Innovative Solutions for Fast Autonomous Navigation (SINAV)

Patrick Roncagliolo [(1)], Gabriele Berardi [(1)], Piergiorgio Lanza [(1)], Andrea Merlo [(1)], Davide Graziato [(2)], Giuseppe D'Amore [(3)]

[(1)] *Thales Alenia Space Italy* [(2)] *Politecnico di Torino* [(3)] *Italian Space Agency*

This paper covers part of the developments of SINAV study. In particular, TAS-I and PoliTo designed and prototyped a continuous and autonomous navigation architecture designed for unstructured environments and uneven grounds, also effective in absence of sunlight thanks to active sensors. Moreover, Deep Learning (DL) modules were trained and deployed in support of the rover autonomy.

1. INTRODUCTION

Autonomous navigation is the foundation of robotic exploration of the surface of distant celestial bodies. First of all, it enables to circumvent unavoidable signal round-trip times that do not permit decent human-in-the-loop control of vehicles; secondly, a number of safety features, such as obstacle avoidance, are inherently embodied by the autonomy solution; last but not least, it enables scientists and ground control to focus on more appealing problems, like processing more data.

Traditionally, due to performance limits in radiation-hardened on-board computers and data handling systems, rovers were able to move autonomously, but with a stop-and-go strategy: a perception loop is executed, the 3D information is computed, a traversability map is later derived and a few meters long path is obtained and executed by means of the locomotion system.

However, this allow to travel only a few hundreds of meter each Sol. Continuous Autonomous Navigation consists of a fast looping strategy of perception, mapping and planning algorithms, allowing the rover to safely travel without the need to stop: its understanding of surrounding environment would be always considered up-to-date and sufficient to plan safe paths and being aware of nearby obstacles. Travel time would also benefice from the capability to travel in dark conditions.

Note: in this paper, complementary problems like heat accumulation during continuous actuator operation and power management are not taken into account, since the focus is solely on the algorithmic and computational aspect of a smooth, continuous navigation.

2. COMPANY HERITAGE

The TAS-I Robotics Team started investigating autonomy solutions for mobile robots in 2008, within the STEPS (Systems and Technologies for Space ExPloration) project. In that first iteration, a pressurized mobile rover mockup has been built, with a fully-functional locomotion subsystem and a suite of state-of-the-art sensors and navigation algorithms [1]. A second phase of the project, called STEPS 2, led to detailed evaluation of early models of Time-of-Flight (ToF) cameras for surface mapping and obstacle detection, and the development of a Real-Time OS abstraction layer for robotics, with tests carried on in a newly-built facility called RoXY (ROver eXploration facilitY), featuring a 400m^2 outdoor, mars-like playground [2].

The so-called "TAS-I Robot Management Framework" or "Test Bench for Robotics and Autonomy" (TBRA) consisted in a flexible and modular software architecture (Framework Engine) in which each functional module (representing the GNC subsystems) implements a key functionality of the GNC (Guidance Navigation and Control) [3]. Modules communicate by means of standardised interfaces designed for exchange of necessary information among the modules composing the entire system. This approach permits the interchange-ability of each subsystem without affecting the overall functionalities of the GNC system. Moreover, this framework featured different KAL (Kernel Abstraction Layers) to allow its deploy over Linux, RTEMS or RTAI. While this approach was interesting, the maintenance costs of the framework and the technical debt accumulating for its much needed upgrade recently (2020) led the Team to consider a new framework, not existing cat the time of STEPS projects: Robot Operating System (ROS) 2. All the major R&D projects and breadboards within the group are now supported by state-of-the-art development based on ROS 2.

3. FACILITIES

TAS-I ROvers eXploration facilitY is a technological area dedicated to robotic systems design, development, validation and verification. It is located in TAS-I Turin site, covers an area of about 600m2, including a Mars playground, control room and workshop.

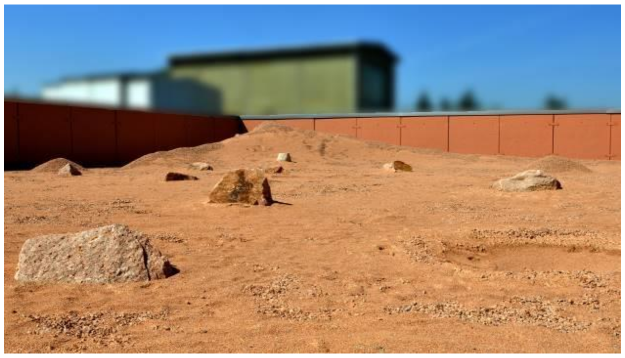

Figure 1 - TAS-I Turin RoXY

The outdoor **playground** covers an area of ~400m2, reproducing Mars-like planetary morphology in terms of color, landscape, boulders, smaller rocks and slopes. The perimeter is surrounded by a uniform background which isolates the terrain from external interferences like peoples and vehicles.

