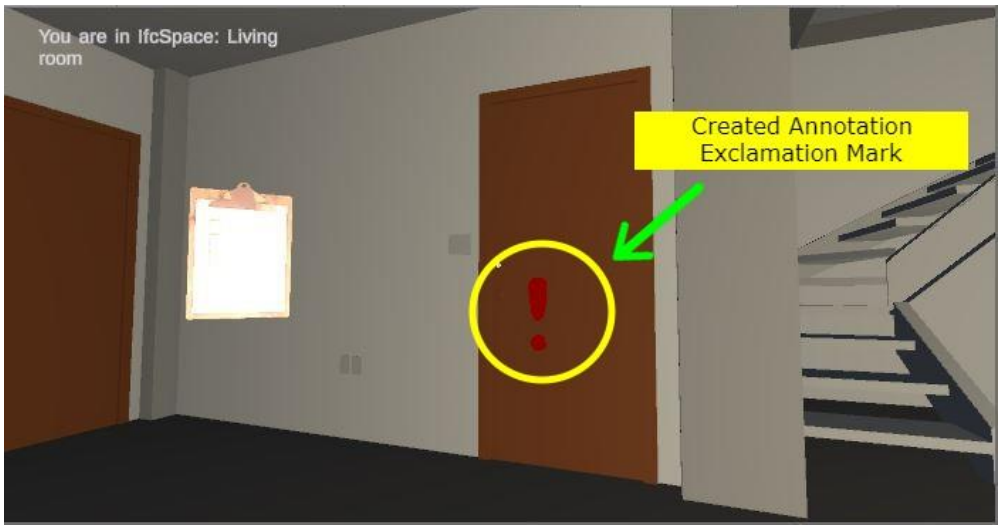


Figure 137: Created Annotation Exclamation Mark.

7.4.6 How To View Annotation

The created annotation exclamation mark can be air tapped to bring up the View Annotation Menu.

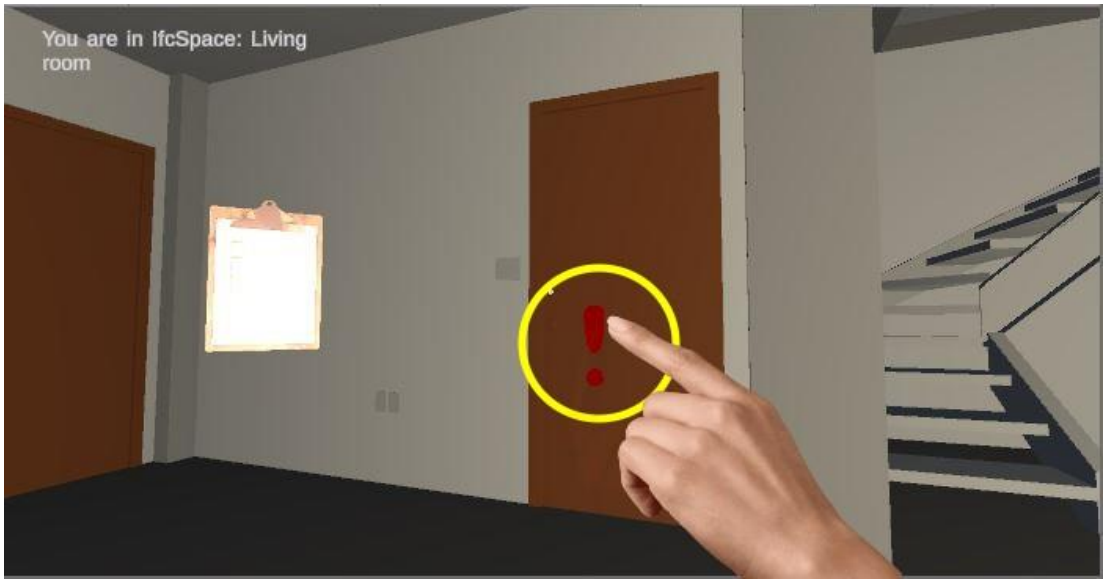


Figure 138: Air tap on the Annotation Exclamation Mark.

The View Annotation Menu is visible in the user's viewpoint.

