

# ugv\_nav4d: Advanced Multi-Surface Navigation for Unmanned Ground Vehicles Using 4D Path Planning Techniques

Arne Böckmann<sup>1</sup>, Janosch Machowinski<sup>1</sup>, and Muhammad Haider Khan Lodhi<sup>2</sup>

<sup>1</sup> Cellumation GmbH <sup>2</sup> DFKI GmbH

DOI: [10.21105/joss.06983](https://doi.org/10.21105/joss.06983)

## Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Tristan Miller](#) ↗ 

## Reviewers:

- [@soraxas](#)
- [@OlgerSiebinga](#)

Submitted: 20 June 2024

Published: 02 December 2024

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## Summary

The `ugv_nav4d` is a global path planner designed for terrestrial robots to navigate complex indoor and outdoor environments. To achieve this, `ugv_nav4d` uses a traversability map (`TraversabilityMap3d`) based on a multi-layered surface map (`MLSMap`) ([Born & Arnold, 2015](#); [Triebel et al., 2006](#)), enabling planning in multi-surface environments. The path is built by selecting motion primitives tailored to the robot's mechanical features, ensuring compatibility with its motion capabilities.

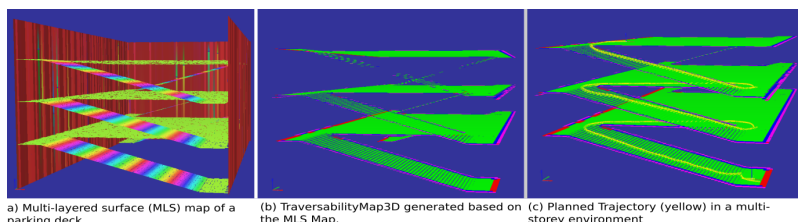


Figure 1: Planned trajectory in a multi-storey environment.

## Statement Of Need

Accurate ground surface representation is crucial for ground-based robots in complex terrains. The ROS2 Navigation Stack (`nav2`) ([Macenski et al., 2020](#)), which uses voxel maps for 3D navigation, often loses detail and accuracy, especially in multi-storey environments, due to its discrete voxelization and separate costmaps for each floor.

We propose `ugv_nav4d`, a path planner that enhances environmental representation with Multi-Layered Surface Maps (MLS) ([Triebel et al., 2006](#)) and a 3D Traversability Map ([Born & Arnold, 2015](#)). `Ugv_nav4d` avoids the “stepping” effect of voxel maps by using a continuous grid and detailed vertical information, providing smoother and more accurate terrain modelling.

Unlike `nav2`, `ugv_nav4d` simplifies planning with a single `TraversabilityMap3D`, which contains detailed ground surface data, offering a superior alternative to `nav2`'s 3D costmaps. For users, MLS maps provide a smoother, more realistic view of terrain compared to the blocky voxel maps, enhancing navigation and decision-making in complex environments.

## Software Components

The core software components of the planner are

- EnvironmentXYZTheta
- PathPlanner
- PreComputedMotions

### EnvironmentXYZTheta

The core of `ugv_nav4d` is based on SBPL (Search-Based Planning Library) (Likhachev et al., 2005). The EnvironmentXYZTheta implements all interfaces needed by SBPL to enable ARA\* based planning. The environment in SBPL is a state space which connects states with associated transition costs. A state is defined by the position (x,y,z) and orientation (yaw) of the robot. The EnvironmentXYZTheta uses a TraversabilityGenerator3d (Machowinski & Böckmann, 2016) to generate a TraversabilityMap3d from a MLSMap, which classifies the MLSMap patches into traversable, non-traversable, and unknown terrain and stores metadata of the ground surface (e.g., slope of the patch, supporting plane, etc.).

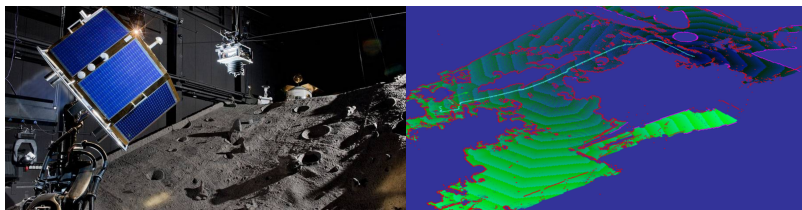


Figure 2: The Moon Crater and its TraversabilityMap3d in the Space Hall at the RIC, DFKI (Robotics Exploration Laboratory, 2024).

### PathPlanner

The search for the shortest path in `ugv_nav4d` is based on the SBPL-provided ARA\* planner. It builds a state space of future robot states and uses a time heuristic to compute the shortest path on the TraversabilityMap3D. Robot and terrain-specific information, including robot dimensions and orientations, collision checks with obstacles, terrain steepness, and motion primitives are incorporated as additional costs in the search for the successive states. The planner generates a complete trajectory for the planned path, which can then be executed by a trajectory follower (Babu, 2016).

