WRS Partner Robot Challenge (Virtual Space) 2018

WRS Partner Robot Challenge (Virtual Space) is the World's first competition played under the cyber-physical environment.

1 Equipment to Prepare

1.1 Equipment

The basic configuration for the WRS Partner Robot Challenge requires a Windows computer to run SIGVerse as well as an Ubuntu computer to run ROS nodes created by the participants. These two computers will be connected to the HUB.

See http://www.sigverse.org for information about SIGVerse.

1.1.1 Windows PC

The competition will prepare the Windows computer to run the programs for the competitive tasks.

No	Item	Description
1	OS	Windows 10
2	Unity	Unity 2017.3
3	IP Address	192.168.0.1
4	Note	The system requirements need to support Unity. The system requirements also need to support Oculus Rift CV1 + Oculus Touch.

1.1.2 Ubuntu PC

The participants need to prepare the Ubuntu computer. Participants need to install the Robot Operating System (ROS) to run the ROS nodes that they create.

(Contact the competition officials if you cannot prepare the Ubuntu computer)

No	Item	Description
1	OS	Ubuntu16.04
2	ROS	ROS Kinetic
3	IP Address	192.168.0.2
4	Note	Wireless communication during competition is prohibited.

1.1.3 Switching Hub

Participants shall use 1000BASE-T with a connection speed of 1 Gbps.

1.1.4 Other Equipment

Participants will also use an Oculus Rift CV1 + Oculus Touch depending on the competitive challenge.

1.2 Software

Participants should download the Unity project for the programs to use in the competitive challenge and the sample ROS nodes for operational verification from the GitHub provided by the competition.

See the Git README for information about operating each software program.

2. Overview of the Handyman

2.1. System Configuration

The system configuration for this competitive challenge is outlined below.

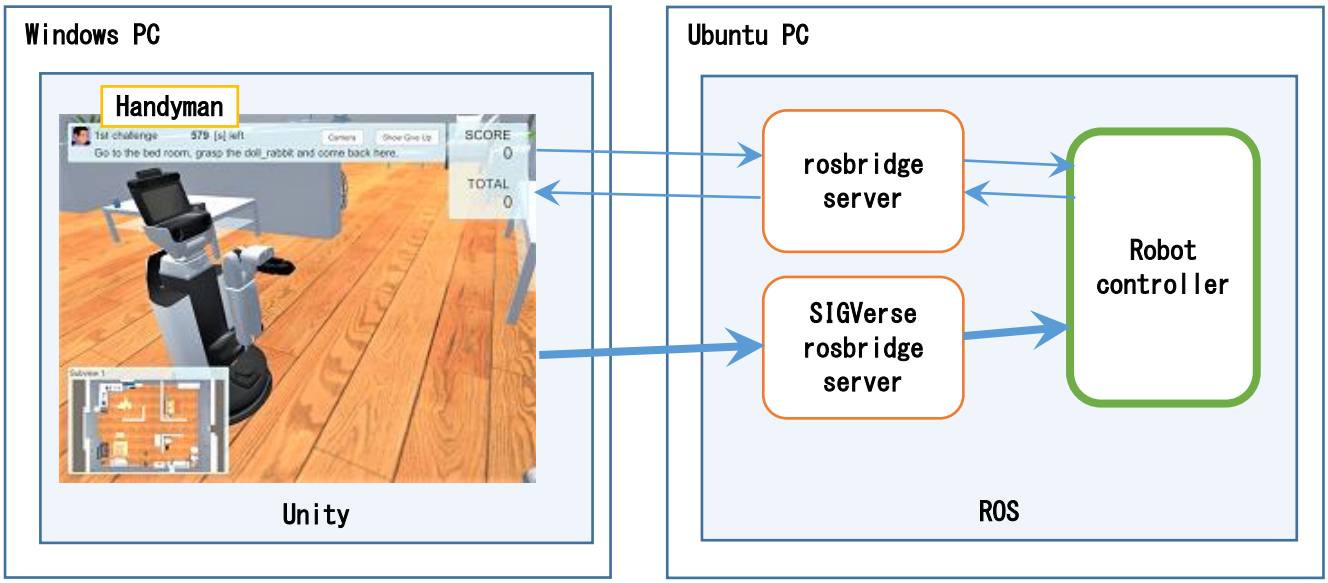

Figure 2.1 System configuration

The Windows computer runs the Handyman program. The Handyman program has been created in Unity.

The Ubuntu computer runs the rosbridge server, the SIGVerse rosbridge server, and the robot controller created by the competition participants.

The Handyman program and robot controller communicate through the basic rosbridge server, but communication with a large amount of data (sensor data, etc.) is transmitted through the SIGVerse rosbridge server.

In the Handyman program, robots move in accordance with the instructions from the robot controller when the human avatar issues commands to the robot.
The robot controller sends ROS messages such as Twist and JointTrajectory to the Handyman program to move the robot in the Handyman program.

The Handyman program distributes JointState, TF, sensor information, and other ROS messages at a regular interval to the Robot Controller.

2.2. Flow of the Competitive Challenge

The flow of the competitive challenge is outlined below.

1. The avatar generates a statement (task) to initialize the position and direction of the robot and object to grasp.
2. The avatar sends the "**Are_you_ready**?" message to the robot.
3. The robot sends the "**I_am_ready**" message to the avatar.
4. The avatar sends the **statement (task)** that is generated to the robot.
5. The robot moves to the room for instruction.
6. The robot sends the "**Room_reached**" message to the avatar.
7. The avatar checks the first statement.
   If successful, points are awarded and the challenge moves on to the next task.
   If unsuccessful, the task ends.
8. The robot grasps the object.
9. The robot sends the "**Object_grasped**" message to the avatar.
10. The avatar checks the second statement.
    If successful, points are awarded and the challenge moves on to the next task.
    If unsuccessful, the task ends.
11. The robot carries out the instruction after grasping.
12. The robot returns to the position of the avatar once all of the instructions are completed successfully.
13. The robot sends the "**Task_finished**" message to the avatar.
14. The avatar checks the position of the robot.
    If successful, points are awarded. The task ends.

- If the task ends (successfully or unsuccessfully):
  The avatar sends a "**Task_succeeded**" (successful task) or "**Task_failed**" (unsuccessful task) message to the robot to start the next task when participants still have attempts left.
  The avatar sends the "**Mission_complete**" message to the robot to end the competitive challenge when participants have no attempts left.
- If the time limit has passed:
  The avatar sends the "**Task_failed**" message to the robot to indicate the task was unsuccessful.
- If users would like to withdraw from the task:
  Users press the Give Up button on the screen to have the avatar send the "**Task_failed**" message to the robot and indicate the task was unsuccessful.

The method to switch rooms will be announced later.
Avatar will send an environment ID to robot at first.

Additional situations would be added in which grasping targets do not exist.

2.3. Time Limits for the Competitive Challenge

- The time limit for each task is 10 minutes.
- Participants have 15 attempts for the tasks.

More detailed information will be announced later.

2.4. Scoring (Each Task)

✓	Arriving at the designated room	+20
✓	Grasping the designated object	+50
✓	Returning to the position of the avatar	+30
✓	Collisions (each time)	-10

The score will be 0 points even if many deductions results in a score below zero.
This is tentative points. More detailed information will be announced later.

2.5. Other

2.5.1. Regarding Statements (Tasks)

Statements are given in free format; always include grasping behavior such as:

Go to the xxx, grasp the *** and come back here,

where xxx indicates the name of the room and *** indicates the name of the object.

2.5.2. Regarding the Position of the Avatar

The avatar shall be positioned at the center of a circle with a 2 m radius.

2.6. Handyman Screen During the Competitive Challenge

Fundamentally, the camera shall track the robot in the screen as shown below.

