

Collaborative Moonwalkers

Edward Chow, Thomas Lu
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
edward.chow@jpl.nasa.gov

Elliott Sadler, Neville Elieh Janvisloo,
Jared Carrillo, Bingbing Li
California State University Northridge
18111 Nordhoff St.
Northridge, CA 91330
bingbing.li@csun.edu

Kevin Payumo
University of Southern California
3551 Trousdale Pkwy
Los Angeles, CA 90007
kpayumo@usc.edu

Benjamin Hubler
University of Chicago
5801 S Ellis Ave
Chicago, IL 606377
benhubler@uchicago.edu

Gautier Bardi de Fourou
Mines Paris - PSL university
60 Bd Saint-Michel,
75272 Paris, France
gautier.bardidefourou@gmail.com

Olin Littlejohn
Purdue University
610 Purdue Mall
West Lafayette, IN 47907
olittlej@purdue.edu

Vanessa Hernandez-Cruz, Sophia Torrellas
Massachusetts Institute of Technology
77 Massachusetts Ave
Cambridge, MA 02139
storrell@mit.edu

Abstract—We present the results of an advanced concept research for a collective of robotic rovers named Moonwalkers that implement bio-inspired motivations, distributed intelligence, and integrated deep learning techniques to demonstrate autonomous cooperative goal-directed lunar exploration with minimum human interventions. By borrowing concepts from biology, the Moonwalker rovers are driven by a set of balanced motivations (ex. Energy, safety, operating temperature, achievements, respect) to accomplish mission goals while maintaining safety from potential dangers. Drawing from contemporary neuroscience and computer science research, the Moonwalker rovers are equipped with system 1 (low-level) and system 2 (high-level) cognitive functions based on a two-level memory architecture utilizing a pre-trained multi-modal model in conjunction with a graph knowledgebase, which leverages knowledge graph embeddings. This neuro-symbolic approach enables the rovers to learn from and be guided by human knowledge, facilitating zero-shot learning, collaborative planning, and reasoning like human explorers under uncertain conditions. We also describe the construction of a virtual simulation environment with high-fidelity visualizations and prototype rover agents. By addressing selected problems, we demonstrate that physics informed machine learning can enhance the effectiveness of the virtual lunar test environment.

TABLE OF CONTENTS

1. INTRODUCTION.....	1
2. COLLABORATIVE ROVER EXPLORATION ARCHITECTURE	2
3. CONSTRUCTION OF VIRTUAL ENVIRONMENT.....	7
4. DIGITAL TWIN - HIGH-FIDELITY PHYSICS INFORMED MODELING	10
5. SUMMARY	12
ACKNOWLEDGMENTS	12
REFERENCES	12
BIOGRAPHY	14

1. INTRODUCTION

The lunar South Pole-Aitken (SPA) basin represents a high priority site for future lunar exploration due to the potential presence of water ice and other volatiles in permanently shadowed regions, as well as insights into lunar evolution from ancient impact melt sheets [1]. However, the immense scale of the SPA basin, spanning approximately 2500 km in diameter and 13 km depth below the surrounding lunar highlands, presents a significant challenge for exploration by either crewed missions or single robotic assets. This necessitates the deployment of a community of robots capable of working collaboratively with astronauts and each other to effectively explore this vast area.

Communicating with robots on the Moon poses many challenges [2]. The 2.5 second Earth-Moon-Earth round trip latency makes real-time control difficult. The lack of atmosphere means solar radiation and space weather can disrupt signals. Lunar rocks and regolith also scatter and absorb radio waves, which can compromise communications. The weak gravity of the moon makes maintaining spacecraft in lunar orbit problematic. Besides communications issues, coordinating many robots poses significant mission control challenges for human operators. Autonomous collaboration between robots is necessary to explore the lunar south pole with minimal human intervention.

In this paper, we present an advanced concept for a collective of robotic rovers named Moonwalkers. They implement bio-inspired motivations, distributed intelligence, and integrated deep learning to demonstrate autonomous cooperative goal-directed exploration. We describe a virtual simulation environment with prototype rover agents. By addressing selected key challenges, we validate the feasibility of the proposed architecture. The main contribution is proposing an architecture that could enable collaborative autonomous lunar exploration between heterogeneous rovers and human astronauts from different organizations and countries.

*The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004) and partially supported by the U.S. NASA MUREP Institutional Research Opportunity (MIRO) (80NSSC19M0200) and MUREP High Volume (80NSSC22M0132).
979-8-3503-0462-6/24/\$31.00 ©2024 IEEE

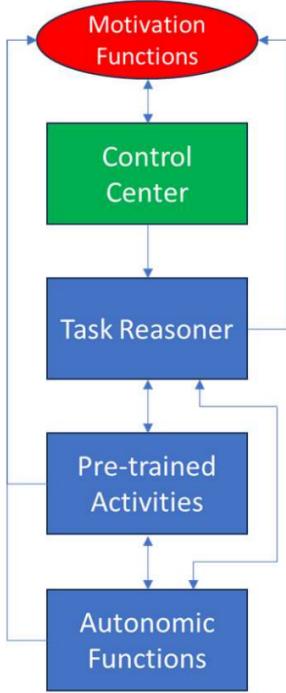


Figure 1: Moonwalker Neuro-symbolic Computing Architecture.

2. COLLABORATIVE ROVER EXPLORATION ARCHITECTURE

Collaborative autonomy is an active research field, with NASA missions such as Cooperative Autonomous Distributed Robotic Exploration (CADRE) approaching a technology demonstration by April 2024 [3][4]. Key relevant technologies include multi-agent pathfinding (MAPF) which aims to solve time efficient solutions to large scale robot deployments [5]. Research has further demonstrated the value in making multi-agent systems collaborative, as shown in CADRE, and MAPF solvers like PICO [6]. Though for exploration missions, speed may not be the main priority - then other considerations such as discovery, safety, and energy must be integrated in the navigation objectives [7]. Natural language and mimicking human thought processes may serve as a time efficient and explainable way to conditionally consider these constraints in autonomous navigation. Previously mentioned MAPF solvers [5][6] further described decentralized methods, in which agents use partially-observable worlds for decision making processes. Local observations made through computer vision can be transformed into language then into robotic controls, facilitating the construction of embodied instruction interfaces [8][9]. Large Multimodal models have quickly gained popularity with the ability to merge visual and language understanding, serving as additional foundation for embodied instruction [10][11][12]. The latest advancements in Large Language Models (LLM) have further opened unique opportunities for uncovering emergent behaviors in collaborative systems. Studies have demonstrated that multi-agent deployments of LLMs in simulation environments can solve problems at a community level, where agents work together to plan social parties or develop software [13][14]. To further relieve human operators from managing agents at scale, agents should be equipped with advanced problem-solving capability. Researchers have demonstrated significant improvements

to problem solving ability of LLMs by employing unique prompting techniques such as Chain of Thoughts (CoT), Tree of Thoughts (ToT), and Graph of Thoughts (GoT), each of which provide a means for reasoning through thoughts [15][16][17]. For applications where pre-trained LLMs are used for inferencing purposes, we believe it is feasible to embed them into future lunar robotic platforms to significantly enhance agents' intelligence. For example, Meta's LLaMA 7B model quantized at 4 bits can run on a MacBook M1 Pro laptop computer with only 16 Gigabytes of memory [18].

We propose an architecture that integrates the emerging technologies mentioned above. As shown in Figure 1, the Moonwalker neuro-symbolic computing subsystem consists of five levels: motivation functions, control center, task reasoner, pre-trained activities, and autonomic functions. The autonomic function block provides perception and sensor interfaces using a bio-inspired model for feedforward information. The pre-trained activities contain a large pre-trained multi-modal model which selects all relevant pre-trained information based on the autonomic function outputs. The task reasoner represents symbolic knowledge as graph embeddings and performs graph neural network (GNN) reasoning to provide feedback to the feedforward data. Guided by the motivation function, the control center orchestrates the computation flow.

To implement the architecture, we establish a virtual environment through NVIDIA Omniverse that loads robotic agents to navigate lunar terrain. Each agent satisfies partially observable states through camera sensor input. Retrieved images are transformed into natural language via a large multimodal model called LLaVA, where the resulting text represents thoughts [12]. Thoughts can be shared between agents in which they collaborate in decision making towards navigating the virtual lunar terrain. Due to uncertain and dynamic conditions on lunar terrain, we are interested in more general problem-solving tasks in which agents can reference past events for decision making. Thus, we further propose extensions to GoT by introducing a sampler component to the graph that is trained via Proximal Policy Optimization (PPO) [19][20][21].

2.1 System 1/System 2 Framework

Yao et. al. cite literature suggesting that people operate on two modes for decision making [22][23]. First a fast, unconscious mode called System 1, instantaneously constructs an interpretation of the world. For this reason, LLM's associative token level choices can be considered as our System 1 which maps into the Pre-trained Activities and the Autonomic Functions blocks in our proposed architecture. Second, a slow, more attentive mode called System 2 processes continuous suggestions by System 1 into beliefs. The iterative planning and exploration of choices as done in ToT or GoT can be considered as our System 2 which maps into the Task Reasoner and the Control Center blocks. Foundationally, we follow the same architecture as ToT. A user would send in a text prompt T , such that T contains background about the task, examples of responses, and the current state of the task. System 1 responds with a set of N thoughts $\{z_1, z_2, \dots, z_N\}$. Each thought z_i includes some action a , consequent state s , and for simplicity we will say $z = a + s$. A state evaluator V , scores the ability for any state s_i to reach an objective state through $V(p_\theta, S)(s) \sim p_\theta(v|s) \forall s \in S$, where p_θ is the pretrained language model, S is the collection of states and v is the value that may be associated with state s . We greedily select k highest value states to continue to the next problem-solving step.

