aerospace

MDPI

Article

EELS-DARTS : A Planetary Snake Robot Simulator for Closed-Loop Autonomy Development

Tristan D. Hasseler, Carl Leake, Aaron Gaut, Asher Elmquist, Robert Michael Swan, Rob Royce, Bryson Jones, Ben Hockman, Michael Paton, Guglielmo Daddi, Masahiro Ono, Rohan Thakker and Abhinandan Jain *

Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Dr., Pasadena, CA 91109, USA; tristan.hasseler@jpl.nasa.gov (T.D.H.); carl.leake@jpl.nasa.gov (C.L.); aaron.gaut@jpl.nasa.gov (A.G.); asher.elmquist@jpl.nasa.gov (A.E.); robert.m.swan@jpl.nasa.gov (R.M.S.); rob.royce@jpl.nasa.gov (R.R.); bryson.jones@jpl.nasa.gov (B.J.); benjamin.j.hockman@jpl.nasa.gov (B.H.); michael.paton@jpl.nasa.gov (M.P.); guglielmo.daddi@jpl.nasa.gov (G.D.); masahiro.ono@jpl.nasa.gov (M.O.); rohan.a.thakker@jpl.nasa.gov (R.T.)
* Correspondence: abhinandan.jain@jpl.nasa.gov

Abstract: EELS-DARTS is a simulator designed for autonomy development and analysis of large degree of freedom snake-like robots for space exploration. A detailed description of the EELS-DARTS simulator design is presented. This includes the versatile underlying multibody dynamics representation used to model a variety of distinct snake robot configurations as well as an anisotropic friction model for describing screw–ice interaction. Additional simulation components such as graphics, importable terrain, joint controllers, and perception are discussed. Methods for setting up and running simulations are discussed, including how a snake robot's autonomy stack closes the commands and information loop with the simulation via ROS. Multiple use cases are described to illustrate how the simulation is used to aid and inform robot design, autonomy development, and field test use throughout the project's life cycle. A validation analysis of the screw–ice contact model is performed for the surface mobility case. Lastly, an overview of simulation use for planning operations during a recent field test to the Athabasca Glacier in Canada is discussed.

Keywords: dynamics; simulation; robotics; exploration; rovers; mobility; contact; graphics; perception; DARTS; NASA JPL

Citation: Hasseler, T.D.; Leake, C.; Gaut, A.; Elmquist, A.; Swan, R.M.; Royce, R.; Jones, B.; Hockman, B.; Paton, M.; Daddi, G.; et al. EELS-DARTS: A Planetary Snake Robot Simulator for Closed-Loop Autonomy Development. *Aerospace* **2024**, *11*, 795. https://doi.org/10.3390/aerospace11100795

Academic Editor: Roy Lichtenheldt

Received: 7 August 2024
Revised: 12 September 2024
Accepted: 24 September 2024
Published: 27 September 2024

1. Introduction

Exobiology Extant Life Surveyor (EELS) is a large degree of freedom (DOF) snake-like robot in development at the Jet Propulsion Laboratory (JPL) for the exploration of icy ocean worlds such as Enceladus [1–3]. Such environments offer limited communication infrastructure and highly unstructured terrains with uncertain properties. Because of this, EELS is designed to be largely autonomous without the need for extensive human operator feedback. EELS features a versatile autonomy stack, NEO, that allows the robot to sense its environment, perform high-level risk assessment and planning, and employ varied locomotion gaits [4–6]. EELS employs bend and twist actuators to alter its shape, enabling so-called shape-based locomotion gaits [7,8]. The robot also features active skin actuation in the form of rotating screws that allows mobility through tight channels where shape-based gaits are not available and over unconsolidated materials such as sand or snow [9–12]. Unlike traditional wheeled rover systems, EELS is designed to excel at moving not only in surface environments but subsurface environments as well. In the latter case, the robot can manipulate its many degrees of freedom to fit itself in between the walls of canyons, crevices, tubes, and so forth. The robot can then press itself against the walls and dig its screws into the icy material in order to keep itself from falling and traverse up or down. This article follows the content of a conference paper that was presented previously [13].

Designing and testing a robot that is intended to operate in inherently uncertain environments presents unique challenges. In many cases, the desired environment is

difficult to recreate experimentally, for example, low gravity environments or icy channels with complex shapes. Moreover, various environment variables such as terramechanics properties, terrain geometry, or lighting may simply be unknown or known with large degrees of uncertainty. Such challenges motivate the development of computer simulations to work alongside experiments in the development of snake robots [14–18]. A sufficiently versatile and high-fidelity physics simulation can empower engineers to rapidly test a variety of operational scenarios. Gravity and terrain can easily be modified in simulation. Unknown variables can be appropriately dispersed and fed into physics-based models. A new robot design can be realized in simulation by changing model properties. Moreover, while hardware is limited to serial tests one after another, simulations can be used by many engineers in parallel at once. Thus, while hardware testing is very important, a capable simulator can be a valuable asset in complimenting and accelerating the research toward the development of such robots.

In this paper, we present EELS-DARTS, a versatile dynamics simulator used by the EELS project as a closed-loop autonomy and perception algorithms test bed. Figure 1 shows a high-level overview of how the EELS autonomy stack closes the loop around the EELS-DARTS simulator. The autonomy stack has been developed to provide extensive controls [10], perception [6], and risk-aware task and motion planning [5] capabilities. In this paper, we focus on the development and features of the EELS-DARTS simulator (highlighted in green in Figure 1) and how it is used by the autonomy team. A core feature of the EELS-DARTS simulator is the ability to support the project in rapidly prototyping new robot designs and traversal scenarios. It offers the ability to ingest a standard robot description in the Universal Robot Description Format (URDF) and optimizes the multibody structure as needed. The simulator is designed to be a drop-in replacement for actual EELS hardware. A ROS [19] interface publishes and receives messages in the form of sensor data, actuation commands, and simulation actions to close the loop with the EELS onboard software in place of the actual hardware. This simulation analogue empowers engineers to rapidly prototype autonomy stack features and shake out issues before any code is deployed on the hardware. A wide variety of environments can be simulated in EELS-DARTS. Synthetic terrains can be used for both surface and subsurface robot traversal scenarios. Meshes from field-test scans of glaciers can be imported. Placement tools such as robot tethering and inverse-kinematic-based placement algorithms are available to assist in initializing the large number of degrees of freedom for arbitrary terrain shapes.

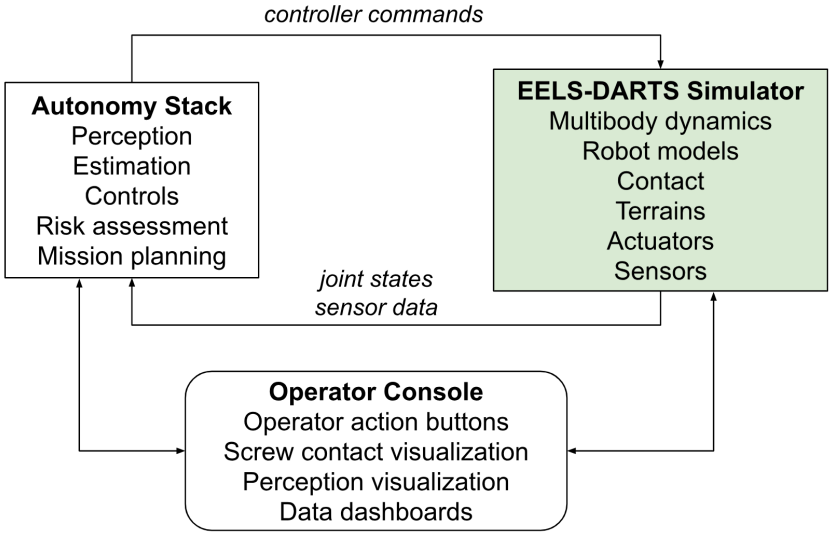

Figure 1. High-level diagram showing how the EELS autonomy stack closes the loop around the EELS-DARTS simulator (this work). An operator panel further provides useful actions and information for simulation operators.

EELS-DARTS builds upon the JPL Dynamics Algorithms for Real-Time Simulation (DARTS) framework, which is a general, multi-mission, flexible multibody dynamics engine [20]. The DARTS dynamics engine uses a minimal coordinates multibody representation and the Spatial Operator Algebra (SOA) methodology for low-cost recursive algorithms to solve the multibody dynamics [21]. DARTS has been in active development since the early 1990s and has been used for a variety of NASA and industry projects for ground vehicles, autonomous robots, rotorcraft, spacecraft, entry descent and landing (EDL), atmospheric balloons (aerobots), molecular dynamics, and other applications [22]. For a robot-level simulation, EELS-DARTS further uses the so-called DARTS Dshell framework [23], which is a collection of middleware that is used in concert with DARTS to build up fully-featured simulations for a particular application with relevant component models (actuators, sensors, avionics, etc.), terrains [24,25], and graphics [26]. A benefit of the Dshell framework is the ability to reuse tried and tested components that have been developed over decades of JPL projects without needing to start from scratch. EELS-DARTS is a novel DARTS/Dshell-based simulation with a combination of new and existing capabilities that have been adapted for snake robot applications. Our contributions in this work include new robot multibody models (Section 2.1.2), screw contact models (Section 2.2.2), terrains (Section 2.2.1), closed-loop interfaces (Section 3.2), and other features that will be discussed in detail in this paper.

The DARTS/Dshell framework provides a unique capability to the EELS project as a closed-loop autonomy testbed. It offers an extensive catalog of pre-existing capabilities in terrain modeling, GPU-based perception, and actuator models and can be readily extended with new capabilities through its C++ or Python 3 API. This capability was crucial for the project to provide a simulation framework that could expand and adapt with the project's needs. While other existing simulation tools address some aspects, none provide all of the simulation capabilities required by the EELS project. For example, while Adams [27] is widely used in engineering applications for rigid/flexible body physics simulation, it is not well suited for closed-loop controls and perception use. Recently, the NVIDIA Isaac Lab toolkit [28] has demonstrated impressive GPU-accelerated rendering and sensor modeling capabilities as well as a modular and user-friendly approach to scene creation. However, the underlying PhysX physics engine does not provide an anisotropic friction model amenable to screw–terrain interaction modeling for mobility studies. MuJoCo [29] is a recently open-sourced multibody physics engine that offers a fast, recursive dynamics formulation and anisotropic contact models but offers somewhat limited options to simulate more sophisticated screw–terrain interactions. Gazebo [30] is another popular open-source simulation framework that has seen heavy use in research and academia. However, similar to MuJoCo, it lacks the ability to rigorously simulate screw–terrain interaction.

The structure of the remainder of this paper is as follows. In Section 2, we discuss the unique design of EELS-DARTS as a snake robot simulator. This includes a discussion of the versatile multibody representation to support various distinct snake robot designs. We also discuss simulation components such as importable terrain, screw–ice contact models, joint controllers, force-torque sensors, perception models, and robot placement. In Section 3, we discuss the user interface and the ROS autonomy stack interface. Section 4 describes simulation results. In Section 4.1, we show simulation usage examples from throughout the design life cycle of the project. In Section 4.2, we perform a quantitative comparison of simulation predictions against experimental results for a surface traversal experiment on hard, synthetic ice in the lab. We also discuss the limitations of the present screw–ice contact model. Section 4.3 provides a discussion of EELS-DARTS use throughout field test planning during a recent trip to the Athabasca Glacier in Canada [31]. We end with concluding thoughts and a discussion of future work.

