Ref. Ares(2022)7548777 - 31/10/2022

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 D7.3 – V1.0

VR Multi-Robot Component

Dissemination level	PUBLIC
Submission Date	2022-10-31
Work Package	WP7
Task	T7:5
Type	Report
Version	1.0
Author	PaleBlue
Approved by	Consortium

DISCLAIMER:
The sole responsibility for the content of this deliverable rests with the authors. It does not necessarily reflect the opinion of the European Union. Neither the Directorate-General for Communications Networks, Content and Technology, Artificial Intelligence and Digital Industry nor the European Commission are responsible for any use that may be made of the information contained therein.

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

Executive Summary

This document describes multi-robot interaction in the virtual environment, and related networking mechanisms. Real world setup, and difference with simulation setup are covered, along with box exchange mechanism, and other relevant entities such as grapes and branches. Furthermore, the latencies occurring in different network setups are measured, for single computer and multi computer configurations.

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

Table of Content

1 Introduction 5
2 Multi Robot setup 5
2.1 Real world setup 5
2.2 Simulation setup 6
2.2.1 Robot pose synchronization 8
2.2.2 Force exchange between robots 9
2.2.3 Wireless communication between robots 10
2.3 ROS architecture 11
2.4 Harvesting and pruning 12
2.5 Box Exchange Mechanism 12
2.5.1 AprilTag 12
2.5.2 Lidar Reflectors 13
2.5.3 The box exchange mechanism - farming robot side 14
2.5.4 The box exchange mechanism– logistic robot side 14
2.5.5 Exchanging the boxes 15
3 Measurements 16
3.1 Single computer, multi-robot setup 16
3.1.1 Laptop setup 16
3.1.2 Desktop setup 17
3.2 Multi computer, multi-robot setup 17
3.2.1 Networking bandwidth 17
3.2.2 Network latency 18
4 References 20

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

Abbreviations and Acronyms

CPU	Central Processing Unit
GPU	Graphics Processing Unit
PNG	Portable Network Graphics, a graphic file format
RGB camera	Red, green, blue camera. Also: colour camera
RGB-D camera	Red, green, blue and depth camera
ROS	Robot Operating System
URDF	Unified Robot Description Format
VR	Virtual Reality

1 Introduction

This is a report describing VR Multi-Robot interaction and related networking mechanisms. Briefly, this document is organized as follows: Chapter 2 describes the Multi Robot setup, including simulation setup, box exchange mechanism, and other relevant entities such as grapes and branches, while Chapter 3 goes into measuring latencies occurring in different network setups for single computer and multi computer configurations.

2 Multi Robot setup

For a multi-robot setup, we try to follow a real multi-robot setup as closely as possible. As a reference, we will give a simplified description of a real multi-robot setup here first.

2.1 Real world setup

Figure 1 Multiple robots in the real world

Primarily, each real robot will have access to its own input (like sensors) and output (like actuators) only. These are read and controlled though a ROS environment running on that robot. The robot can interact with the environment containing grapes, vines, vineyard support structure, humans, and other robots using only these inputs and outputs.

Additionally, the robots may be able to communicate with each other through a wireless communication channel.

2.2 Simulation setup

The simulation setup follows the same line as the real-world setup. That means that in the ROS environment of a single robot normally only the input and output for that specific robot is available with the addition of a communication option through simulated wireless communication. Each simulation environment corresponds to one simulated robot. Multiple simulation environments can run in parallel on the same machine but can also run on different computers.

For each robot we have a separate local simulation environment which simulates everything which is relevant for that robot. We call this robot the *local robot*. This enables us to limit the simulation of the sensors, topics, joints, actuators, and dynamic physics to only those directly related to that robot. It should be noticed that within the local simulation environment, it has no added value to simulate the functionality of devices on another robot, like a lidar, as robots cannot perceive the working of these devices on other robots other that their physical appearance[1]. These other robots in the local simulation environment are called the *remote robots* (see Figure 2).

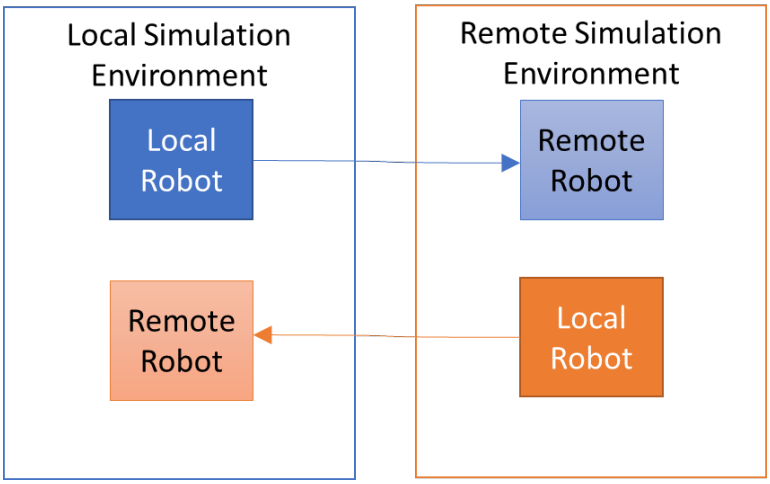


Figure 2 Projecting local robots on remote simulation environments

The synchronization of the simulation environment is managed through a central server which acts as a message broker (see Figure 3). All communication goes through this central server, as most communication is about broadcasting the actions of the local robot to the other simulation environment. When a new simulation environment is launched and added to the setup, it will contact the central server to announce its presence with a new local robot. The central server will forward the information about the new robot to all other existing simulation environments. These existing simulations will then create a new remote robot based on the information received from the new simulation environment. After this, the new robot is synchronized like all other existing robots. When

