Robotics Engineering

RICE

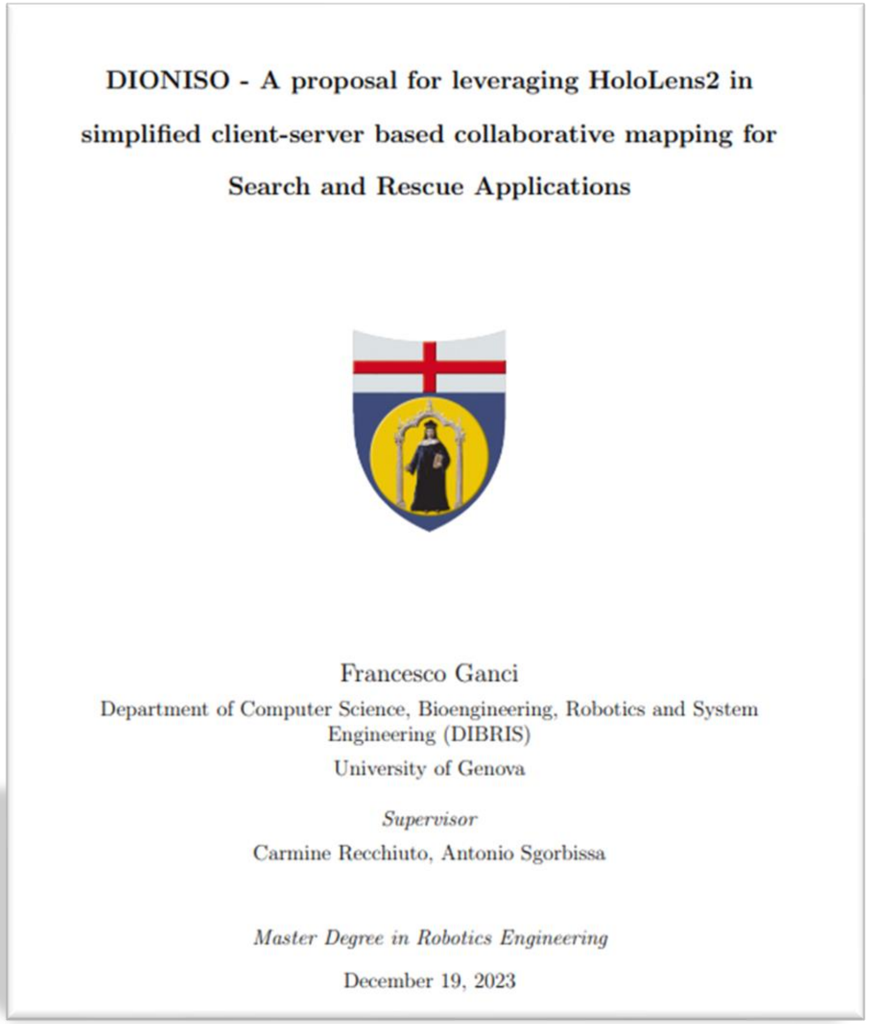
HoloLens2 in Search&Rescue: a P.o.C.

Master’s Degree in **R**obotics **E**ngineering

December 19, 2023

Francesco Ganci, S4143910
Supervisors: Carmine Recchiuto (RICE lab), Antonio Sgorbissa (RICE lab)

Università di Genova



DIONISO - A proposal for leveraging HoloLens2 in simplified client-server based collaborative mapping for Search and Rescue Applications

Francesco Ganci
Department of Computer Science, Bioengineering, Robotics and System Engineering (DIBRIS)
University of Genova

Supervisor
Carmine Recchiuto, Antonio Sgorbissa

Master Degree in Robotics Engineering

December 19, 2023

Key Concepts

Main Research Fields:

- *Search and Rescue*
- Mixed Reality

Main Project Topics

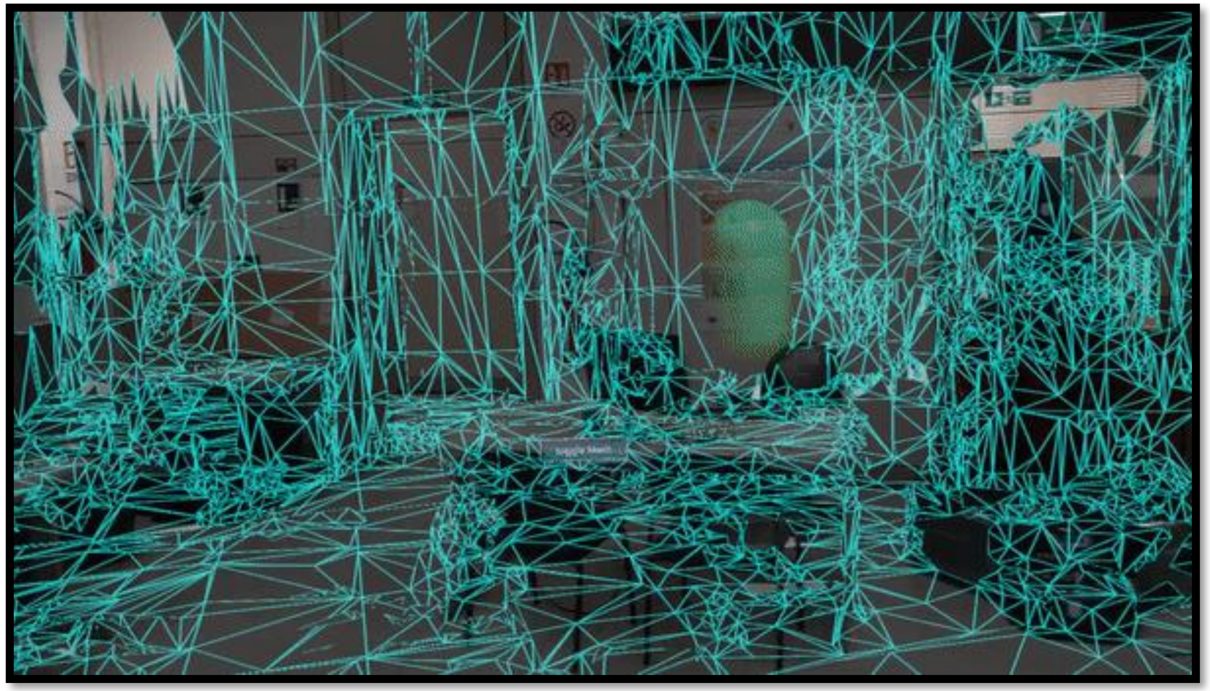
- Localisation and Mapping
- Client-Server based System
- *Server-side collaborative Mapping*

Objectives

- **Main Objective**
  - Augmented/Mixed Reality applied to Search&Rescue context
  - Supporting *First Respondents* to explore a vast disaster area
- **First Disaster Assessment**
  - First Seach of survivors
  - First Search of victims
  - First Search of ways for rescuers' vehicles and people
- Next emergency management phases will *rely on* first assessment informations.