The **control room** hosts the software development validation and verification infrastructure, as well as five work stations for developers and operators.

The **workshop** provides a secure area where to store the robots and to perform integration, test and maintenance activities.

4. STUDY OVERVIEW

The SINAV study is an ASI co-founded project started in September 2021, with a final demo executed in RoXY in September 2023. It is one of the biggest projects ever co-founded by ASI (Contract N. 2021-13-E.0) in Autonomous Navigation and Artificial Intelligence subjects, with a total budget of approximately 1.5 M€. The Italian acronym stands for “Innovative Solutions for Fast and Autonomous Navigation)”, and the project comprised both a surface segment (rover) and an aerial/orbital segment (drone/cubesat): TAS-I Robotics group led the development of the surface segment, starting from a complete hardware retrofitting of the STEPS 2 mobile robot, a skid-steered MobileRobots Seekur Jr and proceeding then with software architecture and algorithm development and deploy. PoliTo strongly contributed to algorithm selection, development and testing.

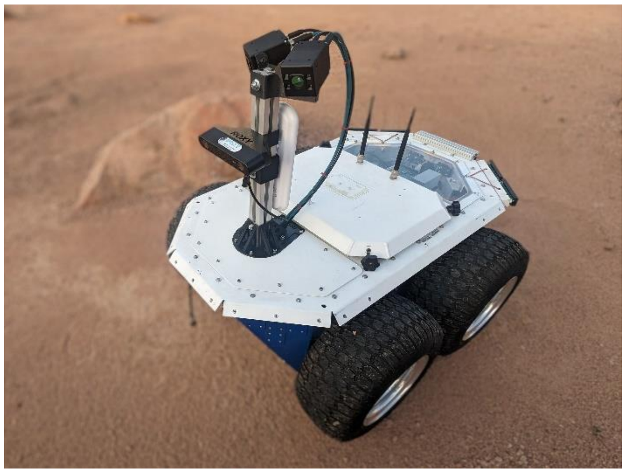

Figure 2 - Retrofitted MobileRobots Seekur Jr

The SINAV project primarily aimed to:

- grow average rover speed tenfold w.r.t. current missions;
- enable autonomous continuous navigation;
- enable navigation in absence of light;
- leverage active sensors such as ToFs;
- use DL to enhance rover navigation;
- allow fusion of rover/drone/satellite mapping data;
- demonstrate rover & drone cooperation.

The ToF cameras are particularly performant in case of untextured surfaces such as the lunar surface, can perceive in a completely dark scenario and require less computing w.r.t. classical stereovision pipelines. [4] This is why the study targeted their exploitation and integration in an autonomous navigation system designed for unstructured environments.

5. SETUP OVERVIEW

The SINAV rover features two OBCs:

- a Versalogic VL-41 EBX mainboard with an Intel i7-3615QE and 16 GB DDR3 RAM;
- a Nvidia Xavier NX with 6 Nvidia Camel cores and 8 GB of RAM.

and the following sensors:

- a Stereolabs ZED 2i stereo camera, with 2K resolution and polarized lens; the SDK provides CUDA-accelerated visual tracking and normal estimation;
- two LUCID Vision Labs Helios2+ ToF cameras with VGA resolution and 100 Hz framerate;
- an Xsens Mti 620 VRU with built-in filtering;

The environment is mainly perceived via 3D Point Cloud representation (or the correspondent 2D depth image, to save memory and bandwidth). The SW framework can be configured to process structured point clouds, which are clouds associated to a matrix (just as a common RGB image): this typically allows to apply optimized (faster) algorithms at a cost of slightly higher memory usage.