[1] We ignore here the possible interference between different infra-red emitters and receivers like lidars and RGB-D cameras. It should be noticed that the sensors that are being used were selected for their low sensitivity for external infra-red light as they are used in an outdoor setup with high infrared light intensity from the sun. We expect that simulating the interferences of IR sources and sensors does not have much additional value for this reason and it does not justify the increasing complexity that would be added to the simulation environment.

a simulation environment is stopped, the central server will notify the left-over simulation environments such that they can remove the applicable remote robots.

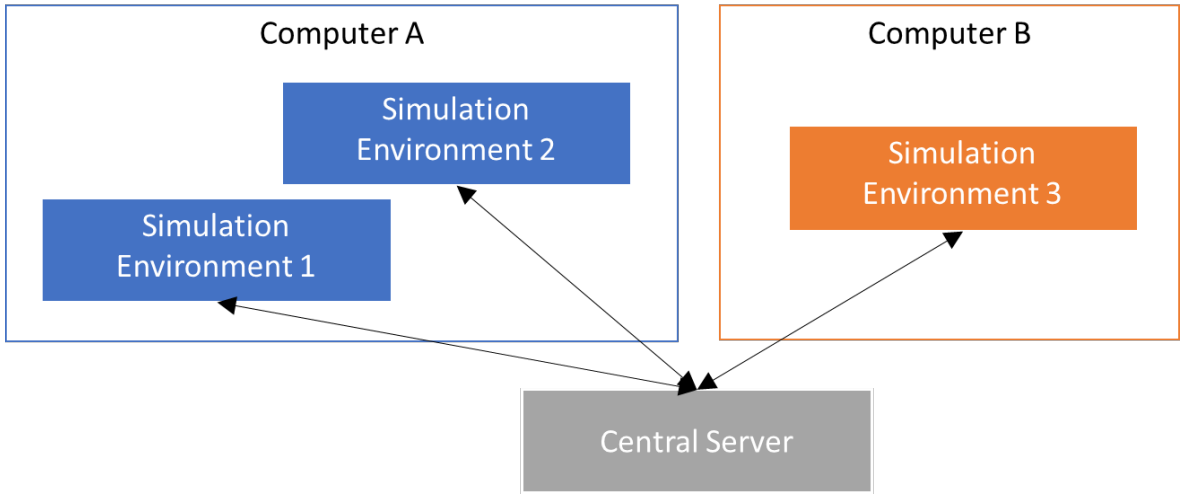


Figure 3 Architecture of the networked simulation

This approach enables us to scale the simulation to multiple robots without a high impact on performance needs as each robot could be simulated on a separate computer. Next to that, the same architecture can be used to include multiple humans in the simulation in a similar manner. The inclusion of humans will be described in the future deliverable D8.4.

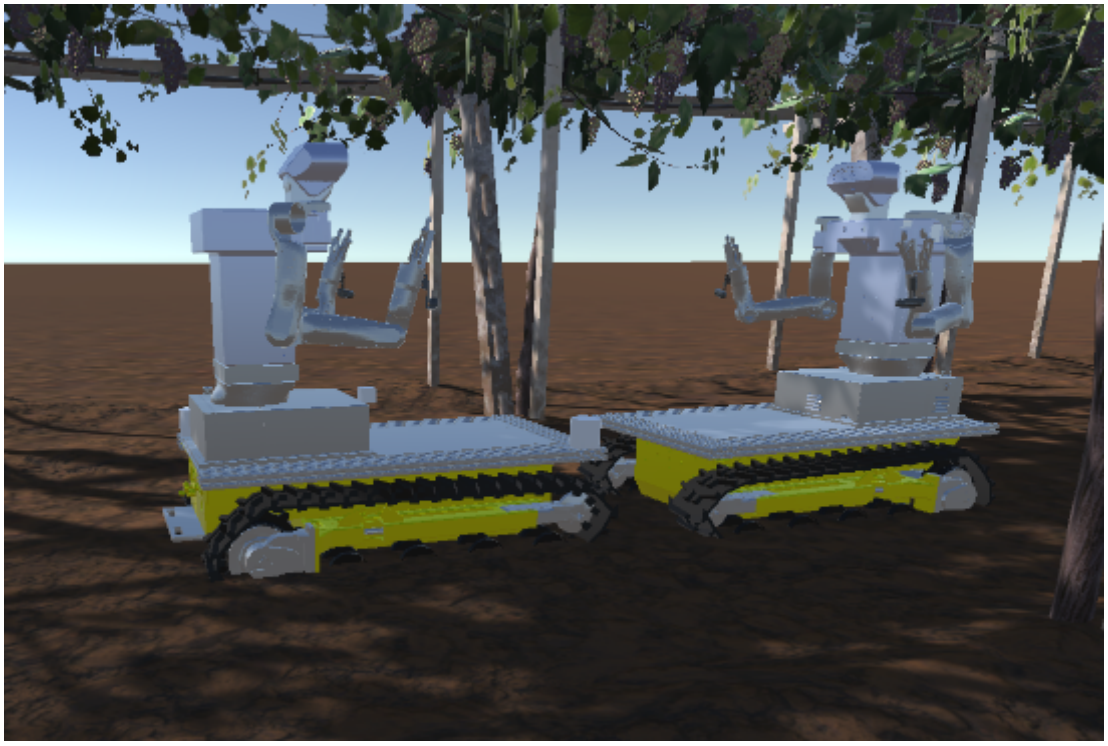


Figure 4 Multiple robots in the simulated world

As this physical appearance of these robots can be perceived with the sensors, we do need to ensure that the perceived pose of these other robots is corresponding to the actual pose. This is like how it