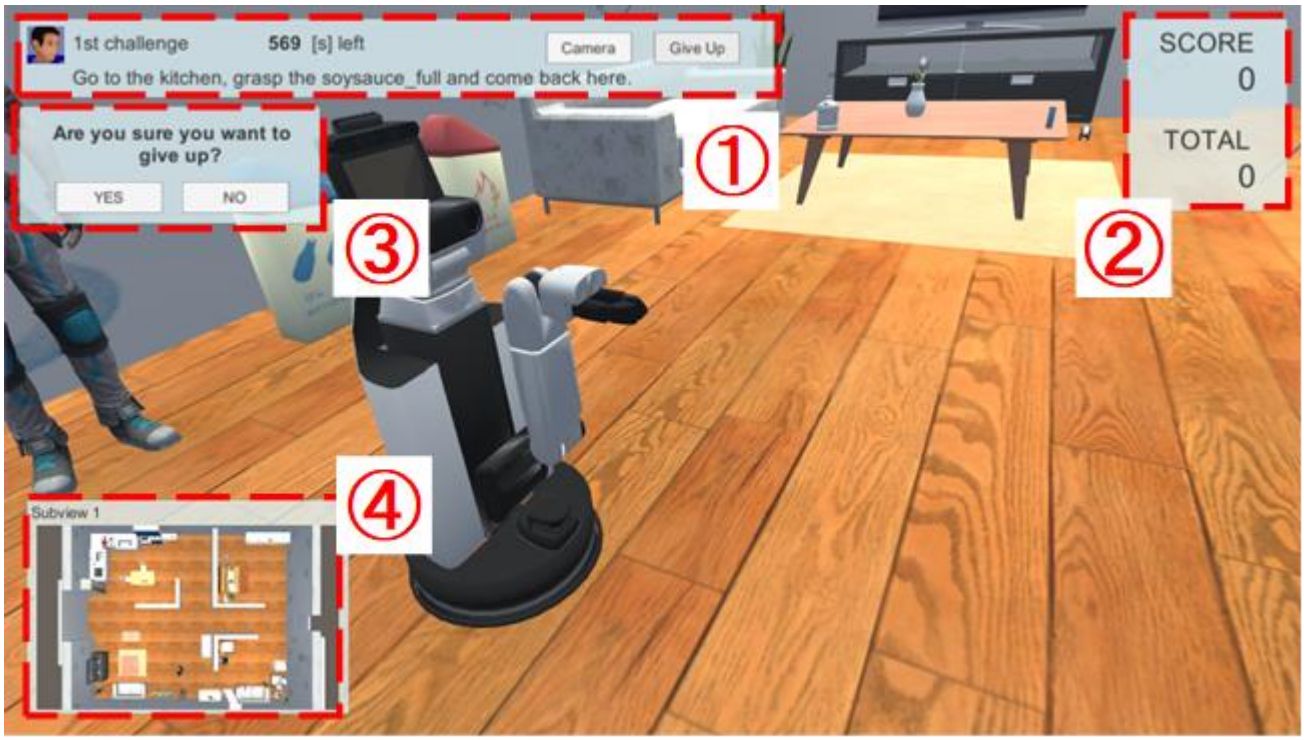

Figure 2.2 Main window

(1) Status information

Status information displays the number of attempts left, the time remaining, a description of the task, the [Camera] button, and the [Give Up] button.

Press the [Camera] button to change the perspective of the camera.

Press the [Give Up] button to display the Give Up menu shown in (3).

(2) Score information

Score information displays the score for each task and the total score for all of the tasks.

(3) Give Up menu

The Give Up menu forces the failure of a task by pressing the [YES] button. This menu closes by pressing the [NO] button.

(4) Overhead view

The overhead view displays the room from a top view.

2.7. Room List

The list of room names used in the challenge is as follows:

Table 2.1 Room list

No	Name
1	bed room
2	kitchen
3	living room
4	lobby

Figure 2.3 An example of room layout

2.8. Graspable Object List

A list of potential objects to grasp during the challenge is shown below.

Table 2.2 Graspable object list

No	Name	No	Name
1	apple	7	sugar
2	toy_D	8	soysauce_full
3	doll_rabbit	9	sauce_full
4	doll_bear	10	ketchup_full
5	doll_dog	11	tumblerglass
6	cannedjuice	12	cup_c01

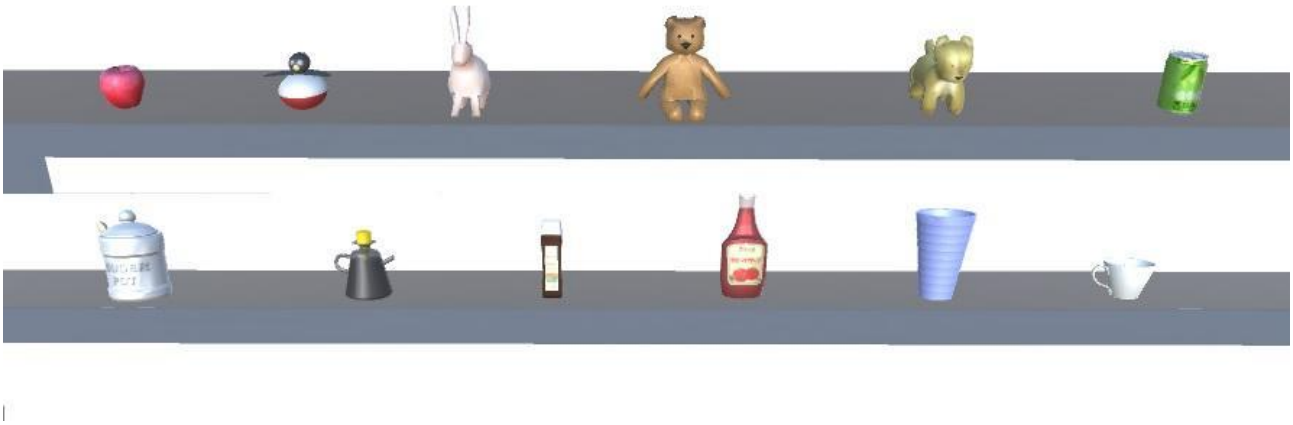

Figure 2.4 Graspable objects

3. Detailed software of the Handyman

3.1. Start-up of Various Programs

3.1.1. Starting the Handyman Program

Download the Unity project from the Git provided by the competition and start the Handyman program by referring to the READEME file.

The Unity project files are located in the directory below:
/Assets/Competition/Handyman/Handyman.unity

The default settings are as follows:

- Participants are given 15 attempts.
- The layout of potential objects to grasp is random.
- The object to grab is also random and the task statement is automatically generated.

See the description of the Handyman settings file (HandymanConfig.json) included hereafter for details about the settings.

3.1.2. Sample Program for the Robot Controller (Ubuntu Computer)

Download the sample robot controller for Handyman from the Git provided by the competition and start the sample program by referring to the README file.
The execution method for this program is indicated below.
$ roslaunch handyman handyman_sample.launch

The behavior of the sample program moves to grasp objects in space with the right hand regardless of the task.
Therefore, the attempt on that task immediately ends if the robot enters the wrong room. The task fails of course because the robot cannot grasp the object correctly.

3.1.3. Starting the Tool to Move the Robot via Keyboard Control (Ubuntu Computer)

Participants can control and move the robots with a keyboard by starting this tool instead of the robot controller.

Download the teleoperation tool for Handyman from the Git provided by the competition and start the teleoperation tool by referring to the README file.
The execution method for this program is indicated below.

```
$ roslaunch handyman tool_teleop_key.launch
```

The operational controls for this program are as follows:

```
$ arrow keys : Move the robot
$ s          : Stop the robot
$ ---------------------------
$ u  : Move Arm Up
$ j  : Stop Arm
$ m : Move Arm Down
$ ---------------------------
$ a : Make Arm - Vertical
$ b : Make Arm - Horizontal
$ c : Make Arm - Low position
$ ---------------------------
$ g : Grasp/Open hand
$ ---------------------------
$ 1 : Send Message "Room_reached"
$ 2 : Send Message "Object_grasped"
$ 3 : Send Message "Task_finished"
$ ---------------------------
$ h : Show help
```

Furthermore, start tool_teleop_key_with_rviz.launch to boot other programs such as rviz simultaneously.

3.2. ROS Messages for Sending and Receiving Between the Avatar and Robot

3.2.1. ROS Topic List

The avatar and the robot send and receive data by using the ROS topics below.