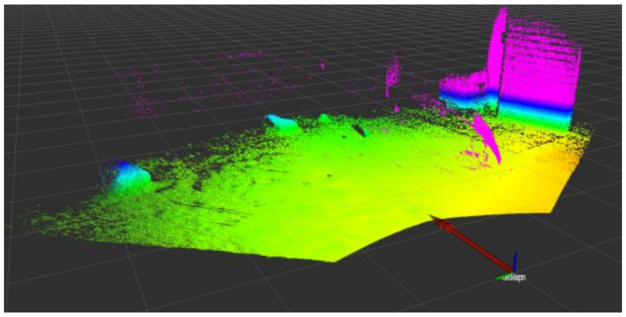

Figure 3 - Example raw point cloud

6. SW ARCHITECTURE

The SINAV architecture is conceptually simple; it consists of few layers:

- the hardware, comprising both sensors and actuators of the robotic platform;
- the low-level abstraction code, with the Hardware Abstraction Layer (HAL) harmonizing the read/write access to sensors and actuators, and the Locomotion/Actuation block that wraps motors to abstract the platform maneuvering (SINAV robot has only one locomotion mode: skid-steering);
- the autonomous navigation layer, strictly subdivided in:
  - localization
  - mapping
  - planning (path planning)
  - control (path tracking)

  which enables to process sensor inputs and command vehicle maneuvers to reach a navigation goal;
- the high-level, deliberative layer which can take short term mission decisions in lieu of the mission control center, if needed. It implements Behavior Tree (BT) reconfigurable logic.

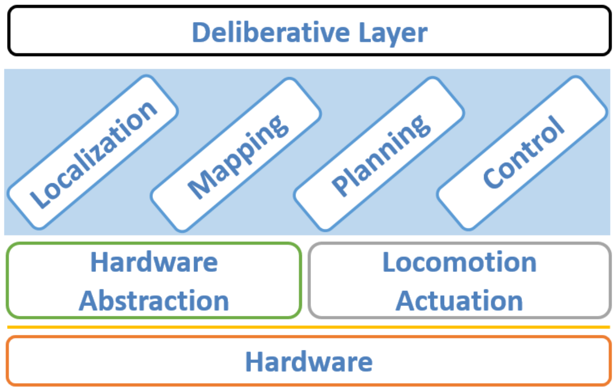

Figure 4 - Reference architectural schema

Notice that Localization and Mapping are explicitly separated instead of being replaced by a “SLAM” block. The idea is to keep the software as modular as possible, for a variety of reasons such as logic and fault isolation, maintenance and even didactic purposes: this platform and framework will be used in R&D for the upcoming years to train roboticists, so fairly monolithic software solutions are avoided if possible. SLAM is still possible to achieve by proper selection of suitable localization and mapping blocks, which are free by design to communicate / exchange data.

Each of the architectural blocks typically comprises many software blocks, called "components".

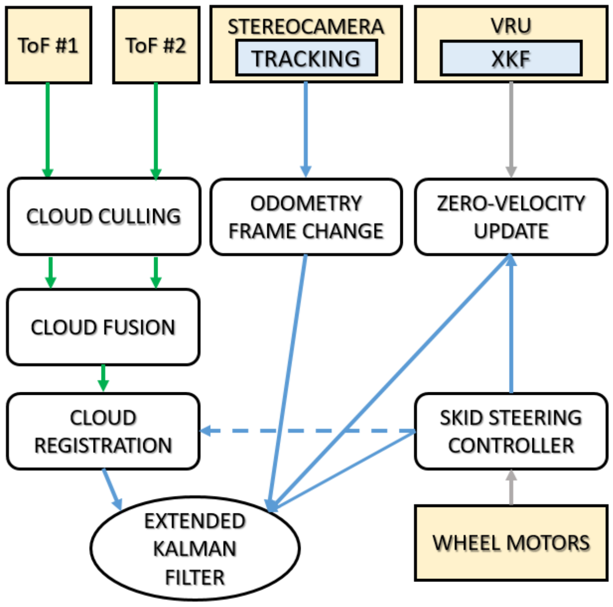

Figure 5 - Localization subsystem

The localization subsystem is in charge to provide position an orientation update of the robotic platform and of every dependent reference frame. An EKF filter takes into account mechanical odometry, orientation estimation coming from onboard Vertical Reference Unit (VRU), visual odometry provided either by the onboard Stereo camera visual tracking algorithm or by point cloud registration techniques applied on the fused point cloud from the two ToFs. The choice between the two visual odometry sources usually depends on the presence or absence of light. On the VRU orientation estimate, a zero-velocity update filtering [5] is made to suppress yaw drift when the vehicle is not moving (no magnetometer is used to get absolute heading).

