A Tree-Based World Model for Reducing System Complexity in Autonomous Mobile Manipulation

Ryo Sakagami[1], Andreas Dömel[1], Peter Lehner[1], Sebastian Riedel[1,2], Sebastian G. Brunner[1,2], Alin Albu-Schäffer[1], and Freek Stulp[1]

***Abstract*—Mobile manipulation tasks in unstructured environments remain challenging for autonomous robots. The need to employ a diverse set of software and hardware components to solve the various subtasks inevitably increases system complexity. Knowledge exchange among such diverse components renders them highly coupled, reduces communication efficiency, and makes the knowledge less accessible. To overcome these challenges, we propose AIMM-WM, a central world model as a single source of truth having an abstracted geometric tree structure. Despite its concise, efficient state representation, AIMM-WM is able to provide a wide range of information from low-level geometries to highly abstracted symbols and is interfaced with diverse components for navigation, motion planning, perception, decision-making, and mission control. We evaluate the performance of AIMM-WM from the real use case of our Lightweight Rover Unit during the four-week Moon-analogue demo mission on Mt. Etna, Italy.**

***Index Terms*—Software Architecture for Robotic and Automation; Mobile Manipulation**

I. Introduction

EVEN with the major advancements in robotic hardware and artificial intelligence methods, mobile manipulation in unstructured environements – from homes to extraterrestrial surfaces – remains challenging. We consider the main reason for this to be the inevitable system complexity. As no single component can solve the entire mobile manipulation problem, mobile manipulation platforms inevitably have a variety of hardware and software components.

This complexity is exemplified by the four-week ARCHES demo mission we conducted on Mt. Etna, Italy [1], [2], illustrated in Fig. 1. In this Moon-analogue environment, we demonstrated the successful completion of autonomous, collaborative planetary exploration tasks. The mission involved a heterogeneous team of 3 robots and the preparations involved 70 researchers. The Lightweight Rover Unit 2 (LRU2) [3] alone already has a high system complexity. It is equipped with stereo and color cameras on a pan-tilt unit, and a manipulator with a docking interface for tool exchange and box manipulation. In total, 89 software components were run to solve diverse subtasks such as platform and arm control, navigation, mapping, localization, object detection, scene analysis, motion planning, decision-making, and mission control.

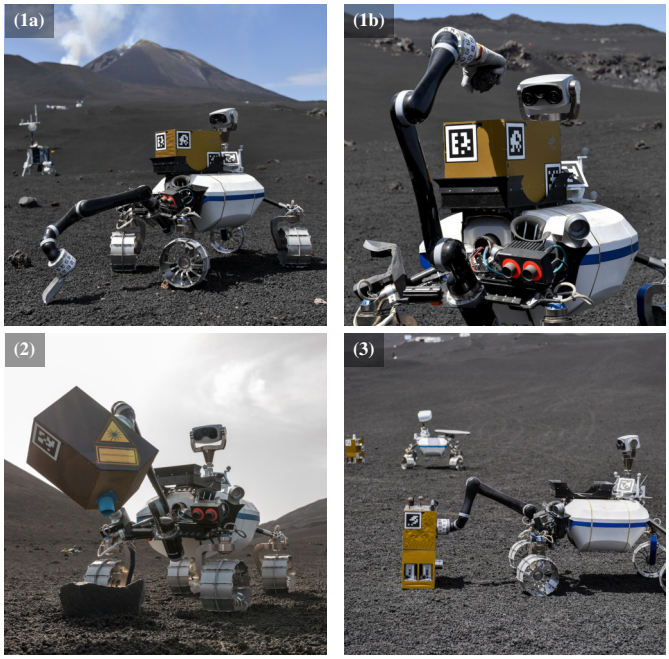


Fig. 1: Snapshots of autonomous task execution by our LRU2 during the ARCHES demo mission; (1a) sand sample collection with a shovel; (1b) stone sample collection with a robotic hand; (2) LIBS measurements; (3) a network of LOFAR boxes deployment.

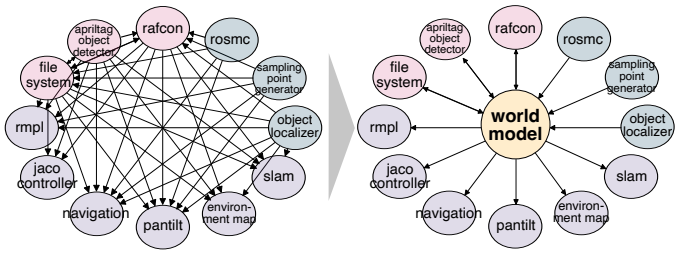


Fig. 2: AIMM-WM reduces data coupling of components within our LRU2 and provides a scalable and efficient system architecture.

To address such system complexity, this letter proposes a central robotic world model [4] for autonomous intelligent mobile manipulators (AIMM-WM). The first contribution of AIMM-WM is to reduce data coupling of components within

Manuscript received: December 21, 2023; Revised March 21, 2024; Accepted May 23, 2024.

This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Helmholtz Association project iFOODis (contract number KA2-HSC-06) and by the European Union's Horizon Europe research and innovation framework project IntelliMan (grant agreement No. 101070136).

