This project is funded by the European Union
G.A No 101016906

Project Number: 101016906
Start Date of Project: 2021/01/01
Duration: 48 months

Type of document D6.4 – V1.0

# VR Human-Robot Interaction Component

| | |
|---|---|
| Dissemination level | PUBLIC |
| Submission Date | 2022-06-30 |
| Work Package | WP6 |
| Task | T6.5 |
| Type | Report |
| Version | 1.0 |
| Author | PaleBlue |
| Approved by | Consortium |

## Executive Summary

This document describes the support for human-robot interactions in the virtual reality simulation environment of Canopies. It describes the general networking environment which connects together the robotic and human clients. Furthermore, the functional possibilities and effects of humans and robots interacting through touch, forces, sound, gestures, and intentions are discussed. The applications used by robotic and human users and the specific features of each of those are discussed. Finally, the synchronization between objects, states, actions, and sound across the network are detailed.

## Table of Content

## Abbreviations and Acronyms

| | |
|---|---|
| RGB camera | Red, green, blue. Also: colour camera |
| RGB-D camera | Red, green, blue and depth camera |
| ROS | Robot Operating System |
| SteamVR | API and devices supporting that API for VR |
| TF | Transform |
| VR | Virtual reality |
| WindowsMR | Windows Mixed Reality: API and devices supporting that API for VR |

This project is funded by the European Union G.A No 101016906

# 1 Introduction

In this document we described the possible human-robot interaction in the simulation environment.

The remainder of the Deliverable is organized as follows. Section 2 gives an overview of how interaction between users like robots and humans is realized. Section 3 describes the various ways humans and robots can interact in the simulation environment. In Section 4 the various client applications intended for the users are described. Finally, Section 5 gives technical information on how the network synchronization is realized.

# 2 General overview

## 2.1 Basic principles

Robots and humans will be able to interact in the simulation environment like in the real world: like talk, listen, push, grab and look. To achieve this, we need to simulate input to the regular sensors: like ears, hands, eyes for humans and Joint States, RGB(-D) cameras and microphones for robots. The sensor inputs for the robots have been discussed in [D4.3]. For the sensory input of human participants, we use VR hardware as is described in [D2.4].

To test the control of the robots adequately, the simulation environment should respond to the actions quickly: when the end-effector of the robot touches a grape bunch, it should start to move directly. If not, the control loop of the robot will not work like in the real environment, and this will decrease the added value of the simulation environment.

To achieve this, all interactions with the environment will be calculated locally on the local computer and only the results of these interactions will be synchronized across the network. This means that when a human touches a robot arm, the touch event will be generated on the local computer and thus will not require network communication. However, when this touching results in a force on the robot arm, this force will then be communicated to the robotic client such that the robot can notice this force.