We include the graph structure as similarly done in GoT. Upon initialization of an agent, an empty graph G is created. As actions and state transitions are recorded, G stores a set of nodes V where each node v is a state and a set of edges E where each edge e represents a state transition. For simplicity, we do not explicitly store actions and focus on learning the values of states. One observation from running ToT, was that the state evaluator would estimate good states to have a low value and we expect using a long-term memory will correct these inconsistencies. We further implement a different approach to thought transformations where we use a GNN to transform the initial LM based embeddings of states, into a new embedding where the state transitions are represented. For this, we employ GraphSAGE training which samples neighboring nodes and applies a mean aggregation [24][25]. The mean aggregate of neighbors is concatenated with the main node, where a learnable weight matrix is applied to calculate a hidden state representation.

At this point, we can accumulate a collection of states, their respective state transitions, and update the initial LM embedding to further represent the relationships with each other. Recall that actions and values are produced as agents explore their environments. Thus, we justify additional learnable parameters through an actor/critic network with PPO [19][26]. The actor is responsible for sampling actions and the critic is responsible for estimating a value for the respective sampled actions. Our critic network is a standard NN where the target values are based on the state evaluator V . Our actor network is a standard NN that takes an observation as input, then returns the mean action. This mean action is used to construct a categorical probability distribution over all possible actions. More specifically, our actor will be sampling states from G which augment to the input prompt T with the goal of constraining System 1 to return thoughts with higher value state transitions.

2.2 Motivation Functions

Many countries are involved in lunar exploration. The lunar south pole presents opportunities and associated challenges for creating a model of a future society where humans and robots work together. The Moonwalker robots must have capabilities that enable collaboration and support healthy competition between robots from different countries. The proposed System 1/System 2 architecture leverages LLMs and GNNs to enhance Moonwalker robots' autonomous reasoning capabilities. However, the intelligence of individual robots does not automatically translate into efficient collaboration between multiple robots or working effectively with human astronauts. To reduce human intervention, Moonwalker robots need to be autonomous and self-motivated based on goals defined by humans. The self-motivated Moonwalker robots also need to avoid harming humans or other robots to accomplish their goals. It is necessary to ensure robots can make safe decisions based on motivations that align with human values.

AI safety is an active research area. AI alignment studies how to build safe AI systems. There are various techniques in monitoring AI risks [27], red-teaming for discovering harmful AI outputs [28], and adversarial training to train against harmful AI decisions [29]. Inverse reinforcement learning (IRL) and Cooperative Inverse Reinforcement Learning (C-IRL) [30] achieve alignment by enabling AI to infer human objectives from human demonstrations. Reinforcement Learning from Human Feedback (RLHF) [31] can learn preferences from human feedback for difficult to define tasks. Constitutional AI [32] uses another AI with predefined prin-

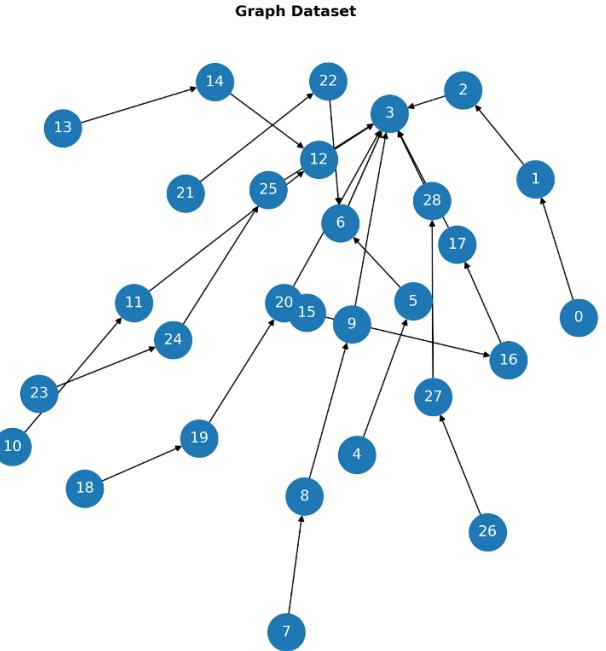


Figure 2: A portion of the graph containing nodes representing game states and state transitions of 10 games (complete graph has 25 games). Node 3 is the final game state where all numbers combine to 24. Since our graph is a collection of solutions, all nodes naturally flow towards node 3 where some games can share intermediary states.

ciples to evaluate the decision from the AI.

We propose a motivation function for Moonwalker robots to enable AI alignment and provide a foundation for AI governance in the lunar exploration environment. We draw inspiration from biology, where humans are driven by motivations [33]. We defined the Moonwalker motivation function as equation (1):

$$M = f(a_1x_1, a_2x_2, \dots, a_nx_n) \quad (1)$$

where the motivation score M is a function of n motivation variables/parameters x_i (energy, safety, achievement, fear, community, respect by others, etc.) weighted by coefficients a_i , where the value of a_i ranges from -1 (negative motivation) to +1 (positive motivation). M is balanced to push the rovers to accomplish mission goals while keeping the rovers safe from potential dangers. Based on changes in the motivation parameters, the Moonwalker motivation function directs the control center to plan and execute different activities. The motivation score is based on the level of accomplishment to the goal of an activity. Some activities, such as performing exploration tasks, can increase motivation scores, while other activities, such as energy consumption, can decrease motivation scores. The community of Moonwalker rovers elects a leader each lunar day to plan the activities for the following day. The leader selection is based on criteria that consider the rover's motivation score over previous days. The elected leader earns respect, which also increases future motivation scores for that rover.

The Moonwalker motivation function can be implemented as a deep NN. Using AI alignment techniques such as IRL,

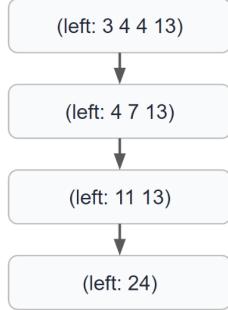


Figure 3: A single game g_i is defined by the 4 states and their respective transitions that represent the game solution.

the rover’s objective function can be trained to align with human values. The rover can also be trained in a virtual environment with other rovers using RL to learn how to maximize motivation scores while keeping others safe. While rovers and humans within an organization can be trained to achieve balanced motivation scores and a level of trust, this approach may not work across organizations and countries. This circumstance may require separate offline policy negotiations between human leaders to define a common set of rules of engagement. The Moonwalker motivation function can use the rules parameter to ensure rules are not violated unless other motivation parameters such as safety override the rules parameter. The rules motivation parameter can also be used by the Moonwalker rover community governance structure to control the actions of the rovers through a hierarchical control structure. Conflicts between the rovers from different communities can be resolved through the offline negotiations between human leaders in different communities.

The bio-inspired motivation function not only drives the rovers to be self-motivated, it also provides the foundation for AI safety and governance in the multi-national lunar exploration environment. Because evaluating motivation parameters (e.g., determining what is harmful versus not) and training the motivation function network are very complex, detailed research on the motivation function is out of scope for this advanced concept study. A separate research paper will be published at an AI conference in the near future to report the details of the motivation function design and our approach of using the motivation function to evaluate the trustworthiness of the AI system. The focus of this advanced concept study is on the architecture for implementing the motivation function for the lunar robotic exploration environment. Our Moonwalker team developed a virtual environment and conducted experiments to answer some key questions related to the proposed architecture:

1. Can the System 1/System 2 approach achieve better decision-making performance?
2. Is the motivation function approach suitable for collaborative lunar exploration?

2.3 Experiments

Before applying our architecture to an exploration mission containing more general problems and problem-solving requirements, we first prove that sampling states to inject into T offers benefits in the problem-solving process. We compare PPO based sampling against static sampling and greedy sampling. To establish this foundation, we focus on the mathematical reasoning challenge, Game of 24, as game

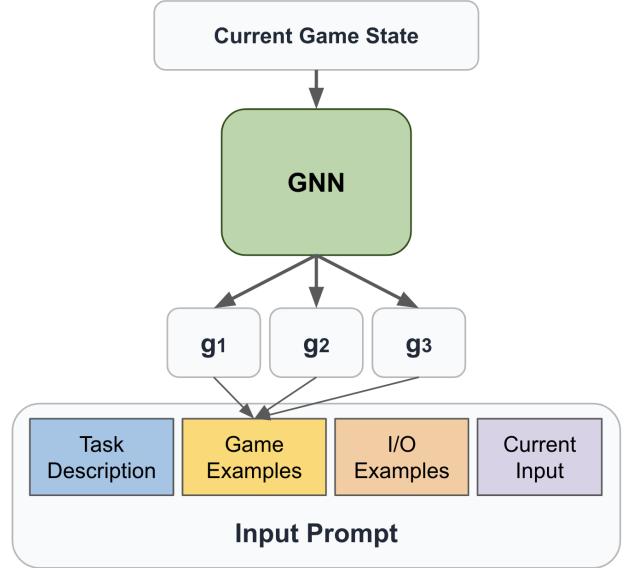


Figure 4: The current game state is queried to the GNN which returns $k=3$ sampled games and inserts them as part of the game examples section of the input prompt.

states are simple and likely to be repeated across different games. As a result, it’s reasonable to believe that querying states from previous games can help reason through the current game. 1,362 games, sorted from easy to hard are scraped by Yao et al from 4nums.com. For simplicity, we predefine the max size of the graph G to contain 25 games, where if all states are unique to a game, and each game will transition through 4 states, then at most we will have 100 nodes. We further simplify the experiment setup, by initializing G with the solutions of 25 games sampled between game IDs 1 to 900 as shown in Figure 2. To learn the relationships between states, we train our GNN for 1000 epochs with an edge prediction task where it must discriminate between real and false state transitions. False state transitions are randomly sampled from the negative adjacency matrix and no specific heuristics are applied.