Table 3.1 ROS Topic list

No	Topic Name	Direction	Message Type	Description
1	/handyman/message/to_robot	SIGVerse->	handyman/HandymanMsg	Event message from Robot to Avatar
2	/handyman/message/to_moderator	ROS->	handyman/HandymanMsg	Event message from Avatar to Robot
3	/hsrb/opt_command_velocity	ROS->	geometry_msgs/Twist	To move Base (base_footprint) of Robot
4	/hsrb/head_trajectory_controller/command	ROS->	trajectory_msgs/JointTrajectory	To move Head (head_pan_joint, head_tilt_joint)
5	/hsrb/arm_trajectory_controller/command	ROS->	trajectory_msgs/JointTrajectory	To move Arm (arm_lift_joint, arm_flex_joint, arm_roll_joint, wrist_flex_joint, wrist_roll_joint)
6	/hsrb/gripper_trajectory_controller/command	ROS->	trajectory_msgs/JointTrajectory	To move Gripper (hand_l_proximal_joint, hand_r_proximal_joint)
7	/hsrb/joint_states	SIGVerse->	sensor_msgs/JointState	State of joints
8	/hsrb/base_scan	SIGVerse->	sensor_msgs/LaserScan	Value of Laser range sensor
9	/hsrb/head_rgbd_sensor/rgb/image_raw	SIGVerse->	sensor_msgs/Image	Image of Xtion rgb camera
10	/hsrb/head_rgbd_sensor/depth/image_raw	SIGVerse->	sensor_msgs/Image	Image of Xtion depth camera
11	/hsrb/head_center_camera/image_raw	SIGVerse->	sensor_msgs/Image	Image of the head camera
12	/hsrb/hand_camera/image_raw	SIGVerse->	sensor_msgs/Image	Image of the hand camera

Note: SIGVerse->: SIGVerse to User's ROS Node, ROS->: User's ROS Node to SIGVerse

A detailed description of 'Event Message' in No.1-2 is provided hereafter.
Furthermore, the TF information of robots is distributed at a regular interval.

3.2.2. Event Messages

Sending and receiving of event messages between the human avatar and robot is performed using original ROS messages (handyman/HandymanMsg).

[handyman/HandymanMsg]
string message
string detail

Table 3.2 List of HandymanMsg

No	Event	Direction	message	detail
1	Send Are_you_ready?	Avatar->	Are_you_ready?	(blank)
2	Send I_am_ready	Robot->	I_am_ready	(blank)
3	Send the task message	Avatar->	Instruction	task message (e.g.: Go to the XXXX, grasp the YYYY and come back here.)
4	Send Room_reached	Robot->	Room_reached	(blank)
5	Failed Room_reached	Avatar->	Task_failed	Room_reached
6	Send Object_grasped	Robot->	Object_grasped	(blank)
7	Failed Object_grasped	Avatar->	Task_failed	Object_grasped
8	Send Task_finished	Robot->	Task_finished	(blank)
9	Failed Task_finished	Avatar->	Task_failed	Task_finished
10	Succeeded the task	Avatar->	Task_succeeded	(blank)
11	All tasks finished	Avatar->	Mission_complete	(blank)
12	Time is up	Avatar->	Task_failed	Time_is_up
13	Click GiveUp button	Avatar->	Task_failed	Give_up

Note: Avatar->: Human avatar to Robot, Robot->: Robot to Human avatar

3.3. Handyman Settings File/Log File Configuration

A list of the settings file and log file configuration is as follows:

/ SIGVerse.log
/ SIGVerseConfig / SIGVerseConfig.json
/ SIGVerseConfig / TeamLogo.jpg
/ SIGVerseConfig / Handyman / HandymanConfig.json
/ SIGVerseConfig / Handyman / HandymanScore.txt
/ SIGVerseConfig / Handyman / EnvironmentInfoXX.json
/ SIGVerseConfig / Handyman / PlaybackXX.dat

3.3.1. SIGVerse.log

SIGverse.log is a common SIGVerse execution file.
A comprehensive execution log is output while Handyman is running.

3.3.2. SIGVerseConfig.json

SIGVerseConfig.json is a common SIGVerse settings file.

Table 3.3 Parameters of SIGVerseConfig.json

No	Name	Type	Example	Description
1	rosbridgeIP	string	"192.168.1.101"	IP address for rosbridge
2	rosbridgePort	int	9090	Port number for rosbridge
3	sigverseBridgePort	int	50001	Port number for SIGVerse rosbridge
4	useSigverseMenu	bool	true	Using SIGVerse menu or not
5	isAutoStartWithMenu	bool	true	Start the program automatically or not
6	setUpRosTimestamp	bool	true	Set up Time stamps of ROS message

3.3.3. TeamLogo.jpg

TeamLogo.jpg is the logo image provided by the competitive participants.
This image is displayed in the Handyman screen to identify the participating teams.

3.3.4. HandymanConfig.json

HandymanConfig.json is a dedicated Handyman settings file.

Table 3.4 Parameters of HandymanConfig.json

No	Name	Type	Example	Description
1	maxNumberOfTrials	int	15	Max number of the trials
2	isScoreFileRead	bool	false	Read HandymanScore.txt or not. It is for recovery in case of some failure. **true:** Start from the continuation of the score file. **false:** Normal mode.
3	isGraspableObjectsPositionRandom	bool	true	**true:** The positions of graspable objects are decided randomly when executed. And outputs EnvironmentInfoXX.json. **false:** The positions of graspable objects are decided using EnvironmentInfoXX.json. And doesn't output EnvironmentInfoXX.json.
4	isAlwaysGoNext	bool	false	It is for debugging. **true:** Even if it failed, it goes to the next step. **false:** Normal mode.
5	playbackType	int	1	Playback mode **1:** Record motions of scene objects. **2:** Play the recorded motions. It is debug mode.

3.3.5. HandymanScore.txt

Handymancore.txt is a file recording the score output by the Handyman program.

This file records the score for the first task on the first line and the score for the second task on the second line.

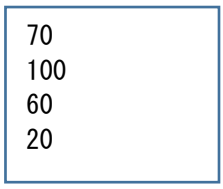

Figure 3.1 HandymanScore.txt

The HandymanScore.txt file is fundamentally output to record the score, but a file already including scores can be used to start the next competitive task if isScoreFileUsed is true in HandymanConfig.json.

3.3.6. EnvironmentInfoXX.json

EnvironmentInfoXX.json is a file recording information about relocatable objects output by the Handyman program.

The names of objects to grasp as well as the position and orientation of the graspable objects and other information are saved in the JSON format.

XX in the file name is the attempt number of the task. EnvironmentInfo01.json would be the name of the file for the first attempt of a task.

If isGraspableObjectsPositionRandom is true in HandymanConfig.json, the position and orientation of graspable objects is determined randomly on execution and output as this file.

If isGraspableObjectsPositionRandom is false, the existing EnvironmentInfoXX.json file is used to arrange graspable objects.

3.3.7. PlaybackXX.dat

PlaybackXX.dat is the file recording the movement of objects during the competitive task output by the Handyman program.

This is a time-series data file that primarily records the position and orientation of objects that can be moved (robot, graspable objects, non-kinematic objects, etc.).

This file is used to replay and confirm the movements of the robot that have been recorded.

XX in the file name is the attempt number of the task. Playback01.dat would be the name of the file for the first attempt of a task.

This file is output if playbackType in HandymanConfig.json is 1.

If playbackType is 2 in HandymanConfig.json, the program loads Playback**00**.dat to play back the robot and movement of the objects in the scene that have been recorded.

Therefore, if you would like to replay attempt 12 of a task, rename Playback12.dat in which the information has been recorded as Playback00.dat to use that file for playback.

3.4. Various Execution Methods changed in the Settings File of the Handyman Program

3.4.1. Method to Set the Layout of Potential Objects to Grasp

The default settings place the potential objects to grasp randomly, but this section describes the method to set the layout of these objects.

If HandymanConfig.json is executed with isGraspableObjectsPositionRandom as true, the layout information is exported to EnvironmentInfoXX.json when executed.

Thereafter, the layout information can be loaded from the EnvironmentInfoXX.json file that was output to perform a simulation with the same layout conditions each time by executing HandymanConfig.json with sGraspableObjectsPositionRandom set to false.
In this case, an error is triggered during execution if an existing EnvironmentInfoXX.json file is not found.

3.4.2. Resuming the Challenge from the Next Task After Ending a Task Before Completion

This section describes how to resume the challenge from the next task if the previous task is ended before completion for some reason.

If isScoreFileUsed is true in HandymanConfig.json, the file already including scores can be used to start the next task in the challenge.

3.4.3. Playback of Robot Movements During the Challenge

This section describes how to easily play back the movements of the robot for later review of the robot movements during the challenge. **Be aware this playback cannot fully reproduce collisions and changes to points during the challenge.**

If HandymanConfig.json is executed with playbackType as 1, the operational information during the challenge is exported to PlaybackXX.dat.

Thereafter, load the operational information from Playback**00**.dat and play back the movements by setting playbackType to 2 in HandymanConfig.json and executing the program.
Make sure the number for the file to load is “00” for playback. The file name needs to be changed to “00” because the number of the attempt is used when recording the operation. Only one file can be used for playback each time the program starts.