## 2.2 Architecture

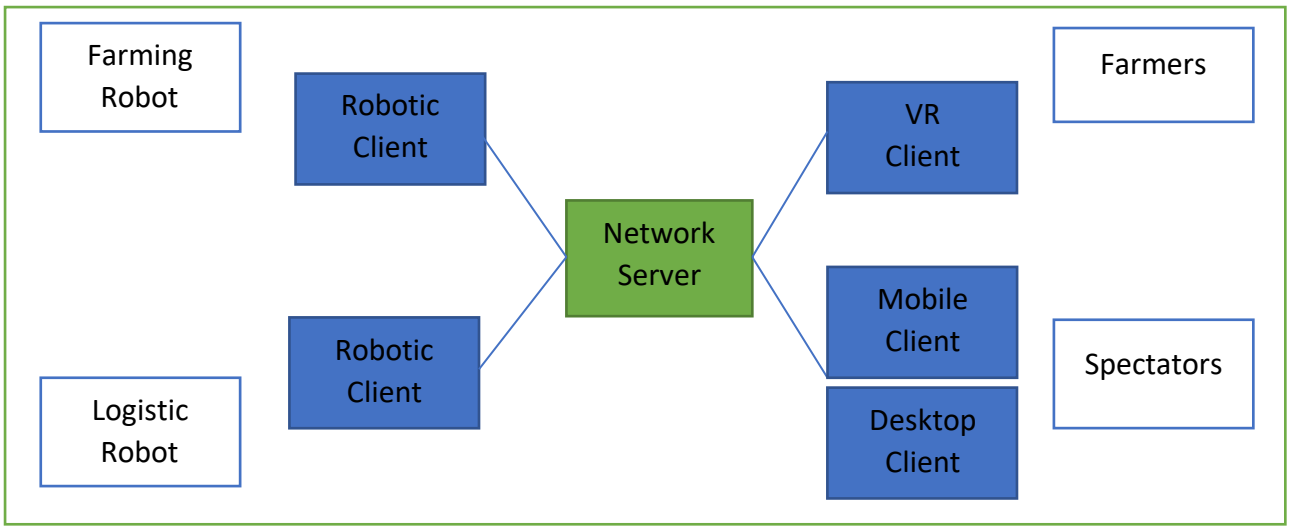
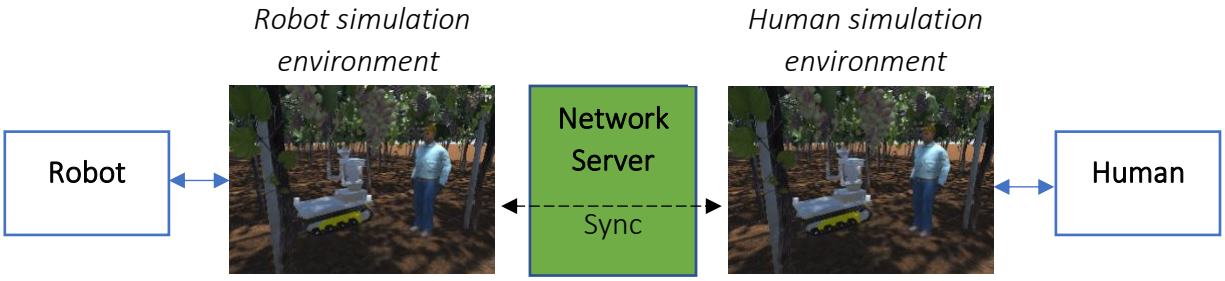



*Figure 1 Level-1 diagram with the users and clients of the system.*

As described in [D2.4], each user, robot, or human, will have a separate client application to interface to the simulation environment (see Figure 1). This client application connects to the other users in the simulation environment to enable interaction and will translate this interaction such that the user can interpret it. For robotic clients this means that it is translated to various ROS interactions, while for human users the interaction is translated into input and output of the VR hardware.

## 2.3 Distributed simulation environment

To enable cooperation between a human and a robot, we need to bring them together such that the actions of one can be observed by the other. This simulation environment is actually distributed across all clients: there is no centralized simulation environment. The network server has only the task to manage the synchronization between the clients. Each client has their own copy of the simulated reality which is synchronized with the copies in the other clients. This setup ensures that each user can interact with the simulation environment directly, with minimal latency. The downside is that different clients may not see the simulation environment in the exact same state as the network synchronization takes time. We consider this as a non-critical limitation.




Another advantage of this approach is that the interaction is more deterministic. The main physics engine of Unity runs single-threaded so non-deterministic effects arising from concurrent processes will not come forward. When running physics concurrently on a distributed network, predictability will become much harder due to the non-determinism caused by the distributed concurrency.

An alternative is to have an authoritative server which does all the calculations and forwards the results to all clients. This ensures that all clients have the same image of the simulation environment, but it increases latencies considerably as every physics interaction requires back and forth communication with the authoritative server. It also puts high calculation requirements on the server which may become a bottleneck when using up to 10 robots at the same time [D2.3]. As a tight control loop for the robotic control with latency as small as possible is required for a representative simulation, we have chosen not to use an authoritative server.

# 3 Interaction

This section describes the functional interactions between robots and humans in the simulation environment.

## 3.1 Physical interaction and compliancy

Humans will be able to physically interact with the robots (or other humans) by grabbing and pushing. When doing this, they will apply forces on the robot's joints which are reported in ROS using the Joint States topic or feedback mechanisms like the Joint Trajectory Controller state. These topics and their behaviour are the same as for the real robot as we try to keep the interface to our simulation the same as the real robot.