2. Simulator Design

In this section we discuss the underlying design of the EELS-DARTS simulator with an emphasis on aspects we believe to be unique or challenging with regards to simulating large-scale snake robots on rough terrain.

2.1. The DARTS Multibody

DARTS uses a minimal coordinates approach to represent the multibody system. Each body is connected to a parent via a joint. The joint type (revolute, prismatic, ball, etc.) determines the number of generalized position and velocity coordinates for that body. This is in contrast to an absolute coordinates approach that relies on bilateral constraints for joints. DARTS uses the Spatial Operator Algebra (SOA) methodology's fast recursive O(N) algorithms to solve the multibody dynamics [21]. While these algorithms are often applied to a tree multibody structure (where each body has exactly one parent), loop topologies and gear constraints (see Section 2.1.3) are readily supported by using the constraint embedding method [32]. To fully specify the multibody model, the mass properties of each body, such as the mass, location of the center of mass frame, and inertia tensor, are required. Currently the robots modeled in the EELS-DARTS simulator use rigid bodies connected via revolute joints, however, the DARTS framework also natively supports flexible bodies should the need arise in the future [33,34]. Further discussion of the minimal coordinates formulation is out of the scope of this paper; however, interested readers are referred to [21,34].

2.1.1. URDF Representation

The EELS-DARTS simulator can ingest Universal Robotic Description Format (URDF) files, which are XML-based robot multibody definition files widely used by the robotics community. DARTS uses a URDF parser to convert these files into its internal multibody representation. Exposing a URDF interface allows rapid iteration by decoupling robot design evolution and the simulator development. A robotics engineer can iterate upon a robot design using tools they are accustomed to without needing to worry about the DARTS-specific implementation details of the multibody.

2.1.2. EELS Robot Models

EELS snake robots are generally configured as a repeating series of modules. Each module is connected to the next one via a revolute joint dubbed a bend joint. Some EELS designs also contain revolute joints in the middle of a module that allows rotation about the axial direction of the robot, dubbed twist joints. The bend and twist joints allow the robot to adjust its shape and enable so-called shape-based locomotion gaits. Each module contains zero, one, or two screws, depending on the robot design. These screws rotate to provide so-called active skin actuation. Figure 2 shows a typical EELS robot design with two screws per module and a perception sensor head as the last module.

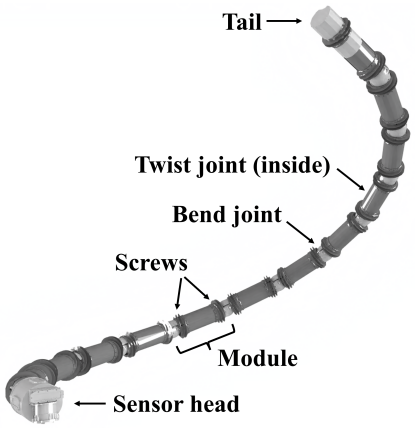

Figure 2. A typical EELS robot configuration consisting of a repeating set of modules connected by bend joints. Screws are included for active skin actuation. A perception sensor suite consisting of an IMU, a lidar, and stereo cameras is attached at the head.

Each new robot design is delivered to EELS-DARTS simulation developers as a URDF file, which is then processed into a DARTS multibody. Once the robot design has been

processed, it is available to run in simulation. The following robot designs are currently implemented in EELS-DARTS:

- EELS—An early EELS design concept with two counter-rotating screws per module. See Figure 3a.
- EELS 1.0, alternating—Version 1.0 EELS hardware. A robot built with 10 modules and an optional perception head. Used for surface traversal and perception experiments. The alternating variant has two counter-rotating screws per module. See Figure 3b.
- EELS 1.0, front only—The "front only" variant removed the rear screw for each module resulting in one screw per module. See Figure 3c.
- EELS 1.5—Version 1.5 EELS hardware. A low-cost EELS robot built with off-the-shelf components. Designed exclusively for subsurface experiments, EELS 1.5 features a cross bar in the middle of the robot with an avionics box. See Figure 3d.
- Mini EELS 1.0—Smaller, low-cost robot with only bend joints and no screws, suitable for tasks such as climbing up small pipes. Primarily used as a test bed for reinforcement learning-based locomotion gaits. See Figure 3e.

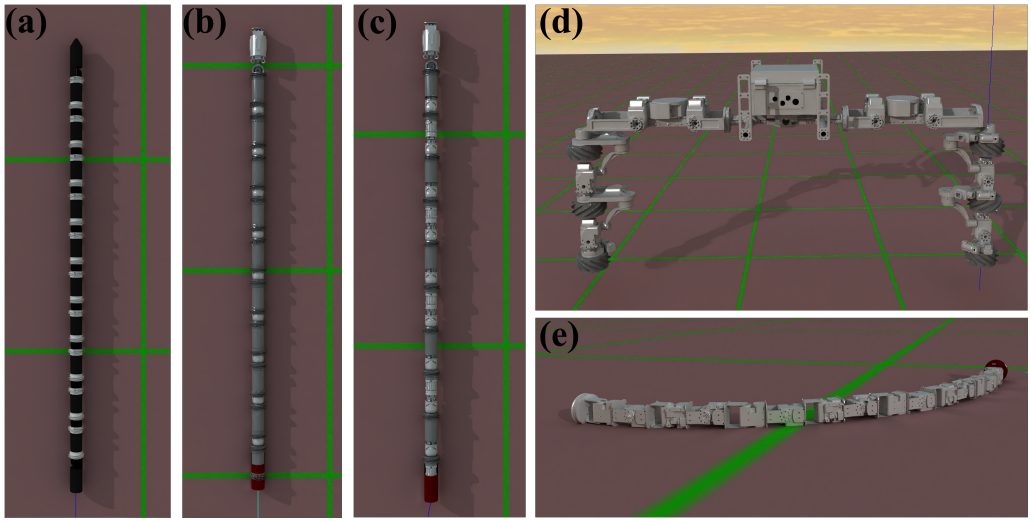

Figure 3. Various distinct EELS robot designs simulated in the EELS-DARTS simulator. (**a**) A preliminary EELS conceptual design with capped head. (**b**) EELS 1.0 configuration with two counter-rotating screws per module and perception head. (**c**) EELS 1.0 configuration with one screw per module. (**d**) EELS 1.5 robot model with cross bar and avionics box, designed for subsurface traversals up and down between channel walls. (**e**) Mini EELS 1.0 robot model.

2.1.3. Constraint Embedding for Counter-Rotating Screws

Some EELS robots are built with counter-rotating screws achieved through hardware transmission coupling. Counter-rotating screws naturally always come in pairs and are referred to as front and rear screws. As previously mentioned, the DARTS framework readily supports constraint embedding [32]. The mechanical coupling constraint between the counter-rotating screws is enforced using DARTS' constraint-embedding technique [32]. As a result, the original two degrees of freedom (DOF) are reduced to one DOF and a counter-rotating constraint. The constraint embedding approach enables the use of the O(N) low-cost minimal coordinate dynamics algorithms even in the presence of such constraints.

2.1.4. Multibody Optimization

Though DARTS uses the recursive O(N) algorithms that arise from using a minimal coordinates approach, the EELS robot multibody can still be relatively large. An EELS robot with ten modules—each module possessing one bend joint, one twist joint, and one screw—has thirty-six DOF (three revolute joints per module plus six free DOFs at the tail). Moreover, robots described by URDFs can contain mass-less links with locked joints for specific locations on a particular body, for example, camera sensor frames and control

frames. For the EELS project, the number of such mass-less links can be large, on the same order of magnitude as the number of actual bodies. The DARTS solver implements an optional multibody optimization step which prunes away such mass-less bodies from the multibody tree and replaces them with simple frames to allow them to reduce the dynamics calculations while still maintaining the frame information. This multibody optimization step also combines bodies that are locked together. This capability allows using the raw robot mass properties followed by optimizing the internal multibody model for maximizing the efficiency in the dynamics calculations. These multibody optimizations led to significant run-time performance gains when simulating such large-DOF snake robots.

2.1.5. Configuring the Robot Definition at Launch Time

The typical workflow for launching the simulator involves selecting a robot from among a set of pre-processed robot models (EELS 1.0, EELS 1.5, etc.). The simulator does provide a number of ways to reasonably modify the existing robot models at launch time. For example, a user may wish to run with the EELS 1.0 robot but change the number of modules from 10 to 4. To accomplish this, the user may pass this option to the simulation via the command line interface (described in detail in Section 3.1). The simulation internally uses a parameterized representation of the multibody model based on the repeating nature of the EELS modules, which allows the specification of the number of modules from the command line. Thus, a robot model with four or ten modules is easily obtained.

The simulator also provides the option to pass in a new URDF file to process at launch time, provided it is sufficiently similar to an existing robot model. This is to ensure that previously defined conventions for the design and structure of the multibody are not accidentally violated. This functionality is intended to allow parameter-level changes such as mass properties, joint positions, joint angles, meshes, and so on. For example, consider a scenario where an autonomy engineer runs an EELS-DARTS simulation and then runs the subsequent experiment on hardware. Upon performing this, the engineer notices that the URDF joint axes are flipped from what they are seeing on hardware. Applying this correction does not require changes to the simulator, and instead can be performed by simply specifying the corrected URDF for the simulator. A permanent change can be folded into the simulator at a later stage. Such flexibility, however, requires the corrected URDF to be structurally similar to the baseline model within the simulator.

2.2. Component Models

In addition to the multibody dynamics, the overall robot simulator includes models for actuator devices and sensors that interact with robot dynamics. Joint controllers, force-torque sensors, inertial sensors, cameras, and lidars are all examples of components that comprise an EELS robot and need to be represented by a mathematical model that is of sufficient fidelity to meet the closed-loop needs of the project. There are also models for the interaction between the EELS robot and the environment it is operating in. We use the DARTS Shell (Dshell) simulation framework [23] for developing component models for these and integrating them into a dataflow for the overall simulation. We discuss each of these Dshell component models in this section, along with models for collision detection, screw–terrain interaction, and robot initialization.

A key aspect of the DARTS design philosophy is model and code reuse across multiple projects whenever possible. For instance, the camera models used by the EELS-DARTS simulator were previously developed, used, and validated by previous projects [26]. Likewise, the screw–ice contact model written for EELS will be available to future projects that might need it.

2.2.1. Terrain

A key design feature of the EELS robot is the ability to operate in a wide variety of environments and scenarios. The environment types used for EELS scenarios are surface and subsurface. Surface scenarios involve the robot traversing across the surface of rough terrain using either screw-based or shape-based locomotion. Subsurface scenarios involve

the robot operating inside a channel, vent, crevasse, or similar enclosed environment, requiring the robot to hold itself and resist gravity by using its shape actuators to push against the enclosing walls. Subsurface scenarios can also involve vertical traversal using the screw actuators. A real-world mission scenario will involve a combination of both surface and subsurface segments.