4 Overview of Interactive Cleanup

4.1 System Configuration

The system configuration for this competitive challenge is outlined below.

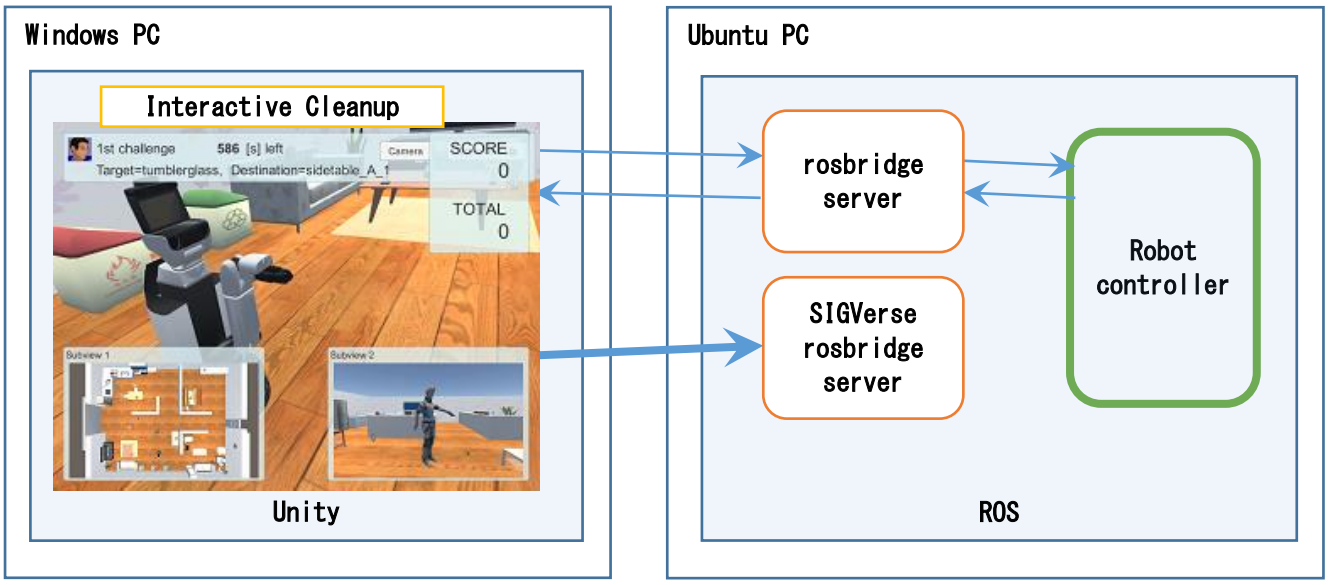

Figure 4.1 System configuration

The Windows computer runs the Interactive Cleanup program. The Interactive Cleanup program has been created in Unity.
The Ubuntu computer runs the rosbridge server, the SIGVerse rosbridge server, and the robot controller created by the competition participants.

The Interactive Cleanup program and robot controller communicate through the basic rosbridge server, but communication with a large amount of data (sensor data, etc.) is transmitted through the SIGVerse rosbridge server.

In the Interactive cleanup program, robots move in accordance with the instructions from the robot controller when the human avatar issues the Cleanup command to the robot.
The instructions for the Cleanup command are determined based on the operations of the human avatar and the messages sent by the human avatar.

The robot controller sends ROS messages such as Twist and JointTrajectory to the Interactive Cleanup program to move the robot in the Interactive Cleanup program.

The Interactive Cleanup program distributes JointState, TF, sensor information, and other ROS messages at a regular interval to the Robot Controller.

4.2 Flow of the Competitive Challenge

The flow of the competitive challenge is outlined below.

(1) Initialize the position and direction of the robot and object to grasp.
(2) The avatar sends the "**Are_you_ready?**" message to the robot.
(3) The robot sends the "**I_am_ready**" message to the avatar.
(4) The avatar issues the Cleanup command.
(4-1) The avatar moves to point to the object for Cleanup.
(4-2) The "**Pick_it_up!**" message is sent to the robot.
(4-3) The avatar moves to indicate the location of the object for Cleanup.
(4-4) The "**Clean_up!**" message is sent to the robot.
(5) The robot moves to grasp the object for cleanup.
(5-1) The robot closes the gripper to clasp the object to grasp.
(5-2) Confirmation of whether the object is correct can be performed optionally.
The robot will send the "**Is_this_correct?**" message after grasping the object, and then the avatar responds either "**Yes/No.**" However, a deduction is taken for each confirmation.
(6) The robot moves to grasp the object for cleanup.
(7) The robot sends the "**Task_finished**" message to the avatar.
(8) The avatar checks the status of Cleanup and assigns points.
The task ends.

- If the task ends (successfully or unsuccessfully):
  The avatar sends a "**Task_succeeded**" (successful task) or "**Task_failed**" (unsuccessful task) message to the robot to start the next task when participants still have attempts left.
  The avatar sends the "**Mission_complete**" message to the robot to end the competitive challenge when participants have no attempts left.
- If the time limit has passed:
  The avatar sends the "**Task_failed**" message to the robot to indicate the task was unsuccessful.
- If users would like to withdraw from the task:
  Users press the Give Up button on the screen to have the avatar send the "**Task_failed**" message to the robot and indicate the task was unsuccessful.
- Results of Cleanup:
  The Cleanup will be deemed a success if the object for clean-up is placed on the table or disposed of in the garbage box pointed to by the avatar.

The method to switch rooms will be announced later.
The avatar will send an environment ID to robot at first.

4.3 Time Limits for the Competitive Challenge

- The time limit for each task is 10 minutes.
- Participants have 15 attempts for the tasks.

More detailed information will be announced later.

4.4 Scoring (Each Task)

✔ Successful Cleanup of the designated object +100

✔ Confirmation of object correctness (each time) -10

✔ Collision (each time) -10

The score will be 0 points even if many deductions results in a score below zero.

This is tentative point. More detailed information will be announced later.

4.5 Other

4.5.1 Regarding Avatar Finger Pointing

The finger for the human avatar to point with is the pointer finger on either the right or left hand.

4.5.2 Regarding the Robot and Avatar

The initial position of the robot and human avatar will be fixed.
The height of the human avatar will be fixed.

4.5.3 Regarding the Publication of room layouts

The layouts of the room used for the competition will be made public before the competition.

4.6 Interactive Cleanup Screen During the Competitive Challenge

Fundamentally, the camera shall track the robot in the screen as shown below.

Figure 4.2 Main window

(1) Status information

Status information displays the number of attempts left, a description of the task, the [Camera] button, and the [Give Up] button.
Press the [Camera] button to change the perspective of the camera.
Press the [Give Up] button to display the Give Up menu shown in (3).
Press the SIGVerse icon on the upper-left to hide the various menus below the icon.

(2) Score information

Score information displays the score for each task and the total score for all of the tasks.

(3) Give Up menu

The Give Up menu forces the fail of a task by pressing the [YES] button. This menu closes by pressing the [NO] button.

(4) Overhead view

The overhead view displays the room from a top view.

(5) Avatar view

The avatar view displays the human avatar.

4.7 Room

An example of a room used in the competition is shown below.