works in the real world: robots can only perceive each other through their sensors or through explicit communication (see Section 2.2.3). The actual pose is known for each robot in their local simulation environment as it is part of the simulation. We then need to ensure that this pose is synchronized between all simulation environments (see Figure 5). This is described in Section 2.2.1 - Robot pose synchronization below.

Because of the action of a robot, the environment may change: a bunch of grapes is moved or harvested, a branch is moved or pruned or a box with grapes is put on a logistic robot. As another robot may cause these changes, these changes will need to be synchronized between the simulation environments too. This has already been described in [D6.4].

The robots may interact with each other physically. This is obvious in the box exchange mechanism where a box is moved from one robot to another. Apart from that, the mobile base or dual arms of the robots may also (unintentionally) collide with each other. These physical interactions will result in dynamic forces being exchanged between the robots. The working of these force exchange is described in Section 2.2.2 - Force exchange between robots.

Finally, the robots can communication with each other wirelessly. The way this is set up is described in Section 2.2.3 - Wireless communication between robots.

2.2.1 Robot pose synchronization

In each local simulation environment, the pose of the robot is continuously calculated based on the input from the ROS environment and the dynamic physics of the robot itself and the environment. This calculation is relatively expensive but is only needed for the local robot.

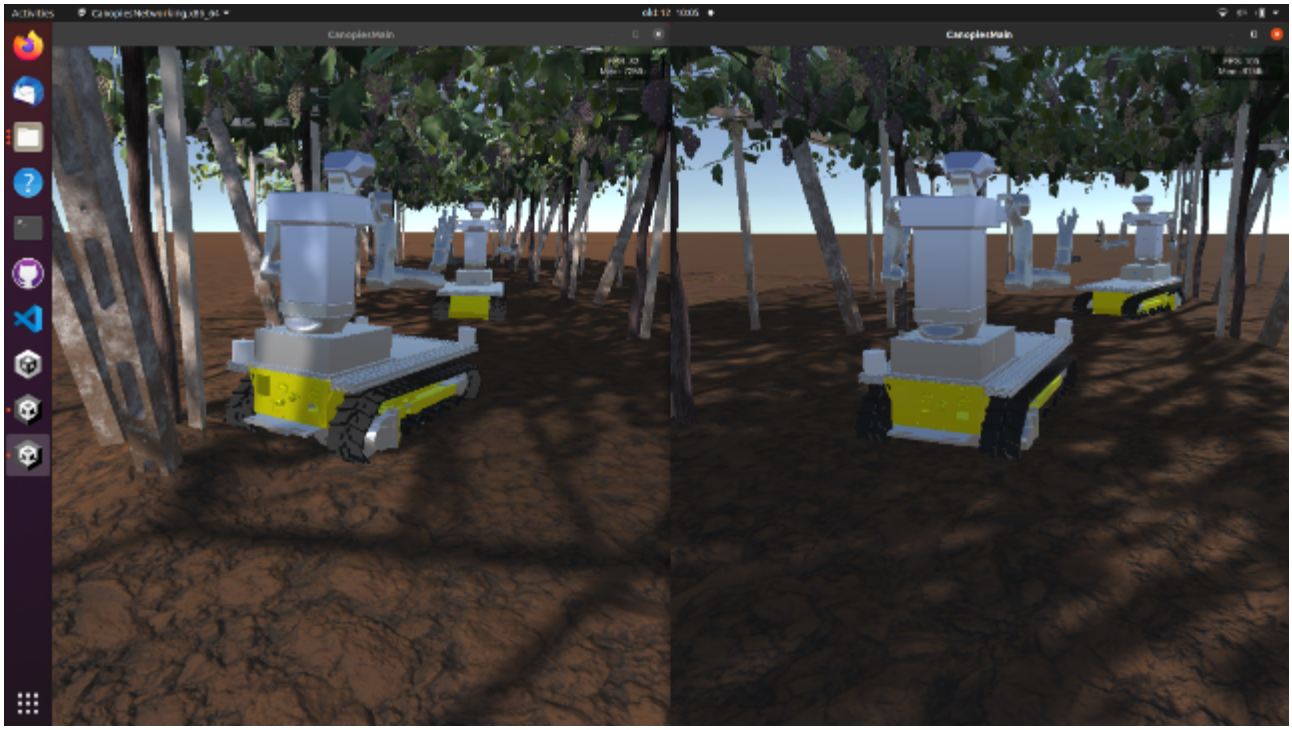


Figure 5 Two simulators with networked pose synchronization

Although the robots themselves can only detect other robots through their sensors or direct communication, the simulation environment still needs the pose of the other robots to be synchronized or the sensor simulations will not see these other robots. For that reason, we need to synchronize the pose of the robots between the simulation environments (see Figure 6).