The varied scenarios that EELS can operate in necessitate a flexible interface for specifying and loading terrain data in the EELS-DARTS simulator. EELS-DARTS leverages the broad capabilities and terrain representations available in the DARTS SimScape middleware [24,25]. An important modeling need is collision detection and contact modeling for the robot/terrain interaction. This requires an interface to query contact location, penetration depth, and contact normal for a desired (x, y, z) location. Figure 4a shows an example of a simple subsurface environment created by importing a primitive cylinder tube with a user-defined radius. Figure 4b shows a surface environment created by importing a Digital Elevation Model (DEM) terrain using SimScape. DEMs in SimScape are serialized and stored in a format that is easily accessible at run-time and portable across projects. For example, once generated and saved in a store, a Mars DEM developed for a rover mission can be used by other projects later on. There are a wide variety of planetary and synthetic DEMs to choose from. Figure 4c shows an example of a user-created mesh that has been imported as a terrain. Users can create a mesh using any 3D modeling software of their choice and then import it into EELS-DARTS. In this case, a parallel wall subsurface environment was imported. Finally, Figure 4d shows an example of a mesh that has been created from lidar scans of a channel in a glacier in Athabasca, Canada. Such scans obtained by the EELS field test team can be converted into a mesh and imported into EELS-DARTS.

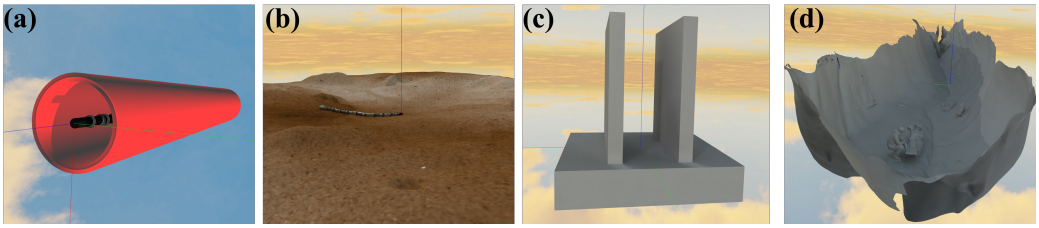

Figure 4. Examples of different terrains imported into EELS-DARTS for a mix of surface and subsurface traversal scenarios: (**a**) Shows a simple hollow cylinder used to simulate a subsurface vent environment. (**b**) Shows a surface DEM topography. (**c**) Shows an example of a parallel-wall subsurface mesh that was created by a user in a 3D modeling software and then imported into EELS-DARTS. (**d**) Shows a field test scan of a glacier channel from the Athabasca Glacier in Canada imported into EELS-DARTS.

2.2.2. Collision Detection and Contact Models

Modeling contact forces can generally be split into two parts: (1) Collision detection to determine contact points between simulation objects; (2) Computation of contact forces at those locations. Collision detection also gathers information on the collision—depth of penetration, relative velocities between the two colliding objects, etc.—that will be used to calculate the contact forces. The second step uses information gathered in the first step to actually compute the contact forces.

EELS-DARTS offers a choice between distinct back-ends to handle collision detection. Both back ends make use of the open-source Bullet library [35]. One back-end uses convex-hulls representations of the meshes, shown in Figure 5. The other uses a signed distance field (SDF) representation of the terrain and a hand-selected set of possible contact points on the screws to enable faster collision detection between the screws and the terrain. This second method ignores body-to-body collisions and instead leverages fast SDF queries for body-to-terrain collisions. Querying collisions in this manner results in significant run-time performance improvements at the cost of some reduction in simulation fidelity.

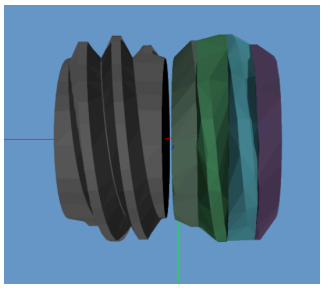

Figure 5. Original screw collision mesh (**left**) and convex hull representation of collision mesh (**right**). Each convex hull in the right mesh is shown using a different color. The convex hull representations of collision meshes are used by one of the Bullet collision back-ends to perform collision detection. The convex hull decomposition process is again another instance where we can trade off accuracy for performance, e.g., reducing the number of convex hulls will improve performance but may cause the discrepancy between the convex hull representation of the collision mesh and the actual mesh to be larger.

At a high level, EELS-DARTS needs to simulate contact forces that result from the interaction of screws with the terrain. In reality, the terrain deforms as a result of these interactions. However, simulating terrain deformation is a computationally intensive process. A trade-off is made to simulate the contact forces on the EELS robots without deforming the terrain. This trade-off is sufficient for screws interacting with hard ice and less so for modeling interaction with loose, unconsolidated materials such as snow or sand. To compute these forces without terrain deformation, a spring–damper normal force plus anisotropic friction model was used. This model allows one to simulate the skewed traction forces that are produced by the threads of screws interacting with ice or other hard surfaces without needing to calculate and simulate terrain deformation.

Figure 6 shows a comparison between an isotropic friction model and an anisotropic friction model. The symbols used in this figure are described mathematically in the following paragraphs.

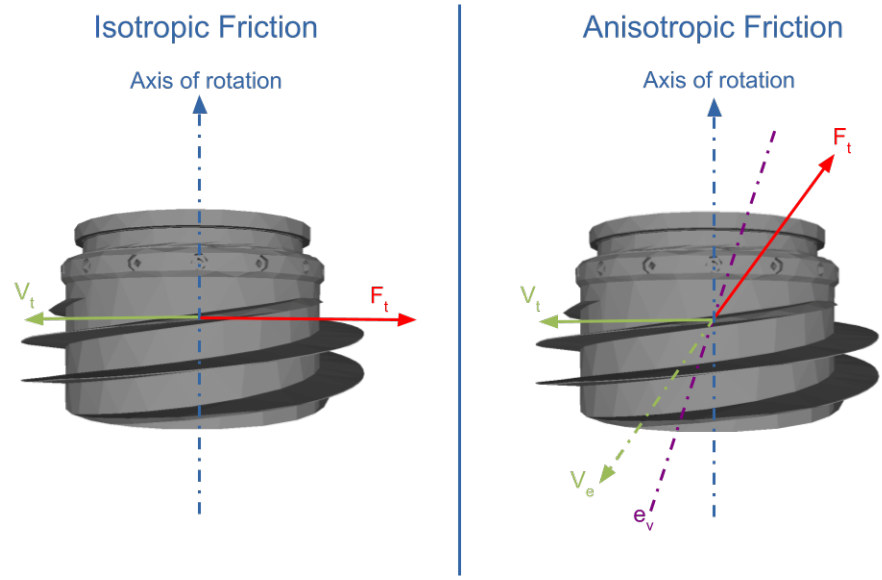

Figure 6. Depiction of anisotropic friction and associated tangential velocities (**right**). For reference, the classic isotropic friction model is shown on the (**left**).

In a classic isotropic friction model, the friction force, F_t, is calculated as simply

$$F_t = -F_n \mu \hat{v}_t \tag{1}$$

where F_n is the normal force at the contact point, μ is the friction coefficient, v_t is the tangential velocity between the colliding bodies at the collision point, and $\hat{v}_t = v_t / ||v_t||$

is the unit vector in the direction of v_t. The friction coefficient between the two bodies is invariant to the direction of the tangential velocity. The penalty method contact force in the normal direction, F_n, is calculated as

$$F_n = (k\,d + c||v_n||)\hat{v}_n \tag{2}$$

where v_n is the component of velocity parallel to the contact normal, k is the linear spring constant for the contact, c is the damping coefficient, and d is the penetration distance.

In contrast, in an anisotropic friction model, the friction coefficient varies depending on the tangential velocity direction, leading to a different formulation for F_t. This anisotropic model defines two friction coefficients: one parallel and the other perpendicular to the axis e_v. The tangential velocity direction is decomposed into components parallel and perpendicular to this axis, which are ultimately used to calculate the friction force. Mathematically,

$$F_t = -||F_n|| \begin{bmatrix} \mu||e_v|| & 0 & 0 \\ 0 & \mu & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ 0 \end{bmatrix}, \tag{3}$$

where t_1 is the magnitude of the component of $\hat{v}_t$ parallel to e_v and t_2 is the magnitude of the component of $\hat{v}_t$ perpendicular to e_v. Substituting the following

$$t_1 = \hat{v}_t \cdot \hat{e}_v, \tag{4}$$

$$t_2 = ||\hat{v}_t - \hat{e}_v t_1||, \tag{5}$$

into Equation (3) and simplifying yields

$$F_t = -F_n \mu \left(\hat{v}_t + (||e_v|| - 1)(\hat{v}_t \cdot \hat{e_v})\hat{e_v} \right) \tag{6}$$

where $\hat{e_v} = e_v/||e_v||$. The effect of this model is to essentially skew v_t to be v_e as indicated in Figure 6. Note the case where $||e_v|| = 1$ recovers the isotropic case given in Equation (1). Values of $||e_v|| \neq 1$ indicate anisotropy is present.

Experimentally, all one needs to determine in order to use this model for F_t are the values of μ and e_v. The direction of e_v is based on the angle of the screw threads since the difference in screw–terrain interaction normal and parallel to the screw threads is what causes the difference in modeled friction. The magnitude of e_v and μ are determined by experiments. These experiments involve dragging a screw along the terrain normal and parallel to the threads and measuring the force it takes to keep it moving at a constant velocity. This is performed with different amounts of weight on the screw to simulate different normal forces as the magnitude of e_v changes with the normal force applied.

Figure 7 shows an example of the aforementioned experimental data and the calculation process for μ and $||e_v||$.

The top-left plot of Figure 7 shows the friction coefficient parallel to the screw thread as a function of normal force. The friction coefficients were determined by dividing the force it took to drag the screw at a constant velocity parallel to the threads by the normal force. The data show that the parallel friction coefficient is approximately the same, regardless of the normal force. Hence, the average of these values was used as μ. The top-right plot of Figure 7 shows similar data for the friction coefficient normal to the screw threads: these data were collected in the same way as the parallel friction coefficient, but by dragging the screw normal to the threads rather than parallel to the threads. The normal friction coefficient changes with the normal force. The $||e_v||$ values in the bottom plot of Figure 7 were obtained by dividing the normal friction coefficient data from the top-right plot by the value of μ obtained from the top-left plot. Then, a linear fit was applied to the data. The results of the linear fit and value of μ are used in the EELS-DARTS anisotropic friction model as μ and $||e_v||$ for contact points. A validation experiment performed to study

how well this simple anisotropic friction model matched real-world behavior for a surface traversal scenario on hard synthetic ice is discussed in Section 4.2.