[1]R. Sakagami, A. Dömel, P. Lehner, A. Albu-Schäffer, and F. Stulp are with the Inst. of Robotics and Mechatronics, German Aerospace Center (DLR), Germany {firstname.lastname@dlr.de}. [2] S. Riedel and S. G. Brunner are with Agile Robots SE and were with DLR at the time of development of AIMM-WM {firstname.lastname@agile-robots.com}.

Digital Object Identifier (DOI): see top of this page.

a system by providing a central world representation, as illustrated in Fig. 2. Data coupling is a type of coupling in which output from one software module serves as input to another module [5]. With AIMM-WM, components do not need to communicate with every other component to maintain world knowledge by themselves.

Secondly, AIMM-WM makes a wide range of knowledge accessible to different components, from low-level geometric information required for motion planners to high-level, abstracted view of the world for decision-making components. In comparison with a pure transformation representation (e. g. ROS TF [6]) or a physical, 3D representation (e. g. Universal Scene Description [7]), AIMM-WM represents not only concrete objects but also abstract ones (e. g. grasps and storages) to make the state rich enough to support interfaces with diverse components.

One of the approaches to achieve these two features, i. e., being central and providing diverse information, is to store heterogeneous data from sensor streams to symbols, e. g. as KnowRob [8], [9] and its extention [10], [11] does. However, this potentially makes the state also heterogeneous, enormous, and redundant. The third contribution of AIMM-WM is thus to keep the state representation concise in a homogeneous tree structure. Nodes represent objects from different abstraction levels while edges represent only geometric transformations. By taking advantage of the tree topology, AIMM-WM can represent high-level symbolic knowledge such as physical dependencies, which are critical for object manipulation. The tree structure also enables efficient integration of geometrical pose updates by localization.

AIMM-WM has been used in our heterogeneous robots in industrial ([12], [13]) and planetary exploration ([14]) domains. This letter abstracts the requirements from these specific applications and describes the whole concepts generically applicable. Furthermore, we evaluate the performance of AIMM-WM from the aforementioned Moon-analogue demo mission, where LRU2 performed long-term autonomous sample collection, laser-induced breakdown spectrometer (LIBS) analyses [15], and deployment of the low-frequency radio array (LOFAR) boxes [16] (see Fig. 1).

The rest of this letter is structured as follows. We first present a concrete use case of AIMM-WM in our LRU2 during the Moon-analogue mission in Section II. Design of AIMM-WM's state representation and its tell/ask interfaces are presented in Section III and Section IV, respectively. AIMM-WM is evaluated in Section VI and compared with related work in Section VII. We conclude with Section VIII.

II. Use Case

To motivate the need for a central representation that stores heterogeneous information, we now first describe the LRU2 software components that will interact with the world model.

In our LRU2, there are 12 software components that interface with AIMM-WM. As is listed and described in Table I, LRU2 employs four different perception components [17], [18], [15], the motion planner RMPL [14], [15], the SLAM system [19], [20], the controllers for the wheels and the manipulator, the state machine execution framework RAFCON [21], and the high-level mission control ROSMC [22].

These components require heterogeneous types of data regarding the world. For instance, the object detector component based on AprilTag fiducial markers [23] queries a geometric structure of how multiple AprilTags are associated to a single concrete object. RMPL requires all concrete objects near the robot to construct a collision environment. RAFCON asks options and conditions for decision making and tells back expected results of decided actions.

As is described in Section I and illustrated in Fig. 1, LRU2 participated in the four-week ARCHES demo mission campaign at a Moon-analogue site on Mt. Etna, Italy [1], [2]. The missions LRU2 conducted were 1) to collect sand and stone samples from unvisited locations, 2) to perform LIBS sample analyses [15], and 3) to deploy the network of the LOFAR boxes in rough terrain [16]. The missions took approximately 2.5 hours, 1 hour, and 3 hours, respectively, and were executed in a region of $1500\,\mathrm{m}^2$ only partially known in advance.

One of the main technological challenges of the mission is that the different software components require different perspectives on the real world. For example, where the tools are located (in the holder or at the end-effector) affects the controller (to compensate the load), the motion planning (to find a collision-free motion), and also the decision making (to attach a suitable tool/box to the end-effector for the current task). AIMM-WM, by providing a central world state in the robot with diverse interfaces, plays a key role within the system architecture to address this technological challenge.

III. State Representation

We define a world model (WM) as a software component in a robot that reflects the real world and shares information about the world with other components. As shown in Fig. 3, a WM is segregated by a boundary and internally consists of a state (which represents the world), and operations (which are used to provide/receive information between the state and the boundary). The boundary has two types of interfaces: "tell" to provide information to the WM and "ask" to query information from the WM. For more detailed descriptions of the definition, refer to [4].

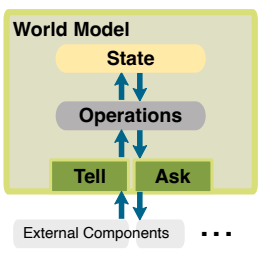


Fig. 3: Graphical representation of a world model in general [4].