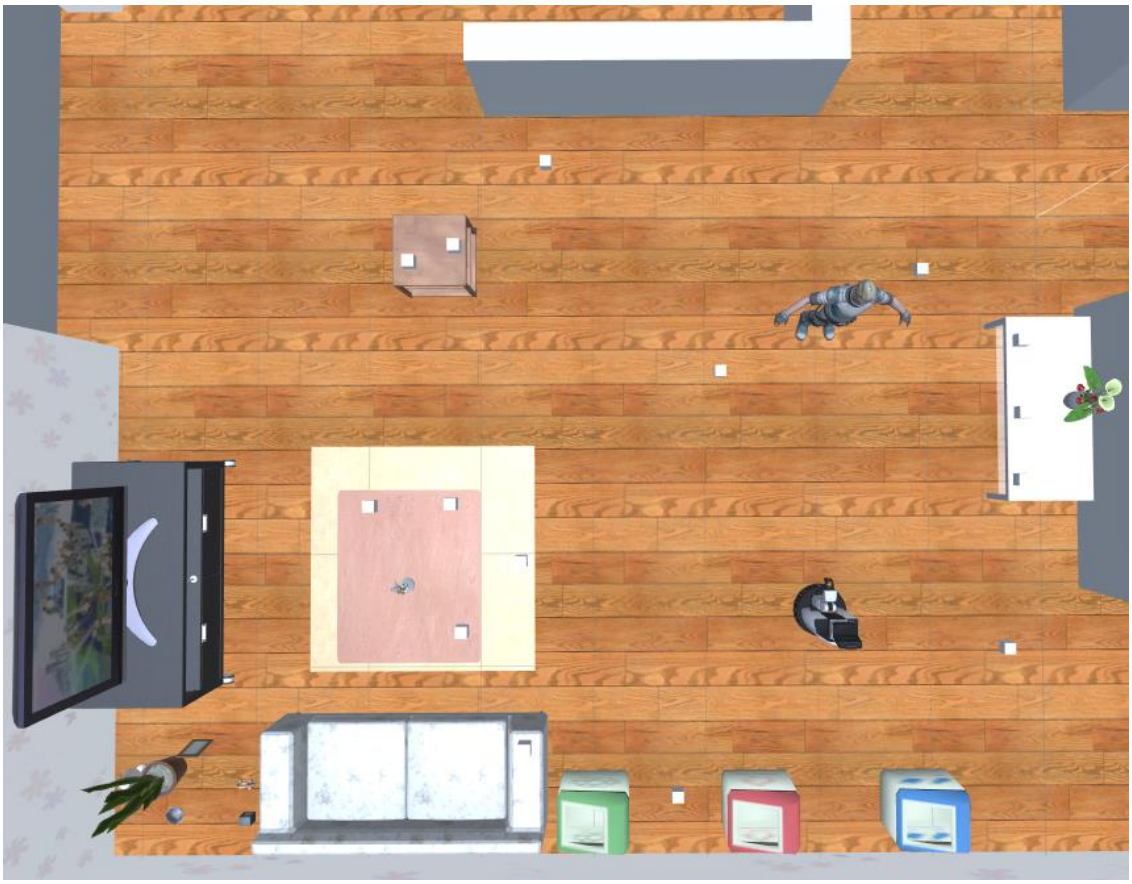

Figure 4.3 An example of the room for the Interactive Cleanup

4.8 Graspable Object List

A list of potential objects to grasp during the challenge is shown below.

Table 4.1 Graspable object list

No	Name	No	Name
1	apple	7	sugar
2	toy_D	8	soysauce_full
3	doll_rabbit	9	sauce_full
4	doll_bear	10	ketchup_full
5	doll_dog	11	tumblerglass
6	cannedjuice	12	cup_c01

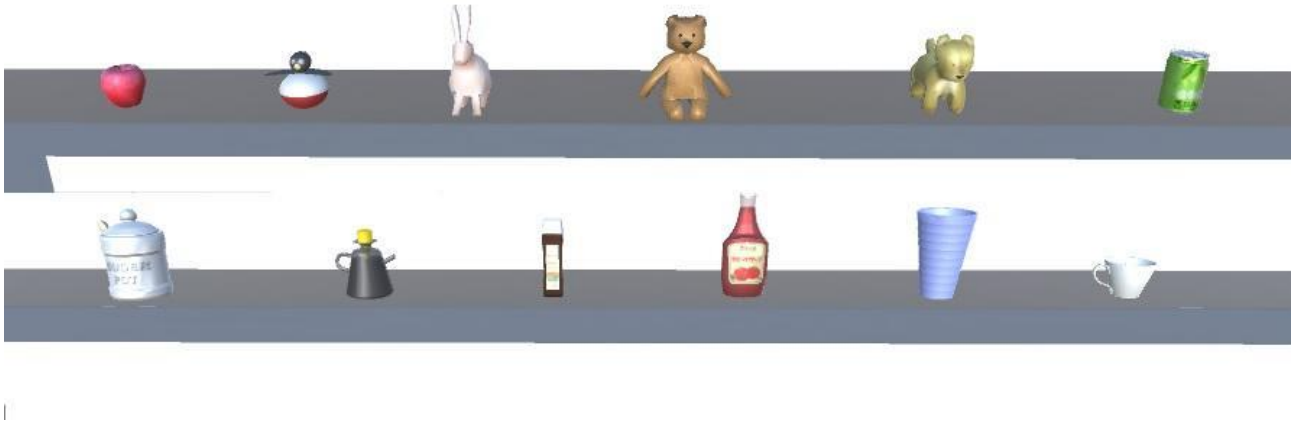

Figure 4.4 Graspable objects

4.9 Destination List

A list of destination during the challenge is shown below.

Table 4.2 Destination list

No	Name	No	Name
1	lowtable_A	4	trashbox_c01
2	wagon_c02	5	trashbox_c02
3	sidetable_A_1	6	trashbox_c03

Figure 4.5 Destination objects

5. Detailed software information of Interactive Cleanup

5.1. Start-up of Various Programs

5.1.1. Starting the Interactive Cleanup Program

Download the Unity project from the GIT provided by the competition and start the Interactive Cleanup Program program by referring to the READEME file.
The Unity project files are located in the directory below:
/Assets/InteractiveCleanup/InteractiveCleanup.unity

The default settings are as follows:

- Participants are given 15 attempts.
- Competition mode

See the description of the Interactive Cleanup settings file (InteractiveCleanupConfig.json) included hereafter for details about the settings.

5.1.2. Sample Program for the Robot Controller (Ubuntu Computer)

Download the sample robot controller for Interactive Cleanup from the Git provided by the competition and start the sample program by referring to the README file.
The execution method for this program is indicated below.
$ roslaunch interactive_cleanup interactive_cleanup_sample.launch

The behavior of the sample program moves to grasp objects in space with the right hand regardless of the task. The robot moves to a suitable place, opens the gripper, and the task ends.

5.1.3. Starting the Tool to Move the Robot via Keyboard Control (Ubuntu computer)

Participants can control and move the robots with a keyboard by starting this tool instead of the robot controller.

Download the teleoperation tool for Interactive Cleanup from the Git provided by the competition and start the teleoperation tool by referring to the README file.
The execution method for this program is indicated below.

```
$ roslaunch interactive_cleanup tool_teleop_key.launch
```

The operational controls for this program are as follows:

```
$ arrow keys : Move the robot
$ s          : Stop the robot
$ ---------------------------
$ u  : Move Arm Up
$ j  : Stop Arm
$ m : Move Arm Down
$ ---------------------------
$ a : Make Arm - Vertical
$ b : Make Arm - Horizontal
$ c : Make Arm - Low position
$ ---------------------------
$ g : Grasp/Open hand
$ ---------------------------
$ 1 : Send Message "Is_this_correct?"
$ 2 : Send Message "Task_finished"
$ ---------------------------
$ h : Show help
```

Furthermore, start tool_teleop_key_with_rviz.launch to boot other programs such as rviz simultaneously.

5.2. ROS Messages for Sending and Receiving Between the Avatar and Robot

5.2.1. ROS Topic List

The avatar and the robot send and receive data by using the ROS topics below.

Table 5.1 ROS Topic list

No	Topic Name	Direction	Message Type	Description
1	/interactive_clanup/message/to_robot	SIGVerse->	interactive_cleanup/InteractiveCleanupMsg	Event message from Robot to Avatar
2	/interactive_clanup/message/to_moderator	ROS->	interactive_cleanup/InteractiveCleanupMsg	Event message from Avatar to Robot
3	/hsrb/opt_command_velocity	ROS->	geometry_msgs/Twist	To move Base (base_footprint) of Robot
4	/hsrb/head_trajectory_controller/command	ROS->	trajectory_msgs/JointTrajectory	To move Head (head_pan_joint, head_tilt_joint)
5	/hsrb/arm_trajectory_controller/command	ROS->	trajectory_msgs/JointTrajectory	To move Arm (arm_lift_joint, arm_flex_joint, arm_roll_joint, wrist_flex_joint, wrist_roll_joint)
6	/hsrb/gripper_trajectory_controller/command	ROS->	trajectory_msgs/JointTrajectory	To move Gripper (hand_l_proximal_joint, hand_r_proximal_joint)
7	/hsrb/joint_states	SIGVerse->	sensor_msgs/JointState	State of joints
8	/hsrb/base_scan	SIGVerse->	sensor_msgs/LaserScan	Value of Laser range sensor
9	/hsrb/head_rgbd_sensor/rgb/image_raw	SIGVerse->	sensor_msgs/Image	Image of Xtion rgb camera
10	/hsrb/head_rgbd_sensor/depth/image_raw	SIGVerse->	sensor_msgs/Image	Image of Xtion depth camera
11	/hsrb/head_center_camera/image_raw	SIGVerse->	sensor_msgs/Image	Image of the head camera
12	/hsrb/hand_camera/image_raw	SIGVerse->	sensor_msgs/Image	Image of the hand camera

Note: SIGVerse->: SIGVerse to User's ROS Node, ROS->: User's ROS Node to SIGVerse

A detailed description of ‘Event Message’ in No.1-2 is provided hereafter.
Furthermore, the TF information of robots is distributed at a regular interval.