*A view of **Amatrice** after the eartquake of year 2017.*

Previous Works – 3D Mapping in MR (1)

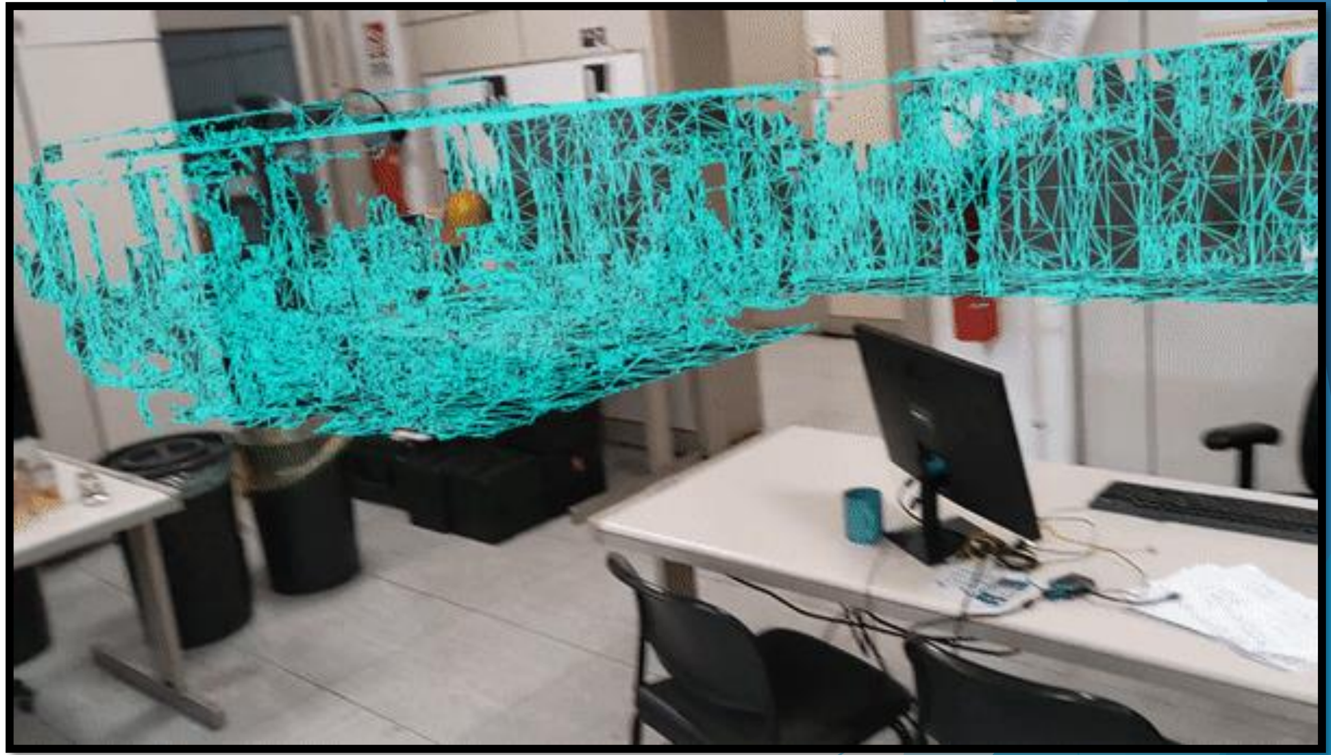


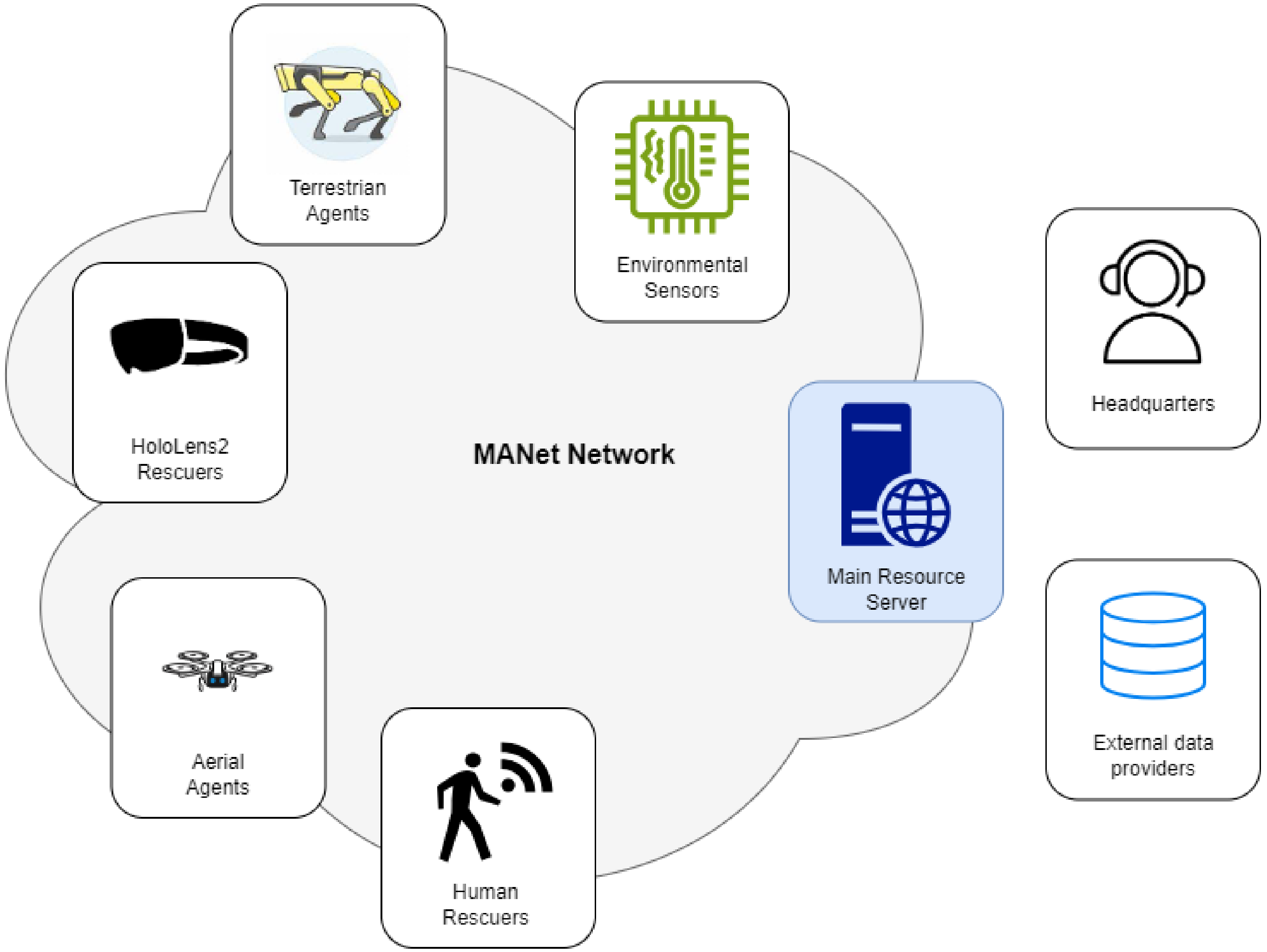
Relevant features from the previous work:

- Laser-scanning of the environment
- Data employed for **Navigation**
- **3D** Reconstruction of the environment

Previous Works – 3D Mapping in MR (2)

- Compact wireframe of the explored space
  - Manipulable map
- The rescuer has a way to see the overall environmental structure
- *Maps can be shared among the rescuers*





A more extended point of view.

SaR organisation is made of **many different agents** coordinated by a **headquartier** that need to share information **using a emergency nework** which is **not assumed to be stable**. **External information** and **On-site Information** have to be mixed together **in near-realtime**.