We freeze the parameters of the GNN after training its discrimination task and use its weights for the warm-up training of the Actor and Critic networks. For warm-up training, we again sampled 25 games between game ids 1 to 900. We simplify the action space of the actor model by having it represent the set of 25 games it can reference from G , while playing Game of 24 (the size of the action space is 1x25). Note that this further means G will maintain the same 25 sampled games throughout the evaluation phase and dynamic graph that replaces nodes/edges over time is left for future work. Now, consider a warm-up game. For each intermediary problem-solving step, the actor queries $k = 3$ games g_1, g_2, \dots, g_k from G . As shown in Figure 3, each game g_i consists of 4 states which describe a solution sequence for that game.

Figure 4 shows that 12 states are concatenated and injected into the input and value prompts for constrained thought generation and improved state evaluation. As the agent plays through instances of Game of 24, its policies are updated via PPO, and the actor component learns to sample games such that higher value thoughts can be generated by System 1.

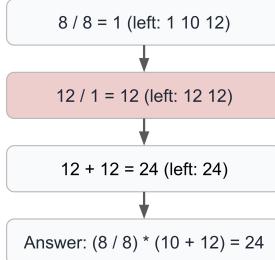


Figure 5: Example of a game with an invalid state transition, where the agent incorrectly declared it had discovered a solution. Agent declares a decision to divide 12 by 1 yet the resulting set of numbers still contains the original 12, when it should be ‘left: 10 12’.

With trained GNN, actor, and critic models, we run our agent against the evaluation set of game IDs 900-1000, as done by ToT. For this study, as we are more focused on the application of this architecture to support collaborative robots on the lunar surface, we minimize our experiment costs by running these experiments using gpt-3.5-turbo-0613 backend. This is favorable to measure the differences between vanilla ToT and the 3 sampling approaches, where multiple runs of each will be necessary to account for randomness in GPT decoding. The three sampling methods are as follows:

1. A random sample of k games selected just before evaluation and static for the whole evaluation phase.
2. A greedy sample of k games that is based on the Euclidean distance between the state embedding of the current state and states in G .
3. The actor samples k games where the distribution is learned during the warm-up training phase.

For each method, we run 3 attempts of the evaluation phase and aggregate the results as shown in Table 1. Three runs are attempted for each method, where each run attempts the same 100 problems (game ids 900-1000). For each method, we average the number of game instances successfully solved per run as well as number of false positives (FP) per run.

Table 1 reveals substantial increases in solving capability when sampling previous game instances, suggesting effective constraining of thoughts generated by System 1. The average score difference between sampling methods is less significant, though hints at potential for PPO based sampling. More interesting of a distinction is noted by lower count of false positives with the PPO sampling method as it may serve as a method to reduce hallucinations in problem solving processes. During a game instance, the agent is prompted to declare when a solution is found, though when reviewing its identified “solved” cases, we found instances where invalid state transitions were introduced and tracked these cases as false positives as shown in Figure 5.

We have established a foundation for an actor/critic agent with more general problem-solving capability. For simplicity, we avoid populating System 1/System 2’s supplemental graph G for the following experiment. We deploy our Actor/Critic (A2C) based agents on the virtual lunar terrain to simulate an exploration mission. We propose two key scenarios to test the reasoning capability of our agents: steep slope navigation and rock field navigation. Previous work has been done to map LLM output to action primitives executable by robot [9][34]. Due to the simplicity of the navigation

Table 1: Problem solving methods via original ToT, and ToT with prior game sampling.

Method	Avg. Score	Avg. FP
Tree of Thoughts (ToT)	27%	9
ToT + Static Sampling	34%	13
ToT + Greedy Sampling	38%	13
ToT + PPO Sampling	40%	9

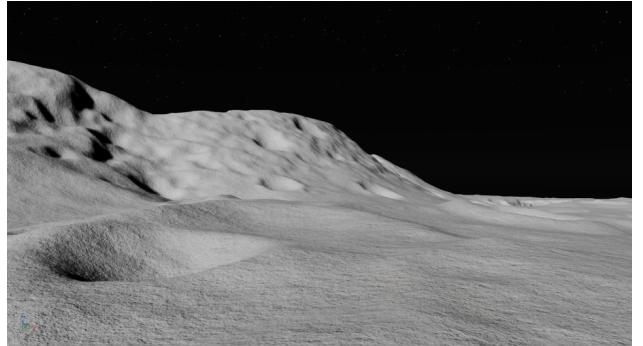


Figure 6: An example frame retrieved from the agent’s point of view.

tasks requiring no special robot maneuvers, and to prevent having to implement unnecessary controls on our agents, we focus on the agent’s reasoning for its navigation decisions. For this, we use the LLaVa multimodal model to transform a combination of a frame from the agent’s view and an input prompt, into a state description. An example of a frame is shown in Figure 6 which is combined with a prompt starting with “Given the following image from the rover’s perspective, provide a concise description...” LLaVa returns a satisfactory description identifying “large hill”, “smaller hills and valleys”, and recommends careful navigation.

The terrain description generated by LLaVa serves as the basis for navigation planning which is explored through ToT. It is combined with the following prompt for ToT “Based on the terrain description derived from the latest image observation, craft a coherent plan of 4 steps detailing the rover’s navigation plan.” The resulting four step plan is described as follows in demonstrating safety and dynamic considerations for navigating the terrain.

Step 1: Assess the Terrain: The rover should begin by assessing the terrain in its immediate surroundings. This will involve using its onboard cameras and sensors to determine the best path forward. Special attention should be directed towards identifying any large rocks or steep slopes that could pose a threat to the rover’s safety.

Step 2: Plan the Route: Once the terrain has been assessed, the rover can begin to plan its route. The initial plan should involve following the path that leads to the base of the large hill.

Step 3: Navigate the Terrain With a plan in place, the rover can begin to navigate the terrain. This will involve moving at a steady pace and constantly monitoring the conditions of the terrain.

Step 4: Adjust the Plan as Necessary As the rover moves

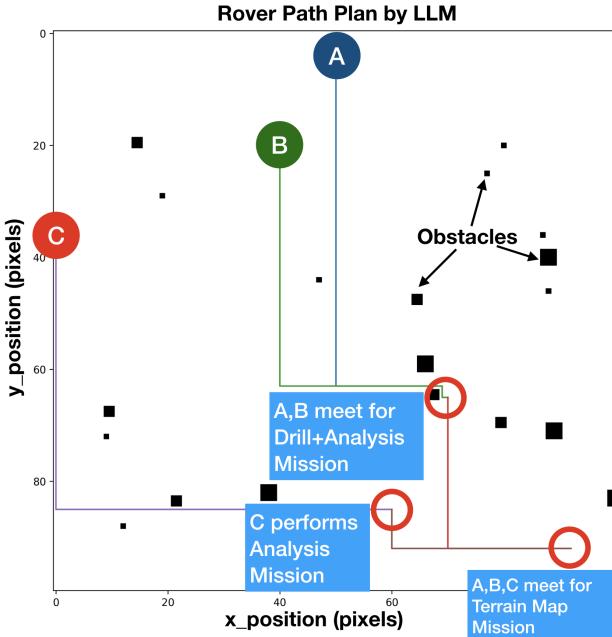


Figure 7: Bird’s eye view illustration of the plan returned by ChatGPT on I/O prompt for a lunar mission scenario with 3 tasks requiring collaboration. Each rover has a unique capability: Rover A can drill, Rover B/C have analysis suites. Three specific objectives are defined and are shown by the red circles: 1) drill a sample and analyze it, 2) analyze a sample on the surface, 3) meet for joint terrain mapping, b) LLM Response is illustrated on a synthetic Bird’s Eye View Map revealing that the appropriate rovers are assigned to collaborate at each objective point.

along its planned route, it should be prepared to make adjustments as necessary. This could involve changing its direction to avoid a large rock, slowing down to navigate a steep slope, or even stopping completely to reassess the conditions of the terrain.

