A novel state estimation framework for a four-wheeled lunar rover with active articulated suspensions

Acta Astronautica 244 (2026) 54–70

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Research paper

A novel state estimation framework for a four-wheeled lunar rover with active articulated suspensions

Giacomo Franchini [a,b,*], Patrick Roncagliolo [c], Davide Graziato [c], Alessandro Ruggiero Chiminelli [c], Andrea Merlo [c], Marcello Chiaberge [a,b]

[a] Department of Electronics and Telecommunications, Polytechnic of Turin, Corso Duca degli Abruzzi, 24, Turin, 10129, Italy
[b] Polytechnic of Turin Interdepartmental Centre for Service Robotics, Corso Francesco Ferrucci, 112, Turin, 10141, Italy
[c] Thales Alenia Space Italia S.p.A., Strada Antica di Collegno, 253, Turin, 10146, Italy

ARTICLE INFO

Keywords:
Moon exploration
Rover odometry
Sensor fusion
ROS 2

ABSTRACT

In recent years, space agencies and private companies have shown a renewed interest in Moon exploration, with the ultimate goal of having a permanent human presence on the surface by the 2030s. A crucial step in this direction involves deploying robotic missions. To this purpose, at Thales Alenia Space Italia S.p.A., a versatile, multipurpose four-wheeled robotic platform with active suspensions has been designed as a common reference mobility system able to perform a multitude of tasks on the Moon, spanning from South Pole exploration to In Situ Resource Utilization. In this context, we propose a state estimation framework based on factor graph optimization to perform sensor fusion and estimation of rover odometry. The implementation relies on the ROS 2 package *Fuse*, which has been optimized and extended with 3D sensors and motion models. The paper contributions are twofold: firstly, we developed a method for estimating the rover's linear and angular body velocities based on data from the wheel-steer-suspension assembly encoders. Three-dimensional components of the body velocity and the associated covariance matrix are computed by accurately determining the plane of instantaneous motion of the rover. Secondly, dedicated sensor models are used to fuse the estimated body velocities with readings from the on-board IMU and with odometry input computed from a visual pipeline. The latter can be obtained both from a stereo camera by matching visual features, or by registering point clouds gathered by time-of-flight sensors, allowing autonomous navigation in any lighting condition. Constraints derived from different sensors are joined by leveraging a motion model that encapsulates the entire span of locomotion modalities allowed by the rover geometry. The method has been validated in simulations built on Project Chrono and with the rover prototype navigating in a representative facility. Results demonstrate that the framework is highly optimized, efficiently facilitating the integration of multiple sensor readings from various sources, delivering fused odometry outputs at a high frequency, and ensuring accurate and real-time state updates.

1. Introduction

Decades after the last human landing, the exploration of the Moon has returned as a central point of interest for all the main international actors in the space sector, not only from a scientific perspective but also from an economic one. Exploration missions from national agencies, such as NASA Artemis, CNSA Chang'e, ESA Argonaut, JAXA Lunar Exploration Program and ISRO Chandrayaan, among the main ones, are being actively complemented by commercial programs pursued by private companies in what is becoming a lunar economy, with the main goal of establishing a long-term human presence on the Moon, opening up plenty of business opportunities and giving the chance to carry out scientific experiments directly from the lunar surface. However, enabling technologies to fully achieve these objectives, such as reliable systems for landing and surface operations, are still missing. In this context, the Domain Exploration and Science of Thales Alenia Space Italia (TAS-I) S.p.A. developed a modular, multipurpose, and autonomous robotic platform with the key idea of adapting a common locomotion system to numerous scenarios, such as lunar South Pole exploration, in-situ resource utilization, and payload deployment, decreasing development time and costs. Different operation profiles are

* Corresponding author at: Department of Electronics and Telecommunications, Polytechnic of Turin, Corso Duca degli Abruzzi, 24, Turin, 10129, Italy.
E-mail addresses: giacomo.franchini@polito.it (G. Franchini), patrick.roncagliolo@thalesaleniaspace.com (P. Roncagliolo), davide.graziato-somministrato@thalesaleniaspace.com (D. Graziato), alessandro.ruggiero@thalesaleniaspace.com (A.R. Chiminelli), andrea.merlo@thalesaleniaspace.com (A. Merlo), marcello.chiaberge@polito.it (M. Chiaberge).