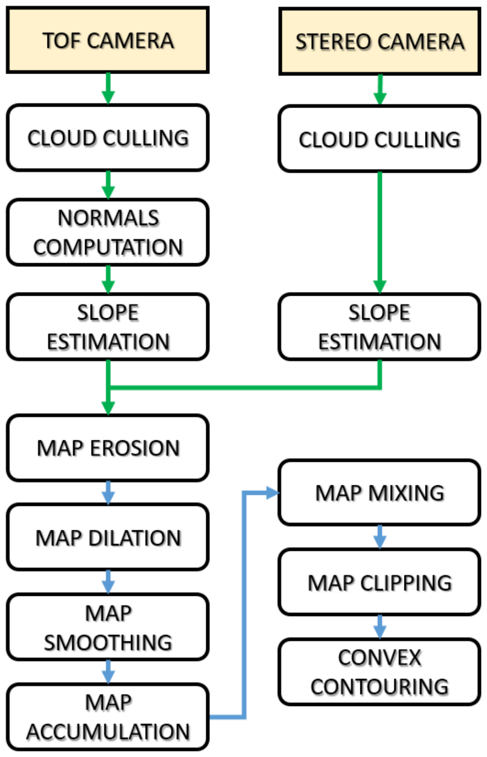

Figure 6 - Mapping subsystem

The mapping subsystem duty is to synthetize a 2D traversability map of the surrounding environment, allowing the planning subsystem to take decisions about the path to generate. Again, this subsystem can either work with point clouds coming from either the stereo camera or from the ToF(s). The points generated from the border of the sensor are discarded (the residual lens distortion is usually higher), surface normals are computed on the cloud if needed (for example, the stereo camera SDK internally compute them, while no similar solution is found on the purchased ToFs). Then, the clouds are collapsed on the ideal 2D plane, binning points on a fixed grid. Each tile gets a slope angle computed from the set of points belonging to it. No mean Z value is computed since this system does not compute an intermediate DEM representation. This means that in this basic pipeline layout, the navigation criteria (cost function of the traversability map) is based on the slope angle. Obstacles can already be intrinsically detected due to the high slope in correspondence to their borders w.r.t. the ground floor.

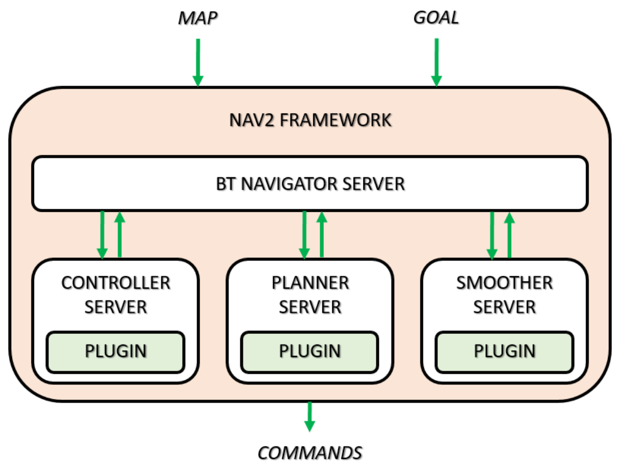

Figure 7 - Planning and control subsystem (Nav2)

The planning and control subsystems are instead inherited by an open source framework called Nav2 [6]. This pluggable framework allows to swap path planner and controller with either a set of pre-defined ones or custom implementations. In the context of SINAV project, PoliTo contributed with the integration of the Shifted Grid Fast Marching [7] planner and the LQR controller. Moreover, Nav2 embeds the runtime for the execution of Behavior Trees [8] policies (the Deliberative Layer of our architecture).

7. ROS 2 IMPLEMENTATION

While space robotics SWs are typically implemented as monolithic (and statically-linked) applications, the preferred approach in the robotic field is to move to distributed frameworks. Although several other solutions exist (eg. YARP, MRPT), the de-facto standard in both R&D and modern industrial applications is called Robot Operating System 2 (ROS 2), a framework abstracting Scheduling, Time management and most importantly data exchange across so called "nodes", either via Remote Procedure Calls (RPC) or Inter-/Intra-Process Communication (ICP) [9]. Using ROS 2 for R&D development drastically widens the amount of skilled developers that can join the project, due to its widespread usage in the robotics community.