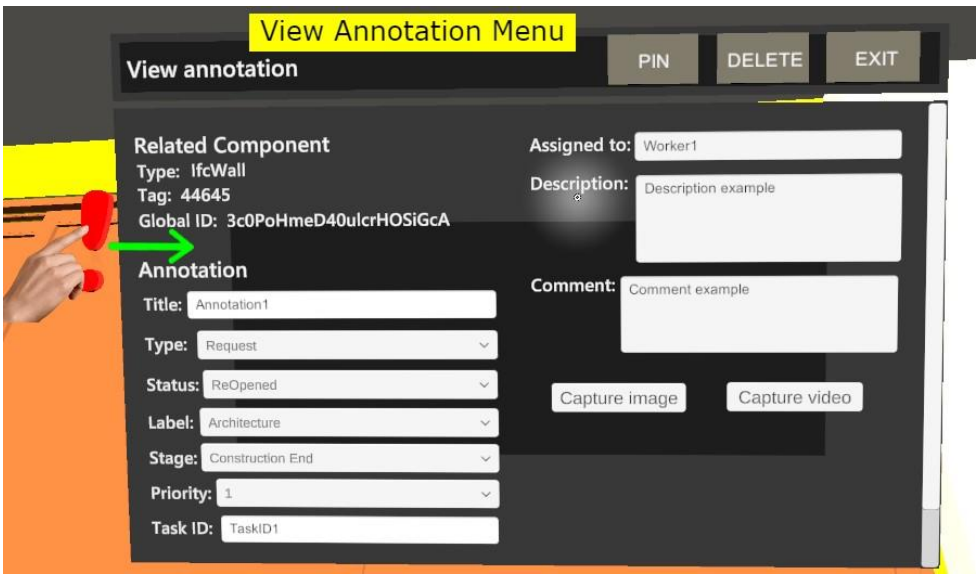


Figure 139: Air tap on the annotation mark to open the View Annotation Menu.

The Dropdown Menus and the Input Fields included in the View Annotation Menu contain the saved info of the annotation and can be air tapped to edit them.

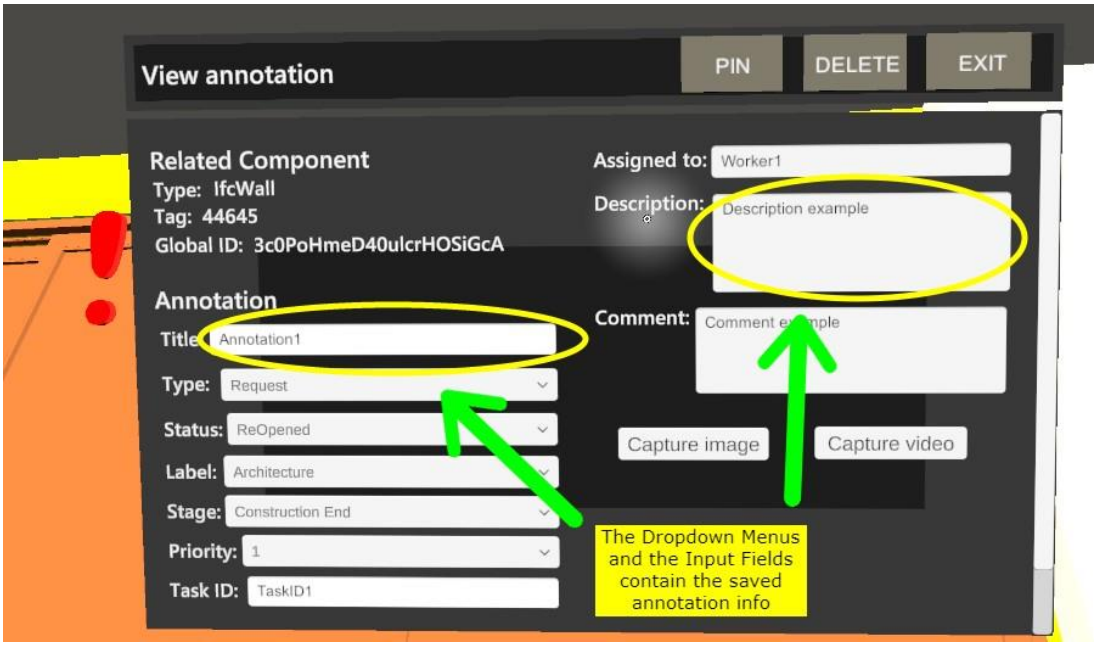


Figure 140: Saved Annotation Text.

The “Delete” button on the View Annotation Menu can be air tapped to delete the annotation and hence, the exclamation mark from the user’s viewpoint.