For the remote robots in the local simulation environment, we primarily need to ensure that the pose of these remote robots is the same as the pose resulting from the calculations in their local simulation environment (see Figure 2). This pose is regularly broadcasted through the network to all other simulation environments.

For the mobile base of the robot, the position and orientation are synchronized together with the linear and angular velocity. The velocities are used to compensate the network transmission time by extrapolating the position and orientation using the received velocity.

For each joint in the dual-arm robot, the position and velocity are synchronized. Like for the mobile base, the velocity is used for compensation of network transmission time.

2.2.2 Force exchange between robots

When a local robot collides with a remote robot, there should be a force exchange between the two robots. If we used the compensated position from the previous section directly, this would not happen, meaning that the remote robots would not respond to forces applied to them and their position is solely determined by the received pose: it is not compliant. Vice versa, the local robot would experience unrealistic forces from the remote robot as it is not compliant.

The solution is to use a spring model to drive the joints of the dual arm. We use the same model as is used to implement the positional interface [D5.6] provided by the Unity Engine:

*Effort= stiffness * (jointPosition - targetPosition) - damping * (jointVelocity - targetVelocity)*[2]

This means that the remote robot will try to maintain its position while still being compliant to the other robot.

The movements of the mobile base work in a different way. The mobile bases are fitted with bumper which detects when the mobile base collides with an object. When this happens, a ROS message is sent which can be used to adjust the movements of this base. This mechanism supports alignment of mobile bases for exchanging boxes.

2.2.3 Wireless communication between robots

Besides communicating physically (by physical interaction), the robots will also be able to communicate electronically with each other. For this, we provide communication channels through ROS. Messages published on certain topics will be forwarded to one or more other robots.

Figure 6 Wi-Fi access point in the field

It should be noticed that in the real-world this communication takes place through a Wi-Fi network in the field (see Figure 7). This communication method has limited quality. For example, objects between and around the sender and receiver and the distance between them have an important effect on the bandwidth, latencies, and stability of the communication channel.

To simulate these effects, a network impairment emulator can be used. Hardware emulator units are available, like the Aukua Network Impairment Emulator[3], but we instead use a simpler approach with a software network impairment tool. Briefly, this allows simulating the needed effects of wireless communication within a controlled environment. It enables control of many aspects of the

[2] https://docs.unity3d.com/ScriptReference/ArticulationDrive.html
[3] https://www.aukua.com/products/network-emulator.html

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

communication, like connection loss, latencies and bandwidth limiting. Besides that, information from the simulation, like the distance between robots can be used to drive the parameters.

2.3 ROS architecture

Each simulated robot will publish and subscribe to resources like topics, parameters, and services. When we simulate multiple robots, we should avoid collisions between the names of these resources. For that reason, each simulation environment needs to use a different namespace. Using namespaces is a well-known approach in ROS and we can naturally and simply inherit it. This namespace can be set in the ROS launch file.

Figure 7 Two simulations using the same ROS environment

It is possible to run multiple simulations on the same computer. These simulation environments can share the same ROS core or each can have a separate ROS core, implementing a ROS multi-core approach. It is even possible to run the ROS core on a different computer than the simulation, providing a maximum of flexibility to distribute workload across available computing power. When sharing a ROS core, the namespaces will ensure that the communication is separated between the simulations.

For the simulations to work, the ROS environments do not necessarily need to be connected. Notably, all communication goes through the simulation environment, routed through the central server. However, it can be that communication between ROS environments also happens in the real-world setup, for example using a multi-master setup. As a matter of fact, this could be replicated in the virtual environment if deemed useful, but it is not technically required for the simulation to run.