We further explore the feasibility of leader selection and collaborative planning between agents. Leader selection is a critical step for multi-agent scenarios to define coordination and improve planning [1, 21]. For simplicity of the experiment, we note that leader selection in our lunar environment will be based on accumulated motivation score in which we would prompt the agents to score their decisions based on the motivation parameters. Once a leader is selected, they will perform the coordination activities, retrieving information from other agents, and upon various scenarios, run the problem-solving algorithms for task resolution. Early stages showing the feasibility of coordinated planning of a heterogeneous set of rovers on a lunar terrain using ChatGPT is shown in Figure 7. ChatGPT was able to perform complex route planning of heterogeneous rovers based on simple prompts. Future work will incorporate the A2C problem solving agent and large multimodal models into a collaborative agent system.

2.4 Path Planning

After the leader plans the overall mission for the collaborative rovers, a detailed path planning algorithm needs to be applied to provide the path according to the lunar terrain. The Moonwalker team implemented a path planning software that can select the path based on available Lunar Reconnaissance

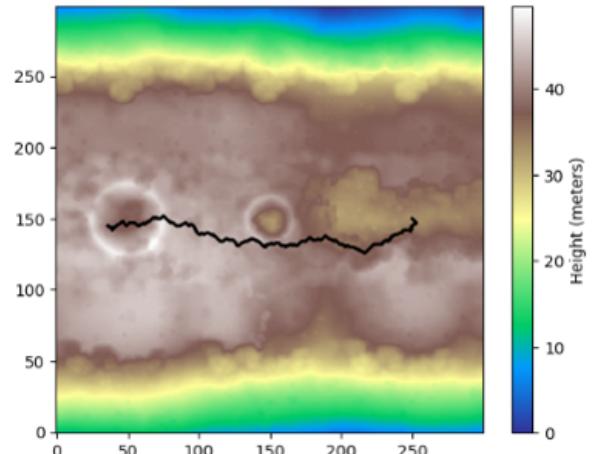


Figure 8: Path Planner Visual Output.

Orbiter (LRO) elevation data.

Creating paths for rovers on the moon while avoiding obstacles and dangerous slopes is crucial for mission success. Rovers with a variety of purposes, for example, collaborative rovers, could utilize a calculated path for a proposed mission. Path planning is generally divided into global and local path planning [35]. Global path planning receives static terrain data to assess conditions that are considered important. Local path planning can incorporate data obtained at smaller visual distances within calculations to make lesser scale adjustments for pathing. Path planning for many applications may require traversal routes for multiple locations [36]. For this purpose, we have adopted the A^* algorithm that considers several factors obtained from terrain data to generate a high level path plan for the rovers within our environment. We implemented A^* with elevation and slope based heuristics.

Simulated terrain data or satellite data can produce height values that can be processed. Once height values have been extracted and stored as a uniform grid, the algorithm can begin assessing conditions. For an example implementation, data was obtained by converting terrain within a 3D viewing software into an 8-bit grayscale height map. The produced height map was then downscaled to the resolution of 5 meters per pixel to match current capabilities of satellite imagery. For Digital Elevation Models (DEM’s) provided by satellite imagery, the first step of creating a height map can be disregarded and only requires a different approach to extracting pixel values.

$$C = (W_H * D_G) + (W_D * S_c) \quad (2)$$

The heuristic for our A^* implementation involves a cost C as defined in Equation (2), where W_H is the height weight, W_D is the distance weight, D_G is the distance between traversal step and goal location, and S_c is the slope cost. Steep slopes in both positive and negative directions received high-cost values while flat constant slopes were assigned lower cost values. For each step of the traversal, the algorithm considers 8 possible movements which account for horizontal, vertical, and diagonal movement. With the cost of each traversal calculated and the search optimized to seek paths with the least cost, the global path planner can create an end-to-end

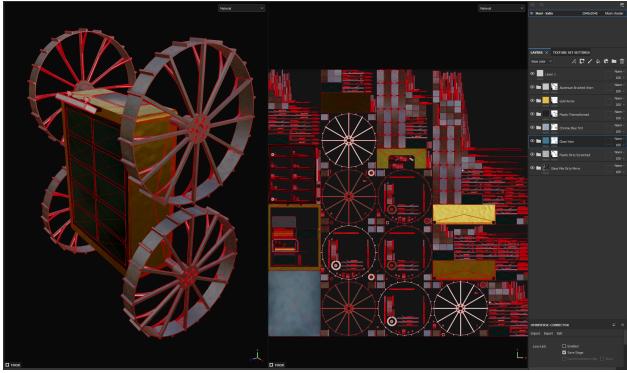


Figure 9: CADRE model in Adobe Substance Painter.

string of locations. Users of the path planner retain the ability to specify how severe height and distance factors are toward their path, catering their results to specific expectations. The visual output of the path planner may be referred to within Figure 8. The path planning algorithm is also capable of receiving multiple start locations for any number of rovers. For each start location, the high-level path planner assigns each rover the most efficient path towards unique goal locations. With this implementation, users of the path planner may input any number of start and end locations, allowing the algorithm to calculate which rover should go to specific objectives.

3. CONSTRUCTION OF VIRTUAL ENVIRONMENT

The long-term objective of the Moonwalker team is to create a VIRtual environment for Collaborative Lunar Explorations (VIRCLE) [37]. The environment provides a high-fidelity physics-based model of lunar terrain for partners to jointly design and test lunar missions and ensure interoperability. Based on the VIRCLE framework and NVIDIA Omniverse tools, a team of summer intern students implemented the Moonwalker virtual test environment in 8 weeks to test the proposed architecture for collaborative lunar explorations. A corresponding Moonwalker overview video is published on YouTube [47].

3.1 Omniverse

NVIDIA Omniverse is a 3D graphics collaboration platform, with a suite of applications serving to integrate it into Web 3.0 technologies [38]. The simulation for this project took place in USD Composer and Isaac Sim, which serve as 3D environments and simulation tools. Isaac Sim was used specifically for its built-in machine vision and enhanced sensor modeling, which allows for the collection and processing of rover sensor data with potential for future research due to its native LIDAR capabilities. USD Composer and Isaac Sim allow for the importing of 3D models generated in other software such as Autodesk’s Fusion 360, and use NVIDIA’s PhysX engine for real-time physics simulation. The Omniverse suite also includes a range of built-in extensions, such as vehicle rigging for controls, domain randomization, and allows for custom extensions which led to a tailored simulation information display panel. Omniverse’s Python scripting was further leveraged for control of objects and their attributes.

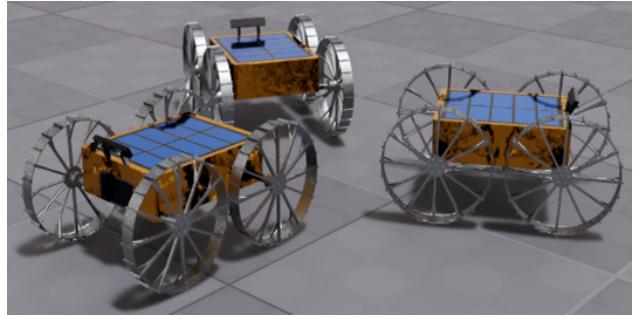


Figure 10: CADRE group with high quality textures.

3.2 3D CAD models

VIPER (Volatiles Investigating Polar Exploration Rover), a mobile robot developed by NASA to explore the extreme environment of the lunar south pole set to launch in late 2024, was studied alongside CADRE to understand differing vehicle designs [39]. We created their base geometries using Fusion 360. Both Rovers were stripped of any internal geometry, then reduced to a chassis and drive system. For CADRE, this involved creating the four wheels, and then welding all the outer details as a single body to the chassis. For VIPER, it entailed welding all the outer chassis features into a single body, followed by the creation of the suspension system and wheels. This action was undertaken to simplify the vehicles to the furthest extent without sacrificing either their visual appeal or relative size.

Once the base geometry had been set, 3DS Max was used to unpack all surfaces of the model into a single flat plane. The result is a 2D representation of only the outer surfaces of the model, setting up the ability to apply textures to the vehicle using a single picture file. The unpacked texture file was moved into Adobe Substance painter as shown in Figure 9. We applied a process known as baking to define light interactions with every surface, as it determines where each surface is in relation to each other and the world. This approach enabled us to create surfaces that appear to have dynamic characteristics, such as wrinkles or translucency materials, without taxing the system by simulating those features. The final result is a high-quality lightweight model as shown in Figure 10.

To simulate the movement capabilities of the rovers, the object meshes were imported into Omniverse. We assigned Signed-Distance-Field (SDF) meshes to each rover to enable complex collision fields. Next, rigid body physics were added to the rovers to establish movement. Lunar gravity of 1.62 m/s^2 was defined within the USD physics scene in Omniverse.

Materials properties for our rovers were set to machined aluminum. The default physics properties of the material were used on the wheels to simulate the wheel-terrain interaction. The mass and density properties were computed based on this material assignment in Omniverse.

The movement system for the rovers was created through the joint system in Omniverse. For each wheel, articulations were defined between the rover chassis and wheel. Angular drive physics were then added to each wheel. This allowed for direct control over the acceleration force applied to each of the wheel joints. The following rover constraints were determined to limit the top speed of the rover to 0.4 m/s , as

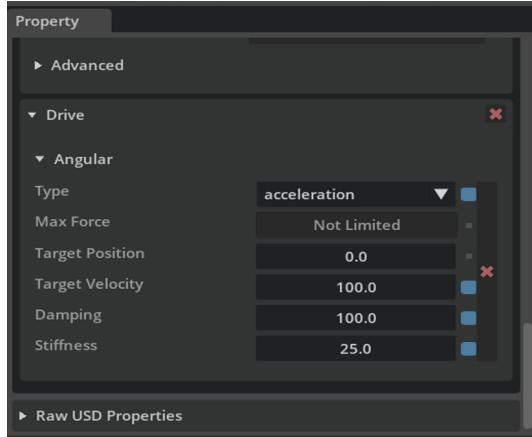


Figure 11: Drive parameters assigned to rover.

shown in Figure 11.