This force feedback can then be used to respond to the human's interactions and make the robot act compliant to the human's actions. When this force-feedback is ignored, the robot will resist change and will try to maintain its pose despite external forces.

Additionally, we plan to add an impedance controller for the robot. This enables the possibility to control the compliance of the robot using stiffness and damping gains. This could be done by using the Tiago impedance controller which will be used for the Canopies robotic prototype. For that we will publish dedicated topics for each simulated Series Elastic Element (SEE) at each joint. If this does not prove to be feasible, a direct implementation of the impedance controller in the simulation will be made.


*Figure 2 Human trying to interact with a robot*

Similarly, when the robot collides with parts of the human, it will result in certain forces in the joints which can then be used to correct the movements of the robot. When the movement is not corrected, the robot's movements will 'fight' with the movements of the human and the strongest one will win (which usually is the robot...)

### 3.1.1 Human hand interaction

The hands of the human are controlled using dynamic physics and can therefore exchange forces. Other parts of the human like the legs and head are using kinematic physics and can therefore not be moved using external forces. This means that the mobile base or the dual arm cannot move the legs or head of the human for example. When the mobile base collides with the legs of the human, the mobile base will stop, even though the mobile base is much stronger than the legs of the human.

*Figure 3 Human hand without collision: the controller is located in the hand palm.*

*Figure 4 Human hand while colliding: the controller is not at the location of the hand palm. The yellow arrow indicates the force applied to the robot.*

The force the hands apply to the environment is derived from the difference between the position of the hand controller and the position of the avatar's hand in the simulation space. Normally, when the hands of the avatar do not collide with the environment, this distance is zero: the avatar's hand palm will continuously match the position of the hand controller (see Figure 3). When the hand of avatar collides with the environment, a difference can appear. For example, when the avatar's hand collides with the robot, the avatar's hand will not move into the robot. On the other hand, the real hand controller held by the user is not stopped: the user can actually move his real hand with the controller inside the robot (see Figure 4). The difference in position is translated proportionally to a force which is applied to the object to which it is colliding. The maximum force is set by default to 100 Newton for a difference of 1 meter.

In the case of collisions, the user can notice this from haptic feedback (vibrations) of the controllers held in the hand. Besides that, the user may notice that the position of his or her own, real hand does not match the position of the hand of the avatar. However, it is our experience that the user does not notice this usually. The reason for this is that the visible behaviour of the avatar's hand colliding with the object is normal and expected for humans. As humans use their vision as their primary sense, they are inclined to reject conflicting sensory input from the muscles and joints (the proprioceptive sensory system). That input is telling them that the hand is actually inside the object, which is not possible in real life. Only when users are paying explicit attention to the sensory input from their muscles and joints, they will notice that the separation exists.

Besides the ability to exert forces, users can also experience force feedback. Although the muscles themselves do not experience it, users will still get a sense of force feedback. Our experience with this is that it is caused by two factors:

1. The resulting behaviour of the object from the interaction;
2. Internal muscular 'fatigue'.

When applying a force to an object, the object will respond in a certain way which is mostly related to their mass and weight. Heavy object will have a high inertia and when they are on top of other objects, have a higher resistance from friction. Therefore, heavy objects will be harder to move than light objects. This will be observed by the user visually and it will tell them something about the weight of the object.

As forces are derived from the difference between the avatar's hand and the user's hand, users need to make larger movements to achieve a certain effect for a heavier object. These larger movements are usually not perceived directly but will translate in a kind of effort needed to achieve their goal. The heavier the object, the bigger the required effort. This bigger effort is experienced through the proprioceptive system (muscles and joints), but in a different way than direct weight sensing as it is not a force, but a movement.

All of this is empirically tested in many demos in the past years and from this we concluded this leads to a natural physics interaction with the environment[1].

## 3.2 Voice and sound interaction

Humans using the simulation environment will be able to speak and talk to other users in the simulation. That means that when multiple humans have joined the simulation environment, they will be able to talk with each other as they are used to in the real world.

Robotic users will be able to hear other users through a simulated microphone. This microphone will receive the voices and sounds in the direct environment. On the other hand, the robots will have a simulated speaker which can play waveforms such that other users can hear that.

[1] It would be an interesting topic for scientific study, but no steps in that direction haven been taken yet.