AIMM-WM represents the state as a tree structure whose nodes represent concrete objects (e.g. a LOFAR box) and abstract objects (e.g. an appropriate grasp for a concrete object) and whose edges represent a 6-DOF transformation

TABLE I: The software components on LRU2 that communicate with AIMM-WM to perform the mission of sample collection, LIBS sample analyses, and LOFAR box deployments.

Components	Description	Data types	Examples of data exchanged with WM
AprilTag object detector	Calculates poses of objects annotated with AprilTags, based on a camera image and the geometric structure of the AprilTags.	Generic	Tells poses of objects
		Visual feature	Asks a size and a geometric structure of AprilTags composing an object
Object localizer [17], [18]	Detects unknown objects on the terrain surface and calculates the poses of their center.	File	Tells an instance of the detected object type from a configuration file
ROSMC [22]	Sepcifies and controls a high-level mission for a heterogeneous robotic team.	File	Tells a new scene at the poses of points of interest
File system	Provides an initial world state as a file and stores the current state dumped into a file	File	Tells an initial state and asks the current state
Environment Map [15]	Models the surrounding area except for known objects as a collision map	Physical	Asks a region of interest for mapping and existing objects inside of it
SLAM [19], [20]	Creates a geometrical map for navigation and exploration and estimates the current robot pose	Physical	Asks if landmark objects with AprilTags are being or have been manipulated by the robot. [a]
RMPL [14], [15]	Plans paths of the Jaco2 manipulator to avoid collisions with the environment and the robot	Physical	Asks all concrete objects in the current scene to construct a collision environment
Sampling point generator [15]	Computes poses on the surface of stones usable for LIBS measurements	Generic	Tells poses of sampling points with respect to the stones
Pan/tilt unit	Controls the pan and tilt joints, whose end-effector is equipped with the navigation cameras	Generic	Asks poses of objects for the robot to look at
Navigation	Controls the wheels to drive the robot to a goal pose	Generic	Asks navigation location poses
Jaco controller	Controls the Jaco2 manipulator to reach a targeted joint configuration	Physical	Ask the total mass and the center of gravity of objects attached to the end effector
RAFCON [21]	Makes decisions of behaviors using state machines to achieve local autonomy	Generic	Asks properties of nodes; tells new poses and properties; tells to reassign a node to a new parent node, etc.
		Decision-making	Asks available storages, suitable grasps/approaches, etc; tells to update the storage availability, etc.

[a]Information is told/asked not by synchronizing robots' poses but via common landmark objects.

between two objects. A schematic example of the state is shown in Fig. 4.

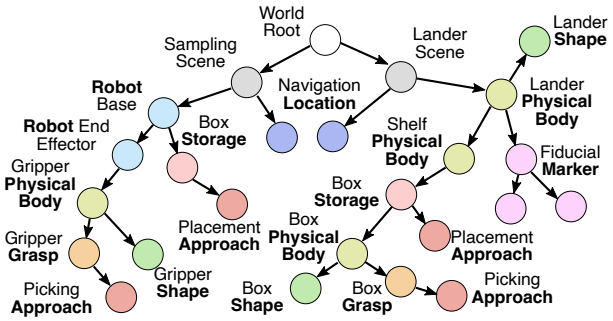


Fig. 4: Schematic example of AIMM-WM's state representation.

A. Node Types

Both concrete and abstract objects are represented as nodes of the tree. We identified the following object types to represent different properties of the world based on our previous review [4] and the requirements of our robots.

1) Concrete Objects:

a) Robot: This type of object represents the robot itself. The concept here is to represent in an abstracted manner how the robot relates itself to the surrounding environment. Therefore, every robotic part that has physical interaction with the environment should be defined as an object with this `Robot` type.

For a mobile manipulator, it is typical to define a robot's base and a manipulator's end effector as independent objects. In this manner, all objects related to the robots base, e. g., objects placed on the platform, can be attached as children of the robot base node. Objects grasped by a manipulator can be attached as children of the end effector node, since they have a physical dependency on the manipulator. Relocation of the robot base can be represented by updating the relation from the robot base to its parent.

Note that providing a detailed robotic model is not the scope of this `Robot` type since the motivation of AIMM-WM is to reduce complexity of communication among diverse components. For instance, the kinematics and dynamics of the manipulator are only relevant for the controllers, and thus it should be represented in a separate robot model module. If necessary, AIMM-WM can still interface to such an external knowledge representation and update the relation between `Robot` nodes, e. g. to continually update the pose relation of the robot base node and the end effector node.

b) Fiducial Marker: Annotating concrete objects with fiducial markers enables accurate and robust localization [24]. The object type `Fiducial Marker` is a specialized one for modelling a fiducial marker pose as well as its size. By having the `Fiducial Marker` objects as children of a `Physical Body` object (see below), it is possible to encode the geometric

relation between the fiducial markers to the concrete object itself.

c) Physical Body: This type represents concrete objects that do not compose the robot. `Physical body` objects describe the physical properties, e. g., a mass, an inertia, and a material. Therefore, this type is close to the common understanding of the term "object". Note that the `Physical Body` object can have one or more children with the `Shape` type (see below) to describe the geometrical information.

2) Abstract Objects:

a) Shape: The objects with this type are abstract in a sense that they define only a geometric shape at a pose. The form of the shape is flexible; it can be primitive shapes such as boxes or cylinders, a voxel, or complex meshes[1].