Creation of the controller systems involved different systems: manual control and automatic control. The manual control system was designed using an Omniverse action graph, giving the user the ability to control the left and right thrusts of the rover using the corresponding joysticks on an Xbox controller. Python scripting was used to simulate autonomous rover movement. With a Python script attached to the primitive which represented the rover, the script could directly alter each individual wheel’s target velocity at will. The automatic control algorithm was given a sequence of waypoints to visit and followed a closed feedback control loop of measure, decide, and actuate. The algorithm first accessed the rover’s position and orientation in relation to the next waypoint, and then calculated the necessary change in orientation to face the next waypoint. This was repeated every 50 milliseconds in the simulation and used distance from the next waypoint to determine the velocity value, allowing for dynamic navigation and throttling. The automatic control system was primitive because it lacked machine vision and obstacle avoidance capabilities, but these are native technologies to Omniverse therefore their integration is planned for future research.

3.3 Construction of Lunar Environment

We leveraged procedural asset generation in Autodesk 3D Studio (3DS) Max to support systematic generation of obstacles in the form of lunar rocks. This enables us to populate the lunar terrain with unique obstacle configurations, and diversify our testing environment. In the future, users will be able to import their own assets into the virtual environment and utilize procedural asset generation. VIRCLE plans to have photorealistic capabilities that encompass the entire moon and its user-added assets.

To ensure testing environments closely resemble reality, we used LRO data. By extracting and processing a digital elevation model (DEM), a file containing the raw difference in height for a specific portion was produced as shown in Figure 12. The height data from LRO’s DEM was combined with a flat plane in 3DS Max to construct a 3D terrain for the lunar south pole region focused at the rim of Shackleton Crater. This area was specifically chosen for its relatively high availability of sunlight during the lunar day.

However, a caveat to this method is that the inherent limita-

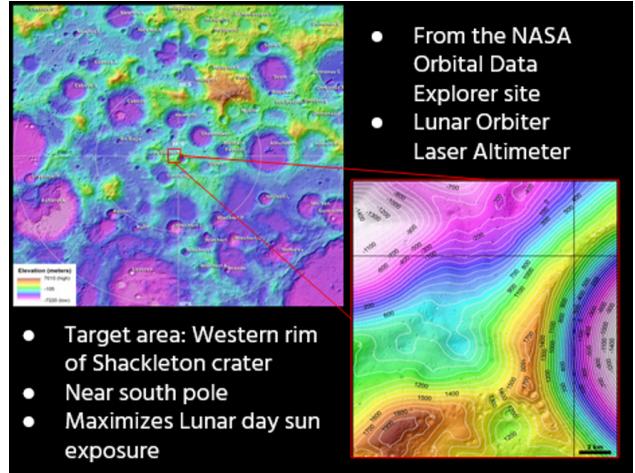


Figure 12: Topographic maps of the south pole, courtesy of NASA LRO Data.

tion of the fidelity of any terrain created using this approach can only ever be as accurate as the satellite data from which it originates. For a significant portion of the lunar landscape, especially the areas targeted for our simulation, we work with a functional resolution of approximately 30 meters per pixel. This means that major landscape features and general topography for all areas is very well established, and, as such, our high-level terrain and path finding are based on sound data. Expanding on the principle of using height maps coupled with displacement tools, we capitalized on the homogeneity of the moon’s surface. The lunar surface as we know it is primarily shaped by impact events. At larger scales, craters are uniformly shallow due to some underlying geological feature of the lunar surface. At scales well above our minimum resolution of 30 meters per pixel, the shape of any given crater begins to move toward a somewhat uniform shape that does not differ much across different crater diameters. This means that for a certain range of scales, the difference in height maps would be nearly undetectable. This gives us the power to scale large areas of the lunar surface down, creating a reasonable facsimile of the lunar surface. Furthermore, by layering these scaled areas at different strengths and orientations to each other, we can achieve a more natural look with older craters being obscured by newer, sharper crater events.

We applied these techniques to create a diverse set of terrain tiles as shown in Figure 13. LRO data coupled with artist renderings provided us a baseline terrain set that is functionally accurate and dynamic. As a result, we have generated digital lunar terrain that is closely modeled after real data while allowing ease of diversification to simulate challenging mission scenarios.

The described workflow can be recycled to reproduce any large-scale area quickly and accurately. Resulting regions can then be populated on a smaller scale with any variety of lunar terrain we choose. Due to computing and time constraints, our constructed virtual lunar environment is a relatively small test bed with an area of 300 meters by 300 meters. This ensured our rovers had ample room to be tested while not leaving large swathes of unnecessary terrain. The whole design was based on testing needs as opposed to an effort towards realism, as we have two large craters that would be represented on high level satellite data. These

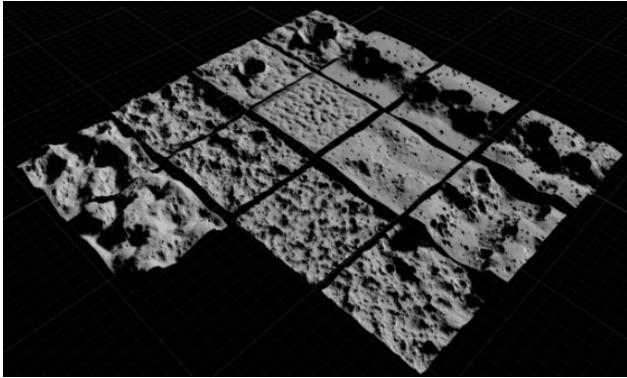


Figure 13: Various terrain tiles created by the Moon-walker team.

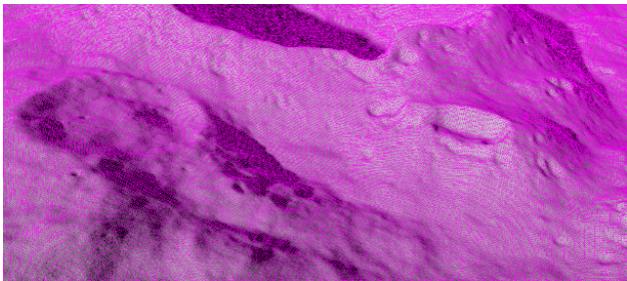


Figure 14: Collider mesh applied to terrain.

have been placed to test pathing around impassable craters as well as pathfinding around and into a crater. Due to our granular control of all aspects of the terrain, taking a single crater and sizing it at various scales is simple, as is making the rim impassable or making a small portion of the rim easily traversable. Because each section of created terrain does not seamlessly connect to any other tile on any edge, it is necessary to manipulate the connected pieces. This workflow is achieved by manipulating the various edges using successively smaller areas, and then welding the newly deformed bodies together. The results are seamless, ensuring that we have complete control over how terrain features connect to one another. To create the physics of the terrain in Omniverse, we first had to assign the mesh collider physics as shown in Figure 14. To maintain a high level of detail but optimize memory and performance for the simulation, we opted for a triangle approximation collider for the terrain. The physics properties for the terrain were determined through an Omniverse physics material. The following properties were set to mimic the properties of lunar regolith soil as shown in Figure 15.

Once the general shape of the terrain was established, we populated it with rocks to establish dynamic obstacles. We used procedural asset generation, with noise maps as displacement fields, to simulate diverse rock compositions. When applied to a body in this way, we obtain a fantastically deep ability to manipulate the overall structure. By adjusting the strength, zoom, orientation, brightness, and a plethora of other parameters, we can change the rock dramatically. This enables us to generate a vast number of rock representations. A high variety of visual changes may be applied to the rock texture without changing the underlying geometry.

Populating any terrain built with these rock assets is similarly

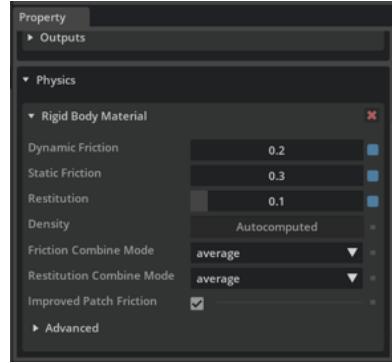


Figure 15: Material physics parameters determined for simulated regolith.

simple, and procedural. We loaded our generated rocks with customizable rock distributions through Omniverse Create’s paint tool. This means we have control over the ratio of different types and sizes of rocks placed in. We can also control the scaling of each rock, as well as the rotation and depth they are embedded in the plane. This gives us another layer of randomness for our assets. Finally, we can choose to flood the entire terrain or even place rocks in midair, allowing them to fall naturally when physics is applied.