The voice loudness will be dependent on the distance between the user representatives (avatars, simulated robots) so that representatives far away from each other will not interfere with voice communication between representatives close to each other.

## 3.3 Light indications

For notifications, alarms, and warning from the robot to the human it will be possible to combine sounds as described above with indication lights. These lights will light up on the robot in specific patterns to indicate various situations: notifications, warnings and/or alarms. The humans and other robots will be able to perceive these light indications in the natural way.

## 3.4 Gestural interaction

The hands of the avatar can take various poses. Using the button input of the controller it is possible to control the thumb, index finger and other fingers. This enables the human to make 6 different gestures (see Figure 6).

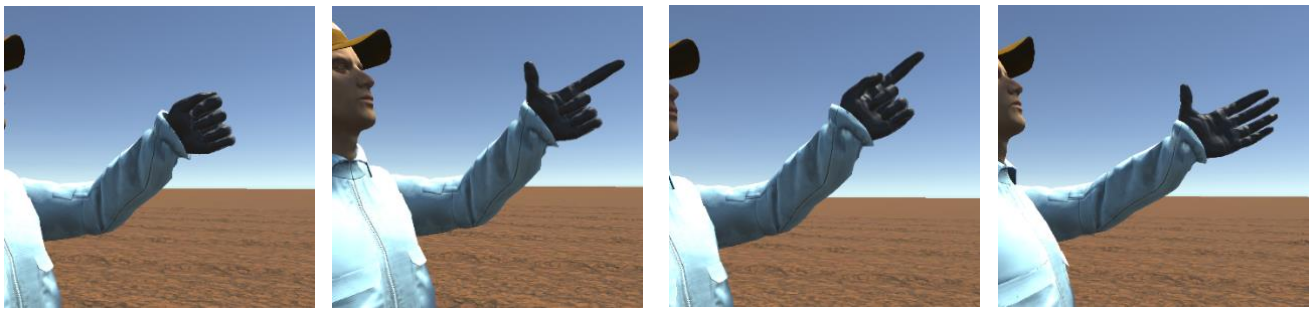


*Figure 5 Various hand poses with controller input*

It is also possible to assign custom designed hand poses to buttons of the controller. This will then replace the poses shown above. However as only three buttons are available for hand poses, the number of poses is limited. Combinations of buttons can be used, but often become confusing quickly. If more hand pose control is needed, we advise to the hand tracking.

Various options exist for optical hand tracking without controllers which can support more variations in the hand pose. However, this has some drawbacks:

1. Hand tracking is less stable than using controllers. The range of motion is limited to the view of the camera and the camera may not see the complete hand because it is covered, for example by the other hand.
2. Using menus to navigate options or indicate intentions is harder with hand tracking. When using controllers, button input can be used to navigate menus. Touching virtual buttons is harder to do reliably.
3. The chosen VR device, Oculus Quest using Oculus Link does not support hand tracking natively. Although the Oculus Quest has good hand tracking, it is only supported when running standalone without a PC. We do not run standalone because of the high requirements for rendering and physics interaction which requires PC power (see also [D2.4]).

4. Hand tracking is not cross-platform. Although we chose to use the Oculus Quest, it is still relatively easy to support other headsets like SteamVR (HTC Vive, Valve Index) or WindowsMR (HP Reverb) headsets. Hand tracking may be supported on other devices, but every device has a different API and a different tracking quality. When we rely on hand tracking, these other devices may not be supportable anymore. Another option is to use an external hand tracking solution like Leap Motion (see Figure 7), but this complicates the setup as this additional device is then required for each VR setup. If hand tracking is required, the last option seems to be the best way to go forward though.


*Figure 6 HTC Vive with Leap Motion hand tracking device attached*

The body and hand plus fingers of the human will be synchronized across the network. This means that the robots should be able to perceive gestures like pointing, touching and body poses using the available sensors, especially the RGB-D cameras.

## 3.5 Intention indication

For training purposes, it may be desirable to add a way to indicate the intention of the human to the ROS environment. As this is meta-information about the movements of the human, the best way to do this is to add a user-interface which can be used to indicate the intention (see Figure 8).