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

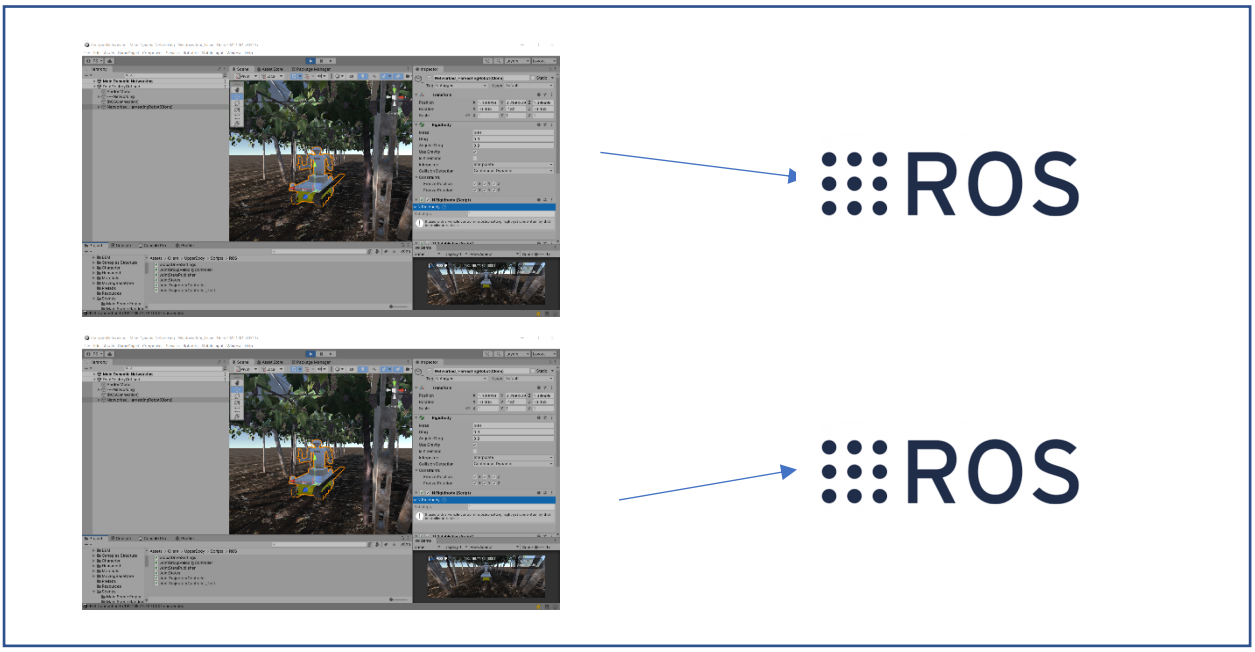


Figure 8 Two simulations using different ROS environments

2.4 Harvesting and pruning

The networking synchronization when harvesting grape bunches or pruning branches has been described in [D6.4], sections 5.1.1 and 5.1.2, respectively. In short, their position and orientation are only synchronized when a robot (or human) is interacting with it or when they are moving. As soon as they reach a stable position, synchronization is stopped.

2.5 Box Exchange Mechanism

Although the final design of the box exchange mechanism is not available at the time of writing this document, we will describe the current view on this simulation here.

2.5.1 AprilTag

On each robot, an AprilTag fiducial marker can be attached to help aligning the two robots (Figure 10, [D7.1]).

The position of this AprilTag on the robot can be adjusted using an updated URDF at runtime (see [D5.6]). It is also possible to change the specific AprilTag on the robot by providing an alternative PNG image. This AprilTag image will then be visible to other robots using their RGB(-D) cameras.

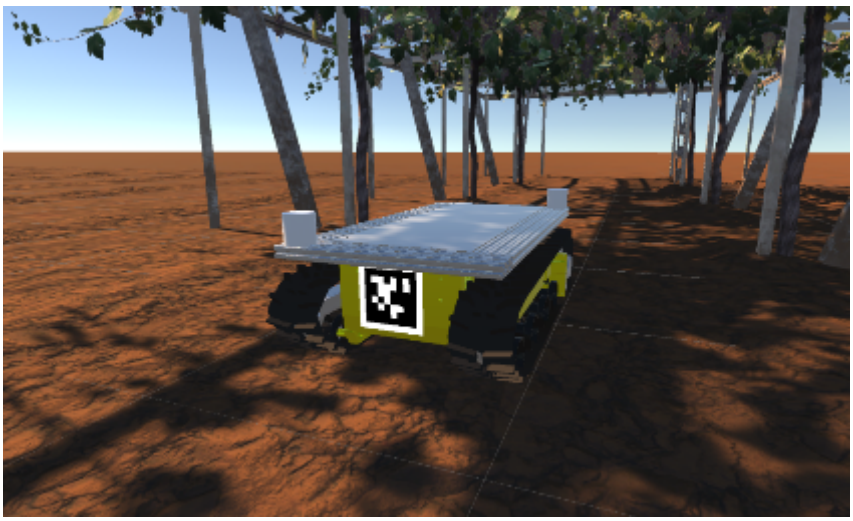


Figure 9 Logistic robot with AprilTag attached

2.5.2 Lidar Reflectors

Additionally, the robots will be fitted with reflectors (see [D7.1]) to help alignment using lidar data. In the simulation, these reflectors will result in very high intensity values in the data coming from the simulated lidars (see [D3.4]), just like they do for the real lidars. This makes it possible to detect the distances to the other robot reliably. The position of these reflectors can be updated using the URDF to support experiments finding optimal placement of these reflectors.

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

Figure 10 Box exchange mechanism on the farming robot. The dual arm is left out here.

2.5.3 The box exchange mechanism - farming robot side

The mechanism to move boxes on the farming robot does not have a lot of moving parts (Figure 11). This makes the simulation relatively easy. The boxes can be moved forward and backward on a conveyor belt. While this is active, the boxes will be dynamic physics objects which means that they will collide with objects and fall to the ground when there is not enough support. This will make the exchanging of boxes more realistic.

However, for the purpose of development and testing, it will also be possible to simulate the box exchange in a kinematic way, without using dynamic forces. In this option, the boxes will not collide or drop to the ground which makes the exchange much easier. The movement between the two robots will then be very similar to an animation. The animation will only play completely when the two conveyor belts from the farming and logistic robots are aligned well enough: i.e., within a certain distance and difference in angle.

2.5.4 The box exchange mechanism– logistic robot side

The logistic robot will exchange boxes with the farming robot with a conveyor belt mechanism in the same way as is described in the previous section.

The logistic robot will have two stacks of boxes: empty boxes and full boxes (Figure 12). Empty boxes will be lowered from the stack when a new empty box is needed and then transported to the front and the farming robot using conveyor belts.