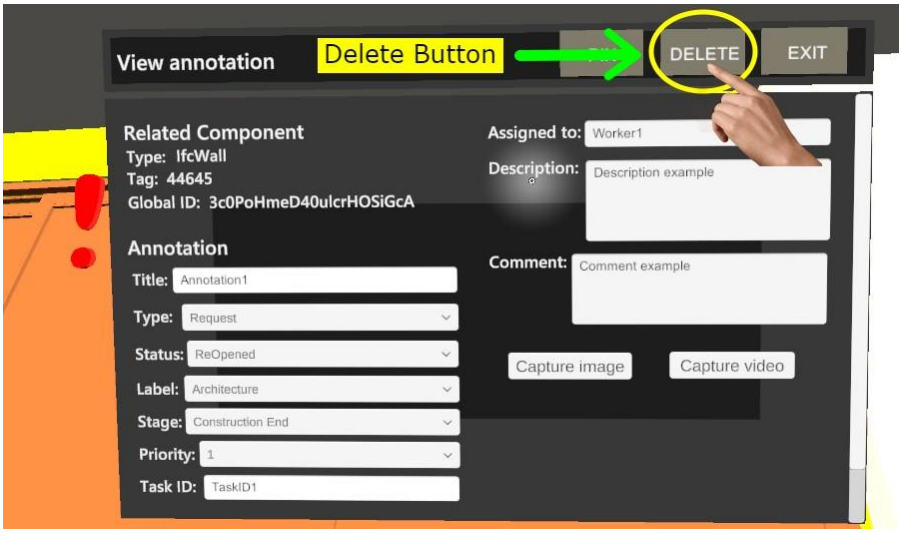


Figure 141: Air tap the “Delete” button.

7.4.7 How To Perform MEP Component Detection

To perform object detection, the Hololens should be firstly paired to a local computer. When the application starts, the user is asked to insert the IP of the computer to which the Hololens will be paired. The user air taps on the Input Field and a virtual keyboard opens-up. The user inserts the IP by typing on the visual keyboard using the air tap gesture.

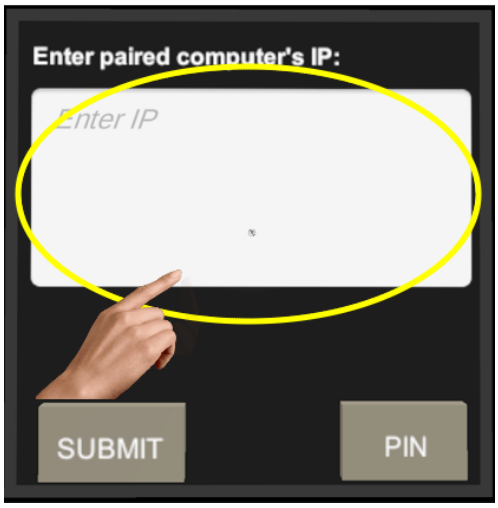


Figure 142: Air tap to insert the paired computer's IP.

To submit the inserted IP, the user selects the “Submit” button using the air tap gesture. By tapping this button, the IP is also stored in Hololens internal storage to be used in subsequent sessions. The user can open this menu at any time during runtime using the speech command “IP”.

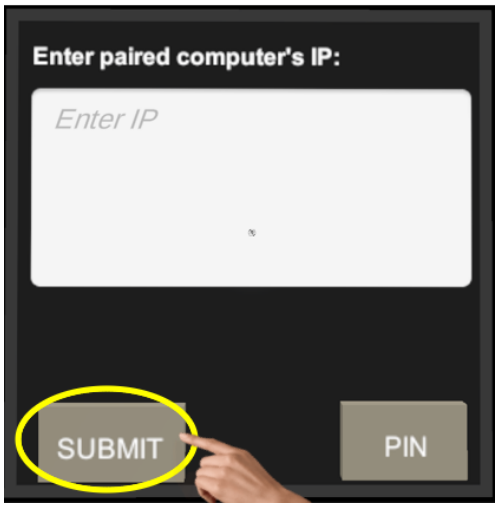


Figure 143: Air tap on the "Submit" button to save IP and store in Hololens storage.

The user can now open the DesktopARIBFA application on their computer. The user already knows the Hololens IP by following the procedure described in Section 7.1.1.3 and illustrated in Figure 98. The user types the Hololens IP in the “Host Name” field and clicks on the

“Connect” button. Subsequently, the camera stream from Hololens is depicted in the main window of DesktopARIBFA, where the detected MEP components at each frame are depicted using 2D bounding boxes.