5.2.2. Event Messages

Sending and receiving of event messages between the human avatar and robot use original ROS messages (interactive_cleanup/InteractiveClanupMsg).

[interactive_cleanup/InteractiveCleanupMsg]
string message
string detail

Table 5.2 List of InteractiveCleanupMsg

No	Event	Direction	message	detail
1	Send Are_you_ready?	Avatar->	Are_you_ready?	(blank)
2	Send I_am_ready	Robot->	I_am_ready	(blank)
3	Send Pick_it_up!	Avatar->	Pick_it_up!	(blank)
4	Send Clean_up!	Avatar->	Clean_up!	(blank)
5	Send Is_this_correct?	Robot->	Is_this_correct?	(blank)
6	Send Yes or No	Avatar->	Yes / No	(blank)
7	Send Task_finished	Robot->	Task_finished	(blank)
8	Failed Task_finished	Avatar->	Task_failed	Task_finished
9	Succeeded the task	Avatar->	Task_succeeded	(blank)
10	All tasks finished	Avatar->	Mission_complete	(blank)
11	Time is up	Avatar->	Task_failed	Time_is_up
12	Click GiveUp button	Avatar->	Task_failed	Give_up

Note: Avatar->: Human avatar to Robot, Robot->: Robot to Human avatar

5.3. Interactive Cleanup Settings File/Log File Configuration

A list of the settings file and log file configuration is as follows:

/ SIGVerse.log
/ SIGVerseConfig / SIGVerseConfig.json
/ SIGVerseConfig / TeamLogo.jpg
/ SIGVerseConfig / InteractiveCleanup / InteractiveCleanupConfig.json
/ SIGVerseConfig / InteractiveCleanup / InteractiveCleanupScore.txt
/ SIGVerseConfig / InteractiveCleanup / EnvironmentInfoXX.json
/ SIGVerseConfig / InteractiveCleanup / AvatarMotionXX.dat
/ SIGVerseConfig / InteractiveCleanup / PlaybackXX.dat

5.3.1. SIGVerse.log

SIGverse.log is a common SIGVerse execution file.
A comprehensive execution log is output while Interactive Cleanup is running.

5.3.2. SIGVerseConfig.json

SIGVerseConfig.json is a common SIGVerse settings file.

Table 5.3 Parameters of SIGVerseConfig.json

No	Name	Type	Example	Description
1	rosbridgeIP	string	"192.168.1.101"	IP address for rosbridge
2	rosbridgePort	int	9090	Port number for rosbridge
3	sigverseBridgePort	int	50001	Port number for SIGVerse rosbridge
4	useSigverseMenu	bool	true	Using SIGVerse menu or not
5	isAutoStartWithMenu	bool	true	Start the program automatically or not
6	setUpRosTimestamp	bool	true	Set up Time stamps of ROS message

5.3.3. TeamLogo.jpg

TeamLogo.jpg is the logo image provided by the competitive participants.
This image is displayed in the Interactive Cleanup screen to identify the participating teams.

5.3.4. InteractiveCleanupConfig.json

InteractiveCleanupConfig.json is a dedicated Interactive Cleanup settings file.

Table 5.4 Parameters of InteractiveCleanupConfig.json

No	Name	Type	Example	Description
1	maxNumberOfTrials	int	15	Max number of the trials
2	isScoreFileRead	bool	false	Read InteractiveCleanupScore.txt or not. It is for recovery in case of some failure. **true:** Start from the continuation of the score file. **false:** Normal mode.
3	executionMode	int	0	**0: For the competition.** This mode uses EnvironmentInfoXX.json and AvatarMotionXX.dat. **1: For data generation.** This mode generates EnvironmentInfoXX.json and AvatarMotionXX.dat.
4	isAlwaysGoNext	bool	false	It is for debugging. **true:** Even if it failed, it goes to the next step. **false:** Normal mode.
5	playbackType	int	1	Playback mode **1:** Record motions of scene objects. **2:** Play the recorded motions. It is debug mode.

5.3.5. InteractiveCleanupScore.txt

InteractiveCleanupScore.txt is a file recording the score output by the Interactive Cleanup program.
This file records the score for the first task on the first line and the score for the second task on the second line.

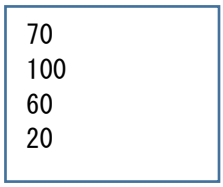

Figure 5.1 InteractiveCleanupScore.txt

The InteractiveCleanupScore.txt file is fundamentally output to record the score, but a file already including scores can be used to start the next competitive task if isScoreFileUsed is true in InteractiveCleanupConfig.json.

5.3.6. EnvironmentInfoXX.json

EnvironmentInfoXX.json is a file recording information about relocatable objects output by the Interactive Cleanup program.
The names of objects to grasp as well as the position and orientation of the graspable objects, the position and orientation of objects for cleanup and other information are saved in the JSON format.
XX in the file name is the attempt number of the task. EnvironmentInfo01.json would be the name of the file for the first attempt of a task.
This file is generated if the execution mode (executionMode) is the data generation mode. (Described hereafter)

5.3.7. AvatarMotionXX.dat

AvatarMotionXX.dat is a file recording the time-series motion data of the human avatar output by the Interactive Cleanup program.
XX in the file name is the attempt number of the task. AvatarMotion01.dat would be the name of the file for the first attempt of a task.
This file is generated if the execution mode (executionMode) is the data generation mode. (Described hereafter)

5.3.8. PlaybackXX.dat

PlaybackXX.dat is the file recording the movement of objects during the competitive task output by the Interactive Cleanup program.
This is a time-series data file that primarily records the position and orientation of objects that can be moved (robot, graspable objects, non-kinematic objects, etc.).
This file is used to replay and confirm the movements of the robot that have been recorded.

XX in the file name is the attempt number of the task. Playback01.dat would be the name of the file for the first attempt of a task.

This file is output if playbackType in InteractiveCleanupConfig.json is 1.

If playbackType is 2 in InteractiveCleanupConfig.json, the program loads Playback**00**.dat to play back the robot and movement of the objects in the scene that have been recorded.
Therefore, if you would like to replay attempt 12 of a task, rename Playback12.dat in which the information has been recorded as Playback00.dat to use that file for playback.

5.4 Various Execution Methods Changed in the Settings File of the Interactive Cleanup Program

5.4.1 Changes of Execution Mode (executionMode)

The program starts in the data generation mode if executionMode is **0** in InteractiveCleanupConfig.json and the mode for the competitive challenge is **1**.

In the mode for the competitive challenge, the existing EnvironmentInfoXX.json file is used to relocate objects and AvatarMotionXX.dat is used to reproduce the movements of the human avatar.
In this case, an error is triggered during execution if these files are not found.
The competitive challenge can be performed by setting up the position and orientation of the relocatable objects and movements of the human avatar in the same state every time by generating these files in advance to load.

In the data generation mode, need to prepare Oculus Rift + Oculus Touch. The position and orientation of relocatable objects is determined randomly upon execution, and the graspable objects are determined by the objects pointed to by the human avatar. In addition, EnvironmentInfoXX.json and AvatarMotionXX.dat also records and outputs the movement of the human avatar.

5.4.2 Generate the Competition Data Using Oculus Rift

Generating competition data using Oculus Rift CV 1 + Oculus Touch is described below.
Competition data means EnvironmentInfoXX.json and AvatarMotionXX.dat.

1. Set the executionMode of InteractiveCleanupConfig.json to 1 and start the program.
2. Start the ROS side program.
3. Proceed as usual until receiving "I_am_ready" message.
4. Perform the pointing action using Oculus Touch as follows:

4.1 Press the A button or the X button. (Start recording the avatar motion)
4.2 Press the middle finger trigger to display the laser pointer.
4.3 Select the grasping target with the laser pointer and press the index finger trigger.
4.4 Select the destination with the laser pointer and press the index finger trigger.
4.5 Press the A button or X button. (recording end of avatar motion)

Although the competition data is output in the above procedure, it is possible to continue the ROS program to perform the cleanup task.
In this way, it is possible to verify in real time how the robot moves according to the movement of the person who is pointing.