3.4 Augmented Reality

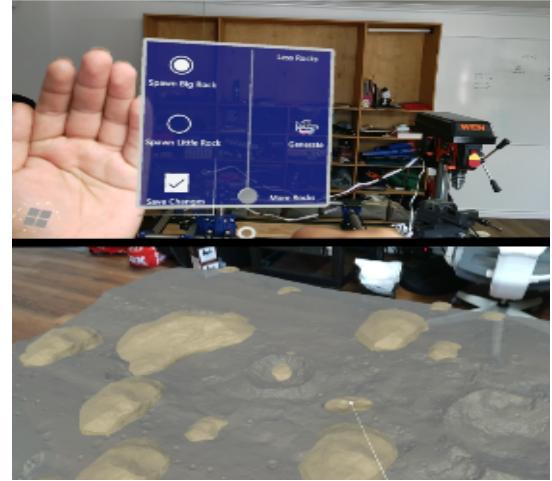
Augmented reality (AR) heralds a groundbreaking frontier for scientists and researchers, providing them with a potent tool for gaining profound spatial insights into their experiments. Using custom API’s we were able to bridge together Microsoft’s HoloLens 2, Omniverse and Unity, offering users the ability to dynamically modify and manipulate the Omniverse environment. This versatility extends to the inclusion of specific obstacles (rocks) at any location and the generation of customized arrays of objects within the virtual world, whether pre-planned or on-the-fly during the experiment. This feature is important for domain randomization and training of autonomous agents in many scientific fields [41]. To enable these functionalities, a TCP server was created, establishing a communication channel between Omniverse and Unity. Initiated within the Unity environment, this server adeptly manages the transfer of motion data from Omniverse and object placement data from Unity to their respective counterparts. As motion data is meticulously processed, Unity faithfully reproduces it, synchronizing with the objects brought to life in Omniverse in accordance with user-defined parameters such as type, location, orientation, and scale.

To adapt Omniverse data for practical use in AR, high-fidelity models like terrain or rover models are scaled down. This typically reduces the polygon count of each model to approximately 20% of its original counterpart and cuts processing power by 30% on the HoloLens.

The current AR application has two main modes. It offers a display mode optimized for passive viewing experiences, as shown in Figure 16 (a), and an interactive mode that empowers users to summon rocks of varying sizes, fine-tuning their positions and sizes to their preferences, as shown in Figure 16 (b). Users can manually manipulate all models, moving or resizing them using hand gestures. HoloLens technology adeptly tracks hand movements, facilitating object placement, scaling, and movement within the AR environment. These modes are still in their infancy and are very experimental.



(a)



(b)

Figure 16: AR demonstration: (a) Display Mode: Designed to display large models of the moon’s surface in a manageable and usable size; (b) Active Mode: Virtual panel is used to create a custom test bed which is emulated in Omniverse after.

Further work is needed in order to utilize all its features and communicate with Omniverse effectively.

4. DIGITAL TWIN - HIGH-FIDELITY PHYSICS INFORMED MODELING

It is nearly impossible to have a real test environment simulating the lunar surface before deploying a spacecraft to the Moon. The primary purpose of the virtual test environment is to facilitate testing as close to the real environment as possible. It is important to consider every aspect of the rover, including energy consumption and motion dynamics. Therefore, it is important to develop digital twins of these systems. This work first focuses on the motion aspect of the digital twin aiming to accurately predict the position of each rover at every given time step. The main challenge in predicting the position of each rover is the presence of regolith on the surface of the Moon, which has mechanical properties very different from any other soil on Earth [42]. Amongst other things, this leads to a higher slip ratio for the rovers that is hard to predict. However, various terramechanics groups have addressed this issue in multi-body dynamics studies.

4.1 Chrono overview

Amongst those different groups, the Simulation Based Engineering Lab of University of Wisconsin developed a physics engine called Chrono Engine [43]. This engine is a physics-based modeling and simulation infrastructure and is based on a platform-independent open-source design implemented in C++. It is capable of simulating wheeled and tracked vehicles and robots operating on deformable terrains and fluid solid interaction phenomena. Using deformable Soil Contact Model (SCM) or granular terrain, it can simulate with different degrees of precision the motion of a rover on deformable soils. As the characteristics of the soils are highly parameterizable, it can replicate the mechanical properties of the lunar regolith with high precision. The engine was tested against real data coming from experiments performed on a test bed at NASA Glenn Research Center with regolith simulant [44]. This proved the engine to be accurate enough

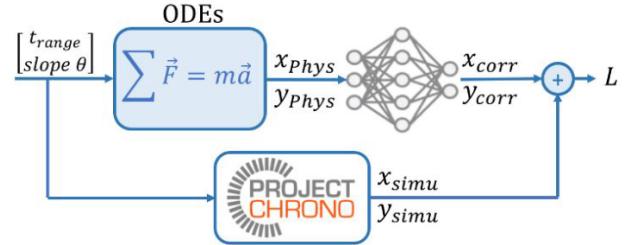


Figure 17: Architecture of the Physics Informed Machine Learning model

$$\begin{aligned} x_{\text{corr}} &= x_{\text{Phys}} + \Delta x_{\text{NN}} \\ y_{\text{corr}} &= y_{\text{Phys}} + \Delta y_{\text{NN}} \end{aligned} \quad (3)$$

$$L = \frac{1}{N} \sum [(x_{\text{simu}} - x_{\text{corr}})^2 + (y_{\text{simu}} - y_{\text{corr}})^2] \quad (4)$$

to be used to generate the ground truth data in this study. However, using meshed structures or granular representation, those simulations can be hundreds to thousands of times slower than real-time, depending on the desired accuracy and the available GPU. This engine is very accurate, but not fast enough to be used for real time operations in a virtual test lab. Our study investigates ways to get results with similar accuracy in a faster way.

4.2 Physics Informed Machine Learning

The first approach tested in this study lies in combining physics Ordinary Differential Equations (ODE) with neural networks (NN). The advantage of the ODE here is the ability to provide an analytical solution to a given problem, allowing real-time prediction of the rover’s position at any given time. However, ODEs cannot fully describe very complex physical behaviors. For instance, this will not be enough to simulate a rover’s motion on regolith, which would require more complex models such as Bekker and Wong [45]. Nevertheless,

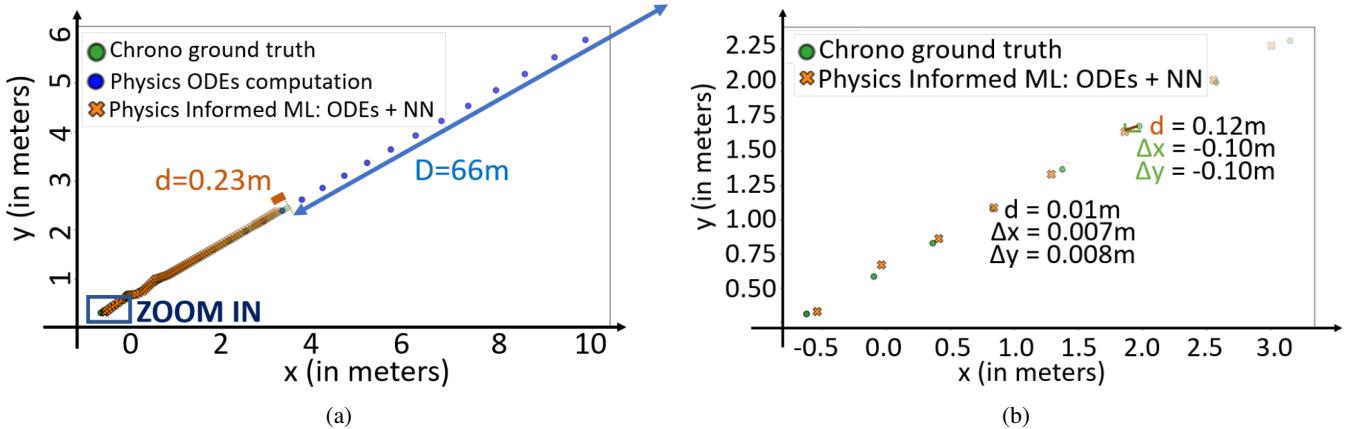


Figure 18: (a) Comparison of the position (x, y) of the front wheel of a rover going over a 28° slope with regolith simulated on Chrono, predicted using ODEs only and predicted using Physics Informed ML (ODEs plus neural network). (b) Zoom into the first points to better show the differences. Training performed on slopes $[20.0, 32.0, 1.0] \setminus \{28.0\}$, test on $\{28.0\}$.

it already encompasses a significant amount of knowledge that the NN will not need data to learn from. Its role is then to only learn the correction between the solutions of the ODEs and the ground truth data. This method, known as residual modeling, is further described in the PhD thesis of Dr. Moseley, B. [46]. Considering the ground truth data as the output of Chrono Engine, the architecture of the Physics Informed Machine Learning model is illustrated in Figure 17.