*Figure 7 User interface to indicate the human's intention*

# 4 Networked clients

This section describes the different network clients which represent actors like robots and humans in the networked simulation environment. These network clients are in fact applications which connect to the simulation environment. We identify three clients:

1. Robotic clients: used for logistic and farming robots;
2. Human clients: for humans using VR to join the simulation as a full body avatar;
3. Spectator clients: which provide a view of the simulation on mobile or desktop computers.

## 4.1 Robotic clients

Robotic clients normally represent a single robot. It is possible to have multiple robots in a single client, but this complicates the interface between the simulated robots and the ROS interface as every topic needs to be prefixed with the identity of the robot to which it is connected. For that reason, we prefer to have a single client per robot.

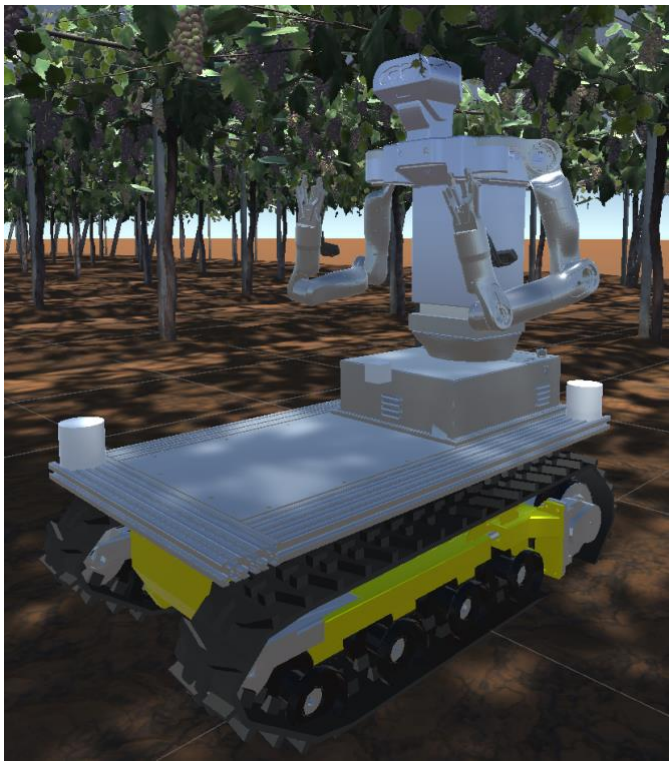


*Figure 8 Simulated Robot*

Each robotic client has a connection to the ROS environment like it is described in [D5.6]. This means that it can be controlled from that ROS environment and the output like that of sensors is sent to that ROS environment.

Apart from the direct robotic input/output, the environment is communicated for testing purposes. This means that a TF tree is published on a separate topic /tf_environment which includes:

1. Pole positions of the support structure;
2. Positions of the grapes on the vines;

3. Positions and skeleton of human avatars present in the environment.

Besides the global objects present in the environment like the support structure and the vines with grapes, other clients will also be present and visible in the scene and will for that be detected by the available sensors on the robot, like Lidars, RGB(-D) camera's, force feedback and microphones. This enables interaction in a similar way as in a real-life setup.

The robotic client can include a desktop view of the robot in the simulation environment, but this view can be disabled to improve the performance of the robotic client simulation.

## 4.2 Human clients

Each human participating in the interaction within the simulated vineyard will use a VR headset to visualize and interact with the simulation. In this way, humans can behave in the simulated environment in a natural way and interact with the robots like would be done as in a real-life situation. The robots will be able to detect these interactions using their simulated sensors as described in the previous section.

The human client software runs of Windows computers and preferably use the Oculus Quest 2 VR hardware (see [D2.4]). It is likely that we will be able to support other VR hardware too (SteamVR for Vive headset support, WindowsMR for HP headset support), but this may change if specific requirements like hand tracking is needed.

*Figure 9 Oculus Quest 2 VR hardware*

## 4.3 Spectator clients

Besides robots and human clients, we have the option to have spectators for the simulation environment. Spectators can only view and hear the simulation but cannot participate and are not visible in the simulation. They will have the possibility to change the view arbitrarily by 'flying' through the simulated environment. If necessary, it is possible to add voice interaction such that the human and, optionally, robot actors will hear the spectators. It is possible to disable the robot hearing

spectators while other humans are still able to hear the spectator. In that case, the robot cannot respond to the voice of the spectator.