5.4.3 Resuming the Challenge from the Next Task After Ending a Task Before Completion

This section describes how to resume the challenge from the next task if the previous task is ended before completion for some reason.

If isScoreFileUsed is true in InteractiveCleanupConfig.json, the file already including scores can be used to start the next task in the challenge.

5.4.4 Playback of Robot Movements During the Challenge

This section describes how to easily play back the movements of the robot for later review of the robot movements during the challenge. **Be aware this playback cannot fully reproduce collisions and**

changes to points during the challenge.

If InteractiveCleanupConfig.json is executed with playbackType as 1, the operational information during the challenge is exported to PlaybackXX.dat.

Thereafter, load the operational information from Playback**00**.dat and play back the movements by setting playbackType to 2 in InteractiveCleanupConfig.json and executing the program.
Make sure the number for the file to load is “00” for playback. The file name needs to be changed to “00” because the number of the attempt is used when recording the operation. Only one file can be used for playback each time the program starts.

6 Human Navigation

6.1 Rules

[Overview]

- The purpose of this competitive challenge is to generate vocalization statements for a person rather than the control of the robot itself. EGPSR reverses the stance of the robot and speaker. The robot states the location of an object the person is searching for as a statement such as, “The object you are looking for is in the second drawer of the kitchen.” A person (test subject) from the general participants is recruited to follow that statement, logs into the avatar in VR, and then goes to take the object. The time until the object is retrieved is measured. The team that generates the easiest and most natural statement to understand for a person to retrieve the object the fastest earns points.
- Reference video for competitive challenge https://drive.google.com/file/d/0BxjNl2PRT1F_WFpFMG1VZUFWY0U/view?usp=sharing

[System Configuration for the Competitive Challenge]

- The computers and each program are connected in the configuration shown in Figure 2.

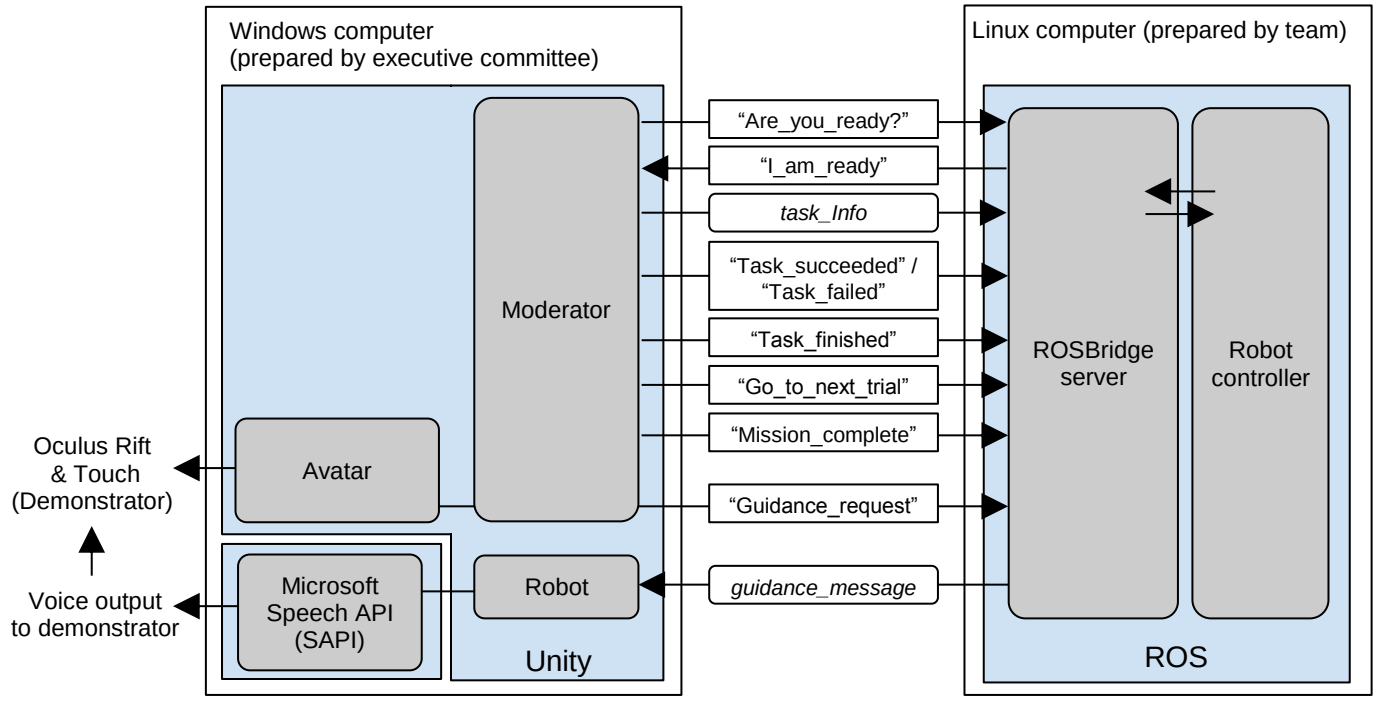

Figure 6.1: System Configuration for the Competitive Challenge

[Flow of the Competitive Challenge]

1. Team member launches the ROSBridge server, SIGVerse ROSBridge server and robot controller.
2. The operator (Competition committee) launches human_navi.exe
3. The demonstrator puts on the Oculus Rift & Touch.
4. The operator presses the task start button.
5. Load the environment for the competitive challenge to initialize the position and direction of the avatar
6. Moderator sends the “Are_you_ready?” message to the robot controller.
7. The robot controller sends the “I_am_ready” message to the moderator.
8. The moderator sends the *TaskInfo* to the robot. The *TaskInfo* includes the target object and

destination.

9. The robot controller generates and sends the *guidance_message* vocal statement to the robot. The vocal statement notifies the demonstrator. The robot controller can send *guidance_message* in an arbitrary timing.
10. The demonstrator follows the vocal statement to take the target object and to place it on the designated location (destination). The demonstrator can request a guidance message. After the demonstrator requested a new guidance message, the “Guidance_request” message is sent to the robot controller.
    - If the target object is grasped, points are awarded.
    - If the wrong object is grasped before the target object is grasped, a deduction is taken.
    - If the target object is placed on the destination, points are awarded.
11. Each trial is finished by events below.
    - The target object is placed on the destination: The Moderator sends the “Task_ succeeded” message to the robot controller.
    - Give_up button is pressed: The Moderator sends the “Task_faild” message to the robot controller.
    - Time is up: The Moderator sends the “Task_faild” message to the robot controller.
12. Attempt ends.
    - If no attempts are left, the moderator sends “Mission_complete” message to the robot controller.
    - If any attempts are left:
        i. The demonstrator moves to the next team area and puts on the Oculus Rift & Touch.
        ii. The operator presses the Go_to_next_trial button. The Moderator sends “Go_to_next_trial” message to the robot controller.
        iii. Return to 5.

[Regarding the Environment for the Competitive Challenge]

- See Figure 6.1 for an example of the configuration.
- A sample environment is scheduled for release before the competition.
- The environment to use during the competitive challenge is scheduled for release before the competition (the day before, etc.).
- The preparation of multiple types of environments is planned.
    - The environment for the competitive challenge includes furniture not provided in the sample, and changes such as the quantity may also be made.