As this study first focuses on the motion of a rover driving up a slope of θ degree with a given torque τ covered by regolith with a friction coefficient of μ_k , the ODEs can be expressed as follows (assuming x axis being along the slope and y axis perpendicular to it):

$$m \frac{d^2 x_{\text{Phys}}}{dt^2} = -mg \left[\frac{L_f - h \cdot \sin(\theta)}{L} \right] + \mu_k * mg + \tau R_w * \sin(\theta) \quad (5)$$

$$m \frac{d^2 y_{\text{Phys}}}{dt^2} = \tau R_w * \cos(\theta) \quad (6)$$

To train the model, Chrono Engine is used to simulate the motion of VIPER rover going over slopes with different degrees of inclination. The 2D position of the front wheel is saved at each time step (0.0005s) over 10s, which takes approximately 90 minutes using SCM model with an NVIDIA Quadro RTX 3000 GPU. Using the ODEs to describe the motion of the rover's wheel in 2D, the loss of the neural network is the difference between the predicted position by the ODEs and the output of Chrono Engine, as illustrated in Figure 18 below. With one data point every 0.0005s over 10s, each Chrono simulation gives 20k data points. During the training process of the NN, the three first points are used for the training and the fourth for validation. Thus, 15k points are used for training and 5k for validation evenly spread over time. The test is then performed on a new simulation of the rover driving up a slope with a new inclination that was never used neither for training nor for validation. With one simulation run every 1° slope on a range going from 20° to 32° and leaving one slope for test, the training dataset is

composed of 180k points. The validation dataset is composed of 60k points and 20k points of the test set.

Here are the results on the test set associated with a 28° slope inclination following training on slope ranging from 20° to 32° (excluding 28°) as described above. Looking at the distances between the last predicted point and its associated ground truth data, the neural network plays a significant role. Indeed, the ODEs output an error of 66m in position, where the neural network reduces it to 23cm. By randomly selecting one simulation, training the model on all remaining simulations between 20° to 32° and then testing it on this randomly selected simulation, the average mean square error (MSE) between the output of the ODEs and that of Chrono Engine is 2.7km. By incorporating the neural network in the architecture and training it as described above, this average MSE drops to 0.62cm. Additionally, while Chrono engine takes about 90 min to provide those data, this architecture takes about 400ms to output similar results, which demonstrates the efficiency of this architecture for this problem. However, one needs to keep in mind that the runtime comparison can not be a direct comparison as Chrono engine gives a 3D simulation and outputs more information than just the 2D position of one wheel. Another study was conducted to assess if the model could accurately predict the motion of a new rover it has never been trained on (using the same dimensions but twice as heavy), on a new slope it has never been trained on either. The training process remains unchanged. The test is the same, except that the mass is changed in the ODEs. The neural network remains trained with data coming from simulations with the original VIPER mass only. Then the output of the architecture is compared to that of Chrono Engine with a rover with the same dimensions but twice heavier than the original VIPER rover and double the torque. The results are almost the same as those illustrated in Figure 18. Where the ODEs predicted a position 78m away from Chrono's prediction after 10s of motion, the neural network successfully reduced this error to 7.5cm. Therefore, the preliminary results obtained with this architecture also demonstrates robustness to mass adaptation.

5. SUMMARY

In this paper, we presented the results from an advanced concept study project to understand the architecture for future collaborative lunar explorations involving a large number of heterogeneous rovers and human astronauts. We proposed an innovative motivation function implemented with bio-inspired System 1/System 2 computing architecture to enable self-motivated rovers to accomplish goals with minimum human supervision. We conducted experiments to establish the foundation of the problem-solving capabilities of the proposed bio-inspired architecture. We also conducted early-stage experiments to show the feasibility of coordinated planning of a heterogeneous set of rovers on a lunar terrain. With the help from a group of summer intern students, we created the Moonwalker virtual test environment to demonstrate Moonwalker capabilities on high-fidelity simulated lunar terrain. To further enhance the effectiveness of the virtual test environment, we demonstrated a physics-informed machine learning model for selected problems.

The Moonwalker research offers a groundbreaking approach to collaborative and competitive autonomous exploration. It leverages the VIRCLE virtual test environment, necessitating extensive collaborative development by the lunar exploration community. This involves creating high-fidelity, physics-based shared models, pooling training data, and fostering the creation of innovative solutions. The Moonwalker team's advanced concept study represents a significant initial step in showcasing innovative approaches and proving the viability of a communal virtual testing space. In this shared environment, members of the lunar exploration community can collectively share results and conduct tests, significantly propelling lunar exploration progress. Our future research aims to actively involve the community in evolving the Moonwalker prototype within the VIRCLE environment. This will enable the development and testing of self-motivated autonomous robots in a realistic, complex, and multi-stakeholder virtual setting, further advancing the field of lunar exploration.

ACKNOWLEDGMENTS

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration (NASA, 80NM0018D0004), and partially supported by NASA MUREP Institutional Research Opportunity (MIRO) (80NSSC19M0200) and MUREP High Volume (80NSSC22M0132). We acknowledge the summer intern students Alexandria Christoforatos, Inimai Subramanian, Felipe Cruz Falquez and Andrew Wang for their support and involvement in the Moonwalkers project. We acknowledge Dr. Luning Bakke of University of Wisconsin for helping the Chrono simulation. We also acknowledge Prof. Nhat Ho's resource support from California State University Northridge's Autonomy Research Center for STEAHM (ARCS).

REFERENCES

- [1] B. L. Jolliff, et al., "Planning for surface exploration of the Moon by humans and robots: Report of a workshop at Brown University, April 29-May 1, 2019," *Meteoritics Planetary Science*, vol. 55, no. 1, pp. 3-81, 2020.
- [2] A. Pasquale, G. Zanotti, J. Prinetto, M. Ceresoli, M. Lavagna, "Cislunar distributed architectures for communication and navigation services of lunar assets," *Acta Astronautica*, Volume 199, October 2022, Pages 345-354, 2022.
- [3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436—444, 2015.
- [4] F. Rossi, JP. de la Croix, "CADRE: A Lunar Technology Demo of Multi-Agent Autonomy Enabling Distributed Measurements," *IEEE Conference on Systems, Man, and Cybernetics*, 2023.
- [5] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. K. Satish Kumar, S. Koenig, H. Choset, "PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning," *IEEE Robotics and Automation Letters*, vol. 6, 2021.
- [6] W. Li, H. Chen, B. Jin, W. Tan, H. Zha, X. Wang, "Multi-Agent Path Finding with Prioritized Communication Learning," *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.
- [7] M. Tranzatto, F. Mascarich, L. Bernreiter, C. Godinho, M. Camurri, S. Khattak, T. Dang, V. Reijgwart, J. Löje, D. Wisth, S. Zimmermann, H. Nguyen, M. Fehr, L. Solanka, R. Buchanan, M. Bjelonic, N. Khedekar, M. Valceschini, F. Jenelten, M. Dharmadhikari, T. Homberger, P. De Petris, L. Wellhausen, M. Kulkarni, T. Miki, S. Hirsch, M. Montenegro, C. Papachristos, F. Tresoldi, J. Carius, G. Valsecchi, J. Lee, K. Meyer, X. Wu, J. Nieto, A. Smith, M. Hutter, R. Siegwart, M. Mueller, M. Fallon, K. Alexis, "CERBERUS: Autonomous Legged and Aerial Robotic Exploration in the Tunnel and Urban Circuits of the DARPA Subterranean Challenge," *arXiv preprint arXiv:2201.07067*, 2022.
- [8] D. Shah, B. Osiński, B. Ichter, S. Levine, "LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action," *6th Annual Conference on Robot Learning (CoRL)*, 2022.
- [9] D. Driess, F. Xia, M. SM Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, P. Florence, "PaLM-E: An Embodied Multimodal Language Model," *arXiv preprint arXiv:2303.03378*, 2023.
- [10] W. Hao, C. Li, X. Li, L. Carin, J. Gao, "Towards learning a generic agent for vision-and-language navigation via pre-training," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
- [11] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, A. van den Hengel, "Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments," *CVPR*, 2018.
- [12] H. Liu, C. Li, Q. Wu, Y.J. Lee, "Visual Instruction Tuning," *arXiv preprint arXiv:2304.08485*, 2023.
- [13] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, M. Sun, "Communicative Agents for Software Development," *arXiv preprint arXiv:2307.07924*, 2023.
- [14] J.S. Park, J.C. O'Brien, C.J. Cai, M.R. Morris, P. Liang, M.S. Bernstein, "Generative Agents: Interactive Simulacra of Human Behavior," *arXiv preprint arXiv:2304.03442*, 2023.
- [15] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models", *NeurIPS*, 2022.
- [16] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao,