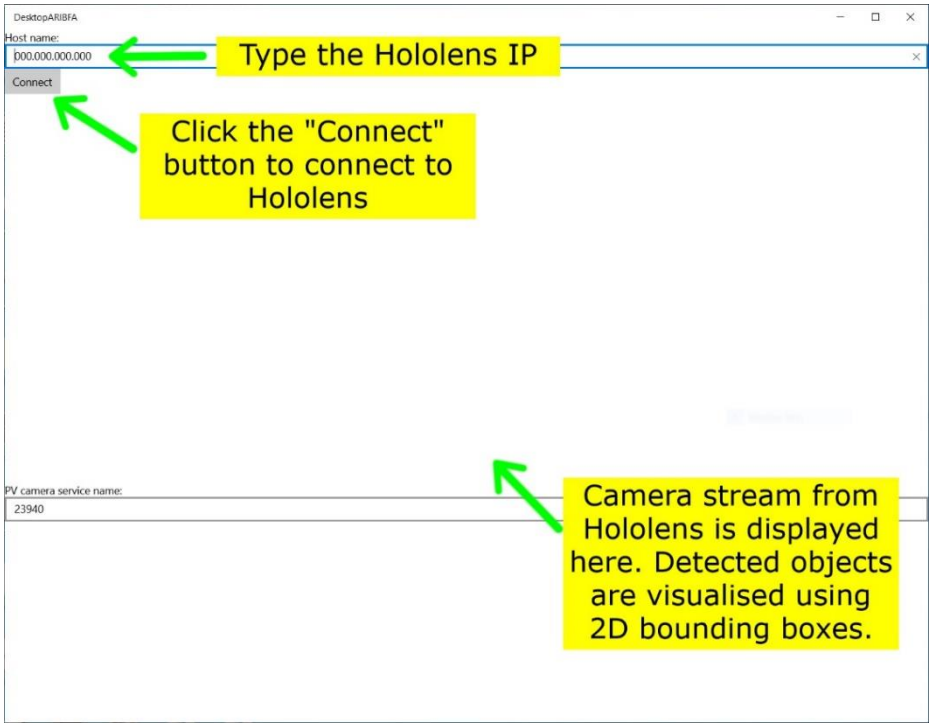


Figure 144: Open DesktopARIBFA application to connect to Hololens and perform MEP component detection.

The user can initialize the detection of MEP components in ARIBFA using the speech command "Detect". The detected objects are visualised in 3D space using 3D bounding boxes. The user can adjust the position, the scale, and the rotation of the bounding box by using the hand manipulation handlers of the bounding box and performing the "air tap and hold" gesture, which is described in Section 7.1.1.2.

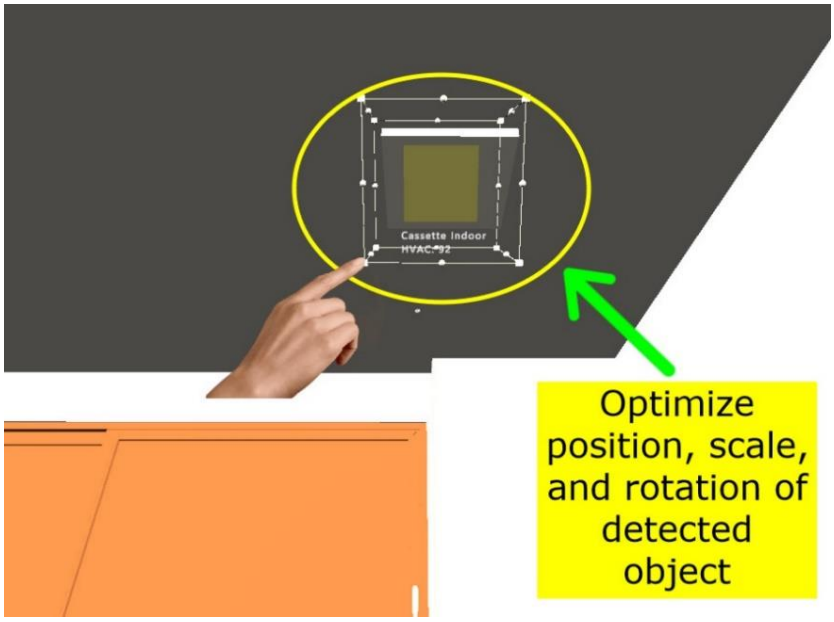


Figure 145: The user can perform the "air tap and hold" gesture to optimize the position, scale, and rotation of the detected object.