Project Cornerstones

- **Localisation and mapping**
  - Many features rely on localization:
    - in-place visualisations (environmental sensors integration)
    - S.O.S. signals (need to know where the operator is)
  - Strictly connected with *interoperability*
- **Information Sharing in near-realtime**
  - Minimum time between the generation of the information and its availability
- **Network stability is *not guaranteed***
  - Devices must work mostly *offline*
  - Sharing data taking advantage of any connectivity spots

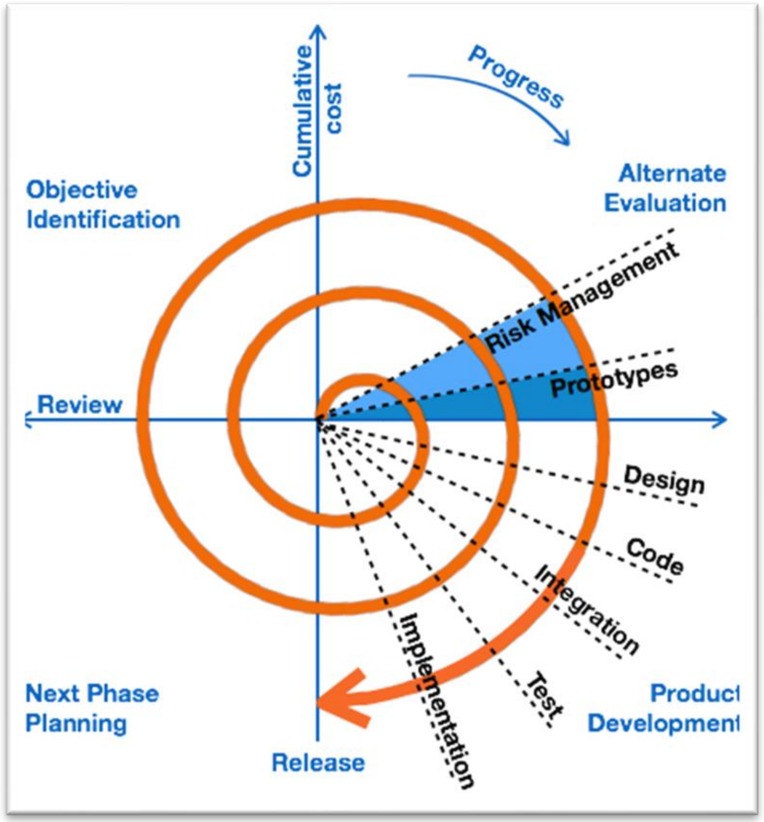
PART 1

HoloLens2 Application Overview

HoloLens2 Characteristics

- 4Gb **RAM Memory**
- **Connectivity**
  - Wi-Fi
  - Bluetooth
- **Battery**
  - 2 Hours Battery life
- **Interactions**
  - Hands Tracking
  - Voice Commands
- **6DoF Position Tracking**
  - Self-localization w.r.t. a *on-the-fly* frame
- ***No support for GPS***

Project Plan: HoloLens2 side



Starting from Scratch using an iterative approach:

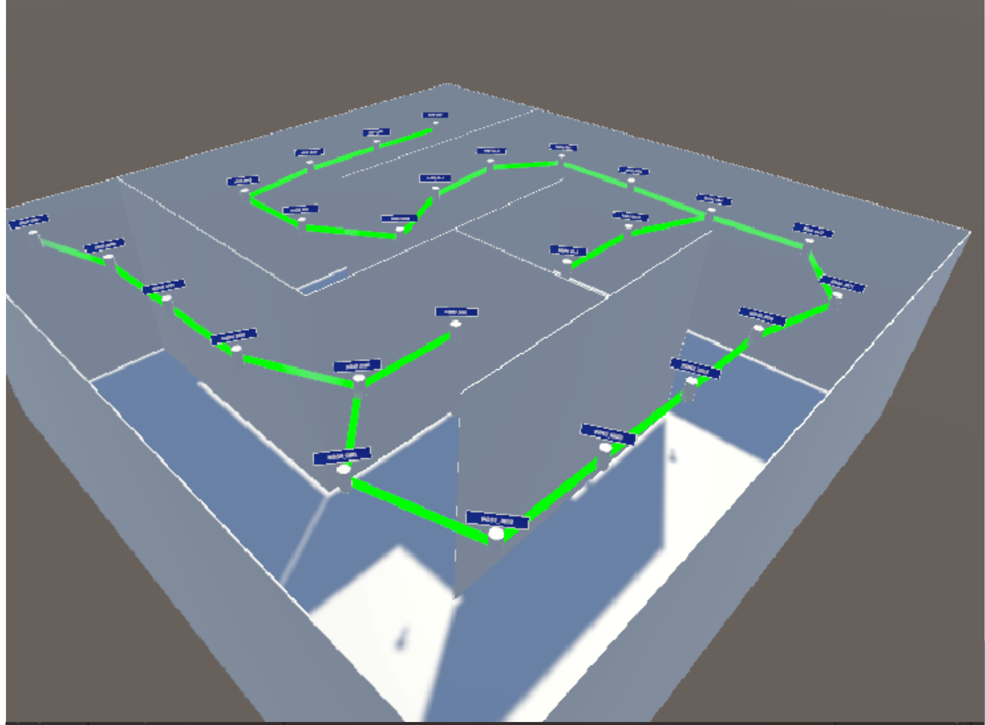
- Modelling user’s activity
- Implement positions tracking
    - Mapping process Management
- Data Sharing
- *Data Visuals and User Experience*
    - Natural-sized point of view
    - Compact point of view