When a new full box has been received from the farming robot, the box is lifted upwards. These movements will be completely animated, as we don't expect these movements need specific testing in the simulation environment.

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

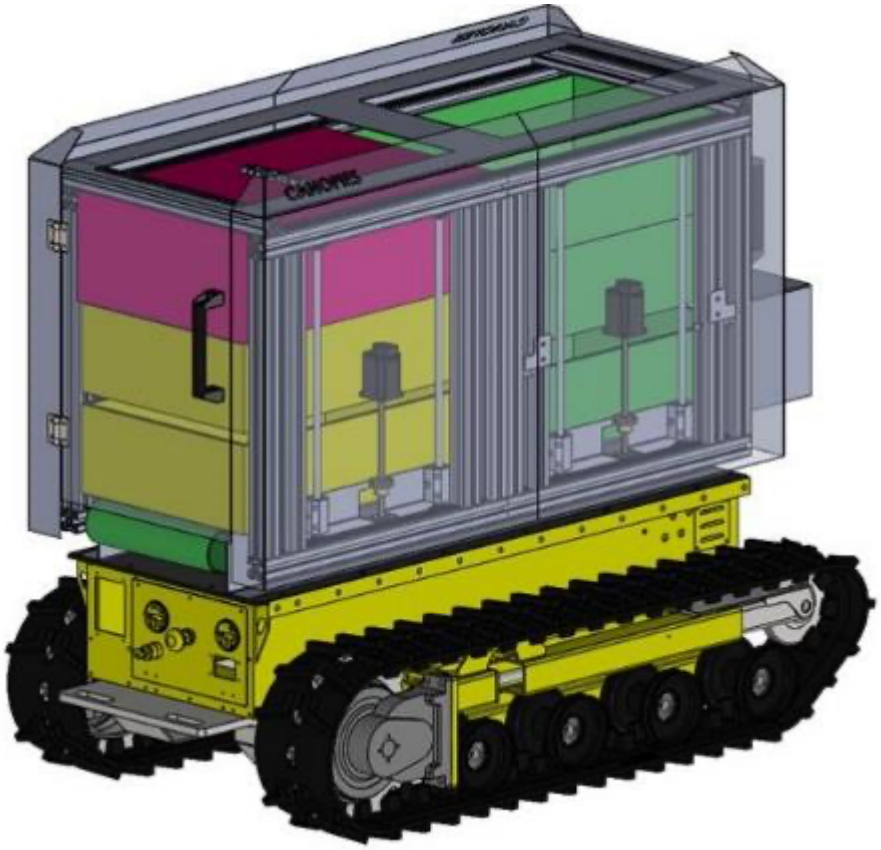


Figure 11 Box exchange mechanism on the logistic robot

Similarly, lowering a new empty box from the empty box stack and moving it to the front is also animated. Only the movement between the logistic robot and the farming robot can be executed using dynamic physics as is described in the previous section.

Should the design change over time due to the needs of the project, the simulation environment will be updated accordingly.

2.5.5 Exchanging the boxes

When exchanging a box between two robots in different simulation environments on the network, one should ensure that only simulation environment controls the properties of that box. The simulation environment controlling these properties is called the owner. When the box is handed over from one robot to another, it moves from the origin robot to the destination robot. At the start of the handover, the simulation environment of the origin robot is the owner of the box. This simulation environment stays the owner until the box has been moved to the destination robot for more than 50%.

This is implemented using the TriggerCollider mechanism in Unity. As soon as the box enters this collider on the destination robot, the ownership will be transferred to the simulation environment of the destination robot. This makes the movement of the box only dependent on the destination, making it possible for the origin robot to move away, disconnect from the simulation of be teleported elsewhere without any consequences for the box transfer.

3 Measurements

3.1 Single computer, multi-robot setup

With the networked multi-robot setup, it is possible to run multiple simulation environments on one machine. We did some initial testing to see the system load for running multiple instances at the same time.

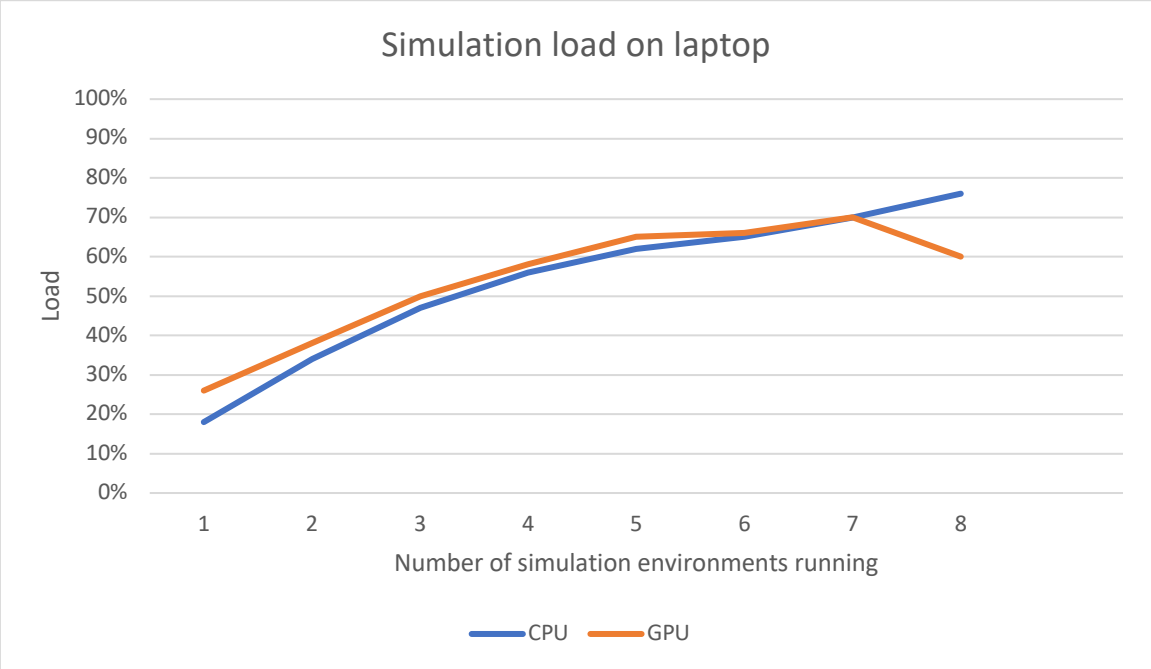
All tests were executed with the main camera off, so no rendering to the screen was done. This lowers the load on the GPU. Enabling the main camera would make the framerate drop significantly, especially when running more than 2 simulation environments on the same computer. If a view on the situation in the simulation is needed, it is best to setup a separate spectator environment to the simulation on a separate machine.