The detected objects that are missing from the 3D BIM model can be added through the Add Detected Object Menu. To open this menu, the user uses the speech command "Add properties". The user can air tap on the various parameter input fields to add them to the newly added building component via the virtual keyboard.

Figure 146: Add Detected Object Menu.

The finalization of the addition of the missing detected MEP component is done via an air tap on the “Add” button. The addition of the detected object to the IFC can also be performed with the speech command “Add to model”.

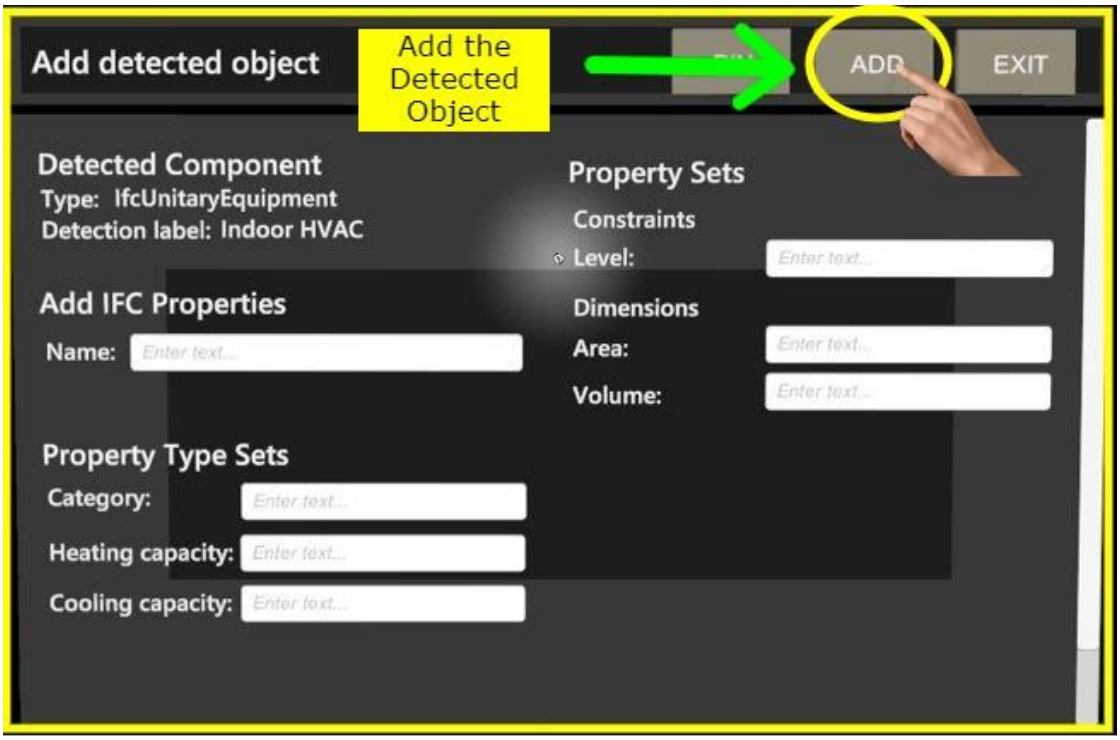


Figure 147: Air tap on the “Add” button to add the detected object to the IFC, along with the inserted IFC properties.

7.4.8 How To Add Missing IFC Properties

The building components with invalid or missing IFC properties are visualised in red color in ARIBFA to notify the user. When the user air taps on a building component with missing IFC properties, the IFC Validation Menu opens-up. This menu displays the invalid/missing IFC property, along with relevant valid IFC properties. The user can add the missing IFC properties using this menu.