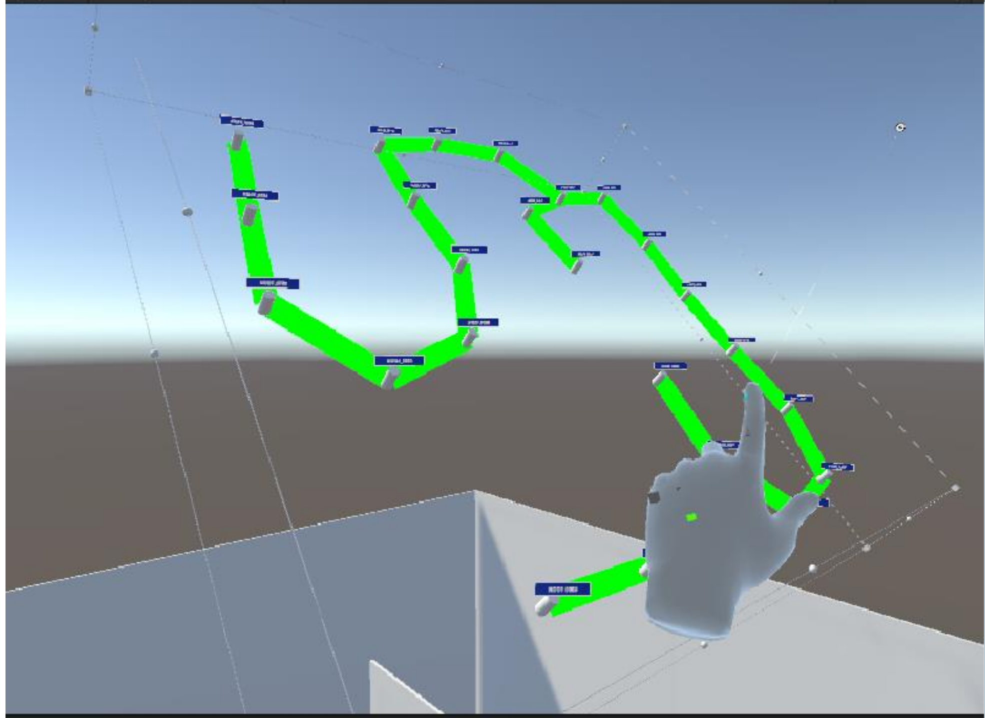
- *… then the server …*

Data Model: *Accessibility Graph*

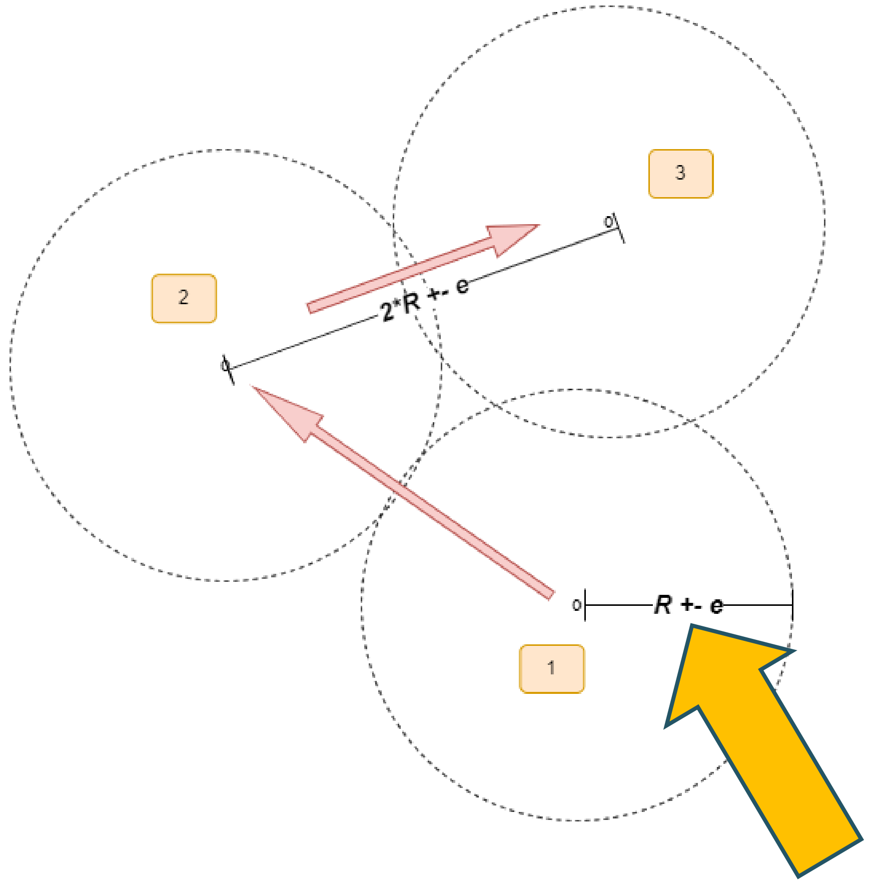
A data structure able to capture the user's movements in the area

- **Waypoints**
  - Positions (sampling)
- **Paths**
  - Physically traveled by the user to go from one Waypoint to another one *(Accessibility)*
- **Paths are recorded once**
  - Only when a new waypoint is created
  - This allows to achieve better clean paths *(data quality)*



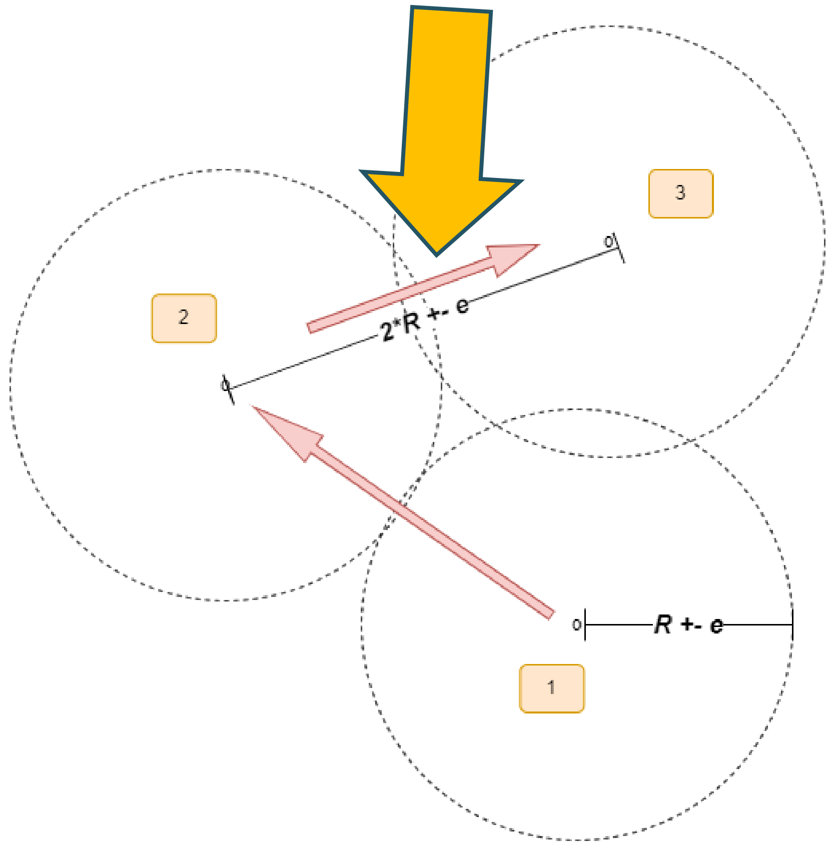


Position **Tracking** *(localisation)*



- Localisation, defined as
  - *Recalling* known positions inside the data structure
- There's a **List of Positions**
  - Another level of indexing
  - Semi-sorted (frame by frame) w.r.t. distance from the current user's position
  - Optimized approach
- Continuous sorting in time
  - Assumption: the user is not moving too fast

Position **Discovering** *(mapping)*



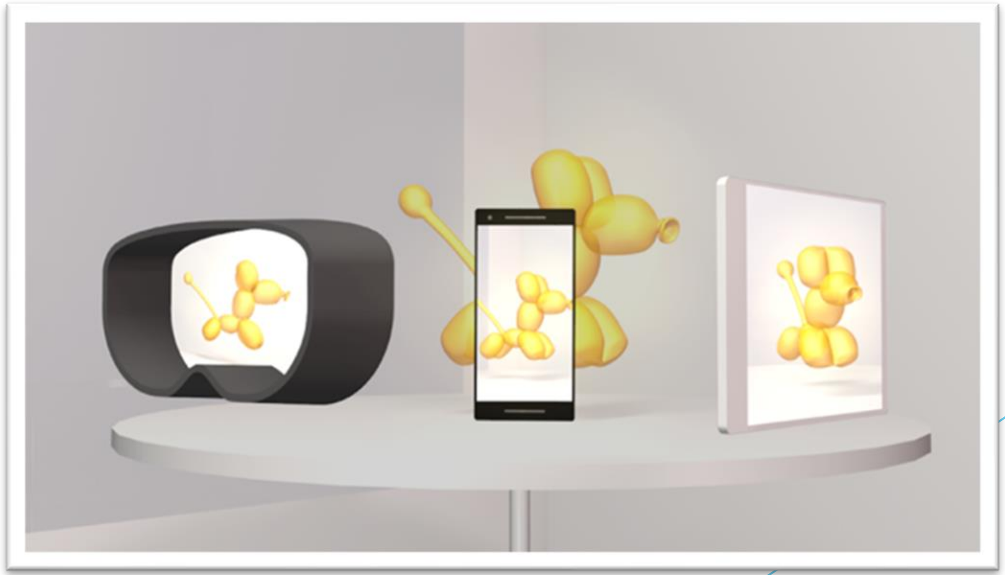
- Each waypoint has a *radius* around it
- A new position is created
  - Each time a distance *doubled compared to* the radius is detected from the identified nearest point
- Tuned Approach
  - Radius : *base distance*