https://doi.org/10.1016/j.actaastro.2026.02.010
Received 3 October 2025; Received in revised form 18 December 2025; Accepted 5 February 2026
Available online 6 February 2026

possible thanks to the rover's kinematic configuration, which consists of four driving wheels, each equipped with active steering and suspension, which allows the accurate control of the chassis attitude and ground clearance. These have been identified as key factors in safely traversing harsh and steep terrains, overcoming obstacles, and satisfying the orientation and pointing constraints required by on-board payloads.

To achieve full autonomous behavior for navigation and mission operations, on-board systems need to be informed accurately and in real-time about the rover state, specifically the chassis velocity and pose. To this end, we based our implementation on the Robot Operating System 2 (ROS 2) [1] package *Fuse* [2,3]: this provides the base framework to gather readings from navigation sensors and to build the factor graph for estimating the robot state. We extended and enhanced this framework by integrating dedicated 3D sensor models to convert on-board Inertial Measurement Unit (IMU) measurements and odometry inputs from a visual pipeline into three-dimensional constraints over the rover state variables inside the factor graph.

Moreover, we developed a method for approximating the local terrain plane and obtaining the chassis linear and angular velocity vectors with the associated covariance matrix, based only on data from the encoders of the four wheel-steer-suspension assemblies. In this way, we exploit rover kinematics as an additional graph constraint for the chassis motion. The resulting factor graph represents a non-linear least squares problem, which is solved using the Google Ceres Solver [4] library to retrieve the optimized rover states. Graph size and optimization time are limited and controlled by marginalizing out variables that are older than a specific lag duration. The described framework allows tracking the evolution of the rover pose and velocity accurately and at high frequency, during all kinds of maneuvers enabled by the platform kinematics. An high-level diagram of the proposed state estimation framework is reported in Fig. 1.

The main contributions of this paper are the following:

- Custom 3D models to constrain the rover state based on measurements from on-board sensors;
- A module for the estimation of the rover's instantaneous ground plane and chassis velocities, based on measurements from the joints' encoder and the rover's kinematic model;
- Their integration into a sensor fusion framework based on factor graph optimization, allowing the estimation of the rover's odometry in real-time.

This paper is organized as follows: in Section 2, a review of the state of the art is reported, with particular focus on state estimation methods for lunar and planetary exploration rovers; in Section 3, an overview of the rover considered in this study, its kinematic model, and the estimation of the chassis velocities are presented. The sensor fusion approach, its mathematical background and the ROS 2 integration are presented in Section 4, while Section 5 describes the test campaigns conducted to validate the framework. Test results are summarized and analyzed. Section 6 closes this paper with conclusions about the work done and future improvements.

2. Related work

In robotics, the state estimation of an agent addresses the problem of estimating state variables that are not directly observable but can be deduced, together with their associated uncertainties, from noisy sensor measurements. There is no unique definition of what the robot state is, and its choice depends on multiple factors, such as the specific robotic platform and the considered use cases. A typical approach, which is followed in this paper, is to define it as the set of the instantaneous velocity of the robot chassis and its pose with respect to a local, fixed reference frame. Estimating the robot state continuously and at a high rate is fundamental, as these predictions are provided as input to the robot's guidance, navigation, and control (GNC) subsystem and exploited to achieve fully autonomous behavior and accomplish mission tasks. This chapter reviews the key contributions in this field. We organize related work into three main areas: first, we review general methods employed for state estimation in robotics. We then narrow down to applications targeted to lunar and planetary surface exploration. Lastly, we focus on techniques for modeling robot velocity kinematics.