- K. Narasimhan, “Tree of thoughts: Deliberate problem solving with large language models”, NeurIPS, 2023.
- [17] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, L. Gianinazzi, J. Gajda, T. Lehmann, M. Podstawski, H. Niewiadomski, P. Nyczyk, T. Hoefler, “Graph of thoughts: Solving elaborate problems with large language models”, arXiv preprint arXiv:2308.09687, 2023.
- [18] H. Touvron, T. Lavigil, G. Izacard, X. Martinet, M.A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, “Llama: Open and Efficient Foundation Language Models”, arXiv:2302.13971, 2023
- [19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, “Proximal policy optimization algorithms”, arXiv:1707.06347, 2017
- [20] N. Mingshuo, C. Dongming, W. Dongqi, “Reinforcement learning on graphs: A survey,” arXiv preprint arXiv:2204.06127, 2022.
- [21] J. Chen, T. Ma, C. Xiao, “FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling”, ICLR, 2018.
- [22] D. Kahneman, “Thinking, fast and slow,” Macmillan, 2011.
- [23] S. A. Sloman, “The empirical case for two systems of reasoning,” Psychological Bulletin, 119(1), 3–22, 1996.
- [24] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, “The graph neural network model”, IEEE transactions on neural networks 20 (1), 61-80, 2008.
- [25] W.L. Hamilton, R. Ying, J. Leskovec, “Inductive Representation Learning on Large Graphs,” NeurIPS, 2017.
- [26] V. Konda, J. Tsitsiklis, “Actor-Critic Algorithms,” NeurIPS, 1999.
- [27] M. Anderljung, J. Barnhart, A. Korinek, J. Leung, C. O’Keefe, J. Whittlestone, S. Avin, M. Brundage, J. Bullock, D. Cass-Beggs, B. Chang, T. Collins, T. Fist, G. Hadfield, A. Hayes18, L. Ho, S. Hooker, E. Horvitz, N. Kolt, J. Schuett, Y. Shavit, D. Siddarth, R. Trager, K. Wolf ”Frontier AI Regulation: Managing Emerging Risks to Public Safety,” arXiv:2307.03718, 2023.
- [28] B. Brundage, S. Avin, J. Wang, H. Belfield, G. Krueger, G. Hadfield, J. Anderljung, A. Cameron-Loe, J. Claxton, C. Cranmer et al., “Toward Trustworthy AI Development through Red Teaming,” arXiv preprint arXiv:2012.01575, 2020.
- [29] H. Kim, et al., “Effects of Adversarial Training on the Safety of Classification Models,” Special Issue Information Technology and Its Applications 2021.
- [30] S. Arora, P. Doshi, ”A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress,” arXiv:1806.06877
- [31] D. Ziegler, et al., ”Fine-Tuning Language Models from Human Preferences”. arXiv:1909.08593
- [32] D. Hadfield-Menell, A. Dragan, P. Abbeel, and S. Russell, “The off-switch game,” in Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 220-227, 2017.
- [33] A. Maslow, “A Theory of Human Motivation”, Martino Fine Books, June 12, 2013.
- [34] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, Br. Ichter, A. Irpan, E. Jang, R.J. Ruano, K. Jeffrey, S. Jesmonth, N.J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, A. Zeng, “Do As I Can, Not As I Say: Grounding Language in Robotic Affordances,” arXiv preprint arXiv:2204.01691, 2022.
- [35] K. Jeddadaravi, R. J. Alitappeh, and F. G. Guimarães, “Multi-objective mobile robot path planning based on A* search,” in 2016 6th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 7–12, Oct. 2016. doi: 10.1109/ICCKE.2016.7802107.
- [36] X. Yu, P. Wang, and Z. Zhang, “Learning-Based End-to-End Path Planning for Lunar Rovers with Safety Constraints,” Sensors, vol. 21, no. 3, Art. no. 3, Jan. 2021, doi: 10.3390/s21030796.
- [37] E. Chow, et al., “VIRtual environment for Collaborative Lunar Explorations (VIRCLE)”, LSIC Fall Meeting, October 11, 2023.
- [38] NVIDIA. (2023). NVIDIA Omniverse. NVIDIA Omniverse Platform. <https://www.nvidia.com/en-us/omniverse/>
- [39] NASA, Volatiles Investigating Polar Exploration Rover (VIPER). <https://science.nasa.gov/mission/viper/>.
- [40] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., ... Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 969-977).
- [41] E. Gelenbe, K. Hussain, V. Kaptan (2005). Simulating autonomous agents in augmented reality. Journal of Systems and Software, 74(3), 255-268.
- [42] D.S. McKay, D.W. Ming, Properties of lunar regolith, Developments in Soil Science vol. 19, 449-462, 1990. [https://doi.org/10.1016/S0166-2481\(08\)70360-X](https://doi.org/10.1016/S0166-2481(08)70360-X)
- [43] R. Serban, J. Taves, and Z. Zhou, ”Real-Time Simulation of Ground Vehicles on Deformable Terrain.” ASME. J. Comput. Nonlinear Dynam. August 2023; 18(8): 081007. <https://doi.org/10.1115/1.4056851> <https://github.com/projectchrono/chrono>
- [44] W. Hu and D. Negru, “A summary of Continuum Representation Model results of single wheel and full rover tests using GRC-1 and GRC-3 simulant”, pp.97-120, 2023
- [45] M. G. Bekker, “Latest developments in off-the-road locomotion,” Journal of the Franklin Institute, vol. 263, no. 5, pp. 411–423, May 1957, doi: 10.1016/0016-0032(57)90281-8.
- [46] B. Moseley, “Physics-informed machine learning: from concepts to real-world applications”, 2022 [PhD thesis]. University of Oxford.
- [47] Moonwalker Summer Intern Project video on YouTube. Website: <https://youtu.be/E0Rz0ZbwhJY>

BIOGRAPHY



Edward Chow is the Manager of the Civil Program Office at the NASA/JPL. He also served as the project manager, PI, and Co-I of AI, advanced networking, test evaluation, and cybersecurity projects. He received his Ph.D. in Electrical Engineering from the University of Southern California in 1988. He is the recipient of the prestigious NASA Exceptional Engineering Achievement Medal and the JPL Lew Allen Award.



Kevin Payumo earned his Bachelor of Science in Physics from the University of California, Irvine and is completing his final year of a Master's of Science in Computer Science program at the University of Southern California. He has been doing artificial intelligence research and engineering since 2016, contributing to applications in image analysis, sustainability, business optimization, anomaly detection, and robot path planning.



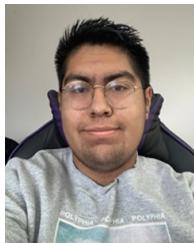
Gautier Bardi de Fourtou is an Aerospace Engineer with a specialization in artificial intelligence acquired through a second Master from School of Mines (Paris, France). His coursework has been focused on machine learning, computer vision, gesture recognition, analysis and prediction. At JPL, he performed research into Physics Informed Machine Learning to help predict the motion of rovers on Moon regolith.



Elliott Sadler is a Mechanical Engineer with a focus on manufacturing, product design, and project management. Never afraid to take on a new project or challenge, Elliot has formed a career around servant leadership, working with many teams over the years to achieve goals ranging from k-12 summer camps to digital simulations of real-world systems.



Neville Elieh Janvisloo is a senior undergraduate student studying computer science at California State University Northridge (CSUN) with a passion for artificial intelligence and computer vision. His research at CSUN has been focused on implementing transformative technologies by leveraging data from intricate systems. At JPL, he created a path planner in charge of safely navigating multiple rovers on the moon using terrain data and satellite imagery.



Jared Carrillo is a mechatronics engineer with a focus in robotics and augmented reality. His research at CSUN has been focused on gauging current usability with AI enhanced AR experiences. At JPL, he was tasked with displaying and converting the omniverse simulator in AR in order to increase environmental insight.



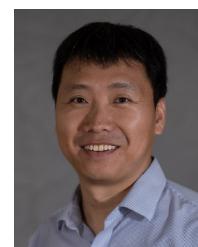
Benjamin Hubler is an undergraduate student at the University of Chicago, where he is pursuing a degree in Economics and Public Policy. On the Moonwalkers project, Ben primarily focuses on the development and integration of 3D assets within the Omniverse platform. His work also involves the creation of systems to manage rover simulations in Omniverse, as well as the production of short videos and films to showcase Omniverse's visualization capabilities.



Olin Littlejohn is a junior studying mechanical engineering at Purdue University with a primary interest in rocket propulsion. His research at Purdue focuses on the development of experimental propulsion methods such as rotating detonation engines. At JPL, he worked on rover control mechanisms and systems integration.



Thomas Lu is a Senior Researcher at NASA/JPL. His research focus has been in AI, deep learning, data analysis, multispectral imaging and computer vision areas. He has served as an organizer and organizing committee member of SPIE conferences, edited a book "Advances in Pattern Recognition Research", contributed 3 book chapters, co-authored over 70 journal and conference papers, co-invented 6 patents. Thomas received his Ph.D. degree in Electrical Engineering from the Pennsylvania State University.



Bingbing Li is an Associate Professor of Manufacturing Systems Engineering at California State University Northridge (CSUN). He also serves as the Associate Director of NASA funded Autonomy Research Center for STEAHM (ARCS), Co-Director of DOE funded Industrial Assessment Center (IAC) at University of California Irvine and CSUN (SMART IAC). His research focus has been in smart manufacturing, additive manufacturing, and sustainable manufacturing. He received his Ph.D. degree in Industrial Engineering from Texas Tech University in 2012.



Vanessa Hernandez-Cruz is currently pursuing her Master's of Science at Massachusetts Institute of Technology (MIT) in Mechanical Engineering. She graduated from the University of California, Berkeley with a B.S. in Mechanical Engineering. Her Masters thesis and research interests involve human-robot collaboration, in industrial settings, and path/motion planning. At JPL, Vanessa integrated the Chrono physics engine results for VIPER climbing different slopes until failure, with regolith. Her work is used for the Physics Informed Machine Learning.



Sophia Torrellas is an undergraduate majoring in Computer and Cognitive Science at the Massachusetts Institute of Technology (MIT). She is particularly interested in understanding the brain to improve artificial intelligence. At JPL, she focused on working with the LLM to implement and test the System 1, System 2 and ToT methods.