Next to that, no ROS environment is used during these tests. Because of that we get a clearer picture about the system requirements for the simulation itself.

3.1.1 Laptop setup

We tested multiple simulations running on a laptop with the following specs:

- AMD Ryzen 7 5800H 8-core
- Nvidia GeForce RTX3060
- 16 GB memory



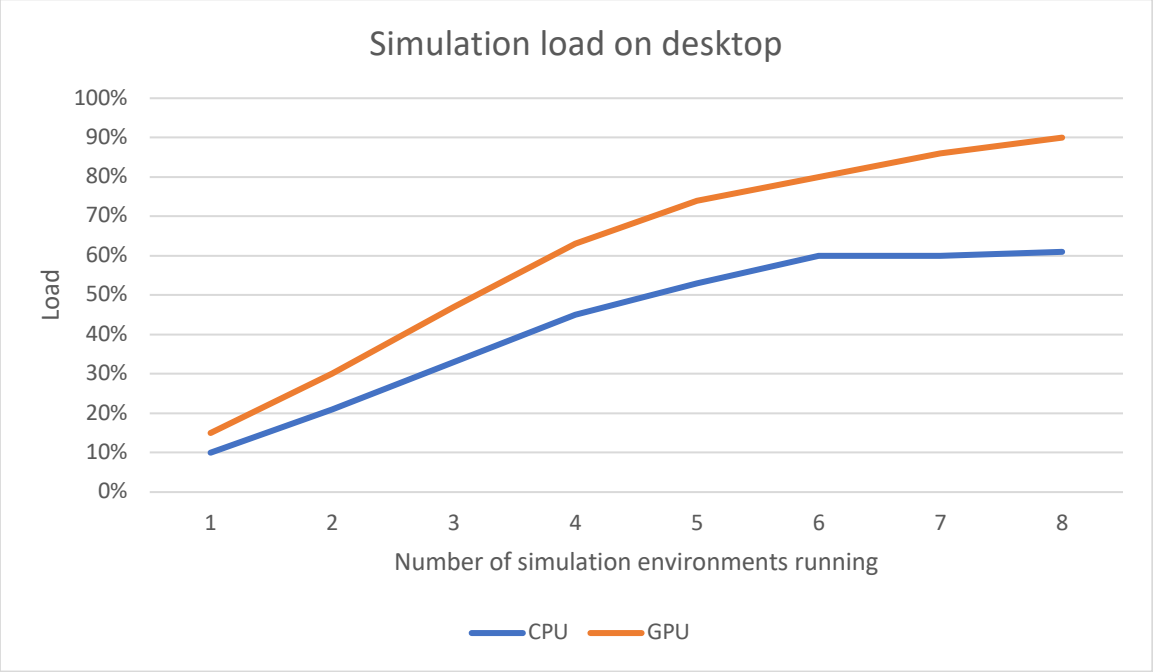
On the laptop setup, we see that the performance consumption goes up until about 4 instances after which the load stays around 60-75%. This is an indication of a non-optimal implementation and makes it clear that there is room for improvement.

The non-optimal performance is probably because there is a bottleneck which makes the CPU wait for data. We already found that readback from the GPU makes the CPU wait for data, this could be

fixed by using an asynchronous readback, but this improvement was not yet realized at the time of writing.

3.1.2 Desktop setup

- AMD Ryzen 9 3900X 12-core
- AMD Radeon RX 5700XT
- 64 GB memory



In the desktop setup we see the CPU load levelling out at 60%, caused by the same limitations as we found for the laptop. Based on these results, we expect that running 8 simulations simultaneously on computer like this will be possible.

3.2 Multi computer, multi-robot setup

It is possible to run the robot simulations on different computers in order to distribute the load. In the examples above, the laptop could simulate 4 robots and the desktop 6, which totals to 10 robots. In general, adding computers to the simulation makes it possible to simulate more robots. This is only limited by the networking communication.