Filtering-based techniques, summarized in [5,6], are among the most widely explored methods for state estimation in robotics. These aim to retrieve the state variables and uncertainties from sensor measurements in a prediction–correction fashion: the next state is initially predicted by propagating the previous one according to a motion model of the system, guided by a control input. Then, the prediction is corrected by integrating the incoming sensor readings. Two main approaches are available: the first is based on the Kalman Filter [7] and its extensions, the Extended Kalman Filter (EKF) and the Unscented Kalman Filter [8] (UKF), which were developed to deal with non-linear models and non-Gaussian uncertainties. Recent works build on these foundations to improve estimation accuracy and filter robustness: in [9,10], the authors perform state estimation of a ground mobile robot using only proprioceptive sensors, with an invariant EKF (InEKF). The latter is an adaptation of the original algorithm that aims to reduce errors introduced by linearizing the robot and sensor models. By tracking the robot state errors rather than the state itself, and parametrizing it using a matrix Lie Group, the models can be considered linear under the small-error assumption, and the error evolution can be tracked on the manifold, where it follows log-linear dynamics.

In the second one, called particle filter [11], the robot state probability density is approximated by a set of random samples, each one representing the state at a specific time point. State updates are performed through a sequential process of sampling and resampling, for the prediction and correction steps respectively, incorporating both time evolution of the system and measurement updates to refine the state estimate.

On the other end, graph-based techniques treat state estimation as a Maximum A Posteriori (MAP) estimation problem over the joint probability distribution function of states at different times. In this family of methods, the joint probability distribution is modeled by a factor graph, where the robot states and constraints generated from sensor and motion models represent the graph variables [12]. Typically, in state estimation, the graph is built incrementally, and inference over its variables is performed frequently [13] to retrieve optimized state variables at a high rate.

Recent works explored integrating data-driven approaches into the described methods. [14] propose to embed a stochastic neural network in an Ensemble Kalman Filter (EnKF) to model the process noise. In this way, the learned module allows the filter to adapt the uncertainty on the system evolution, increasing its flexibility and robustness to missing or noisy observations. Experiments have been performed on both visual odometry and manipulation tasks. In [15], state estimation for an omnidirectional wheeled rover is performed using factor graph optimization, fusing IMU and wheel odometry measurements. Here, the authors employ a neural network that learns the observation and noise models from sensor data and, correspondingly, adapts the factors within the graph, leading to better accuracy estimates.

All the described approaches are discrete-time state estimation methods. This means that robot state variables are inferred only at specific time points, typically when a sensor measurement is available. Recently, some research [16] proposed a continuous-time approach. There, state evolution is modeled as a continuous function, making variables available at any requested time.

State estimation has been extensively studied across a multitude of applications. In the context of space robotics, however, additional challenges need to be considered. Firstly, the lack of position, navigation, and timing systems providing absolute localization signals forces the estimation module to rely solely on local information. Moreover,

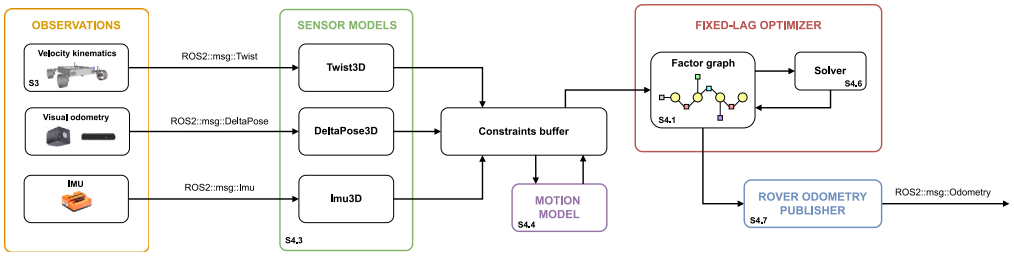


Fig. 1. The proposed rover state estimation framework. In our setup, observation of rover motion comes from three sources: visual odometry from stereo and ToF cameras, chassis twist provided by a velocity kinematics pipeline, and IMU raw measurements. Sensor models process the observations and generate graph constraints between state variables. An omnidirectional motion model provides additional constraints between unconnected variables. The factor graph is optimized at high frequency over a sliding window of recent states. Updated rover poses and velocities are queried and published to the ROS 2 environment as an odometry message.