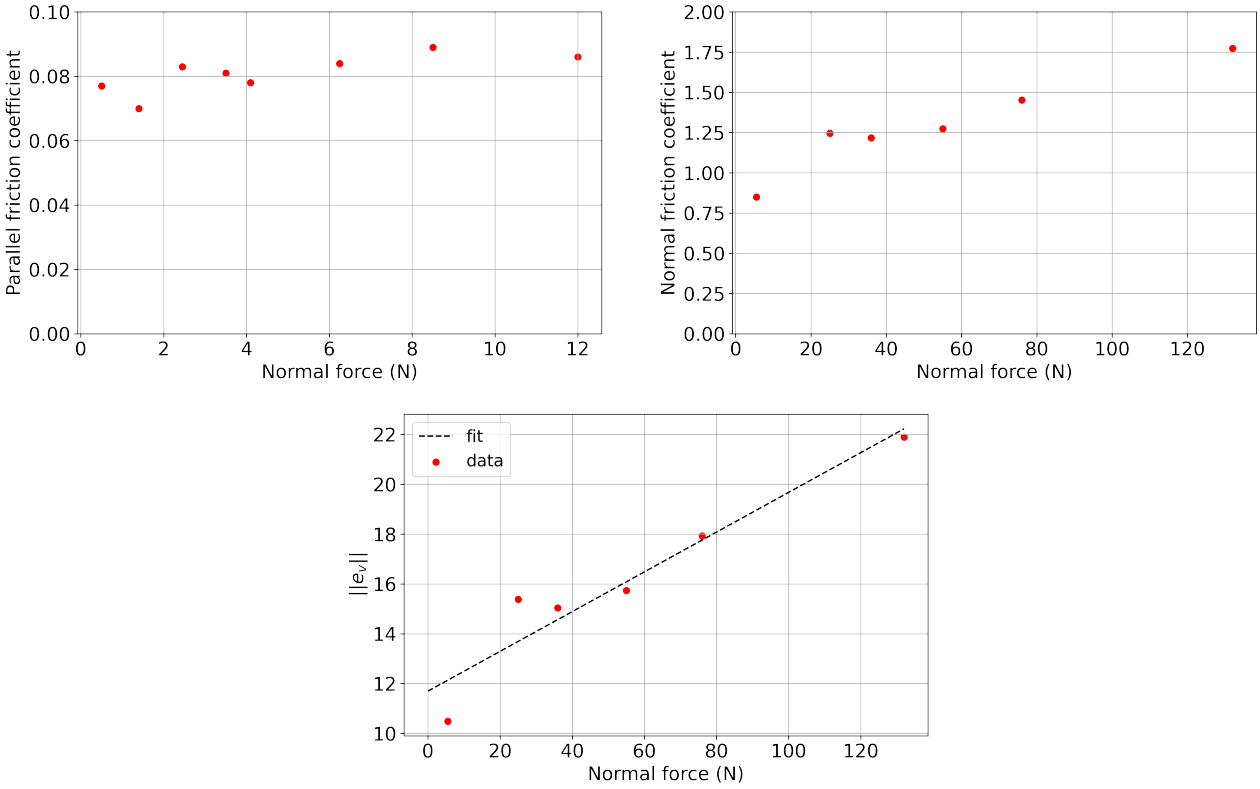

Figure 7. Calculation of $||e_v||$ from experimental data. **Top left**: friction coefficient parallel to the screw threads vs. normal force. **Top right**: friction coefficient normal to the screw threads vs. normal force. **Bottom**: $||e_v||$ vs. normal force.

2.2.3. Graphics and Perception Models

Development and testing of guidance, navigation, and control (GNC) for closed-loop autonomy in the loop with perception requires simulation of raw sensor data. EELS-DARTS uses IRIS-DARTS [26] for camera and lidar sensor simulations. IRIS-DARTS provides high-fidelity, real-time camera, and lidar simulation via a versatile and extensible ray-tracing pipeline for accurate engineering quality sensor modeling.

IRIS-DARTS uses the GPU-accelerated ray-tracing OptiX library [36] for real-time rendering, with implementations of Whitted ray tracing and path-tracing for varying levels of lighting fidelity. Path tracing optionally includes a machine-learned (ML) denoiser that eliminates sampling noise with limited stochastic ray samples. This pipeline allows for adjustment of lighting fidelity within the same virtual environment to provide sufficient accuracy-speed trade-off adjustment for a wide range of perception tasks.

IRIS-DARTS supports multiple camera models, including pinhole, fish-eye, and CAHV/CAHVOR/CAHVORE models. The CAHVORE family of models [37] are used by the machine vision community to capture non-idealities due to optical distortion and sensor alignment. The first four parameters (C, A, H, V) define the camera extrinsics and sensor plane offsets for cameras that deviate from the pinhole assumption. The CAHVOR model extends this to include a non-ideal optical axis, O, and three radial lens distortion parameters captured by R. The lens distortion is further extended by CAHVORE to support a non-central entrance of light through the aperture. The entrance vector in this model is captured by E. Examples of these models are shown in Figure 8. In addition to lens distortion, IRIS-DARTS supports camera noise, lens and aperture vignetting, color, and gamma adjustments via a general-purpose and extensible filter pipeline to support arbitrary image processing algorithms as part of any sensor model.

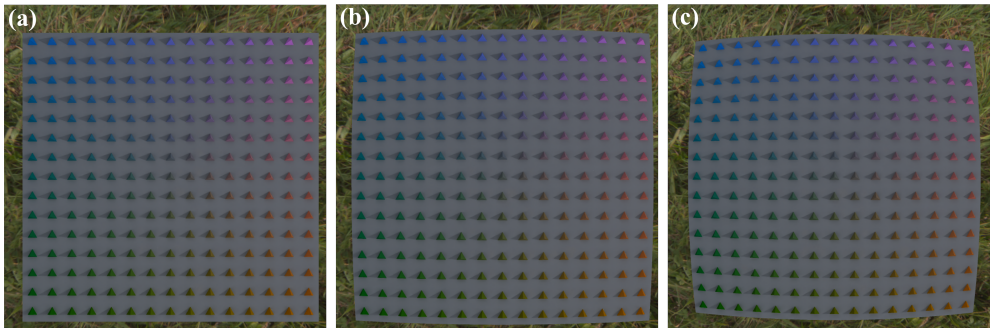

Figure 8. Example (**a**) CAHV, (**b**) CAHVOR, (**c**) CAHVORE images showing camera distortion with C = (0, 0, 35), A = (0, 0, −1), H = (734.264, 0, −310), V = (0, −714.264, −320), O = (0, 0, −1), R = (0.05, −0.21, 0.01), E = (0.12, −0.14, 0.13).

The lidar in IRIS-DARTS is implemented using the same ray-tracing architecture as the camera, ensuring consistent generation of multi-modal sensor data. Since physical lidars have a wide variety of beam patterns, IRIS-DARTS supports both a parametric grid, as commonly found in scanning lidars, as well as a user-defined beam pattern specified by an array of origins and directions for the beams. EELS-DARTS currently implements a number of different beam patterns to match lidar units installed on various versions of EELS hardware. The data produced by the lidar model include the points as well as the intensity of the hit-point, modeled with diffuse reflectance. More complex reflection models are planned for future work.

IRIS-DARTS provides visualization capabilities within EELS-DARTS to assist in simulation setup and autonomy development by providing debug information such as frame axes and labels, wheel tracks, and live sensor data within the scene (see Figure 9). Viewing the sensor data from a third-person perspective can assist with the development of sensor coverage and provide insights and context to the sensor's perspective and data. To ensure the accuracy of sensor data while simultaneously providing debugging and context information to the user, objects can be masked as physical or ornamental, with sensors only capturing physical objects, whereas a visualization viewport can show physical and ornamental geometry.

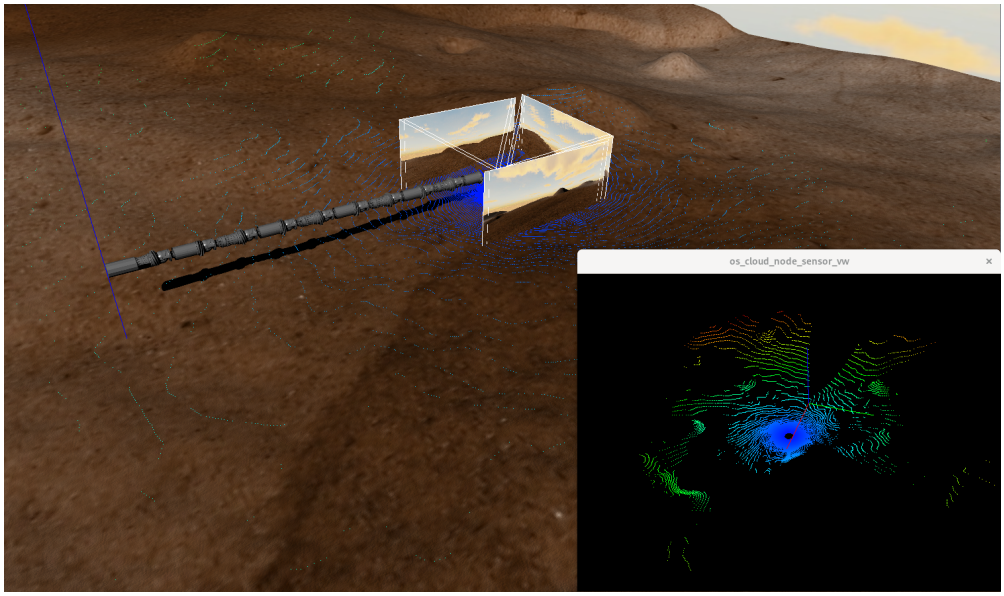

Figure 9. EELS-DARTS simulation with three stereo camera pairs and a lidar, with data displayed in a planview viewport, providing insight and context. A second viewport (lower right) exclusively displays the simulated lidar point cloud. The color of the points encodes distance from the sensor, blue being the closest and red being the farthest.

2.2.4. Force-Torque Sensor Models

Some EELS robots, such as EELS 1.5, are built with 6 DOF force-torque sensors (FTS) installed between some of the modules. The purpose of the FTS array is to provide control algorithms with feedback about the state of the contact between the robot and the walls when operating in subsurface environments. In EELS-DARTS, each FTS is treated as a separate body that is connected to its parent via a locked zero DOF hinge. The hinge and body are positioned to match the sensing frame of the transducer. Using this approach, the FTS component models implemented in EELS-DARTS can compute the FTS wrench by directly extracting the inter-body spatial forces between the FTS body and its parent body. Inter-body spatial forces are readily available within the DARTS dynamics solver, and so these queries for the FTS model incur little extra computational cost. The force-torque wrench is then published by the ROS server (see Section 3.2) as a standard WrenchStamped message. The wrenches are currently published as ground truth measurements without noise, however, noise characteristics can be easily implemented to suit project needs in the future if desired.