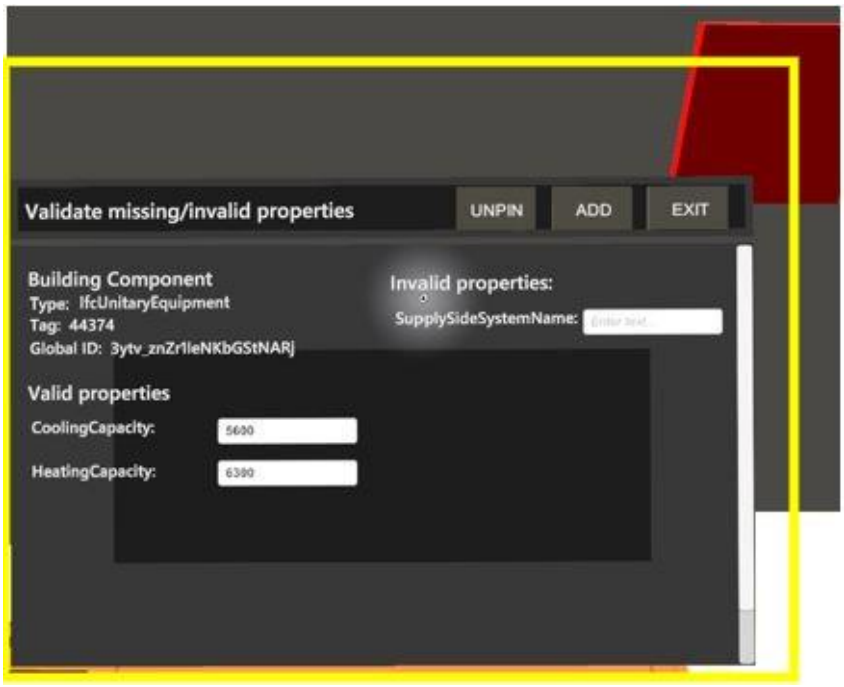


Figure 148: IFC Validation Menu.

The Input Fields present in this menu can be tapped to enable the editing via the virtual keyboard.

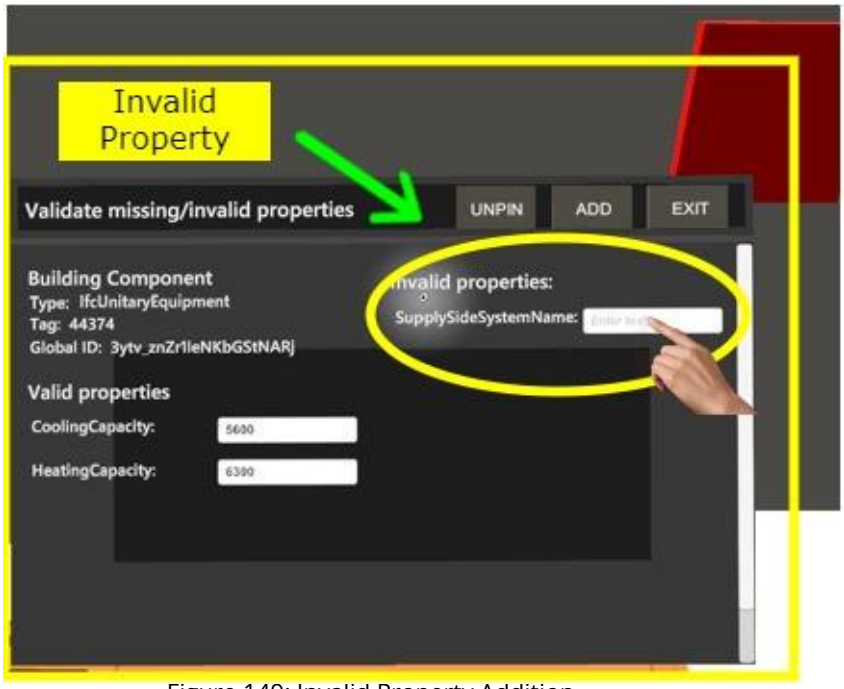


Figure 149: Invalid Property Addition.

The addition of the missing properties is finalized by air tapping on the “Add button” on the IFC Validation Menu.

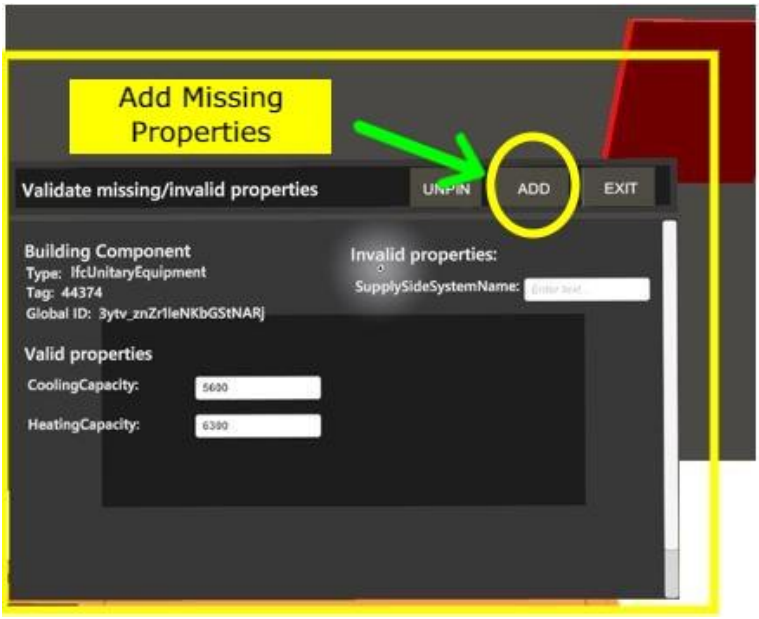


Figure 150: Add Missing Properties.