Due to its simple definition, this type can be used for different purposes. For instance, it is useful for defining a region of interest for a perception component, for modelling safety zones, for defining goal regions of delivery tasks, and for specifying collision obstacles for the motion planner.

b) Grasp: The `Grasp` objects represent contact information for grasping a concrete object. A grasp has a geometric relation to its concrete object and thus is modelled as a child node of the `Physical Body` objects. Besides this geometric relation, a grasp stores various process parameters such as a gripper width and force.

c) Storage: The `Storage` objects represent contact information for placing a concrete object on another one. Similar to the grasp, a storage has a geometric relation to a concrete object that can sustain another concrete object. Therefore, a storage is modelled as a child node of the `Physical Body` or `Robot` objects.

d) Manipulator Approach: This type of objects represents poses where the manipulator should reach. An approach has a geometric relation to its grasps and storages, and thus is modelled as a child node of the `Grasp` objects as well as the `Storage` objects.

e) Navigation Location: Similar to the `Manipulator Approach`, this type of objects represents poses where the robot base should reach.

B. Tree Structure

The tree structure is employed due to the following advantages.

a) Physical Dependencies: Mobile manipulation tasks by nature involves activities of picking and placing objects. Depending on physical contacts – a box on a shelf, or in the gripper – objects move together. Physical contact introduces invariants w. r. t. relative poses.

Representing such physical dependencies between objects in a WM provides substantial advantages for mobile manipulation tasks. Utilizing the parent-child relations in the tree structure is a concise approach to do so. In the example above, the box node can be represented as a child node of the shelf before picking, and then the box node can be reassigned to the gripper node at the end-effector after picking.

[1]Only a pointer, e. g., a path to a mesh file, is stored in the node's property.

The advantage of the tree structure against the physics simulation approaches [25], [26], [10] is that the model does not require parameters difficult to estimate in reality, e. g., friction coefficients and exactly correct meshes, nor heavy computational effort as simulators do. The disadvantage is that cyclic physical dependencies (e. g. when an object stands over the edge of different objects) cannot be directly represented in the tree. Introducing an abstract common parent node to group the object nodes having physical dependencies is one of the practical solution to overcome this conceptual limitation.

b) Local Reference: With the tree structure, every object has exactly one parent object with a specified transformation. Changing this transformation leads implicitly to change poses of all children objects with respect to the world coordinate frame. This concept has been employed in many other world model approaches such as ROS TF [6] and Robot Scene Graph [27] and we find it especially useful for localizing objects and a robot. For example, when a robot moves its platform, the only transformation which needs to be updated is the one from the robot node to its parent, regardless of the objects that the robot is carrying. If a list representation is used, all the objects on the robot must update their pose with respect to the world coordinate frame.

c) Scene Concept: For mobile manipulation tasks, there are many cases that a robot needs to consider only its local workspace. We call such a local workspace a *scene*. With the tree structure, scenes are easily represented by dividing the entire tree into sub-trees and thus a search space can be reduced to be inside a scene where the robot belongs to. This is typically beneficial for interfacing with a motion planner, since it performs more efficient by neglecting geometrical constraints far away from the manipulator. Other components also take great advantage of the scene concept if a robot works in an environment with a large number of objects, e. g. in a manufacturing factory.

Although the local scene can also be extracted by calculating all the objects within a certain distance, utilizing the tree structure is more advantageous in efficiency for performance and in transparency for human understanding.

IV. Tell and Ask Interfaces

As is highlighted in Section II, an autonomous mobile manipulation robot requires various components for different subtasks, and they require information about the world from a different perspective. Thus the challenge for having a central WM is to have an extendable boundary so that heterogeneous, wide-range information about the world can be provided/requested. Since some components have a common interest in a certain aspect of the world (e. g. only physics-related information), we group domain-specific queries into separate interfaces to make the system architecture well-structured and extensible. In this section, we describe the following five types of tell/ask interfaces required for AIMM-WM.

1) Generic Interface: Through the generic interface, components can tell/ask generically useful data to/from the world model, such as to add, remove, change, and get nodes and edges of the state. This interface also provides data containing

the information of the state's tree structure as well. For instance, asking pairs/triplets of parent and child nodes satisfying a certain condition is generically useful for abstracting physical dependencies. We call them pair/triplet queries and include them in this interface.

2) File Interface: Through the file interface, components tell/ask a state in a human readable file such as YAML. This interface is used 1) to integrate prior knowledge such as a topological/geometric structure of the entire world or sub-parts composing it, and 2) to store the state into a file for logging and debugging purposes.

3) Physical Interface: This interface provides physical and geometrical data from the world model. This interface extracts concrete objects from the state and builds a 3D collision representation, which is used e. g. by motion planners.

As another example, this interface provides the overall mass and the center of gravity of an object taking all children object nodes into consideration. This is especially useful for the impedance controllers of manipulators, since an estimation of the weight of workpieces at the end effector is essential for them.