To explain how ROS 2 true potential can be unlocked, take as reference the following figure. Two ROS 2 applications, or nodes, can be developed, built and executed independently; communication would then happen via IP over DDS protocol (red blocks example). However, ROS 2 nodes crafted as "components" can be also seamlessly loaded in the same process space at runtime (orange blocks example). If at this stage, the user enables a specific flag, intra-process communication can be enabled, so the RMW layer is completely bypassed: data is exchanged via zero-copy protocol (green blocks example).

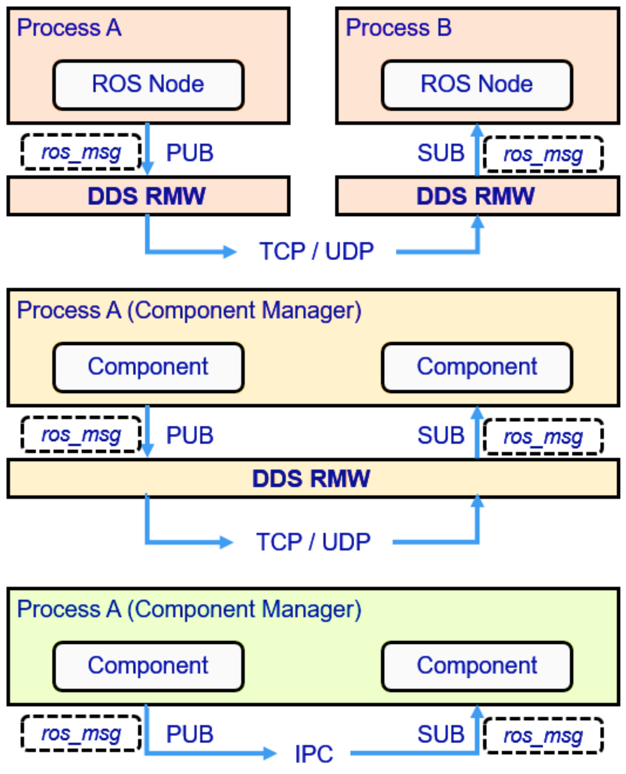

Figure 8 - Intra-process optimization in ROS 2

This way, ROS 2 enables developers to separate and isolate concerns in development, test and debug phase, allowing easily to switch to an optimized runtime with near zero overhead [10]. In the SINAV projects, many small components were developed, each wrapping usually a very simple task so that the user can choose at runtime how to build its perfect data processing pipeline, for example.

Another optimization that can heavily reduce runtime cost consists in type adaptation: due to the ROS IDL message representation, commonly native structures, such as OpenCV matrices, needs to be converted to POD structures back and forth when sending or receiving messages (red blocks case here under). TAS-I Team developed a set of custom data adapters supported by ROS 2 API, allowing the framework to directly exchange native structures of choice via IPC without conversion overheads. This proved extremely useful to process and exchange heavy data such as structured point clouds.

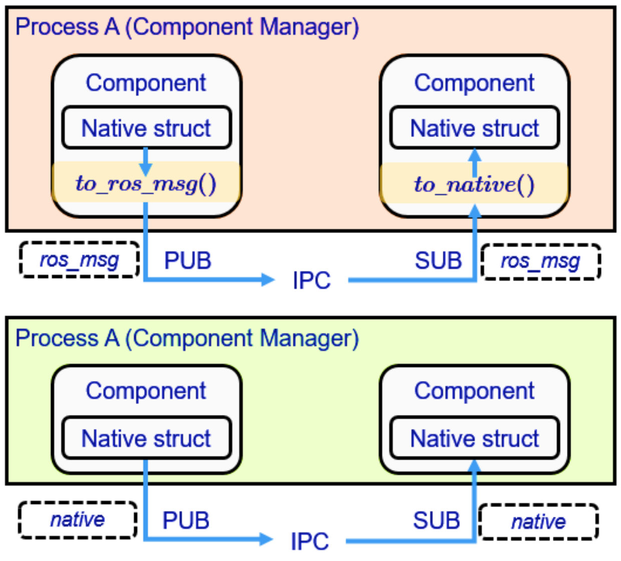

Figure 9 - Message adapters in ROS 2

TAS-I Team carefully partitioned the AutoNav runtime across the two rover OBCs, minimizing the amount of data exchanged over IP. Then, on each of the two OBCs, the majority of the component are grouped together dynamically in the same component container, effectively maximizing the data exchanged via intra-process (where only tokens - addresses- are exchanged, with the data effectively never copied).