This creation rule ensures a *good distribution of points* across the space

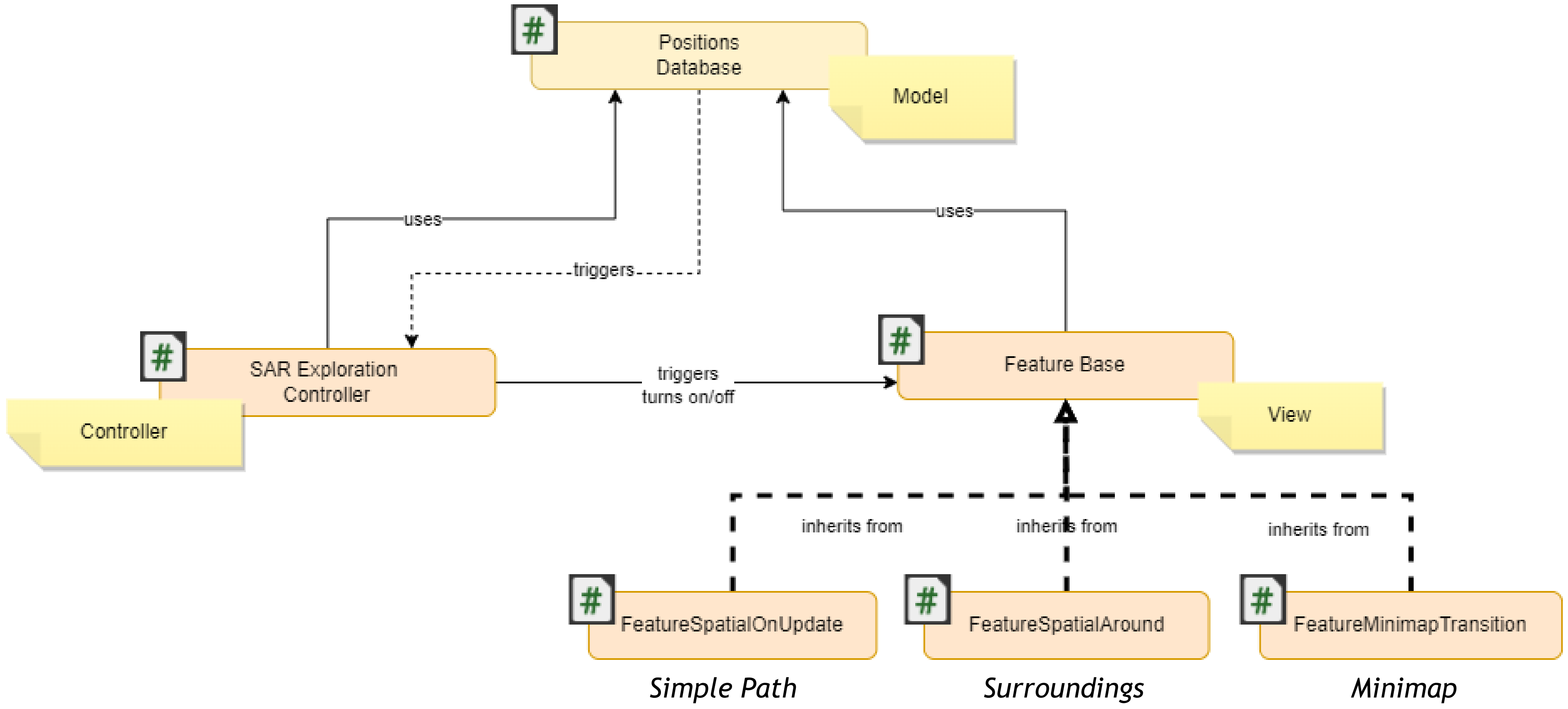
Sharing data – *Device Calibration*

- **Operative Frame Transformation**
  - Users *agree* on a point in the space as **common origin**
  - Looking in one precise direction, users can agree on **orientation**
  - This step allows to **make sharable the data from the mapping**
- A user procedure for calibration have been developed.

▶*Calibration is required for interoperability at localization and mapping level*



Visuals Design Pattern



- Derived from *Model-View-Controller* design pattern
- Feature → *visual*

PART 2

Server Application Overview

System Base Requirements

Design from scratch

- Using a Linux-based remote machine (CINECA ADA Cloud) with Docker Engine

Information Collection

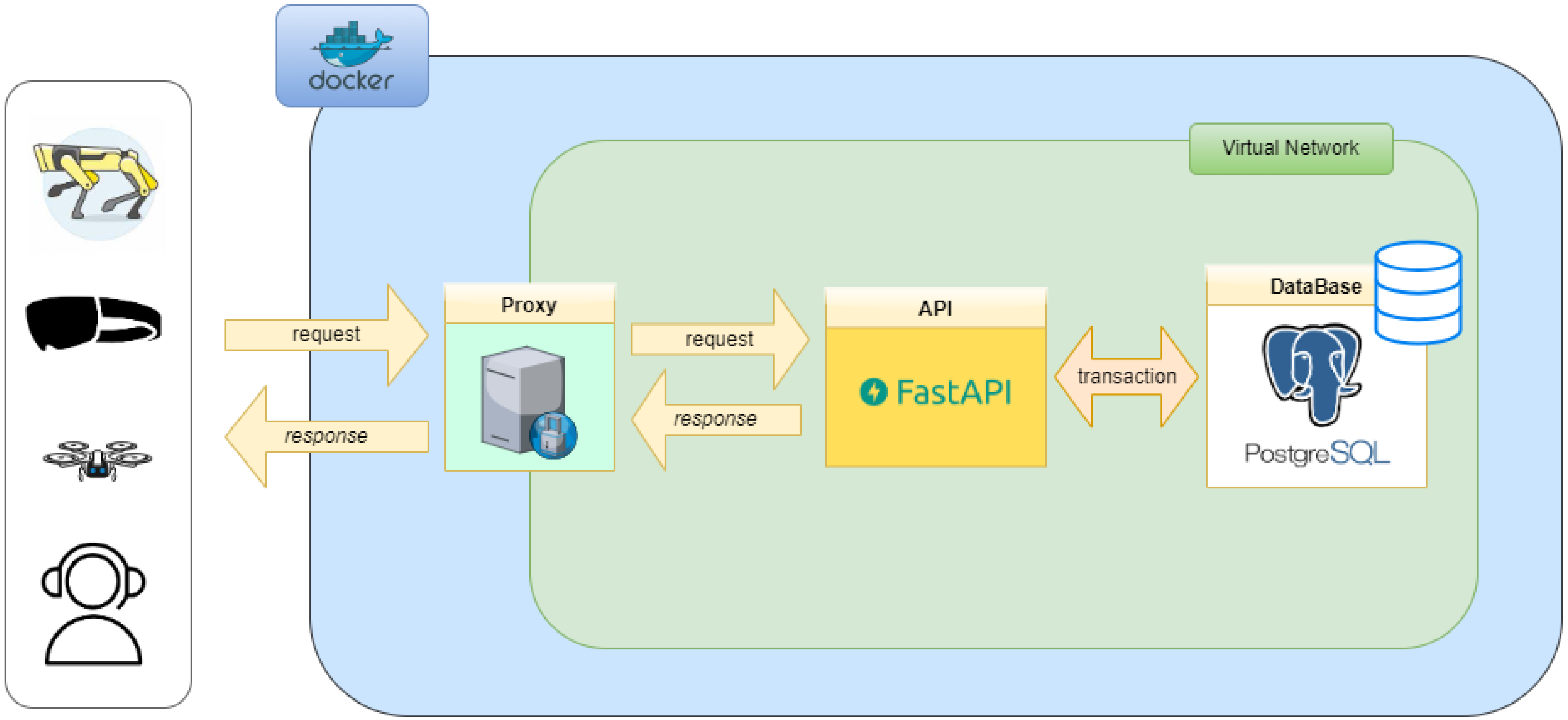
- Unified data model
- Integrating different sources/agents

Information Sharing

- Server has to be a *central hub* for sharing informations
- Not important which device generated the information → ***collaboration***