4) Perception Interface: Through this interface, perception components tell/ask data necessary and useful for them to run their algorithms and to provide new information regarding the world. Components based on fiducial markers [24] require geometric information of markers attached to objects. Other feature-based perception components need the model of features of objects, i. e. how these features are related to each other. The interface extracts such visual features and their relation stored in the state and provides them to the various perception modules.

5) Decision-Making Interface: Through this interface, decision-making components ask from AIMM-WM a list of certain objects or specific conditions so that the robot can decide its behavior, and tell to AIMM-WM the expected results of the decision.

The list of objects are often used for iterating certain skills (e. g. to transport all cups) or for error handling (e. g. to use other grasps if one has lead to a failure). For example, as shown in Fig. 5, this interface uses specialized pair/triplet queries to provide all the empty storages in the scene (1), all the storages on the robot base (2), and all the storages on a physical body object having fiducial markers (3).

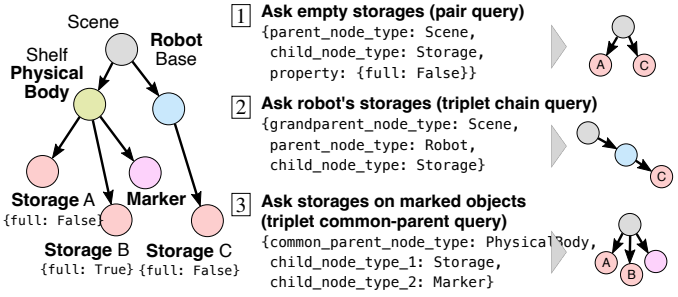


Fig. 5: Examples of pair and triplet queries for decision making.

A typical example of certain conditions relevant for picking tasks is to check where the targeted object is located. Depending on the type of the sustaining object, the robot needs to choose a proper strategy and tool for manipulation.

V. Implementation

The implementation of AIMM-WM is shown in the software architecture in Fig. 6, which is according to our WM template shown in Fig. 3. AIMM-WM consists of a database and five processes where external components tell/ask information to/from AIMM-WM.

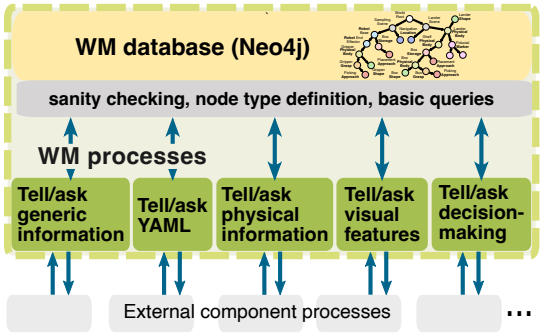


Fig. 6: Software architecture of AIMM-WM.

The database process maintains the state of AIMM-WM. We employed a Neo4j graph database because it provides 1) data persistence essential for long-term autonomy, 2) reliable data handling (e. g. to manage race conditions), 3) browser-based visualization tools, and 4) maintainability due to a wide-range of users and an active open-source community.

Since Neo4j is a generic graph database, it is tailored into the state representation described in Section III by implementing a Python library. It ensures that the internal structure of the database is a tree, defines the different node types as classes, and associates transformation information with edges. Furthermore, sanity checks are executed after each writing operation so that the state satisfies certain conditions, e. g., that no edges would lead to a cycle. When the checks fail, the database safely rollbacks the operation.

The tell and ask interfaces are implemented by five processes for each of the five data types described in Section IV. To increase robustness, these interfaces are implemented as independent client processes on the robotic operating system. In case of unexpected errors during runtime, only the corresponding processes can be restarted without affecting the database nor other running processes.

Another advantage of making the tell/ask processes independent is to keep the runtime environment minimal. Depending on the communication protocol used by the external components, the tell/ask processes require a different runtime environment as well. Although we used ROS topics and services to communicate with external components, this can be flexibly adapted per process to suit system requirements.

VI. Experiments

In this section, we evaluate the performance of AIMM-WM utilizing data from the ARCHES demo mission, which took place at a Moon-analogue field on Mt. Etna in Italy [1], [2]. We logged the initial world state as well as the sequence of

ROS service calls on the tell/ask interfaces. Recording the inputs/outputs of the services as well as the name of the service caller component enables us to reconstruct and analyze the world state at each execution step of the mission.

The following features of AIMM-WM are evaluated: 1) providing a central WM within the robot architecture, 2) providing diverse tell/ask interfaces, 3) having the tree structure for local reference, and 4) the inherent ability of representing a scene in the tree structure.

A. Experiment 1: Data Coupling Reduction

One of the motivations of having a WM was to reduce the data couplings among components, as is highlighted in Fig. 2. To quantify the impact of introducing a WM, we analyzed the system architecture of LRU2 with and without AIMM-WM, and compared the number of data couplings (which we denote by c) w. r. t. the number of software components exchanging world information (which we denote by n). The results are shown in Fig. 7.