Moreover, the DDS communication was limited to each of the host, allowing only specific data streams to flow on the network via a more efficient protocol called Eclipse Zenoh, relying on PtP tunnels instead of broadcast communication [11].

8. DL ALGORITHM INTEGRATION

Four different task-specific Deep Neural Networks (DNNs) were developed and trained and easily integrated in the ROS 2 architecture of the rover, demonstrating the flexibility of the SINAV software stack in presence of additional SW modules or agents in the same local network.

DNNs design is based on open-source SOTA architectures while training has been performed on Thales custom datasets. To answer the low availability of specific datasets, realistic 3D simulations of the stack deploy environment have been created and used for training. Synthetically trained DNNs have nonetheless achieved good results in real testing environments. Inference is based on open-source libraries (ONNX-runtime) and has been tuned to work within a low computational resource environment.

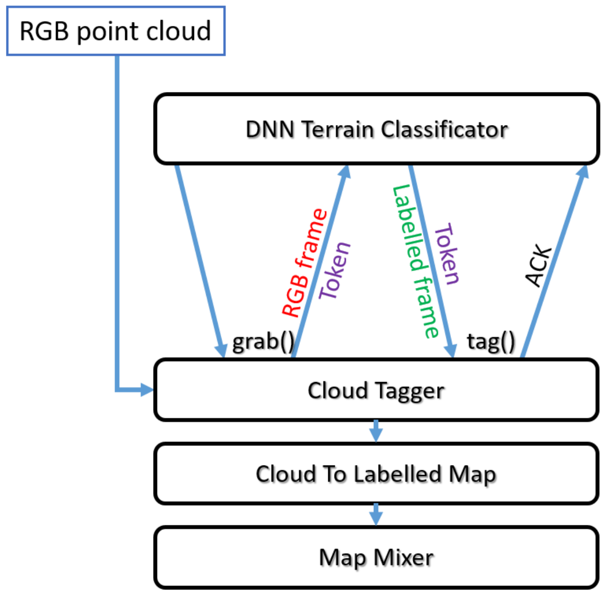

Figure 10 - Integration of terrain segmentation

The task covered by the four networks are:

- terrain segmentation;
- detection of mission relevant objects;
- monocular depth estimation;
- image super-resolution.

In particular, segmentation output has been considered by the stack to update the cost map for navigation. In this paper, the integration approach is shown to demonstrate the flexibility of the architecture. The DNN Terrain classification node, which can be executed even on a remote machine, sends a *grab()* request to the Cloud Tagger component, which resides in the same component container where the stereo camera HAL component is (this is to reduce overhead at minimum). A point cloud is retained in memory, RGB data is extracted and sent back to the DNN module, so that only the minimum amount of information is shared over the IP network. The DNN assigns a label to each pixel of the retrieved image, and calls the *tag()* service sending back a 1-channel image with pixel values corresponding to such labels. The Cloud Tagger retrieves the source point cloud from memory, adds a “label” channel to it dumping all the labels from the received raster, and releases the tagged cloud downstream. Another component then receives this tagged cloud, applies grid binning and assigns to each cell a class (depending on the labels of the points inside the corresponding bin). The originated traversability map can then be mixed with other maps (such as the default one encoding slopes) with different numerical criteria.

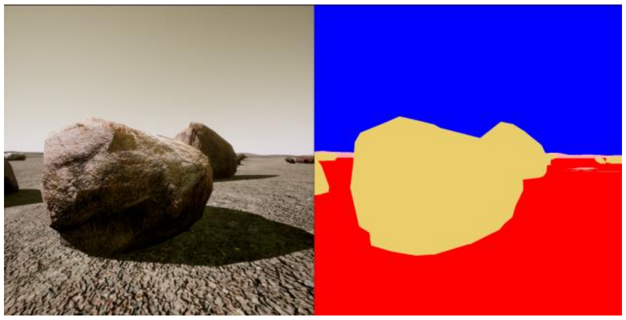
Figure 11 - Segmentation output (synthetic dataset)

9. TEST CAMPAIGN

The surface segment test campaign took place in the RoXY facility in July 2023, and was successfully executed by TAS-I and PoliTo Teams. The rover navigated autonomously to waypoints defined at runtime by a remote console, sent via wireless PtP link to the rover itself. Whenever the rover reached a goal, a new one was chosen and sent to it until the minimum required demo mission duration (10 minutes) was reached. The rover navigated continuously, detecting and avoiding obstacles along its path by re-computation of its path towards the requested goal. For each traverse, travel time and distance have been logged to extract average speed as the final metric. Four runs were executed in different light conditions (sunrise, clear sky, cloudy, sunset) and two more were executed at dusk and by night. The average speed has been always above 10 cm/s, which is way above the initial target (6-8 cm/s). Moreover, an additional run has been done processing additional segmentation data coming from the additional DL terrain segmentation module, demonstrating avoidance of sandy craters that are present in the RoXY facility.