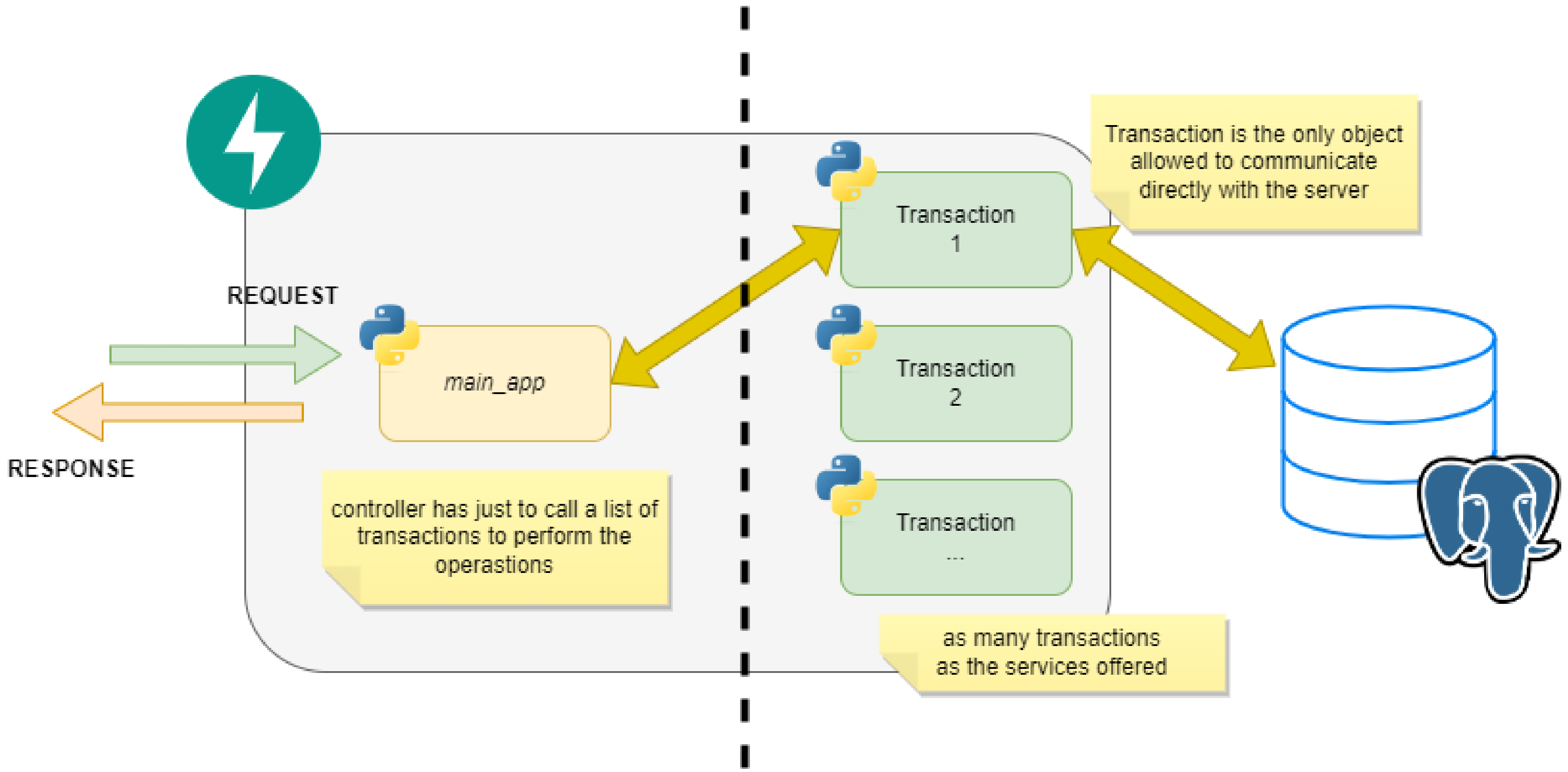
Data Management Challenges

- Similar Positions Matching
  - Tuned approach based on distance
  - **IDs negotiation and reconciliation**
- Efficient networking usage
  - Only unknown data should be exchanged
- Efficient Storage
  - Need for storage with as less redundancies as possible
  - Efficient measurements mix → Collaborative mapping, realtime information sharing



Server Side

- Three-tier architecture based on HTTP
- RDBMS storage *(Relational approach)*
- Implementation as combination of microservices


REQUEST
RESPONSE
main_app
controller has just to call a list of transactions to perform the operastions
Transaction 1
Transaction 2
Transaction ...
Transaction is the only object allowed to communicate directly with the server
as many transactions as the services offered

API Module Structure

API encloses all the Logics of the server application. It is a **transaction-based architecture** with a control module as interface (*main_app*) and a set of independent classes implementing the database queries and operations.

For instance, Login, Upload, Download Acquire device, are all examples of *transactions*, each of them having its own class.

Download Upload Protocol

DOWNLOAD *(first operation)*

- *Given a center position and a radius*
- Extraction of waypoints inside that radius
  - Unreachable waypoints are excluded
  - Exchanged only unknown waypoints
- Extraction of paths linking the selected waypoints
  - **Minimum amount of data**

UPLOAD *(anytime)*

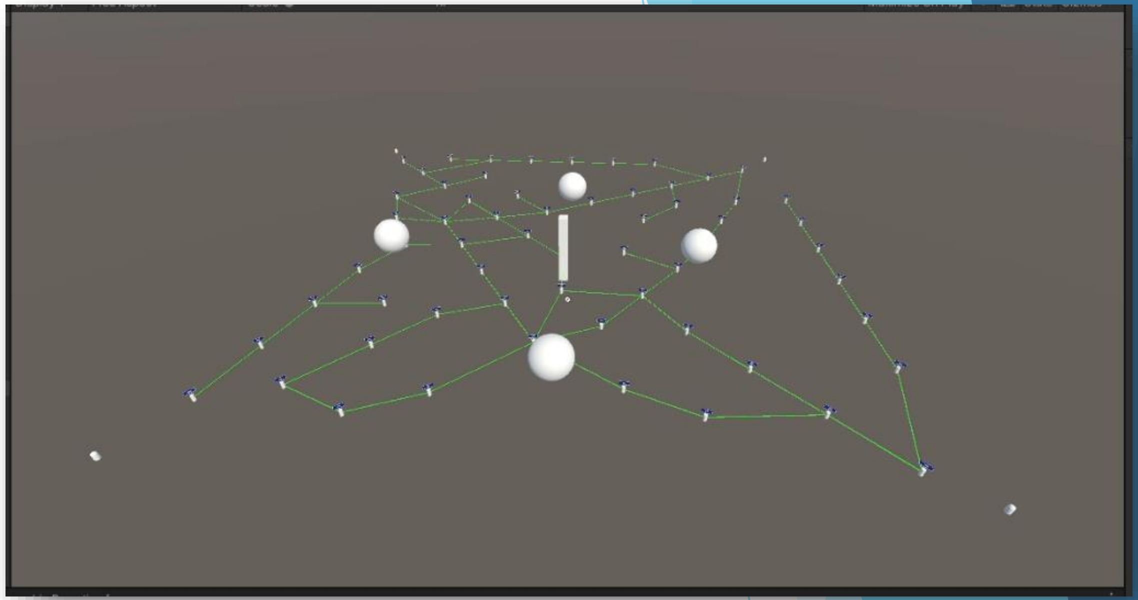
- *Given a set of waypoints and paths to upload*
- **Waypoints alignment and recording**
  - **Tuned matching**
  - IDs matching and negotiation
- Paths pre-filtering
  - Due to the alignment operation
- **Paths recording**

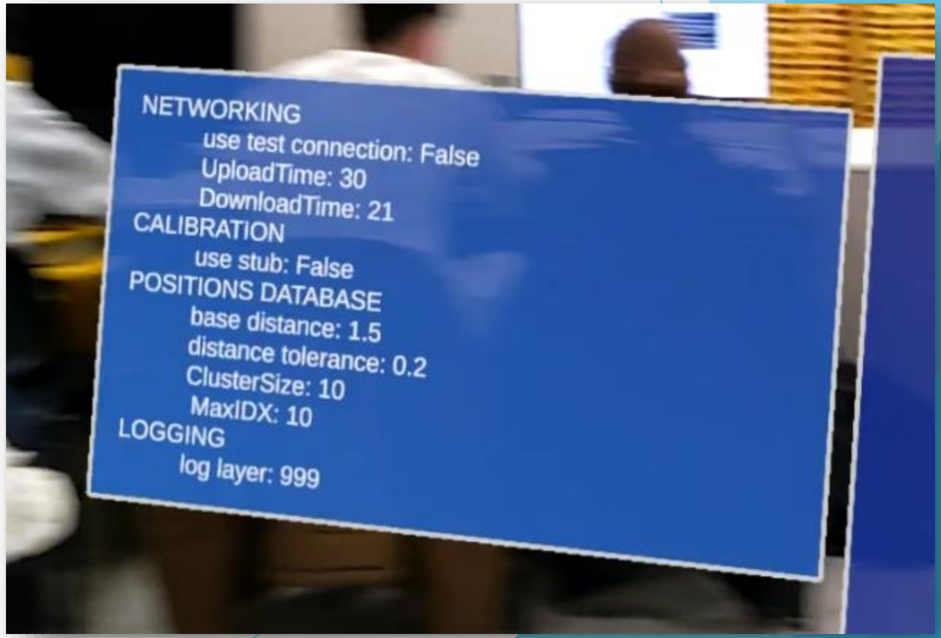
PART 3

Application Testing

Application Testing (1)