the available sensor suite is restricted by constraints typical of space segments, such as low mass, limited power consumption, and computational resources. Additionally, space robots typically operate in unstructured environments and in the presence of extreme lighting conditions, making reliable perception more challenging. A number of filtering methods are available in the literature: an early approach in this sense is the one proposed in [17], where a state estimation pipeline for the Mars Exploration Rovers is introduced. In particular, an EKF is implemented to fuse rover wheel odometry with motion estimations derived from sequential visual matching of terrain maps generated online. In this case, the state representation accounts only for the rover's planar motion. An augmented rover state vector is considered in [18], where the wheel slippage ratio is jointly estimated along with the six-dimensional rover pose within an UKF. Again, wheel and visual odometry are provided as input to the filter, and wheel slip is obtained by comparing the velocities computed by the two pipelines. Moreover, input state vector components are modified adaptively by estimating the instantaneous accuracy of visual odometry and introducing a gyro-odometry model. In [19], a purely inertial navigation for planetary rovers is proposed to reduce the high computational cost of the visual odometry pipeline. This is achieved by fusing IMU measurements within an EKF, adding additional constraints derived from the rover's kinematic configuration and from state-aware zero-type pseudo measurements. This allows for bounding the characteristic drifts inherent to inertial navigation, but at the cost of performing periodic stops of the rover to integrate zero-type measurements within the filter. A comparison of outlier-robust Kalman filters for improved wheel-inertial odometry of a planetary rover is illustrated in [20]. Three filter adaptations have been implemented and tested: Huber Regression Kalman Filters, which minimize a Huber cost function instead of a classical squared error function to decrease measurement outliers' influence; Covariance Scaling Kalman Filters, where a chi-squared test is performed on the Mahalonobis distance between the actual and the predicted measurement to determine if it is an outlier. The test returns a factor used to scale the measurement covariance accordingly. Variational Kalman Filters use an auxiliary variable within the measurement model to obtain a measurement distribution with heavier tails after marginalization, which highlights outliers. Since the obtained distribution is non-Gaussian, the posteriors are approximated with variational inference. Field experiments performed with a testbed rover platform in analogue terrain demonstrated improved odometry accuracy for all these adaptations compared to the base EKF.

[21] presents the continuous relative localization of an autonomous planetary rover based on a particle filter, employed as the tracking module of a full Simultaneous Localization And Mapping (SLAM) system. At every step, each particle state is first propagated according to the delta transformation provided by the rover wheel odometry. The weights are updated based on scan-matching results between the current point cloud and the local map constructed by the rover. Point clouds come from an on-board stereo camera, and matching is performed through a single Iterative Closest Point (ICP) iteration. The particle filter tracks only the rover's plane motion: the authors assume that the on-board IMU provides accurate roll and pitch values, while the rover's z component of pose is retrieved during localization in the global map.

Factor graph approaches to state estimation for planetary rovers are also present in the literature: [22] proposes an inertial navigation framework that limits IMU drift by adding motion estimation constraints to the graph, generated from visual and LiDAR odometry. However, at present, LiDAR sensors are not viable for on-board use in space exploration rovers due to their high power consumption. [23], instead, introduces an improved version of Huber loss factor graph that screens anomalous measurements via multi-state coupling, achieving more robust and accurate estimations. The framework has been tested with simulated data for a Mars rover equipped with a navigation camera, an IMU, and a sun sensor.

Together with on-board sensor measurements, in rover state estimation algorithms, it is typical to fuse the predictions of the robot velocity computed from the sensed wheel rotation velocities. This is achieved through the definition of the kinematic model, which, for wheeled rovers, is typically described as a set of open kinematic chains, one for each wheel, starting from the rover body and ending at the contact points with the ground. In lunar exploration, addressing harsh, uneven terrains while fulfilling mission objectives requires designing complex rover kinematic models. In the literature, such modeling for advanced wheeled rovers builds upon the concepts introduced by [24]. Here, the authors present a generic method for formulating rover kinematic equations that enable the computation of chassis velocities as a function of joint velocities by constructing Jacobian matrices that encapsulate the robot's kinematic chains. This foundational work, however, is limited to motion on planar surfaces, and several approaches that extend its principles have been proposed. In [25], the authors derive the kinematic model for a generic, articulated wheeled rover navigating on non-flat terrains by employing the Denavit–Hartenberg convention. Jacobians for velocity kinematics are obtained by differentiating the model. [26] generalizes the procedure of [25] by performing coordinate-free vector differentiation, with the advantages of not being limited to a specific coordinate choice for the links in the chain, and avoiding the complexity of differentiating coordinatized homogeneous transformation matrices. In [27], the same authors enhance the velocity kinematic model with a method for predicting body-level slip.