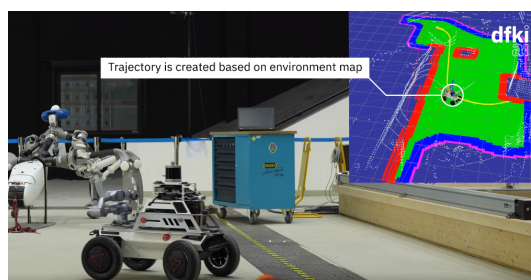


Figure 3: The planned trajectory (yellow) using the robot Hunter-SE in the final demonstration of the project KIMMI-SF.

### PreComputedMotions

Motion primitives are pre-defined movements stored as spline trajectories, representing feasible robot motions from a given pose (x, y, z, yaw). The planner uses these primitives to find successor states, ensuring paths are traversable and collision-free. Each motion has a base cost for flat surfaces, with additional penalties for factors like steepness. Primitives fall into four categories: Forward, Backward, Lateral, and Point-Turn. These primitives can be tailored for

different robot types: Ackermann drive robots use forward and backward primitives, differential drive robots can use point-turn primitives, and omni-directional robots utilize all four types. Point-turn primitives are unique as they don't use splines, unlike the others generated by the `SbplSplineMotionPrimitives` library (Böckmann, 2016).

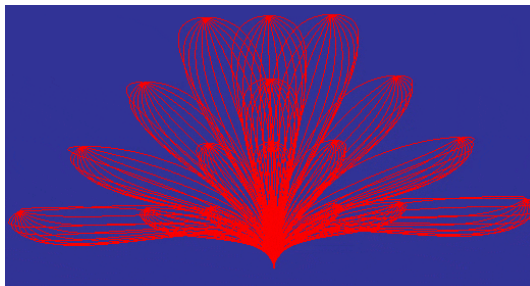


Figure 4: A pool of forward motion primitives by the `PreComputedMotions` module.

## Debugging

### PlannerGui

A GUI for `ugv_nav4d` allows debugging and testing by experimenting with planner parameters on a static map. The `PlannerGui` can load point clouds from ply or serialized MLS maps, using left and right clicks to set start and end locations. Errors can be logged to a file and `ugv_nav4d_replay` can be used to analyze the state and execute planning in a controlled environment.

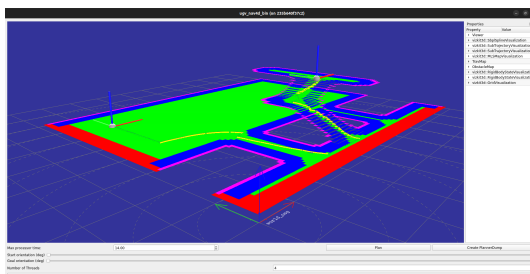


Figure 5: GUI for debugging and testing of `ugv_nav4d`.

## Field Tests

`ugv_nav4d` has been used extensively in research projects for almost a decade. It has been utilized for autonomous navigation in several projects, including [Entern](#), [ANT](#), [VIPE](#), [KIMMI-SF](#), [HiSE](#), [PerSim](#), and [CoRobX](#). As a versatile planner, it has also been used for various terrestrial robots, such as [Artemis](#), [Coyote-III](#), [Crex](#), [Charlie](#), [Hunter-SE](#), [SherpaTT](#), and [Asguard-IV](#).

## Acknowledgements

The `ugv_nav4d` library was initiated and is currently developed at the Robotics Innovation Center of the German Research Center for Artificial Intelligence (DFKI) in Bremen, together with the Robotics Group of the University of Bremen. The development was started in the scope of the Entern project (50RA1406), which has been funded by the German Aerospace Center (DLR) with funds from the German Federal Ministry for Economic Affairs and Climate Action (BMWK).

We would also like to acknowledge the contributors of the `ugv_nav4d` repository, including the developers visible on the project's GitHub contributors page, for their valuable efforts and dedication.

## References

- Babu, A. (2016). *Trajectory Follower*. [https://github.com/rock-control/control-trajectory\\_follower](https://github.com/rock-control/control-trajectory_follower)
- Böckmann, A. (2016). *SBPL Spline Primitives*. [https://github.com/rock-planning/planning-sbpl\\_spline\\_primitives](https://github.com/rock-planning/planning-sbpl_spline_primitives)
- Born, A., & Arnold, S. (2015). *SLAM Maps*. <https://github.com/envire/slam-maps>
- Likhachev, M., Ferguson, D., & Gordon, G. (2005). *SBPL: Search-Based Planning Library* (Version 1.3). <https://github.com/sbpl/sbpl>
- Macenski, S., Martín, F., White, R., & Clavero, J. G. (2020). The Marathon 2: A Navigation System. *CoRR*, *abs/2003.00368*. <https://doi.org/10.1109/IROS45743.2020.9341207>
- Machowinski, J., & Böckmann, A. (2016). *Traversability Generator3d*. [https://github.com/dfki-ric/traversability\\_generator3d](https://github.com/dfki-ric/traversability_generator3d)
- Robotics Exploration Laboratory. (2024). <https://www.dfki.de/web/technologien-anwendungen/living-labs/robotics-exploration-laboratory>
- Triebel, R., Pfaff, P., & Burgard, W. (2006). Multi-level surface maps for outdoor terrain mapping and loop closing. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2276–2282. <https://doi.org/10.1109/IROS.2006.282632>