2.2.5. Joint Control

The EELS robot's autonomy stack is responsible for higher-level tasks such as screw velocity allocation control, motion planning, proprioception, exteroception, risk estimation, and mission planning. A detailed description of the autonomy stack is out of the scope of this paper. However, interested readers are referred to [10] for a detailed description of the joint control strategies that exploit the anisotropic properties of screws to enable complex motion control, to [6] for a description of the perception and localization algorithms, and to [5] for a discussion of the risk-aware task and motion planner. At the lowest level of the autonomy stack, these actions are translated into desired joint positions and/or velocities. These commands are then published and sent to EELS-DARTS as input to joint controllers implemented within EELS-DARTS. Because EELS-DARTS is intended to be a simulation analogue for hardware, the joint controller options implemented in EELS-DARTS are relatively basic, the default being a standard PID controller similar to the ones implemented in hardware. The gains for these controllers are configurable by the users and were generally tuned ad-hoc to yield low tracking error. This allows autonomy engineers to focus on autonomy stack development and not worry about the lower-level control issues. It is common for DARTS simulators to interface with the full flight software (FSW) stack during later stages of a project, and this will be done in the future for EELS-DARTS as the autonomy stack matures.

2.2.6. Robot Initialization and Placement

The start of a simulation scenario involving operation on rough terrain requires the computation of an initial state for the robot that places it appropriately at the start location and pose. Solving for the placement state for the EELS snake robot of arbitrary shape and size within arbitrarily shaped environments (glaciers, channels, vents, crevasses, etc.) is non-trivial since there can be many contact points required that are not known a priori. EELS-DARTS includes a placement algorithm that solves for the 30+ DOF state values that set the robot's shape so that it is in non-penetrating contact with the rough terrain surface. The placement algorithm can be broadly viewed as solving the inverse kinematics (IK) for the robot's state and terrain contact points, satisfying the user-specified initial constraints and those from the terrain topography.

EELS-DARTS breaks the placement problem down into two distinct parts: (1) Initializing with a preliminary shape guess; (2) Followed by a settling phase that "pulls" the robot into the terrain. As a first step, the snake robot is initialized with an initial guess of the shape. This initial guess necessarily requires some form of domain knowledge about what shape is optimal for the given scenario and is informed by autonomy engineers. For example, if the robot is starting inside a cylindrical channel, the initial guess may be a helix shape, similar to what is shown in Figure 2. The second step then proceeds by "pulling"

the snake robot into the terrain by disabling gravity and applying attraction forces to many points on the robot. For a given point, the attraction force acts in the direction between the given point on the robot and the shortest distance to the terrain. The force applied is proportional to the distance to the terrain, similar to a spring. This process is simulated until equilibrium has been reached and the robot is in a sufficient state of contact with negligible movement. The autonomy system then takes over and commands the robot to actively maintain this initial state at which point gravity is re-enabled. Figure 10 shows an example of this placement process inside a rough cylindrical terrain.

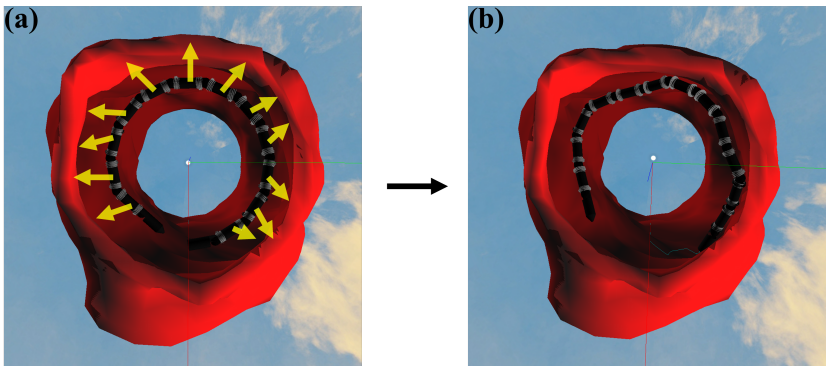

Figure 10. Example of an initialization and placement process inside a rough cylindrical subsurface environment. The red, green, and blue lines represent the x, y, and z axes of inertial coordinate system, respectively. In this view gravity is acting into the page: (**a**) Shows the robot initialized as a helix. The yellow arrows represent attraction forces that pull each module of the robot in toward the terrain. (**b**) Shows the final placement of the robot.

There is also an option to attach an arbitrary number of "tethers" to help support the robot during initialization, similar to how the robot would be rigged up with ropes during an actual field experiment (see example test setup shown later in Section 4.3.1). These tethers are implemented as simple stiff spring–damper models that can be attached to arbitrary points on the robot and anchored to fixed points in space. This placement framework greatly improved the success the EELS autonomy team had in initializing the simulator state to match the prototyping and testing experiments in the field.

3. Using the Simulator

In this section, we discuss how users launch and interact with an EELS-DARTS simulation. A command-line interface is provided to launch and specify options, and a Robot Operating System (ROS) interface is used to close the loop with the robot control and autonomy software.

3.1. Command Line Interface

The EELS-DARTS application provides an extensive command line interface (CLI) to run and configure the simulation. The CLI referred to as Dclick builds upon the open source click Python package. Most of the CLI options are used to specify settings and parameters to configure the simulation and set up a desired scenario. For example, a small subset of the available CLI options are as follows:

- Sim—select robot to use, toggle interactive command-line mode, etc.
- Integrator—select integrator type, tolerances, step size, etc.
- Graphics—select graphics back-end to use, path-tracing options, skybox settings, etc.
- Contact—select contact model to use, set parameters for contact models.
- Terrain—select terrain to use.
- Perception—enable/disable cameras and lidar, set parameters for perception models.
- Joints—enable/disable spring–damper models for joints, set spring–damper parameters.
- ROS—enable/disable ROS interface, set ROS topic publishing rate.

Fully configuring a simulation can involve specifying tens or hundreds of settings and parameters. To avoid the need to write everything out at the command line and to enable repeatability, Dclick supports saving and loading configuration files. Once a simulation has been run, a full configuration file can be saved, edited, and loaded later as needed. A combination of command-line arguments and configuration files can be used as input to run the simulation, and precedence is given to the options specified at the command line. Any setting that has not been specified by a user reverts to a default value.

3.2. ROS Interface

Once the simulation has been configured and launched, it can be interacted with in real time using ROS. ROS is a widely used open-source robotics software development platform. EELS-DARTS implements an interface that uses the ROS topics and messages system to send and receive standardized packets of information to the EELS autonomy stack. Since the EELS-DARTS simulator is intended to be a sufficiently similar drop-in replacement for actual robot hardware, EELS-DARTS publishes ROS messages such as joint states and sensor outputs (IMUs, cameras, lidars, etc.) that are published by the hardware. Moreover, EELS-DARTS can process ROS messages from the autonomy stack to control various aspects of the simulation in real time, for example, joint controller set points, placing/moving the robot, toggling gravity, and so on. Figure 11 shows a simplified graph of published and subscribed ROS topics for EELS-DARTS.

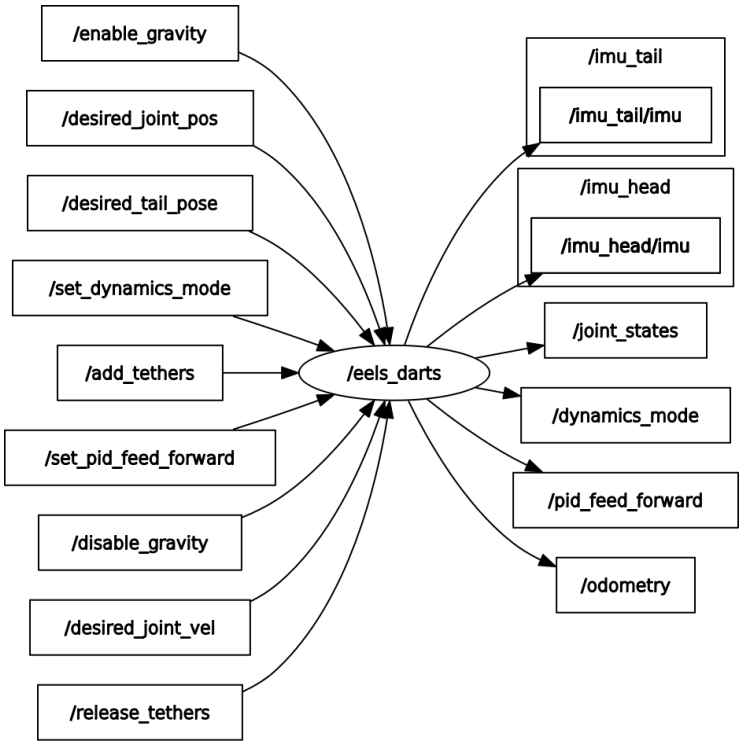

Figure 11. Graph of ROS topics that EELS-DARTS publishes and subscribes to. Some topics, such as /joint_states, provide information about the state of the robot, while others, such as /enable_gravity, allow users to interact with the simulation in a uniform manner.

4. Results

In this section, we discuss a variety of simulation results from the EELS project development. We first present a selection of simulation scenarios that were run to aid and inform the design process. We then perform a brief validation analysis for the anisotropic friction contact model. We finish with a description of a recent field test to the Athabasca Glacier in Alberta, Canada, and discuss how the EELS-DARTS simulation was used during field-test operations planning.

4.1. Example Uses of the EELS-DARTS Simulator

The EELS-DARTS development and evolution continued side by side with the rest of the EELS project from the early stages. The simulator has been used to investigate a wide variety of scenarios ranging from simple to complex. For example, one of the first studies performed was a tethered dangling test inside a synthetic cylinder tube, shown in Figure 12a. Next was an investigation of the surface traversal capabilities of the EELS 1.0 robot using screw-based and shape-based locomotion gaits, shown in Figure 12b. Then, scans from field tests were brought in, shown in Figure 12c. Additionally, leading up to the field test discussed in Section 4.3, the EELS 1.5 robot model was tested in a variety of subsurface environments, shown in Figure 12d. As mentioned previously, the simulator was also used to test perception algorithms, utilizing the IRIS-DARTS camera and lidar models available in DARTS. An example of this is shown in Figure 13.