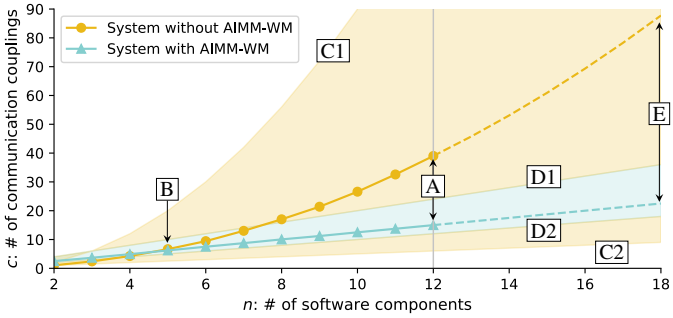


Fig. 7: Data couplings among software components in LRU2 with and without AIMM-WM.

As is listed in Table I, $n = 12$ in LRU2. We represented a system architecture as a graph, where a node represents a software component and a directed edge represents an information flow, as shown in the right diagram of Fig. 2. If we would have not employed AIMM-WM, each software component must have told information to all relevant ones, as shown in the left diagram of Fig. 2. We computed c as the total number of the directed edges in such a graph. In the actual LRU2 system c was 15, though the value would be 39 if AIMM-WM were not used (A). This means that the data coupling is improved by a factor of 2.6 by having AIMM-WM in LRU2.

Secondly, in order to analyze the case where the system were simpler, we removed nodes from the graph of Fig. 2 and computed c. Since c is affected by which node is removed, we randomly chose n_r nodes to be removed and computed the average of c of 200 times. The nodes were chosen so that no node is isolated because such a node does not tell nor ask any world information. This was iterated for n_r from 1 to 10, and as a result, the system with AIMM-WM outperforms if $n \geq 5$ (B).

Thirdly, we derived the upper/lower bound of c by utilizing the graph theory. Without AIMM-WM, the maximum of c is $n(n-1)$ when the system is represented as a complete directed graph (C1). On the other hand, the minimum of c is $n/2$ when each node of the system graph has only one outbound or inbound edge (C2). With AIMM-WM, each software component node can have edges only to/from AIMM-WM. Therefore, the minimum of c is n when each node has either outbound/inbound edge (D2) and the maximum of c is $2n$ when each node has both (D1).

Finally, we simulate a situation where LRU2 could have employed more components. With AIMM-WM, c can be modeled as a linear function of n because an additional software component increments c only by 1 or 2 to communicate with AIMM-WM. The slope was computed to be 1.25 based on the actual LRU2 system. Without AIMM-WM, c can be modeled as a quadratic function of n because an additional software component needs to communicate with some of the other existing components, which increases as n does. The quadratic coefficient was computed to be 0.27 for LRU2. As a result, if n were 18 in LRU2, there would have been four times more data couplings in the system without AIMM-WM than with AIMM-WM (E).

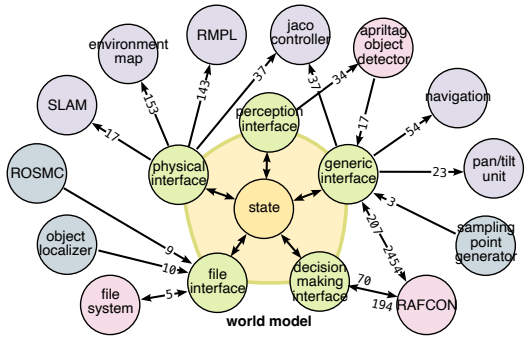
B. Experiment 2: Knowledge Accessibility

The second motivation for AIMM-WM was to make the knowledge about the world accessible to various different components from a single world model, instead of having multiple different world models synchronizing information between them. In Fig. 8a, we show the actual communication diagram with the five interfaces (see Section IV) and the components. The annotated numbers on the directed edges show the total number of ROS service calls hosted by the interface processes during the demo mission.

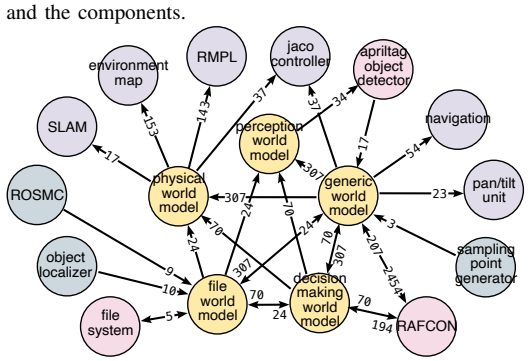
To quantify the advantage of having a single, central world model with diverse interfaces, we simulated a communication diagram if we would have employed five different world models (see Fig. 8b). In this simulated system architecture, each world model needs to synchronize the other world models whenever it receives new information. The expected number of such synchronization calls additionally required would be counted to 1604 in total, while in AIMM-WM the information was directly written into the state without synchronization 396 times. This means that providing diverse information from a single world model by AIMM-WM improved the efficiency by a factor of 4.05.

C. Experiment 3: Efficiency of Tree Structure

One of the reasons why we chose the tree structure as a state representation was to make pose representations among objects efficient by utilizing local references. To evaluate this, we compare the tree structure against a list structure, since some object-centric world models such as [28] employ the list structure. We assume that the list state consists of the same number of objects as the tree nodes of AIMM-WM, and each object has a pose information w. r. t. the global coordinate frame. By replaying the ROS service calls which were executed on LRU2 during the mission, we evaluate the computational effort if the list structure had been employed as a world state.



(a) Actual communication among the AIMM-WM interfaces and the components.