The non-linear components of a skid-steering rover kinematic model are learned with a neural network in [28]. Network parameters that are terrain-independent are first learned offline, while online training is performed by inserting the network into a factor graph and using it to learn hanging features. The network estimated the corrected 2D twist of the rover chassis from proprioceptive sensor inputs.

3. Velocity kinematics

In velocity kinematics, the robot kinematic model is used to establish the relationship between the velocities of the joints and the

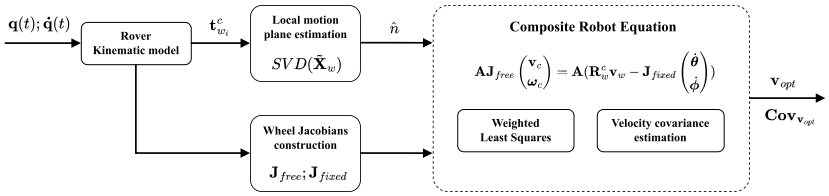


Fig. 2. Diagram of the rover inverse velocity kinematics pipeline. The instantaneous kinematic configuration is computed from the joint states, provided to the local motion plane estimation algorithm, and used to construct the wheel Jacobian matrices. By solving the Composite Robot Equation via weighted least squares, the optimal chassis velocity and its covariance matrix are estimated.

Table 1
Summary of notation conventions used in the paper.

Type	Symbol	Description
Scalar	a	Lowercase, non-bold
Vector	$\mathbf{b}$	Lowercase, bold
Matrix	$\mathbf{M}$	Uppercase, bold
Transformation matrix	$\mathbf{T}^i_j$	From parent frame i to child frame j
Rotation matrix	$\mathbf{R}^i_j$	From parent frame i to child frame j
Translation vector	$\mathbf{t}^i_j$	From parent frame i to child frame j
Linear/angular velocity	${}^l\mathbf{v}_k, {}^l\boldsymbol{\omega}_k$	Velocity of frame k, expressed in frame l

velocity of a reference frame, typically the end effector. In this work, we consider the robot chassis as its base frame, while the wheels are treated as end effectors. We then focus on inverse velocity kinematics to compute the linear and angular velocities of the chassis from the angular velocities of the wheels. The workflow we followed is illustrated in Fig. 2.

As in related works, we also based our implementation on the method introduced by [24]: first, the robot kinematic model is derived and stored. The kinematic configuration of the rover, described by the joint positions $\mathbf{q}(t)$ and velocities $\dot{\mathbf{q}}(t)$, is used to estimate the local plane of motion and to propagate the wheel velocity up to the chassis for every single kinematic chain. The obtained relationships are stacked together to form an overdetermined system of linear equations, which is then solved using a least squares approach. The best-fit chassis velocities $\mathbf{v}_{opt}$ are computed, while estimation residuals $\mathbf{r}$ are leveraged to derive the associated velocity covariance matrix $\mathbf{Cov}_{\mathbf{v}_{opt}}$. The entire process is detailed in the next subsections.

3.1. Conventions

For clarity, we present the conventions for symbols used in this paper here. Scalar quantities are denoted by lowercase non-bold symbols (e.g., a); vector quantities by lowercase bold symbols (e.g., $\mathbf{b}$); and matrices by uppercase bold symbols (e.g., $\mathbf{M}$). Transformation matrices, rotation matrices, and translation vectors from a parent frame i to a child frame j are indicated as $\mathbf{T}^i_j$, $\mathbf{R}^i_j$, and $\mathbf{t}^i_j$. Linear and angular velocity vectors of frame k, expressed in frame l, are denoted as ${}^l\mathbf{v}_k$ and ${}^l\boldsymbol{\omega}_k$, respectively. If the left apex is not reported, then the velocity vector is assumed to be expressed in frame k. Table 1 summarizes symbol conventions.