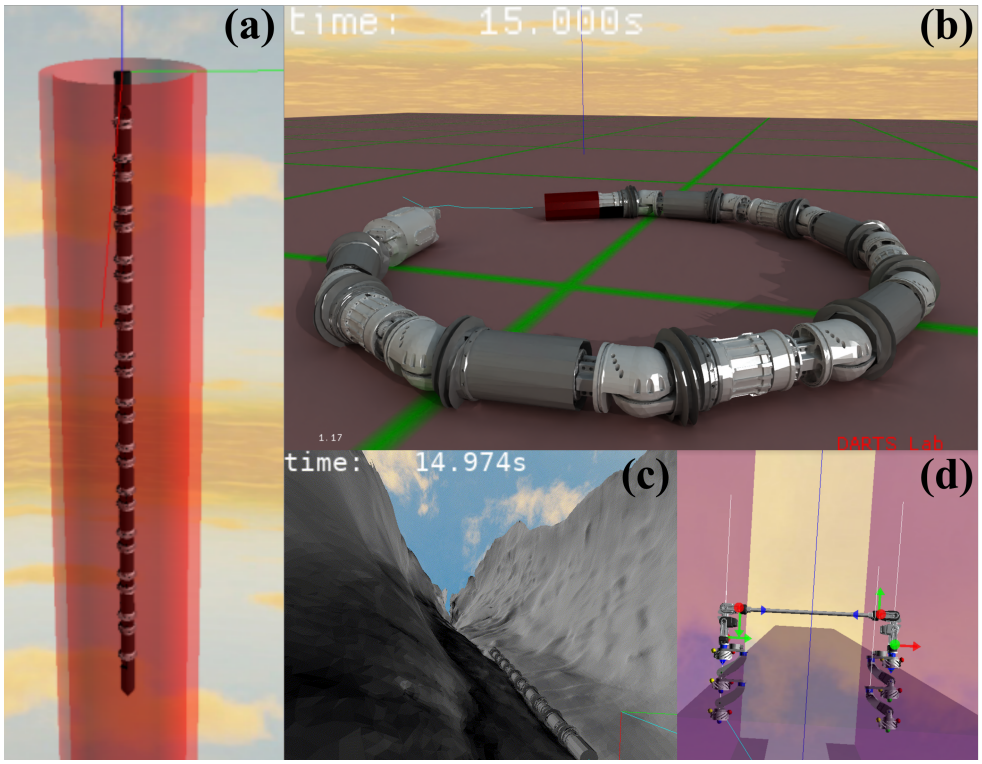

Figure 12. (**a**) An example of a synthetic cylinder-tube subsurface environment. (**b**) EELS version 1.0 initialized into a crescent shape during a surface traversal scenario on flat synthetic terrain. (**c**) EELS 1.0 traversing inside a field-test scan of a glacier channel taken from Athabasca, Canada. (**d**) EELS version 1.5 rigged into a synthetic U-channel subsurface environment. The white lines indicate tethers used to support the robot before the robot has fully supported itself by pushing against the walls. The red, green, and blue arrows represent the x, y, and z axes of each module's coordinate system, respectively.

One important aspect during testing was simulation runtime performance. It was required that the simulation run at least roughly 75% real time, meaning running a 30 s simulation would take 40 s to execute on the computer. Preferably, the runtime performance is closer to 100% real time. This requirement was due to a few reasons. First, closing the loop with the autonomy stack requires that the simulation run in lockstep with the stack, which operates in real time. If the simulation lags too far behind the stack, this causes issues when exchanging commands and information. Additionally, engineers needed to rapidly test the simulation with as little turnaround time as possible. Waiting for simulations to finish running could cause a buildup of delays in testing schedules. We found the improvements that made the most significant speedups were as follows: including a multibody optimization step (discussed in Section 2.1.4), using SDFs to represent terrain

instead of meshes for collision detection (discussed in Section 2.2.2), applying regulators to the anisotropic friction model to prevent integrator thrashing at rest, and selecting an integrator method that was suited for stiff dynamics, due to a large number of contacts and PID controllers handled by the simulation. With these improvements, the simulation was able to run at roughly 75% real time on compute-constrained hardware such as laptops and up to roughly 100% real time on dedicated workstations.

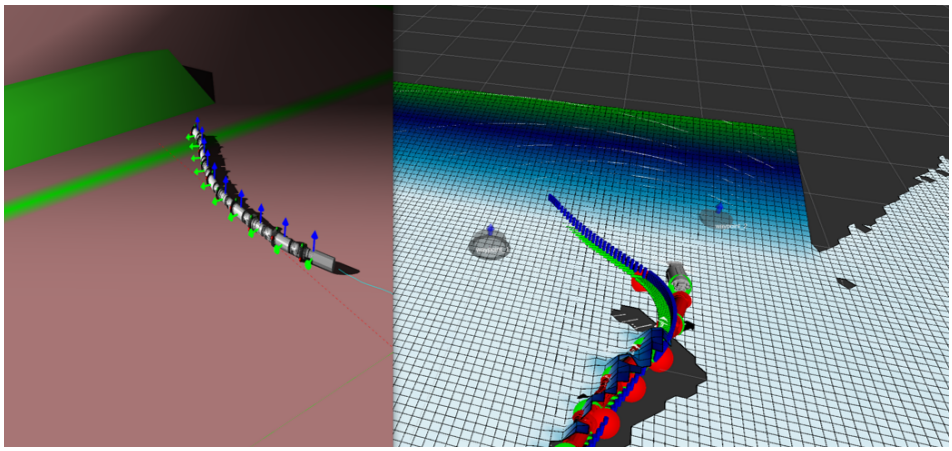

Figure 13. An example of a perception stack test with EELS-DARTS. Shown is the EELS 1.0 robot traversing inside an obstacle course in EELS-DARTS (**left**). The red, green, and blue arrows on the left represent the x, y, and z axes of local coordinate systems on the robot, respectively. The outputs from the lidar model from EELS-DARTS are fed into the perception stack to generate obstacle cost maps in order to test trajectory planning algorithms (**right**). The color grid overlaid on the right represents the terrain elevation estimated by the autonomy system, white being zero, and green being higher elevation. The blue line represents a desired path, and the red spheres indicate where the robot modules are positioned relative to the desired path.

4.2. Contact Model Validation

A brief validation experiment was performed following the development of the anisotropic friction model to quantify how closely the model could capture real screw behavior on hard consolidated ice with negligible screw penetration. The experiment first involved a surface traversal in a lab environment with the EELS 1.0 robot initialized in a crescent-shaped pose, shown in Figure 14. Then, with the bend and twist actuators locked, the robot rotated the screws in various combinations of speed and direction to move itself about the surface. The surface material was hard synthetic (plastic) ice. The position and velocity of each joint were recorded by hardware transducers. The two-dimensional pose (XY position and yaw angle) of the unactuated (free) tail segment was measured with respect to an inertial frame using a VICON system in the lab. The VICON measurements were considered the "ground truth" for the free degrees of freedom of the robot.

The experiment was then repeated in EELS-DARTS. The previously recorded joint positions and velocities were fed as set points into PID controllers in the simulation in order to mimic the screw rotations executed by the robot on hardware. The PID controllers in the simulation were tuned to yield very low tracking errors. Finally, the resulting motion of the free degrees of freedom of the tail in the simulation was compared to the ground truth values measured in the real world. Mimicking the motion of the actuated degrees of freedom of the screws as closely as possible in simulation allowed the macroscopic behavior of the anisotropic friction model to be verified. If the screws rotated the same in simulation as on hardware, it was expected that the resulting motion of the robot should be the same. The results of this comparison are shown in Figure 15.

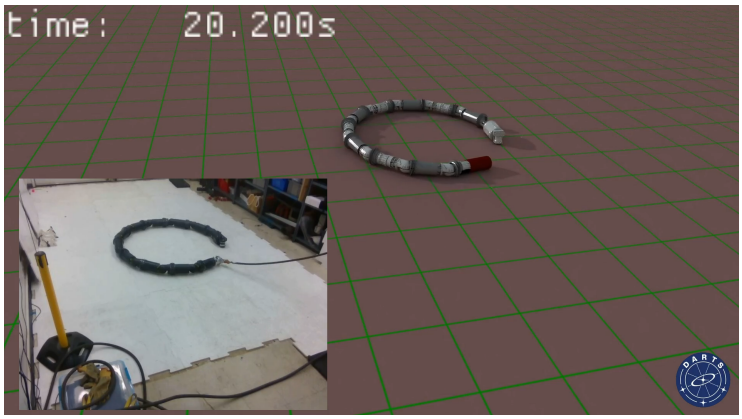

Figure 14. Camera still from a surface traversal experiment on hard synthetic ice with the EELS 1.0 robot at JPL (inset) and the same setup recreated in simulation. The initial joint positions were measured by transducers on the hardware and used to initialize the robot multibody in simulation.

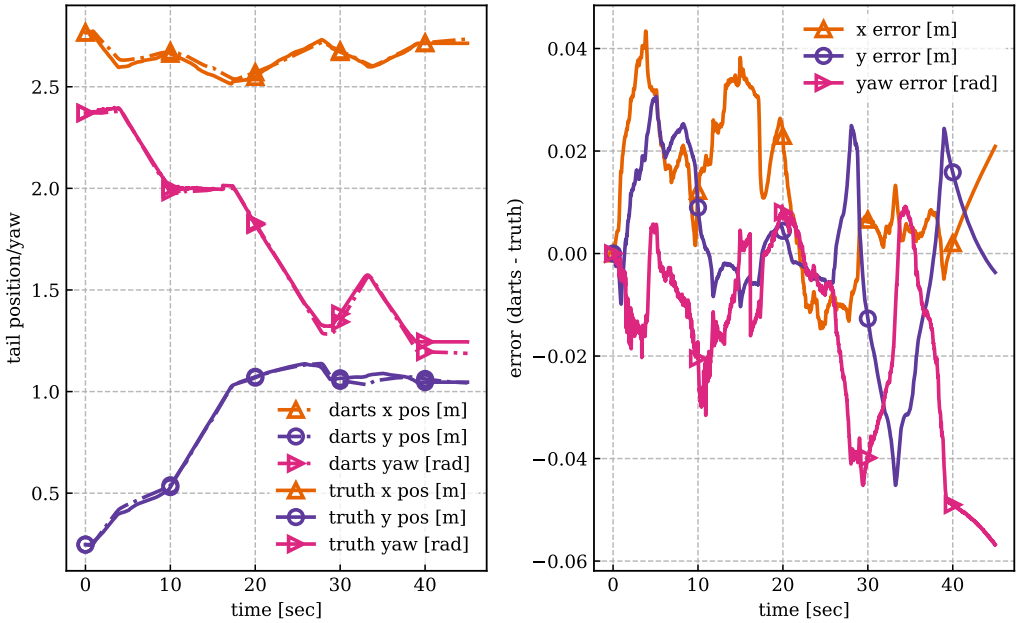

Figure 15. Measured robot motion compared with simulation predictions for a 45 s screw-based surface traversal experiment on hard synthetic ice. The left plot shows measured tail x, y position and yaw as a function of time compared with the same values predicted using the anisotropic friction model in simulation. The right plot shows the error (simulation-truth) as a function of time.

The results shown in Figure 15 indicate that the macroscopic behavior of the screws is captured well by the anisotropic friction model. The resulting motion of the robot in simulation and experiment match well. Over the course of a 45 s traverse, the largest tracking error was slightly more than 4 cm in position and roughly 0.06 radians for rotation. There does not appear to be a systematic bias in the model. Sometimes, the movements are slightly larger than expected, and sometimes, they are smaller. There are a number of second-order effects that are likely not properly captured in the model, including non-uniform surface properties in the lab (dirt, surface roughness, etc.), screw penetration into the ice sheets, imperfect modeling of mass properties of the robot, and so on. However, the general motion trends showed good agreement. Finally, it is important to note that the anisotropic friction model is limited to screw-based locomotion using sharp-edged screw blades on hard ice with minimal penetration. Various factors such as thick, rounded edge screws or unconsolidated snow or sand are beyond the limits of the model and instead

would likely require semi-empirical terramechanics approaches [38]. This is planned for future work.