10. RESULTS AND CONCLUSIONS

The tests demonstrated the capability of the system to navigate continuously in both day and night time, at a mission average > 10 cm/s. By comparison, NASA Perseverance can travel at 4.5 cm/s, while MER rovers were only able to navigate at 1 cm/s, which was the reference for the tenfold target increase of SINAV study. Taking into account European state of the art, Rosalind Franklin has been designed for just 14 m/h (0.4 cm/s) [12].

Of course, this raw comparison is not completely fair, since SINAV relied on powerful COTS avionics; on the other side, the ROS 2 framework still embodies some overhead that penalizes SINAV code, together with the one associated to the accentuated modularization of the processing pipelines. The current status of the system is evaluated as a solid TRL 4 overall, but TAS-I intends to mature it throughout the next years due to the growing interest in European market to create a segment of multi-purpose lunar rovers.

Finally, it is important to highlight that in the future the Team aims to leverage advancements in Space ROS, which is the NASA/PickNik initiative to make ROS 2 compliant to space SW standards, so overall TRL could be boosted by such developments.

References

[1] P. Messidoro, M. A. Perino and D. Boggiatto, "Enabling technologies for space exploration systems: The STEPS project results and perspectives," *Acta Astronautica,* vol. 86, pp. 219-236, 2013.

[2] A. Biggio, A. Sperindé, S. Torelli, E. Simetti, C. Inni, L. Vercellino, F. Salvioli and B. Bona, *Validation and verification of modular GNC by means of TAS-I robot management framework in outdoor rovers exploration facility,* Turin, 2015.

[3] A. Biggio, S. Torelli, A. Sperindé, E. Simetti, C. Ianni, F. Salvioli, L. Vercellino and B. Bona, *Design and implementation of a robot management framework and modular gnc for robotic space exploration,* Turin, 2016.

[4] K. Uno, L.-J. Burtz, M. Hulcelle and K. Yoshida, "Qualification of a Time-of-Flight Camera as a Hazard Detection and Avoidance Sensor for a Moon Exploration Microrover," *Trans. JSASS Aerospace Tech. Japan,* vol. 16, no. 7, pp. 619-627, 2018.

[5] R. P. Suresh, Sridhar, Vinay, J. Pramod and V. Talasila, "Zero Velocity Potential Update (ZUPT) as a Correction Technique," in *3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Bhimtal, 2018.

[6] S. Macenski, F. Martin, R. White and J. G. Clavero, "The Marathon 2: A Navigation System," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, 2020.

[7] P. Roncagliolo, "Evaluation and Implementation of Modern Path Planning Algorithms for Planetary Exploration Rovers," Genoa, 2020.

[8] R. Ghzouli, T. Berger, E. B. Johnsen, A. Wasowski and S. Dragule, "Behavior Trees and State Machines in Robotics Applications," *IEEE Transactions on Software Engineering,* pp. 1-24, 21 April 2023.

[9] S. Macenski, T. Foote, B. Gerkey, C. Lalancette and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics,* vol. 7, no. 66, 2022.

[10] S. Macenski, A. Soragna, M. Carrol and Z. Ge, "Impact of ROS 2 Node Composition in Robotic Systems," *IEEE Robotics and Automation Letters,* 2023.

[11] W.-Y. Liang, Y. Yuan and H.-J. Lin, "A Performance Study on the Throughput and Latency of Zenoh, MQTT, Kafka, and DDS," 2023.

[12] M. Winter, B. Chris, V. Pereira, R. Lancaster, M. Caceres, K. Mcmanamon, B. Nye, N. SIlva, D. Lachat and M. Campana, "ExoMars Rover Vehicle: Detailed Description of the GNC System," in *13th Symposium on Advanced Space Technologies in Robotics and Automation*, Noordwijk, 2015.