# 5 Network synchronization

This section describes the technical network synchronisation between the clients which enables the functional interactions.

## 5.1 Environment

The poses off all non-static objects will be synchronized across the network. The position and orientation of such an object forms the basis for the synchronization. Besides that, it is possible to use velocity and angular velocity to improve the synchronization for fast-moving objects. These velocities are then used to compensate for the communication time of the networking synchronization.

When two users interact with the same object at the same time, the position and orientation of the object may differ between users more than usual (see also Section 2.3). This is because the reaction control loop between the movements of one user and the other user includes the network communication latency which is a considerably longer control loop than when an object is directly controlled without interaction with other users.

Suppose we have two users and one object. User 1 wants to have the object in position 1, user 2 wants to have the object in position 2. Assuming the users have similar strengths, this would result in a tug-of-war with the object somewhere between position 1 and 2 in the real world. Because of the communication times involved in the networked setup, in the simulation the object will oscillate between position 1 and 2. A user will notice and respond to a changed force from the other party only after the communication time as it cannot detect the change in force earlier.

### 5.1.1 Branches

If branches can be synchronized[2], their movement will only be sent to other clients when a human or robots is interacting with it. Other robots or humans will only see these movements when they are close to that same branch.

### 5.1.2 Grape bunches

Grape bunches close to robots or humans will be synchronized (bunches farther away will be static and will not be synchronized, see [D2.4]). When a human moves the bunch still on a branch, a robot on a different client will see these movements when they are close to that bunch. When the human is close to a certain bunch, but the robot is not near that bunch, the bunch will not be synchronized as it is not relevant for the interaction between the human and the robot.

[2] No decision has been made yet on the most effective way to implement this.

*Figure 10 Human avatar interacting with a grape bunch*

When a bunch is detached from the branch, it is synchronized as long as it moves within the local space. When it lies on the ground, it is not synchronized until a human or robot picks it up. Similarity, grape bunches are no longer synchronized when they have been placed in a box.

### 5.1.3 Boxes

The position of the boxes (with their contents) is synchronized when they are moving in their local space. When a box has a fixed place on a robot and this robot moves, the box position is not synchronized as it is not moving relative to the robot.

## 5.2 Robotic state

The state of the robots across all clients should be the same as much as possible. Since network communication takes time, differences will occur.

Each robot will be controlled via ROS on one client, the local client. We call this the local robot. All other clients, the remote clients, will see a copy of the robot which should follow the movements of the local robot as closely as possible. These copies of the robot visible in other clients is called the remote robots.

The local robot can be controlled in three ways: using a position, velocity, or effort interface. While it is basically easy to copy positions from the local robot to the remote robots, it gets more complicated when using velocities or effort to control the robot because the movements using these interfaces are relative to the previous pose.

On each client, the previous pose can be different because they may have had an interaction with a local human or robot which has not yet been communicated to the other clients. The time step is

different for each remote robot because the network transmission time needs to be considered. This can be compensated, but at the cost of increased latency.

To have the remote robot state match the local robot state as closely as possible with a low latency we chose to use positional and velocity synchronization. The actual position and velocity of the joints of the local robot are sent to all remote robots. In the remote robots these values are used to drive the joints using the standard spring-based joint implementation (see Section 3.3.3 in [D5.6]) which will move the joints to this requested position and velocity.

This implementation still enables dynamic physics interaction (see Section 5.4 - Physics dynamic interaction)

## 5.3 Human state

The humans are controlled using VR hardware consisting of a head-mounted devices (HMD) and hand controllers. These three inputs determine the full pose of the human's avatar. Because of this, we basically only need to communicate the root position plus the three inputs to the other clients to reproduce this same pose on the remote clients.