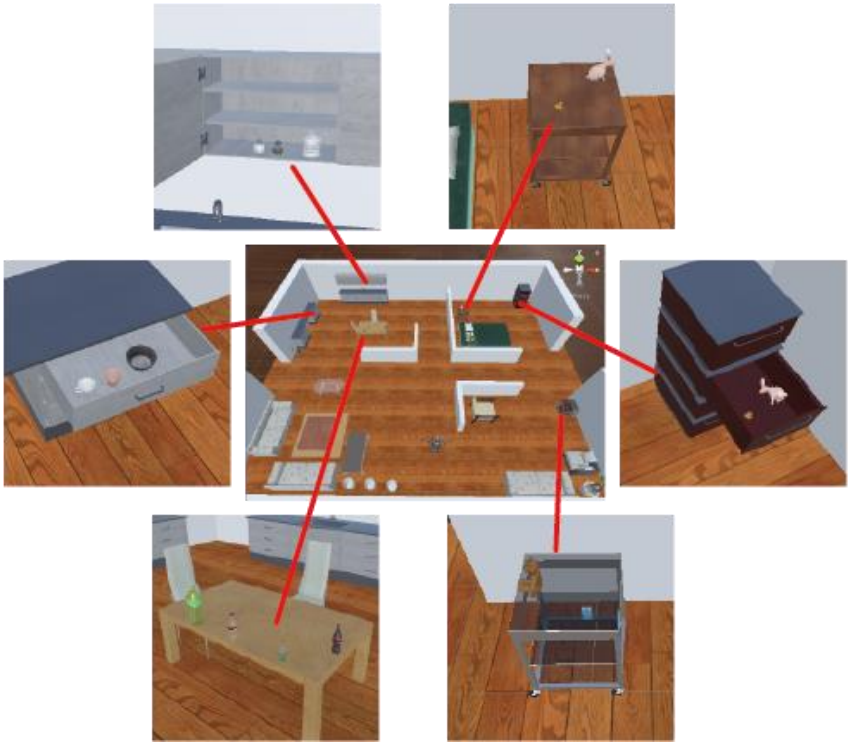

Figure 6.2: Example Environment for the Competitive Challenge

[Object to Manipulate]

- The 3D models of the objects to grasp and the labels (ID) will be released in advance.
- The position of the objects placed in the environment for the competitive challenge will be unknown.
- The position, quantity, and type of objects changes for each attempt.
- The same object may also be placed in the environment.
- The position of each object in the environment for the competitive challenge is sent to the robot at the start of the task as the *task_info* message. However, this message does not include information such as where the object is located in/on which piece of furniture.
- The object name is in the “objectID_xx (XX are numerical figures)” format.
    - Examples: petbottle_2l_full_c01_01, babytoy_duck_01, babytoy_duck_02, etc.

[Protocol of Task_info Message]

- Task_info includes the information below as ROS messages for notification.
    - Environment_id: Environment ID (same as the environment Prefab name in Unity)
    - Target_object: ID and position of objects to find
    - Target_object: Object IDs and positional layout for objects other than the object to find

[Regarding Robot Instructions]

- The vocal instructions are communicated verbally to the demonstrator (uses SAPI).
- The open dialogue between the test subject and robot shall not include.
- The test subject can request the vocal instructions at any time by using the button input on the Oculus Touch.
- The robot can speak the vocal instruction at any time.

[Regarding the Demonstrator]

- The demonstrator is a volunteer who is a member of the audience or team member not participating in this challenge.
- The demonstrator is required to be someone who has no knowledge about the environment.

- Demonstrators will learn the operational procedures to move, grasp objects, as well as open and close doors and drawers in a test environment in advance.
- The demonstrator starts from a place where they cannot look over the environment.

[Regarding Devices]
- The competitive challenge uses the Oculus Rift & Touch.

[Regarding the Implementation Procedure]
- A line of multiple teams will compete (Demonstrators will not be able to know instructions in advance through attempts for other teams)

[Voice Output to Demonstrators]
- The voice generated by SAPI is output to demonstrators.
  - The voices of other teams cannot be heard by demonstrators.
- SAPI is executed on the computer prepared by the Competition committee.

[Time Limits for the Competitive Challenge]
- 90 min
  (5 min for each task x 12 tasks) + 30 min for explanation and practice for test subjects

[Regarding Scoring]

Action	**Score**
Grasp the target object	*20*
Required time until the demonstrator grasp the target object ($\frac{150-required\_time\ [second]}{150\ [second]}\times$)	*30*
Grasp the wrong object (each time until the target object is grasped)	*-5*
Place the target object in/on the destination	*20*
Required time from the grasp of the target object until the demonstrator places the target object in/on the destination ($\frac{150-required\_time\ [second]}{150\ [second]}\times$)	*30*
Collision of the robot (each time)	*-10*
Highest number of points	***100***

Test subjects will understand the scoring in advance.

6.2. Usage Method for Sample Program

Using the Unity-side Sample
1. Download the Unity Project from Git provided by the competition in an arbitrary directory.
2. Start Unity and open the Unity Project that was cloned.
3. Set the ROS-side IP. (see Section 1)
4. Open a Unity scene Assets/Competition/HumanNavi/HumanNaviSample.unity at the bottom-left of the Unity Editor.
5. Execute the scene.

Using ROS-side Sample
6. Execute the command below to launch the sample program.
   $ roslaunch rosbridge_server rosbridge_websocket.launch

$ rosrun human_navigation human_navation

Customizing the Environment for the Competitive Challenge

- Environment for the Competitive Challenge
  - The sample of the environment for the competitive challenge is the prefab below. Assets/Competition/HumanNavi/Prefabs/Environments/Environment_01.prefab, etc.
  - The prefab environment is made up of the environment (e.g.: SIGVerseHouse_01), object list (e.g.: TargetObjects), and Location.
    - Environment_01
      ├SIGVerseHouse_01
      └ …
      ├TargetObjects
      │ ├babytoy_duck_01
      │ └ ...
      └Location
      ├cabinet_A_lower_01
      └ ...
  - The “Graspables” tag is set for each object (confirmation available in top-right of editor). Objects set with the “Graspables” tag in each prefab environment are listed and provided on the ROS side.
- Settings File for the Competitive Challenge
  - The settings file for the competitive challenge is the file below. humannavi_unity/SIGVerseConfig/HumanNavi/HumanNaviConfig.json
  - Descriptive Example of Config File (Excerpts)

```
{
  "isScoreFileUsed": false,
  "maxNumberOfTrials": 12,
  "taskInfo":[
    {"environment":"Environment_01","target":"petbottle_2l_full_c01_01"},
    {"environment":"Environment_02","target":"babytoy_duck_01"},
    ...
}
```

    - maxNumberOfTrials: Number of attempts (this element number needs to be higher than the number of attempts in taskInfo)
    - Environment: Name of environment prefab Environment prefab needs to be registered in “Environment Prefabs” of the Human Navi Moderator (Script). This script is attached on the HumanNaviModerator in the scene. Change “Size” and drag to the prefab to Element xx (e.g. Element 0) in the “Environment Prefabs”.
    - Target: The object to manipulate can be changed by replacing the object name (ID) of the graspable object in the prefab environment.
    - Destination: Name of location The destination can be changed by replacing the object name (ID) of the location in the prefab environment.
- Creating a Unique Environment
  - Editing the Environment
    - Drag the prefab environment to the hierarchy at the top-left of the editor to display the environment in the scene.
    - Please change the layout and quantity of furniture and objects as

necessary.
        - The “Graspable” tag needs to be set to objects. Furthermore, the object name needs to assign a proper name in the “ObjectID_xx (xx are numerical figures) format.
    - Creating a Prefab from the Environment Created
        - Open Competition/HumanNavi/Prefabs/Environments from Assets at the bottom-left of the screen.
        - Drag the environment that was created in the hierarchy (e.g.: Environment_xx) to the above Environments folder.
    - Registering a Moderator of the Prefab Environment
        - Select Hierarchy/HumanNaviModerator.
        - Select the Environment Prefabs in Moderator (Script) at the bottom-right of the editor.
        - Change the size (number of elements), and then drag the prefab to Element.
- Changing the Time Limits for the Competitive Challenge
    - Open HumanNaviMenu in Hierarchy.
    - Change Time Limit of Score Manager (Script) (by seconds).

[Topic List]

No	Topic Name	Direction	Message Type	Description
1	/humannavi/message/to_robot	SIGVerse→	human_navigation/HumanNaviMsg	task message from moderator to robot controller
2	/humannavi/message/to_moderator	ROS→	human_navigation/HumanNaviMsg	reaction message to moderator
3	/humannavi/message/task_info	SIGVerse→	human_navigation/TaskInfo	task information including environment_id, target object, list of the other objects, and destination
4	/humannavi/message/guidance_message	ROS→	std_msg/String	sentences for guidance

Note: SIGVerse→: SIGVerse to User's ROS Node, ROS→: User's ROS Node to SIGVerse

7. Final

[Time Limits for the Competitive Challenge]

- Each 15 min ×*n* teams + α

[Flow of the Competitive Challenge]

- 10 min
  - Preparations for demonstration presentation
  - Demonstration Presentation
- 5 min
  - Question and answers
  - Take down of demonstration

[Note]

- This competitive challenge does not take deductions for restricted items or for the use of special functions.

[Scoring]

- A judging panel with expert knowledge will award points based on the evaluation criteria below.
  - Creativity/presentation of the story
  - Effectiveness of interaction between the person and robot
  - Diversity/universality of system integration
  - Difficulty/completeness of performance
  - Relevance/practicality in daily life