- After detection of **a set of metrics** for describing performances
  - Frame rate, Odometry Vs. identified distance, HIT/MISS ratio, Networking latency, Networking received/sent data
- **User Simulated Tests**
  - Especially for *visuals*, moving inside a test map
- **Benchmark tests**
  - Assessing performances depending on user's velocity and paths "randomness"
  - Trying to estimate a good tuning for practical tests
- **Device Practical tests**
  - Mainly focused on trying the complete experience (*User Acceptance Tests*)





```
NETWORKING
    use test connection: False
    UploadTime: 30
    DownloadTime: 21
CALIBRATION
    use stub: False
POSITIONS DATABASE
    base distance: 1.5
    distance tolerance: 0.2
    ClusterSize: 10
    MaxIDX: 10
LOGGING
    log layer: 999
```

Application Testing (2)

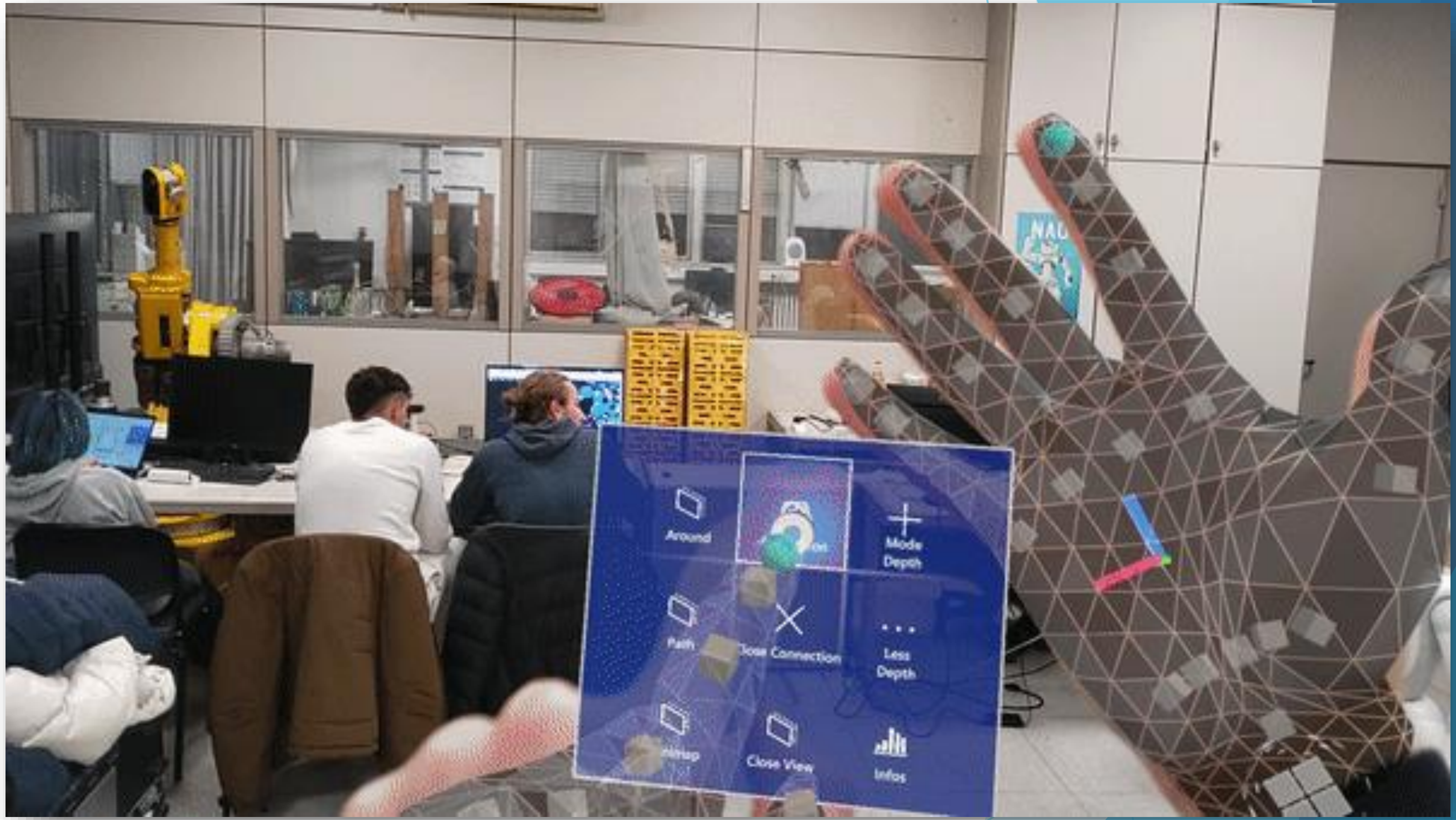
Type of Result	Results
Maximum Frame Rate	*Simulation*: 250fps *Device (no recording)*: 60fps | *Device (recording)*: 30fps
Minimum Base Distance	(in simulated environment) *Without optimisations*: **0.5m** | *Using optimisations*: **0.35m**
Stable Base Distance	**1.5m** (*both in simulation and from device testing*)
Maximum User's speed	*Simulation*: **1.7m/s** (about 6Km/h, i.e. a quick walk) *Device Testing*: **1.4m/s** (from practical tests, no benchmark)
Best Line Benchmark Test	*User's average velocity*: **1.67m/s** (~6Km/h) *Line Length*: **250m** | *Generated points*: **186**
Device Tuning	*Radius*: **1.5m** *with tolerance* **0.2m** *Cluster size*: **10** | *Max Indexes*: **10**

- Results mostly from simulated environment in order to estimate the device app performances
  - In particular, stress tests in simulated environment
- Found a stable configuration, device tests → *tuning validation*