(b) Expected communication among the components if separate world models were employed.

Fig. 8: Communication diagrams annotated with the number of interface calls during the ARCHES demo mission.

During the mission, the services to update the pose information and to reassign a node took place 23 times and 182 times, respectively, typically when LRU2 localized itself and picked/placed objects. If the list structure were employed, all the N children nodes under the manipulated node must have updated their pose information with respect to the global coordinate frame. Therefore, the list structure would have costed at least N-times more computation than the tree structure. The average value of N is calculated to be 22.05 by conducting the simulation using the log data of the operation calls. This means that the tree structure of AIMM-WM improved the operation efficiency by a factor of 22.05.

D. Experiment 4: Efficiency of Scene Concept

The last advantage of AIMM-WM is that its tree structure inherently allows a scene to be extracted as a subtree. This is advantageous in many contexts, and here we focus on the physical interface used by the motion planner for evaluation. The motion planner requires all the physical bodies and shapes in the environment, but only if they are in the current workspace. Therefore, the scene concept improves the performance by restricting the search space into a subset of the current entire state.

The query took place 143 times, and on average, when the motion planner queries, 109.89 nodes existed under the current scene in comparison with 325.90 in the entire state. This means that the search space to query relevant, concrete objects is reduced by a factor of 2.97 on average.

E. Summary of experiments

Table II summarizes the performance improvement by having AIMM-WM with respect to each feature.

TABLE II: Performance improvement factor achieved by AIMM-WM with respect to each of its feature.

Experiment / Feature	Benchmark method	Improvement factor
1. Central WM	Distributed states	2.60
2. Single WM	Multiple WMs	4.05
3. Tree representation	List representation	22.05
4. Scene concept	No scene concept	2.97

VII. Related Work

As is reviewed in [4], [29], [30], [31], the world model has been studied for more than five decades in robotics. After being employed in the first autonomous robot Shakey [32], different research domains in robotics have developed their concepts about the world model independently.

The navigation domain studies the world model to represent a map of the environment and a robot pose. The geometries of the environment are represented in various models [29] and estimated by simultaneous localization and mapping (SLAM) components [33]. Recent approaches are combined with neural networks to integrate semantic information as well [34].

In the object detection and manipulation domain, the world model can take a form of the object-centric knowledge base. This could contain shapes, grasps, approaches, and taxonomical labels identified by categorization [28], [35], which are utilized for hybrid planning and reasoning activities [36].

Being compared to all the domain-specific approaches above, our AIMM-WM addresses the *inter-domain* world modelling to cover the entire mobile manipulation tasks.

As related work of such mobile manipulation WMs, there are Robot Scene Graph (RSG) [27] and KnowRob [8], [9] extended with a simulator [10] and Universal Scene Description [7]. While RSG shares similar design decisions as AIMM-WM does (e. g. to employ an abstracted graph structure), the main focus of RSG was to process diverse data effectively in a central place by caching raw sensor data and intermediate results, rather than to interface with diverse components as AIMM-WM does.

KnowRob incorporates information of diverse abstraction levels from sensory data to high-level events and stores them into a heterogeneous set of databases with potentially redundant or inconsistent information. AIMM-WM in comparison aims to represent the world in a single, consistent state with as concise information as possible, but as rich enough as necessary to interface with heterogeneous components.

VIII. Conclusion

In this paper, we proposed AIMM-WM, a central world model for autonomous intelligent mobile manipulation robots. The main feature of this WM is to provide/receive a wide range of information to/from components required by the autonomous system, such as perception, navigation, motion

planning, and decision-making components. Despite the difference in the type of required information, AIMM-WM is able to cover all the use cases with the single, minimal, concise state represented as a tree.

AIMM-WM contributes to making complex robotic systems more robust and extensible by 1) providing a consistent world state for the entire system instead of synchronizing states among every component, 2) reducing the data couplings and improving the scalability of the system for integrating more software components, and 3) improving efficiency in telling/asking information to/from the state due to the tree structure composed of abstracted objects. Nevertheless, AIMM-WM as a central component increases communication latency due to an additional data transit, and thus low-level components requiring strict real-time data exchange should communicate directly.

In this article, we have highlighted the impact of using AIMM-WM on our LRU2 system during the ARCHES mission. However, we have also integrated AIMM-WM in our other heterogeneous robots such as the lander, the aerial drone, and another rover with science cameras [1] as well as our two different industrial robots [12], [13]. In our future work, we aim to use ontology [37] to represent the object types of AIMM-WM. We also plan to publish AIMM-WM with ROS2 support as open source.

REFERENCES

[1] M. J. Schuster *et al.*, "The ARCHES Space-Analogue Demonstration Mission: Towards Heterogeneous Teams of Autonomous Robots for Collaborative Scientific Sampling in Planetary Exploration," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, pp. 5315–5322, Oct 2020.

[2] A. Wedler *et al.*, "Finally! Insights into the ARCHES lunar planetary exploration analogue campaign on Etna in summer 2022," in *73rd International Astronautical Congress (IAC)*, 2022.

[3] M. J. Schuster *et al.*, "Towards autonomous planetary exploration: The lightweight rover unit (LRU), its success in the SpaceBotCamp challenge, and beyond," *Journal of Intelligent & Robotic Systems (JINT)*, Nov. 2017.

