Figure 6. The Conversational *Agent* was responsible for HRI and ensured continuous responsiveness to user input. The Robot Control *Agent* was focused solely on the execution of the mission. The mission was defined by a human prompt (e.g. “Navigate to the chair”). Its execution was performed through ROS 2 based tooling using actions. The *Agent* was also responsible for deciding when to report completion or failure to the user.

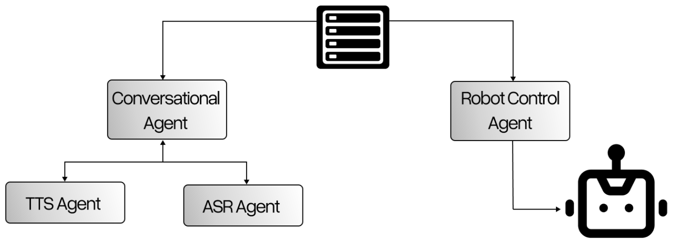

Fig. 6: Multi-agent architecture for S2S and navigation

Conclusions The Robot Control *Agent* successfully achieved navigation and object detection capabilities using the ROS 2 nav2 stack and the open-set detection tools. The system was able to handle commands such as navigating to specified locations or positioning itself relative to environmental objects. The digital twin environment proved invaluable for rapid prototyping, and the subsequent transfer to the real robot confirmed the robustness of the design. In general, the MAS-based setup provided a balanced approach between efficient mission execution and effective human-robot interaction. The framework proved itself to be flexible enough for testing different complex agent setups, where the user can examine the influence of each pipeline component.

Challenges Several challenges were encountered during system integration. Difficulties were experienced in achieving a clear understanding of agent embodiment, and issues were observed in the multi-agent architecture. In particular, inconsistencies in the understanding of embodiment were occasionally observed between the two LLM-based *Agents*. It has also been noted, that the decision to make LLM-based *Agents* responsible for synchronization of information regarding the state of the mission led to some inconsistencies. Similar problems were also encountered with long-running background tasks executed by the robotic stack. This could be addressed by incorporating rule-based synchronization mechanisms in future work. Furthermore, while Robot Control *Agent* could use the navigation stack successfully, error handling and mission success detection were inconsistent.

3.2 Manipulation: Robotic arm

Overview An *Agent* controlling a robotic arm was deployed in O3DE, where it received a task defined using natural language. *Agent's* perception was limited to a camera stream. The received images were processed using an open-set segmentation model [18], which provided additional information about the view. The observed environment included a desk with colorful blocks and vegetables. The interaction with the environment was performed with *Tools* utilizing MoveIt 2 [2] for motion planning and execution.

Experiments To evaluate its performance, the *Agent* was tested in three key manipulation tasks:

1. Sorting Objects – classifying objects before placing them in corresponding groups (Figure 7);
2. Stacking Items – building stable stacks by placing objects on each other;
3. Object Replacement – swapping pairs of objects, strategically using an intermediate position to prevent collisions and ensure proper placement.

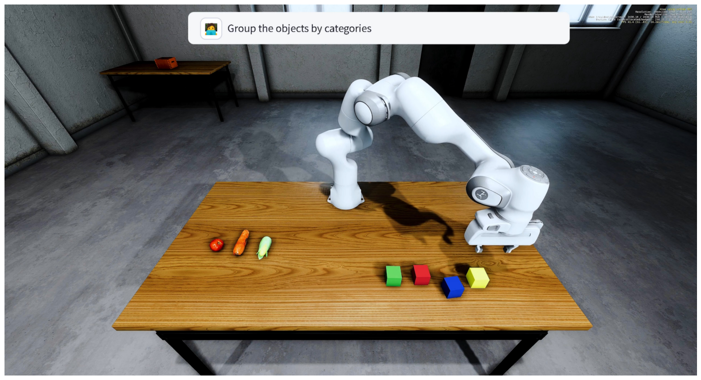

Fig. 7: Scene setup after a sorting task is completed

Various cloud-based multimodal LLMs (GPT-4o, GPT-4o mini, Claude 3.5 Sonnet) were tested as reasoning engines. They were tasked with the interpretation of scene data, planning movement sequences, and verifying task success.

Challenges Although success has often been observed in these tasks, *Agents* repeatedly faced the same difficulties, particularly in spatial reasoning and self-correction. When stacking (task 2), the *Agent* often tried to place objects "inside" each other rather than on top, failing to account for physical boundaries. A similar issue occurred during swapping objects (task 3) – the *Agent* tried to place the moved object directly in the place of the second object, without considering an intermediate position. Despite explicit instructions to verify the completion using camera images, the *Agent* frequently misjudged its success, assuming that the tasks were completed correctly even when errors were present. These problems were observed in multiple LLMs, suggesting general limitations in spatial reasoning, understanding boundaries, and object interactions.

Results The *Agent* succeeded in basic manipulation tasks. However, it frequently failed on tasks that require spatial reasoning and action sequencing. Stacking and object replacement errors were common, as the *Agent* lacked an intuitive grasp of the physical constraints. The *Agent* frequently failed to recognize its mistakes due to poor image understanding within the simulation. Most of the LLMs tested were able to complete the assigned tasks with adequate prompt engineering. However, in many cases, the instructions had to be extremely explicit, essentially guiding the LLM step-by-step through each action. Without such detailed instructions, the *Agent* frequently struggled to perform an effective sequence of movements. This highlights a key limitation: while multimodal LLMs can perform complex tasks, their ability to independently reason about spatial interactions remains limited.

3.3 Agriculture: Handling edge-cases

Fig. 8: Tractor's image used for enhancing Agent's embodiment

Overview An autonomous tractor was deployed in O3DE, a ROS 2-enabled simulation environment. The environment was designed for high-fidelity real-time simulation of an orchard with trees. The *Agent* was specifically tasked with handling unexpected situations in an agricultural setting.

The tractor was equipped with a rule-based autonomous system. When that system detected an anomaly, the tractor's control was transferred to the *Agent*.