4.3. Athabasca Glacier Field Test

A motivation for the development of EELS-DARTS was to provide simulated environments to enable testing and planning for experiments in the field with the actual robots. One such field test was conducted at the Athabasca Glacier in Alberta, Canada [31]. This site was chosen since it is a good terrestrial analogue to Enceladus for demonstrating vertical mobility. Experiments were performed on the glacier in various moulins, which are vertical shafts that form within the glacier by water percolating through small cracks in the surface. The goal of these tests was to evaluate the real-world performance of various locomotion control strategies for vertical mobility in these moulins. The success of the tests was determined by whether the robot was able to support its own weight when attempting to remain in a static position in the vertical shaft, the total distance traveled during an experiment, and robustness to disturbances from rough sections of ice. The robot tested in these scenarios was the EELS 1.5 platform described in Section 2.1.2, which was specifically designed for vertical mobility experiments.

4.3.1. Simulation Testing for Field Test Scenarios

The most important characteristic to capture accurately in the simulation was the screw–ice interaction, namely the anisotropic friction properties described in Section 2.2.2. The robot's locomotion control strategies involve exploiting the screws' anisotropic friction properties, and as such, inaccurate simulations can lead to large discrepancies when deployed in real-world scenarios.

EELS-DARTS provided a critical piece of the infrastructure in the development and verification process of vertical mobility locomotion strategies. During development, following initial unit-testing of each iteration of the controller, the next step would be evaluation in simulation. The main integration test used to evaluate controller performance would sequentially evaluate whether the control strategy was capable of making contact with the environment surface, holding a static position when supporting its own weight for a set time period, and tracking the total vertical distance traversed before a mobility failure.

For simulation, we flagged a mobility error when the angle between the pitch of the body and the gravity vector exceeded 60 degrees, which was determined to be irrecoverable. If this condition was detected, the simulation run was terminated.

The simplest scenario to evaluate the system in was a set of parallel walls, where the system attempted to push into the walls and then climb upward. Figure 16 displays the EELS 1.5 system simulated in this environment in EELS-DARTS. Also shown is a similar scenario recreated experimentally inside a walk-in freezer at JPL. Multiple vertical stability and mobility tests were performed on real ice in the freezer alongside EELS-DARTS simulation runs prior to the field test at the glacier.

Another useful scenario to evaluate the system was to place the robot into a mesh from a 3D reconstruction of a real-world moulin. These 3D reconstructions often come from high-density point cloud scans. Once converted into a mesh format, these can easily be used within the EELS-DARTS environment. Using these reconstructions, we can evaluate the robot system's performance in realistic scenarios that are expected to be seen when deployed with hardware. Figure 17 displays three different views of the EELS 1.5 robot being simulated within an imported moulin reconstruction mesh.

Figure 16. Left: view of the EELS 1.5 robot simulated within EELS-DARTS prior to executing its plan to engage contact with the wall and attempt to climb. White lines represent tethers. The red, green, and blue lines represent the inertial coordinate system. **Right**: image of the EELS 1.5 robot suspended within a vertical ice wall testbed inside a walk-in freezer lab at JPL.

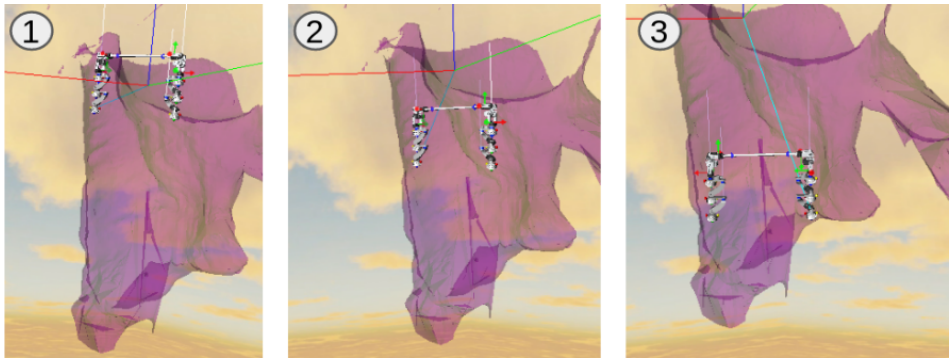

Figure 17. Frames of the EELS 1.5 robot being placed in different locations within a reconstructed moulin scan that has been imported into EELS-DARTS. (**1**), (**2**), and (**3**) show sequential frames of the robot being manually placed further down into the moulin supported by tethers (white lines). The red, green, and blue lines represent the x, y, and z axes of the coordinate system for the moulin scan, respectively.

Once there was confidence in an iteration of the control strategy, the method would be deployed to hardware and tested in a real-world moulin. Figure 18 shows EELS 1.5 being deployed into a moulin by the field test team along with close-up images of the screws making contact with ice. The system was able to down-climb successfully over 1.5 m into a moulin similar to this one.

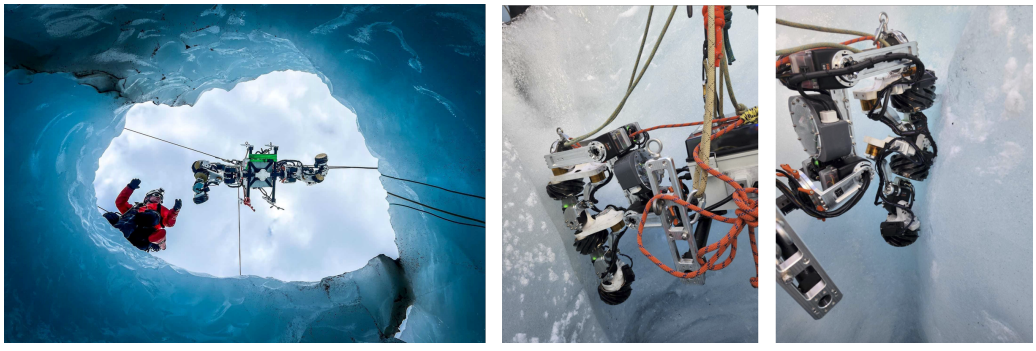

Figure 18. Left: the EELS 1.5 robot being deployed into a moulin in Athabasca. **Right**: close-up images of the EELS 1.5 screws engaging in contact with rough and uneven ice walls.

4.3.2. Challenges

A key challenge faced when transferring control strategies tested in EELS-DARTS to hardware was the temporal aspect of the screw–ice interaction. In order to obtain sufficient screw penetration into the ice, a criterion necessary to sustain meaningful load with the

screws, it was necessary to heat the screws to a temperature between 10 and 40 degrees Celsius, where the temperature was verified with onboard thermocouples. When the screws engaged in contact with the ice, they would immediately start melting into the surface, and if motion did not start quickly enough, the screws risked penetrating so deep into the surface that the controller was incapable of pulling itself out and continuing the climb. This results in a vertical mobility failure that is irrecoverable without human intervention. Future development of the EELS-DARTS screw–ice interaction model may include such temporal thermal aspects to enable the development and iteration of more robust vertical mobility locomotion strategies.

5. Conclusions

In this paper, we have presented EELS-DARTS, a novel simulator built upon the JPL DARTS framework for simulating large-scale snake robots. We discussed various features that allow EELS-DARTS to be a powerful and versatile simulation framework. The versatile dynamics modeling framework provided by DARTS allows users to quickly process and import new snake robot designs into the simulator with minimal effort. The CLI and ROS interface allows a full autonomy stack loop to be closed around EELS-DARTS, providing a fully featured software analogue to actual robot hardware. EELS-DARTS provides a highly configurable simulation environment, including terrain selection, robot initialization, and placement.

EELS-DARTS implements a fast and reasonably accurate anisotropic friction model to provide screw–terrain interactions for hard ice scenarios. We compared the overall robot position predicted in simulation using the anisotropic friction locomotion model with experimental results for a 45 s traverse on hard synthetic ice. The predicted tail position of the robot differed from experiment by at most roughly 4 cm in x, y position and roughly 0.06 radians (3.4 degrees) in yaw. No obvious consistent offsets between simulation and experiment were observed. It was hypothesized that some of the observed differences could be attributable to imperfect modeling of surface properties in the lab, non-negligible screw penetration into the synthetic ice sheets, and imperfect modeling of mass properties of the robot. We determined that the model was sufficiently accurate for the application at hand, which required near real-time performance. We emphasize that the anisotropic friction model is only suitable for use under the following assumptions: hard, non-deformable contacts, minimal penetration, sharp screw edges, minimal surface melting, and uniform surface properties.

We described how EELS-DARTS has been used to simulate multiple evolutions of EELS robot designs and topologies, test the full autonomy stack, including locomotion and perception algorithms, and prepare for field test scenarios.

For future work, we plan to explore terramechanics approaches to model screw–terrain interactions for soft soil or sand scenarios. We also identified that the anisotropic friction model could be extended to include a temporal thermal component to capture ice melting effects seen during experiments. Such contact modeling extensions would allow more accurate simulation in more varied environments. We also plan to continue supporting new robot designs and features as needed by the EELS project.

Author Contributions: Conceptualization, all authors; Methodology, T.D.H., C.L., A.G., A.E., B.J., B.H., M.P., G.D. and R.T.; Software, T.D.H., C.L., A.G., A.E., R.M.S., R.R., B.J., M.P., G.D., R.T. and A.J.; Validation, T.D.H.; Formal analysis, T.D.H., C.L., A.G., A.E., R.M.S., R.R., B.J., B.H., M.P., G.D. and R.T.; Investigation, R.M.S., R.R., B.J., B.H., M.P., G.D. and R.T.; Resources, R.M.S., R.R., B.J., B.H., M.P., G.D., M.O., R.T. and A.J.; Data Curation, T.D.H., C.L., A.G., A.E. and B.H.; Writing—Original Draft Preparation, T.D.H., C.L., A.E., B.J. and A.J.; Writing—Review and Editing, T.D.H., C.L., A.E., B.H., M.P., R.T., M.O. and A.J.; Visualization, T.D.H., C.L., A.E., R.R., B.J., M.P. and G.D.; Supervision, M.P., M.O., R.T. and A.J.; Project Administration, R.T., M.O. and A.J.; Funding Acquisition, M.P., R.T., M.O. and A.J. All authors have read and agreed to the published version of the manuscript.

Funding: 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).

Data Availability Statement: Data are contained within the article. The DARTS simulation software is not freely available. Licensing inquiries can be directed to the corresponding author.