[4] R. Sakagami *et al.*, "Robotic world models – conceptualization, review, and engineering best practices," *Frontiers in Robotics and AI*, 2023.

[5] "ISO/IEC/IEEE international standard – systems and software engineering – vocabulary," *ISO/IEC/IEEE 24765:2017(E)*, pp. 1–541, 2017.

[6] T. Foote, "tf: The transform library," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, ser. Open-Source Software workshop, April 2013, pp. 1–6.

[7] M. A. Bolstad, "Large-scale cinematic visualization using universal scene description," in *2019 IEEE 9th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 2019, pp. 1–2.

[8] M. Tenorth *et al.*, "Knowrob: A knowledge processing infrastructure for cognition-enabled robots," *The International Journal of Robotics Research*, vol. 32, pp. 566–590, 2013.

[9] M. Beetz *et al.*, "Know Rob 2.0—a 2nd generation knowledge processing framework for cognition-enabled robotic agents," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 512–519.

[10] A. Haidu *et al.*, "Knowrobsim—game engine-enabled knowledge processing towards cognition-enabled robot control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4491–4498.

[11] G. H. Nguyen *et al.*, "Translating universal scene descriptions into knowledge graphs for robotic environment," 2023.

[12] A. Dömel *et al.*, "Toward fully autonomous mobile manipulation for industrial environments," *International Journal of Advanced Robotic Systems*, vol. 14, 2017.

[13] F. Steinmetz *et al.*, "Intuitive task-level programming by demonstration through semantic skill recognition," *IEEE Robotics and Automation Letters*, vol. 4, pp. 3742–3749, 2019.

[14] P. Lehner *et al.*, "Mobile manipulation for planetary exploration," in *IEEE Aerospace Conference*, March 2018, pp. 1–11.

[15] ——, "Mobile manipulation of a laser-induced breakdown spectrometer for planetary exploration," in *2023 IEEE Aerospace Conference*. IEEE, May 2023.

[16] E. Staudinger *et al.*, "Enabling distributed low radio frequency arrays – results of an analog campaign on Mt. Etna," in *IEEE Aerospace Conference*, 2023.

[17] M. Durner *et al.*, "Unknown object segmentation from stereo images," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 4823–4830.

[18] ——, "Autonomous rock instance segmentation for extra-terrestrial robotic missions," in *2023 IEEE Aerospace Conference*, 2023, pp. 01–14.

[19] M. J. Schuster *et al.*, "Distributed stereo vision-based 6d localization and mapping for multi-robot teams," *Journal of Field Robotics (JFR)*, 2018.

[20] M. J. Schuster, "Collaborative Localization and Mapping for Autonomous Planetary Exploration: Distributed Stereo Vision-Based 6D SLAM in GNSS-Denied Environments," Ph.D. dissertation, University of Bremen, 2019.

[21] S. G. Brunner *et al.*, "RAFCON: A graphical tool for engineering complex, robotic tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[22] R. Sakagami *et al.*, "ROSMC: A high-level mission operation framework for heterogeneous robotic teams," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5473–5479.

[23] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[24] M. Kalaitzakis *et al.*, "Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag, aruco and stag markers," *Journal of Intelligent & Robotic Systems*, vol. 101, pp. 1–26, 2021.

[25] A. S. Bauer *et al.*, "Probabilistic effect prediction through semantic augmentation and physical simulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 9278–9284.

[26] M. Neumann *et al.*, "URoboSim – an episodic simulation framework for prospective reasoning in robotic agents," 2020.

[27] S. Blumenthal *et al.*, "A scene graph based shared 3D world model for robotic applications," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 453–460.

[28] D. Leidner *et al.*, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, 2012, pp. 429–435.

[29] E. Angelopoulou *et al.*, "World model representations for mobile robots," in *Proceedings of the Intelligent Vehicles '92 Symposium*, 1992, pp. 293–297.

[30] C. Landsiedel *et al.*, "A review of spatial reasoning and interaction for real-world robotics," *Advanced Robotics*, vol. 31, pp. 222–242, 2017.

[31] A. Belkin *et al.*, "World modeling for autonomous systems," *Innovative information systems modelling techniques*, vol. 1, pp. 135–158, 2012.

[32] N. J. Nilsson, "Shakey the robot," AI Center, SRI International, Menlo Park, CA, USA, Tech. Rep., 1984.

[33] T. Taketomi *et al.*, "Visual slam algorithms: A survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, pp. 1–11, 2017.

[34] W. Chen *et al.*, "An overview on visual slam: From tradition to semantic," *Remote Sensing*, vol. 14, p. 3010, 2022.

[35] D. Leidner, "Cognitive Reasoning for Compliant Robot Manipulation," Ph.D. Thesis, Universität Bremen, Bremen, 2017.

[36] D. Leidner *et al.*, "Object-centered hybrid reasoning for whole-body mobile manipulation," in *2014 IEEE International conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 1828–1835.

[37] P. M. Schäfer *et al.*, "Flexible robotic assembly based on ontological representation of tasks, skills, and resources," in *18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*. International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 702–706.