3.2. Modular wheeled robotic platform for moon missions

The multipurpose wheeled robotic platform considered in this study is illustrated in Fig. 3. It has been designed at TAS-I under the ESA's European Moon Rover System (EMRS) program [29], with the goal of developing a modular rover capable of satisfying a variety of use cases and mission profiles. The aim was to overcome the concept of realizing mission-specific platforms to decrease mission development time and costs.

The rover locomotion system consists of four independent wheel assemblies, each one equipped with active steering and suspension systems. Standalone, active suspensions equipped with Series Elastic Actuators (SEA) enable the platform to maintain a desired chassis height and attitude with respect to the ground, while simultaneously minimizing the stresses transmitted to on-board payloads, even when navigating steep and harsh terrains. The rover is capable of driving in a multitude of locomotion modalities due to its independently steered wheels, as illustrated in Fig. 4. This kinematic configuration provides the vehicle with a high level of flexibility, enhancing mission adaptability as payloads with different operational constraints can be integrated on the same platform.

The rover's kinematic model reflects its locomotion system configuration: starting from the chassis' geometric center, four equal kinematic chains extend to the wheel rotation axes. A reference frame is associated with every movable body in the chain, and every frame is connected with a single parent frame by a revolute joint. A schematic representation of one chain is reported in Fig. 5: the child frame of the rover chassis is associated with the suspension joint; from there, the rover body is connected with the steering-driving assembly by a pantograph link, which allows maintaining the parallelism of the two while controlling the chassis attitude and ground clearance. A support frame is defined at the end of the pantograph link. The last two frames of the kinematic tree are associated with the steering and driving joints, respectively. The transformation matrix from parent to child frame is defined in the classical homogeneous way:

$$\mathbf{T}^{parent}_{child} = \begin{bmatrix} \mathbf{R}^{parent}_{child} & \mathbf{t}^{parent}_{child} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{1}$$

Eq. (2) reports the full kinematic chain for wheel i. From now on, link names will be shortened to ensure clarity and conciseness.

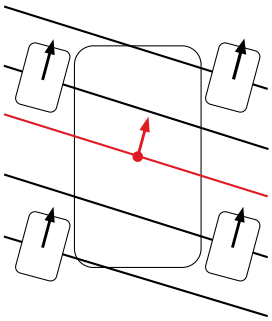
$$\mathbf{T}^{chassis}_{wheel_i} = \mathbf{T}^{chassis}_{susp_i}\mathbf{T}^{susp_i}_{pant_i}\mathbf{T}^{pant_i}_{steer_i}\mathbf{T}^{steer_i}_{wheel_i} = \mathbf{T}^{c}_{sp_i}\mathbf{T}^{sp_i}_{p_i}\mathbf{T}^{p_i}_{st_i}\mathbf{T}^{st_i}_{w_i} \tag{2}$$

3.3. Wheels-terrain contact plane

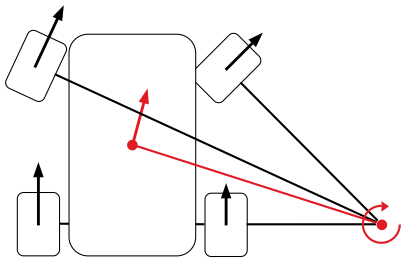
To correctly estimate the direction of motion of the chassis in 3D space via inverse velocity kinematics, it is compulsory to define the directions of the wheel axis velocities. These depend on the geometry of the traversed terrain, and on the positions of the contact points with the ground. Estimating these contact points is a challenging task: a large number of solutions have been proposed, which typically require either additional sensors mounted on the wheel surface [30] or the integration of a vision-based pipeline [31–33].