Acknowledgments: ©2024 California Institute of Technology. Government sponsorship acknowledged. 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)

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CLI	Command-line interface
DARTS	Dynamics Algorithms for Real-Time Simulation
DEM	Digital elevation model
DOF	Degree of freedom
EDL	Entry, descent, and landing
EELS	Exobiology extant life surveyor
FSW	Flight software
FTS	Force torque sensor
GNC	Guidance, navigation, and control
GPU	Graphics processing unit
IK	Inverse kinematics
IMU	Inertial measurement unit
JPL	Jet Propulsion Laboratory
MDPI	Multidisciplinary Digital Publishing Institute
ML	Machine learning
NASA	National Aeronautics and Space Administration
PID	Proportional–integral–derivative
ROS	Robot operating system
SDF	Signed distance field
SOA	Spatial operator algebra
URDF	Universal robotic description format

References

1. Ono, M.; Thakker, R.; Georgiev, N.; Gavrilov, P.; Archanian, A.; Drevinskas, T.; Daddi, G.; Paton, M.; Strub, M.; Melikyan, H.; et al. To Boldly Go Where No Robots Have Gone Before—Part 1: EELS Robot to Spearhead a New One-Shot Exploration Paradigm with In-situ Adaptation. In Proceedings of the AIAA SciTech Forum, Orlando, FL, USA, 8–12 January 2024.
2. Carpenter, K.; Cable, M.L.; Choukroun, M.N.; Ono, H.; Thakker, R.A.; Ingham, M.D.; McGarey, P.; Barchowsky, A.; Iacoponi, S.; Bowkett, J.; et al. Venture Deep, the Path of Least Resistance: Crevasse-Based Ocean Access Without the Need to Dig or Drill. *Bull. Am. Astron. Soc.* **2021**, *53*, 356.
3. Exobiology Extant Life Surveyor Website. Available online: https://www.jpl.nasa.gov/robotics-at-jpl/eels, (accessed on 25 June 2024).
4. Thakker, R.; Paton, M.; Jones, B.; Daddi, G.; Royce, R.; Swan, M.; Strub, M.; Aghli, S.; Zade, H.; Nakka, Y.; et al. To Boldly Go Where No Robots Have Gone Before—Part 4: NEO Autonomy for Robustly Exploring Unknown, Extreme Environments with Versatile Robots. In Proceedings of the AIAA SciTech Forum, Orlando, FL, USA, 8–12 January 2024.
5. Vaquero, T.S.; Daddi, G.; Thakker, R.; Paton, M.; Jasour, A.; Strub, M.P.; Swan, R.M.; Royce, R.; Gildner, M.; Tosi, P.; et al. EELS: Autonomous snake-like robot with task and motion planning capabilities for ice world exploration. *Sci. Robot.* **2024**, *9*, eadh8332. [CrossRef]
6. Talbot, W.; Nash, J.; Paton, M.; Ambrose, E.; Metz, B.; Thakker, R.; Etheredge, R.; Ono, M.; Ila, V. Principled ICP Covariance Modelling in Perceptually Degraded Environments for the EELS Mission Concept. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, MI, USA, 1–5 October 2023; pp. 10763–10770. [CrossRef]
7. Marvi, H.; Gong, C.; Gravish, N.; Astley, H.; Travers, M.; Hatton, R.L.; Mendelson, J.R.; Choset, H.; Hu, D.L.; Goldman, D.I. Sidewinding with minimal slip: Snake and robot ascent of sandy slopes. *Science* **2014**, *346*, 224–229. [CrossRef]
8. Tesch, M.; Lipkin, K.; Brown, I.; Hatton, R.; Peck, A.; Rembisz, J.; Choset, H. Parameterized and Scripted Gaits for Modular Snake Robots. *Adv. Robot.* **2009**, *23*, 1131–1158. [CrossRef]
9. Marteau, E.; Tosi, L.P.; Veismann, M.; Gavrilov, P.; Peticco, M.; Hockman, B.; Ono, M.; Khanum, M.; Abdelrahim, M.; Marvi, H. To Boldly Go Where No Robots Have Gone Before—Part 3: The Screw Mobility System of EELS for Robust Surface and Subsurface Mobility on Highly Unknown Terrains. In Proceedings of the AIAA SciTech Forum, Orlando, FL, USA, 8–12 January 2024.

10. Thakker, R.; Paton, M.; Strub, M.; Swan, M.; Daddi, G.; Royce, R.; Tosi, P.; Gildner, M.; Vaquero, T.; Veismann, M.; et al. EELS: Towards Autonomous Mobility in Extreme Terrain with a Versatile Snake Robot with Resilience to Exteroception Failures. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, MI, USA, 1–5 October 2023.
11. Vadlamannati, A.; Beknalkar, S.; Best, D.; Bryant, M.; Mazzoleni, A. Design, Prototyping, and Experiments Using Small-Scale Helical Drive Rover for Multi-Terrain Exploration. In Proceedings of the ASME International Mechanical Engineering Congress and Exposition, New Orleans, FL, USA, 29 October–2 November 2023; Volume 6: Dynamics, Vibration, and Control. [CrossRef]
12. Beknalkar, S.; Bryant, M.; Mazzoleni, A. Algorithm for Locomotion Mode Selection, Energy Estimation and Path Planning for a Multi-terrain Screw-Propelled Vehicle for Arctic Exploration. In Proceedings of the 2024 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Boston, MA, USA, 15–19 July 2024; pp. 1462–1467. [CrossRef]
13. Hasseler, T.D.; Leake, C.; Gaut, A.; Elmquist, A.; Swan, R.M.; Royce, R.; Jones, B.; Hockman, B.; Paton, M.; Daddi, G.; et al. EELS-DARTS: A Planetary Snake Robot Simulator for Closed-Loop Autonomy Development. In Proceedings of the 7th International Conference on Multibody System Dynamics, Madison, WI, USA, 9–13 June 2024.
14. Wan, Z.; Sun, Y.; Qin, Y.; Skorina, E.H.; Gasoto, R.; Luo, M.; Fu, J.; Onal, C.D. Design, Analysis, and Real-Time Simulation of a 3D Soft Robotic Snake. *Soft Robot.* **2023**, *10*, 258–268. [CrossRef]
15. Transeth, A.A.; Leine, R.I.; Glocker, C.; Pettersen, K.Y. 3-D Snake Robot Motion: Nonsmooth Modeling, Simulations, and Experiments. *IEEE Trans. Robot.* **2008**, *24*, 361–376. [CrossRef]
16. Transeth, A.A.; Leine, R.I.; Glocker, C.; Pettersen, K.Y.; Liljeback, P. Snake Robot Obstacle-Aided Locomotion: Modeling, Simulations, and Experiments. *IEEE Trans. Robot.* **2008**, *24*, 88–104. [CrossRef]
17. Monsalve, J.; Leon, J.; Melo, K. Modular snake robot oriented open simulation software. In Proceedings of the 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, Hong Kong, China, 4–7 June 2014; pp. 546–550. [CrossRef]
18. Sibilska-Mroziewicz, A.; Hameed, A.; Możaryn, J.; Ordys, A. Virtual Reality Simulations of the Snake Robot. In *Proceedings of the Digital Interaction and Machine Intelligence, Warsaw, Poland, 12–15 December 2022*; Biele, C., Kacprzyk, J., Kopeć, W., Owsiński, J.W., Romanowski, A., Sikorski, M., Eds.; Springer Nature: Cham, Switzerland, 2023; pp. 307–313.
19. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3.
20. Jain, A. DARTS—Multibody Modeling, Simulation and Analysis Software. In *Proceedings of the Multibody Dynamics 2019 (ECCOMAS 2019), COMPUTMETHODS Conference, Duisburg, Germany, 15–18 July 2019*; Kecskemethy, A., Geu Flores, F., Eds.; Springer Nature: Cham, Switzerland, 2019; Volume 53, pp. 433–441.
21. Jain, A. *Robot and Multibody Dynamics: Analysis and Algorithms*; Springer: New York, NY, USA, 2010.
22. Dynamics Algorithms for Real-Time Simulation (DARTS) Lab Website. Available online: https://dartslab.jpl.nasa.gov (accessed on 25 June 2024).
23. Cameron, J.; Balaram, J.; Jain, A.; Kuo, C.; Lim, C.; Myint, S. Next Generation Simulation Framework for Robotic and Human Space Missions. In Proceedings of the AIAA SPACE Conference and Exposition, Pasadena, CA, USA, 11–13 September 2012.
24. Jain, A.; Cameron, J.; Lim, C.; Guineau, J. SimScape Terrain Modeling Toolkit. In Proceedings of the Second IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT), Pasadena, CA, USA, 17–20 July 2006.
25. Myint, S.; Jain, A.; Cameron, J.; Lim, C. Large Terrain Modeling and Visualization for Planets. In Proceedings of the Fourth IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT), Palo Alto, CA, USA, 2–4 August 2011.
26. Aiazzi, C.; Gaut, A.; Young, A.; Elmquist, A.; Jain, A. IRIS: High-fidelity Perception Sensor Modeling for Closed-Loop Planetary Simulations. In Proceedings of the AIAA Scitech Forum, San Diego, CA, USA, 3–7 January 2022.
27. Ryan, R.R. ADAMS—Multibody System Analysis Software. In *Multibody Systems Handbook*; Schiehlen, W., Ed.; Springer: Berlin/Heidelberg, Germany, 1990; pp. 361–402. [CrossRef]
28. Mittal, M.; Yu, C.; Yu, Q.; Liu, J.; Rudin, N.; Hoeller, D.; Yuan, J.L.; Singh, R.; Guo, Y.; Mazhar, H.; et al. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. *IEEE Robot. Autom. Lett.* **2023**, *8*, 3740–3747. [CrossRef]
29. Todorov, E.; Erez, T.; Tassa, Y. MuJoCo: A physics engine for model-based control. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 5026–5033. [CrossRef]
30. Koenig, N.; Howard, A. Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, 28 September–2 October 2004; pp. 2149–2154.
31. Paton, M.; Rieber, R.; Cruz, S.; Gildner, M.; Abma, C.; Abma, K.; Aghli, S.; Ambrose, E.; Archanian, A.; Bagshaw, E.A.; et al. 2023 EELS Field Tests at Athabasca Glacier as an Icy Moon Analogue Environment. In Proceedings of the 2024 IEEE Aerospace Conference, Big Sky, MT, USA, 2–9 March 2024; pp. 1–18. [CrossRef]
32. Jain, A. Computing Inter-Body Constraint Forces in Recursive Multibody Dynamics. In Proceedings of the 5th Joint International Conference on Multibody System Dynamics, Lisboa, Portugal, 24–28 June 2018.
33. Leake, C.; Jain, A. FModal: A Flexible Body Dynamics Modeling Pipeline for Guidance and Control. In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 4–11 March 2023.
34. Jain, A.; Rodriguez, G. Recursive Flexible Multibody System Dynamics Using Spatial Operators. *J. Guid. Control Dyn.* **1992**, *15*, 1453–1466. [CrossRef]

35. Coumans, E.; Bai, Y. PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning. 2016–2021 Available online: http://pybullet.org (accessed on 6 August 2024).
36. Parker, S.G.; Bigler, J.; Dietrich, A.; Friedrich, H.; Hoberock, J.; Luebke, D.; McAllister, D.; McGuire, M.; Morley, K.; Robison, A.; et al. OptiX: A General Purpose Ray Tracing Engine. *ACM Trans. Graph.* **2010**, *29*, 66. [CrossRef]
37. Madison, R.; Pomerantz, M.; Jain, A. Camera Response Simulation for Planetary Exploration. In Proceedings of the i-SAIRAS Conference, Munich, Germany, 5–8 September 2005.
38. Nagaoka, K.; Otsuki, M.; Kubota, T.; Tanaka, S. Terramechanics-based propulsive characteristics of mobile robot driven by Archimedean screw mechanism on soft soil. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 4946–4951. [CrossRef]