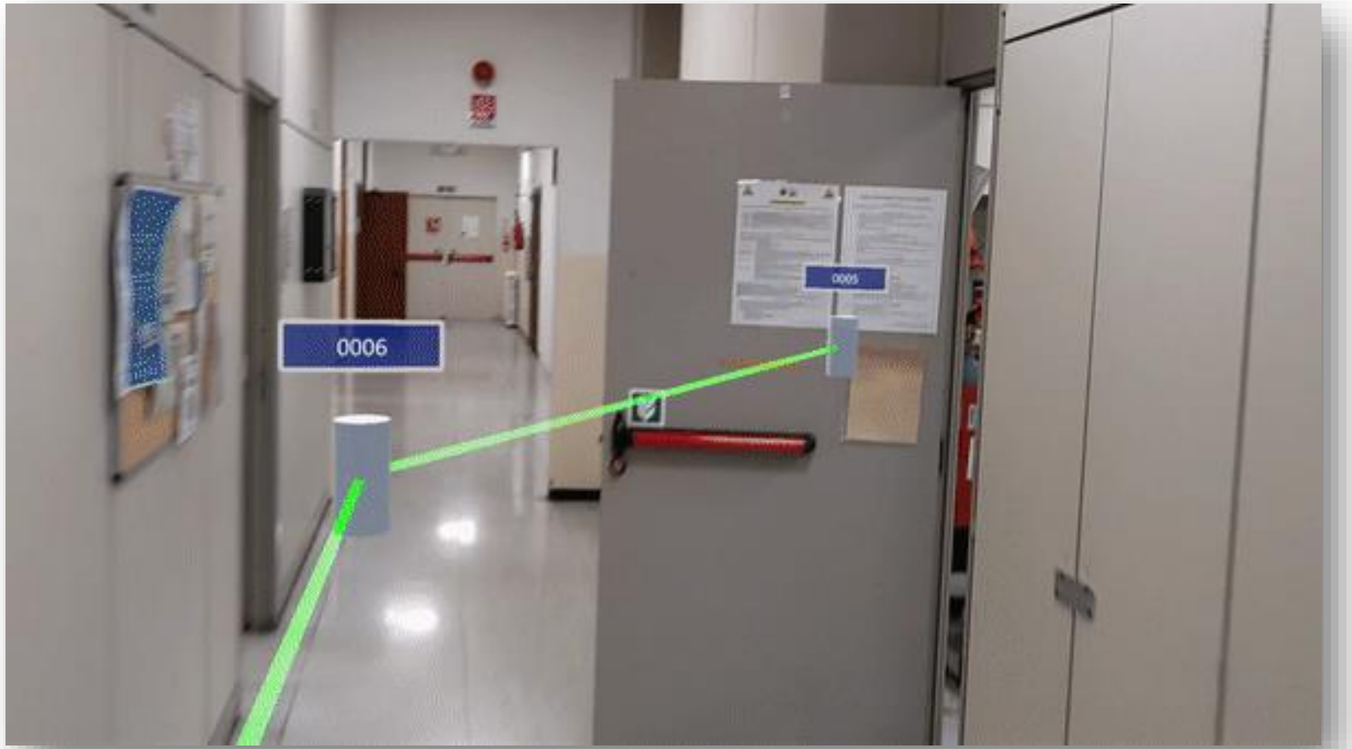
Operative Calibration Procedure

- Calibration is issued by command
- An aim appears on the screen
  - User remains still in a “agreed” place
  - Looking at a “agreed” point
- Snapshot of position and orientation
- Immediately after, the system downloads *and transforms* positions from the server

In the video, Surroundings visual is used to show what have been downloaded from the server.

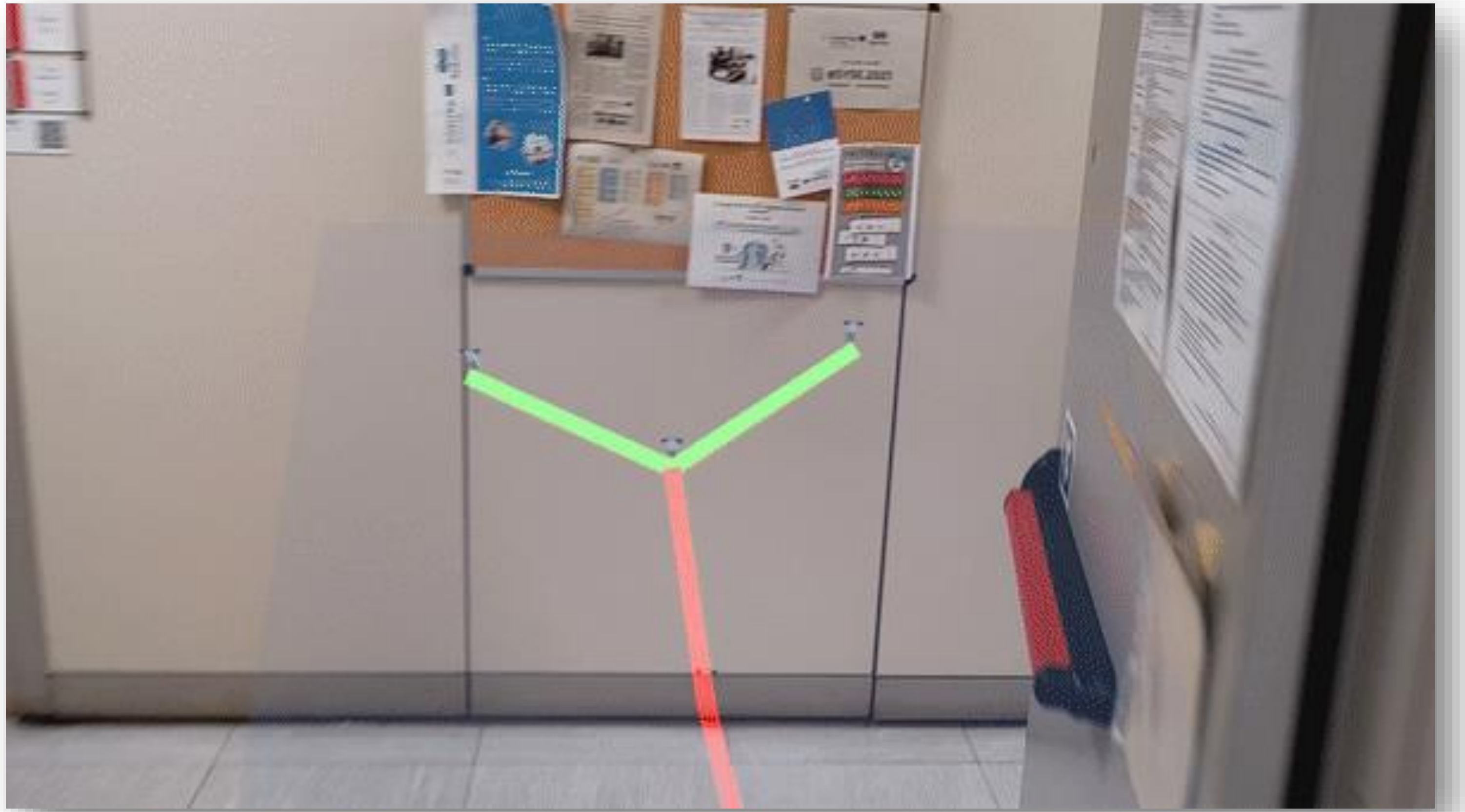


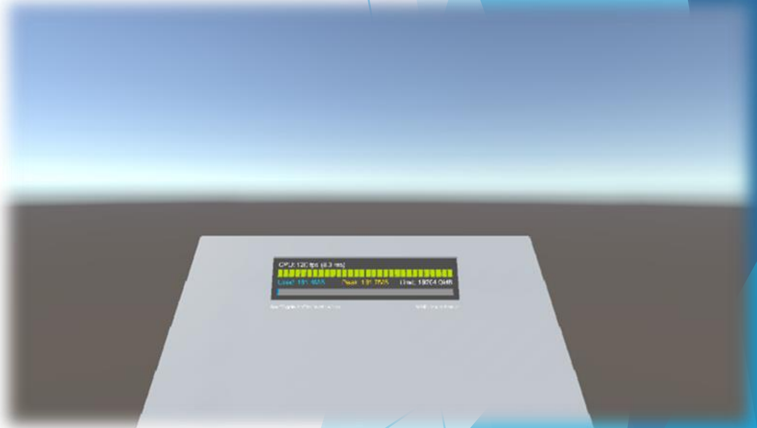
Simple Path Visual & Surroundings Visual



- **Simple Path** *(not in the video)*
  - It draws the path as the user travels it, leaving behind “bread crumbs”
- **Surrounding Visual (*video*)**
  - It shows all the paths around the user (recursive exploration with adjustable tuning)

Minimap Visual





- **Re-adaptation** of Surroundings visual
  - Bounded in a plane in front of the user; important to not cover the line of sight
  - User’s position represented **in red** in the map
- It resembles a “paper map” *(figure on the right)*

Written and Directed
by
QUENTIN
TARANTINO

Thanks!

Francesco Ganci