Unlike other works, we propose a simpler yet efficient method that directly estimates a plane approximating the local terrain profile from proprioceptive information alone. By leveraging only the joint encoder measurements and the rover kinematic model, the method infers a plane that locally approximates the terrain profile under the assumption of continuous wheel–ground contact, without introducing extra sensing modalities. We assume that the rover is continuously in contact with the terrain, meaning that the wheel velocity vectors lie on the local plane. The constraints imposed by the pantograph links then

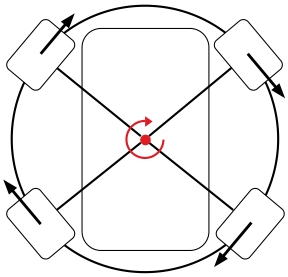
Fig. 3. The multipurpose robotic platform at the Rover eXploration facilitY at TAS-I.



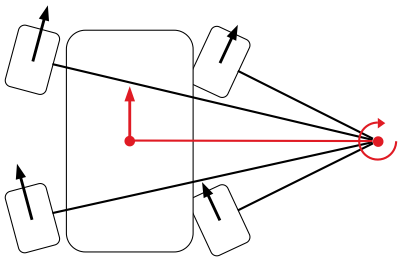
(a) Crabbing mode: the rover maintains a fixed heading and only translates in the chassis plane.



(b) Single-axis Ackermann mode: only the wheels on the front or rear axis are steered.



(c) Turn-in-place mode: the rover can rotate around any desired point in the chassis plane.



(d) Double-axis Ackermann mode: all four rover wheels are steered simultaneously.

Fig. 4. The available rover locomotion modes. For every wheel, the linear velocity and the turning radius are reported. Linear and angular velocities of the chassis center of mass, together with its turning radius, are highlighted in red.

determine the direction of the chassis velocity, which is also parallel to this plane. The method is based on the idea that, in both configurations of active or passive suspensions, variations in terrain inclination will be captured by the rover's suspension encoders and reflected in the vertical distances of the wheel axis from the chassis (see Fig. 6). With this in mind, for every wheel we compute the instantaneous position of its surface lowest point relative to the chassis frame $\mathbf{t}^c_{w_i} = (x_{w_i}, y_{w_i}, z_{w_i})^\top$, using the rover kinematic model defined in Eq. (2), and assume those as the terrain contact points. These are stacked together to create the wheel position matrix $\mathbf{X}_w$:

$$\mathbf{X}_w = \begin{bmatrix} x_{w_1} & y_{w_1} & z_{w_1} \\ x_{w_2} & y_{w_2} & z_{w_2} \\ x_{w_3} & y_{w_3} & z_{w_3} \\ x_{w_4} & y_{w_4} & z_{w_4} \end{bmatrix} \tag{3}$$

We consider as local terrain surface the best-fit plane that minimizes the average squared perpendicular distance from every point in $\mathbf{X}_w$. A plane in space is described by Eq. (4) and can be uniquely defined if its

Link name	Reference	Joint type
Chassis	L1	Fixed
Suspension	L2	Revolute (X)
Pantograph	L3	Revolute (X)
Steer	L4	Revolute (Z)
Wheel	L5	Revolute (Y)

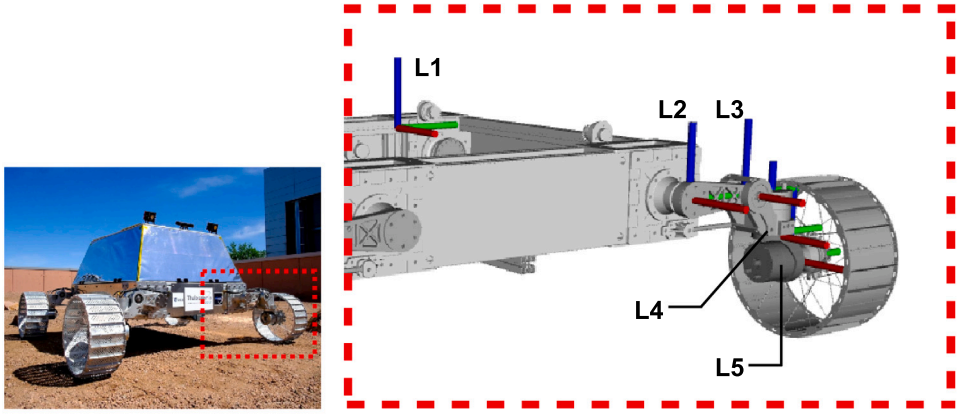


Fig. 5. Detail of one of the rover kinematic chains. Frames X, Y, and Z axes are represented in red, green, and blue, respectively.