8 CONCLUSION AND OUTLOOK

An overview of the final features implemented as part of the ARIBFA application, and its submodules is outlined in this deliverable. Additionally, the user requirements and the use cases have been detailed. Furthermore, the general architecture of the application alongside its submodules has changed compared to the previous version of the application to include the new implementation features. The new features have been presented extensively considering connections to other components and tools in the BIMERR project.

Methods and tools used in the development have been described in their final state while also analyzing the application of them in the implementation stages of the whole development process. Features like the mapping of the 3D BIM model to a building site, information provision of every mapped component of the building, and annotation capabilities based on speech commands, using the annotation data model of BIMMER, have all been overviewed thoroughly in this deliverable as well as in Deliverable D5.9. Moreover, the improvements and new features of the ARIBFA submodules have been includedand detailed.

This final version of the application and its modules have the following Improvements and new features:

- The annotation process now uses the annotation data model of BIMERR.
- Visualisation of Process and Task Details data model
- Visualisation of Error report for missing or invalid IFC properties
- Editing of IFC properties
- Detection of unregistered MEP components
- Development of DesktopARIBFA, a desktop application to handle the connection of a computer with the Hololens in order to perform object detection
- Update of the 3D BIM model with detected MEP components
- Keycloak Integration
- Integration of Rest request to BIF platform

Next, the APIs necessary for the connection of the ARIBFA app to the BIF platform, were presented. Finally, in this deliverable the deployment plan and a user manual have been included.

9 REFERENCES

[1] J. Martin, Rapid application development, Macmillan Publishing Co., Inc., 1991.

[2] [Online]. Available: https://bleuwire.com/advantages-rapid-application-development-rad/.

[3] [Online]. Available: https://unity.com/.

[4] [Online]. Available: https://bimerr.eu/bimerr-tools/process-workflow-modelling-and-automation-toolkit-pwma/.

[5] [Online]. Available: https://docs.microsoft.com/en-us/hololens/hololens1-hardware.

[6] [Online]. Available: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/mrtk-getting-started.

[7] [Online]. Available: https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/Updating.html#upgrading-to-a-new-version-of-mrtk.

[8] [Online]. Available: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/platform-capabilities-and-apis/using-the-hololens-emulator.

[9] [Online]. Available: https://en.wikipedia.org/wiki/Xbox_Wireless_Controller.

[10] S. Lockley, C. Benghi and M. Cerny, "Xbim.Essentials: a library for interoperable building information applications," *Journal of Open Source Software,* vol. 2, no. 20, p. 473, 2017.

[11] [Online]. Available: https://assetstore.unity.com/packages/tools/modeling/runtime-obj-importer-49547.

[12] [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/.

[13] S. Agarwal and B. Brojeshwar, "3D point cloud registration with shape constraint," in *IEEE International Conference on Image Processing (ICIP)*, 2017.

[14] J. Park, Z. Qian-Yi and K. Vladlen, "Colored Point Cloud Registration Revisited," in *IEEE International Conference on Computer Vision*, 2017.

[15] Y. Aoki, H. Goforth, R. A. Srivatsan and S. Lucey, "Pointnetlk: Robust & efficient point cloud registration using Pointnet," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[16] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[17] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan and S. Song, "DeepVCP: An end-to-end deep neural network for point cloud registration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[18] X. Huang, L. Fan, Q. Wu, J. Zhang and C. Yuan, "Fast registration for cross-source point clouds by using weak regional affinity and pixel-wise refinement," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 2019.

[19] C. Kim, S. Hyojoo and K. Changwan, "Fully automated registration of 3D data to a 3D CAD model for project progress monitoring," *Automation in Construction,* vol. 35, pp. 587-594, 2013.

[20] M. Bueno, F. Bosché, H. González-Jorge, J. Martínez-Sánchez and P. Arias, "4-Plane congruent sets for automatic registration of as-is 3D point clouds with 3D BIM models," *Automation in Construction,* vol. 89, pp. 120-134, 2018.