*Figure 11 Multiple humans in the simulation environment*

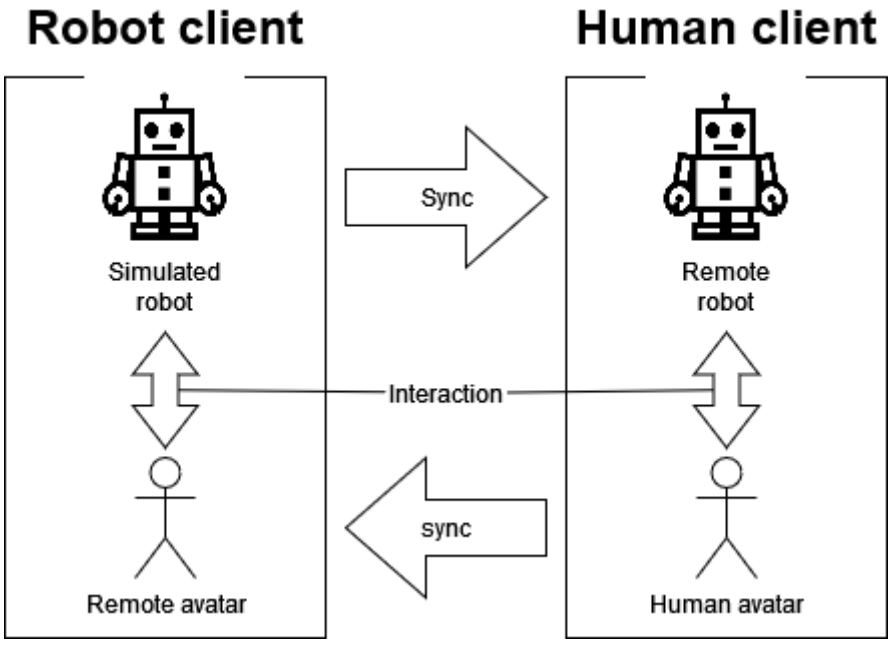
However, as the procedural animation of the legs may result in different poses because of slight differences in the simulation state on each client, it can happen that the pose of the avatar is different between clients. For that reason, we also synchronize the positions of the feet and hips.

## 5.4 Physics dynamic interaction

All dynamics physics interactions are computed locally. This limits the response time of the physics updates to the update rate of the physics engine which is set to 100Hz.

When a human physically interacts with a robot, for example by pushing it, it is therefore actually pushing against the copy of the robot present on the human client.

On the other hand, at the robot's client, the copy of the human will be pushing against the local robot.




On the human client, the human only sees the interaction between the remote copy of the robot. On the robotic client, the robot only perceives the interaction with the remote copy of the human.

This means that there may be slight differences between the perceived actions as the pose synchronization is not instantaneous.

## 5.5 Voice Communication

For voice communication we use the Photon Voice[3] package which synchronizes spoken interaction between clients. The nature of this communication is that the latency for this is higher than for that other communication (up to 500 milliseconds) and may therefore be not completely in sync with the other network synchronization. In non-critical situations this is normally not noticeable, but audio-based alarms may not be as effective as in real-life because of this increased delay. These aspects will be further investigated using a thorough experimental evaluation to better understand if some revision may be required.

For humans, the voice is captured using the microphone embedded in the VR headset while the headset has speakers to produce sounds from other humans, robots, and environment.

[3] https://www.photonengine.com/en-US/Voice

For humans, sounds emitted by the robots or other humans will be heard in the headset or headphones and spoken voice will be digitized and send over to all other clients.

Robotic clients will forward the audio between the simulation and the ROS environment. We plan to use the standard audio_common[4] transport for that. This is currently under investigation as it is not yet clear whether the TCP connector which connects ROS with Unity is able to transport the audio stream in a stable way.

[4] http://wiki.ros.org/audio_common

# 6 References

| Id | Title | Version | Date |
|---|---|---|---|
| [D2.1] | Requirements, Specifications and Benchmarks | 1.0 | 2021-04-30 |
| [D2.2] | Specifications and KPIs for the two CANOPIES robot prototypes | 1.0 | 2021-06-30 |
| [D2.3] | Dimensioning of the Real-World Scenario for Final Demo | 1.0 | 2021-10-29 |
| [D2.4] | VR Farming Environment Specification | 1.0 | 2021-10-29 |
| [D3.4] | Simulated Farming Environment and Basic Robotic Components | 1.0 | 2021-08-31 |
| [D4.3] | VR Agronomic-Oriented Perception Component | 1.0 | 2021-12-30 |
| [D5.6] | VR Mobile-Robot Component | 1.0 | 2021-03-31 |