3.2.1 Networking bandwidth

While testing a setup of 8 robots, the total bandwidth consumed for the pose synchronization was about 1Kbit/s while the robots where not moving. Although it was not practically possible to see the bandwidth used when all robots where moving all joints, we can make a calculation for that:

Mobile base position	Vector = 3 floats	12 bytes
Mobile base orientation	Quaternion = 4 floats	16 bytes
Mobile base velocity	Vector = 3 floats	12 bytes
Mobile base angular velocity	Vector = 3 floats	12 bytes
		52 bytes

Joint position	1 float	4 bytes
Joint velocity	1 float	4 bytes
		8 bytes

As we have 18 joints (2 torso, 2 head, 2 x 7 arm joints), the total message size of a robot pose is 52 + 8 bytes = 60 bytes.

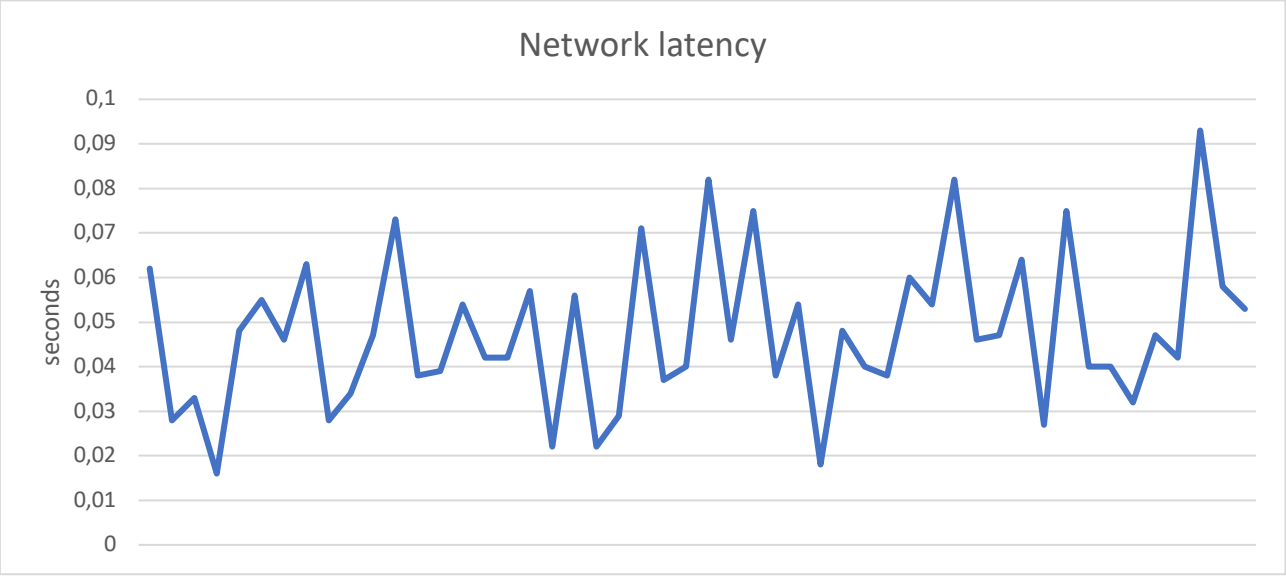
With the default update rate of 25 updates per second and 8 robots in the scene, 60 x 25 x 8 = 12,000 bytes per second in total network bandwidth.

The figure above does not include any networking bandwidth needed for the environment, like grapes and bunches and dedicated communication as described in Section 2.2.3. But it does indicate that the networking bandwidth required for the synchronization of simulation environments is very small. This means that it is easy to scale the whole simulation up to use multiple robots and that it is unlikely that the central server becomes a bottleneck when scaling up the simulation to 10 robots as its required in [D2.3].

3.2.2 Network latency

All networking communication goes through a central server (see Section 2.2) which distributes these between the simulation environments. This central server resides on the internet and is provided by Photon[4]. Because of this, the latency of messages can be significant.

We did a measurement of this latency using a computer connected to the internet with a 150Mbps glass fibre connection with a measured ping of 5ms.



The graph above shows a segment of this measurement which shows that the average latency is around 50ms with a minimum of 15ms and peaks going up to 100ms.

[4] It is also possible to run a private server in the local network, but this has not been investigated yet.

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

Of course, the tested setup is quite ideal, and the latency is dependent on the available network and internet connection but is gives an indication of the latencies involved with the synchronization of the state.

The effect of the latencies is reduced by extrapolating the positions of the objects using the measured lag and the velocity of the object as is described in Section 2.2.1.

It should be noticed that the effect of these latencies could be noticed when robots interact directly with each other, because these latencies are part of the feedback loop. However, based on our experience, we do not expect any issues here. Only when two robots (or humans) directly manipulate the same object, both using forces by holding the same object, the effect becomes noticeable as the object can start shaking. This is the result of both control loops trying the control the same object, while the object's movements are subject to the latency of the network. We do not expect this situation happening with the robotics simulation: we have not identified situations where multiple robots handle the same bunch, box, or branch at the same time. In the case of the box exchange mechanism, we ensured that only one robot controls the box' position (see Section 2.5.5).

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

4 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	2022-03-31
[D6.4]	VR Human-Robot Interaction Component	1.0	2022-06-30
[D7.1]	Coordinated Manoeuvring for Box-Exchange Operation	1.0	2022-08-31