[21] J. Ratajczak, M. Riedl and D. T. Matt., "BIM-based and AR application combined with location-based management system for the improvement of the construction performance," *Buildings,* vol. 9, no. 5, p. 118., 2019.

[22] M. Kopsida and I. Brilakis, "BIM registration methods for mobile augmented reality-based inspection," in *Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM 2016)*, Limassol, Cyprus, 2017.

[23] [Online]. Available: [Online]. Available: https://github.com/microsoft/MixedRealityToolkit-Unity..

[24] [Online]. Available: https://developer.vuforia.com/.

[25] [Online]. Available: https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity.html.

[26] [Online]. Available: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/tutorials/mr-learning-base-02#creating-and-configuring-the-scene.

[27] D. Acharya, M. Ramezani, K. Khoshelham and S. Winter, "BIM-Tracker: A model-based visual tracking approach for indoor localisation using a 3D building model," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 150, pp. 157-171, 2019.

[28] Z. Turgut, Z. G. A. Gulsum and A. Sertbas, "Indoor localization techniques for smart building environment," *Procedia computer science,* vol. 83, pp. 1176-1181, 2016.

[29] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics,* vol. 33, no. 5, pp. 1255-1262, 2017.

[30] D. Acharya, K. Khoshelham and S. Winter, "BIM-PoseNet: Indoor camera localisation using a 3D indoor model and deep learning from synthetic images," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 150, pp. 245-258, 2019.

[31] A. Xiao, R. Chen, D. Li, Y. Chen and D. Wu, "An indoor positioning system based on static objects in large indoor scenes by using smartphone cameras," *Sensors,* vol. 18, no. 7, p. 2229, 2018.

[32] P. Hübner, K. Clintworth, Q. Liu, M. Weinmann and S. Wursthorn, "Evaluation of HoloLens tracking and depth sensing for indoor mapping applications," *Sensors ,* vol. 20, no. 4, p. 1021, 2020.

[33] B. Mahmood, S. Han and D.-E. Lee, "BIM-Based Registration and Localization of 3D Point Clouds of Indoor Scenes Using Geometric Features for Augmented Reality," *Remote Sensing,* vol. 12, no. 14, p. 2302, 2020.

[34] P. Hübner, Weinmann, Martin and S. Wursthorn, "Voxel-Based Indoor Reconstruction From HoloLens Triangle Meshes," *arXiv preprint arXiv:2002.07689 ,* 2020.

[35] P. Hübner, M. Weinmann and S. Wursthorn, "Marker-based localization of the microsoft hololens in building models." International Archives of the Photogrammetry," *Remote Sensing & Spatial Information Sciences,* vol. 42, no. 1, 2018.

[36] F. Baek, I. Ha and H. Kim, "Augmented reality system for facility management using image-based indoor localization," *Automation in Construction,* vol. 99, pp. 18-26, 2019.

[37] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[38] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[39] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767,* 2018.

[40] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934,* 2020.

[41] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition.*, 2014.

[42] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015.

[43] S. Ren, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015.

[44] K. He, G. Gkioxari, P. Doll{\'a}r and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017.

[45] S. Zhi, Y. Liu, X. Li and Y. Guo, "Toward real-time 3D object recognition: A lightweight volumetric CNN framework using multitask learning," *Computers & Graphics,* vol. 71 , pp. 199-207, 2018.

[46] C. R. Qi, O. Litany, K. He and L. J. Guibas, "Deep Hough Voting for 3D Object Detection in Point Clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[47] C. R. Qi, X. Chen, O. Litany and L. J. Guibas, “ImVotenet: Boosting 3D object detection in point clouds with image votes,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[48] S. Shi, W. Xiaogang and L. Hongsheng, “Pointrcnn: 3d object proposal generation and detection from point cloud,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[49] M. Simony, S. Milzy, K. Amendey and H.-M. Gross, “Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[50] G. Li, Z. Song and Q. Fu, “A New Method of Image Detection for Small Datasets under the Framework of YOLO Network,” in *IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference*, 2018.

[51] [Online]. Available: https://onnx.ai/.

[52] [Online]. Available: https://github.com/microsoft/HoloLensForCV.

[53] [Online]. Available: https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/Updating.html#upgrading-to-a-new-version-of-mrtk.

[54] [Online]. Available: https://docs.microsoft.com/en-us/hololens/hololens1-hardware.