normal versor $\hat{\mathbf{n}} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)$ and a point $\mathbf{p}_0 = (x_0, y_0, z_0)$ that belongs to it are known.

$$\hat{n}_x(x - x_0) + \hat{n}_y(y - y_0) + \hat{n}_z(z - z_0) = 0 \tag{4}$$

$$\tilde{\mathbf{X}}_w = \mathbf{X}_w - \mathbf{c}_w = \mathbf{X}_w - \frac{1}{4}\sum_{i=1}^{4}(x_{w_i}, y_{w_i}, z_{w_i}) \tag{5}$$

What we are interested in is the orientation of the plane, so the only variable to be estimated is the normal vector: to this end, we center the wheel position matrix by subtracting the contact points centroid $\mathbf{c}_w$ row-wise, as described in Eq. (5). By setting up the linear, homogeneous, and overdetermined system $\tilde{\mathbf{X}}_w\hat{\mathbf{n}} = \mathbf{0}$, the local plane normal versor is the non-trivial solution that minimizes the residuals in the least-squares sense. This is computed by performing the singular value decomposition of $\tilde{\mathbf{X}}_w$:

$$SVD(\tilde{\mathbf{X}}_w) = \mathbf{USV}^\top \tag{6}$$

The columns of $\mathbf{V}$, called right singular vectors of $\tilde{\mathbf{X}}_w$, represent the three principal components of the best-fit local terrain plane: the unit vector that we search for is the rightmost singular vector corresponding to the least singular value of $\tilde{\mathbf{X}}_w$. The plane orientation is stored in the matrix $\mathbf{R}_{plane}$, which encodes the rotation from the estimated normal vector to the vertical direction of a fixed inertial frame. This frame is assumed to have its xy plane coinciding with an ideal horizontal surface and its z axis defining the upward direction. Assuming the ideal condition of no wheel slippage in both longitudinal and lateral directions, the wheel center velocity ${}^{w_i}\mathbf{v}_{w_i}$ is aligned with the x axis of the wheel reference frame and is obtained from Eq. (7), where w_i and r_i denote the angular velocity and the radius of wheel i.

$${}^{w_i}\mathbf{v}_{w_i} = \mathbf{R}_{plane}\begin{pmatrix} w_i r_i \\ 0 \\ 0 \end{pmatrix} \tag{7}$$

3.4. Chassis velocity estimation

Considering a generic rover motion, the chassis angular and linear velocities expressed in the chassis frame can be written as $\boldsymbol{\omega}_c = (\omega_{c_x}, \omega_{c_y}, \omega_{c_z})^\top$ and $\mathbf{v}_c = (v_{c_x}, v_{c_y}, v_{c_z})^\top$. These are estimated using inverse velocity kinematics by mapping the velocity of each wheel to the chassis itself. To find this mapping function, we start from propagating the angular and linear velocity of the chassis up to the wheel frame using the kinematic equations of rigid bodies described by Eqs. (8) and (9): these are applied to the whole kinematic chain to relate the child and parent link velocities, identified by indices i and i-1, taking into account the relative velocity between the two.

$${}^i\boldsymbol{\omega}_i = \mathbf{R}_i^{i-1}{}^{i-1}\boldsymbol{\omega}_{i-1} + {}^i\boldsymbol{\omega}_{rel} \tag{8}$$

$${}^i\mathbf{v}_i = \mathbf{R}_i^{i-1}({}^{i-1}\mathbf{v}_{i-1} + {}^{i-1}\boldsymbol{\omega}_{i-1} \times \mathbf{t}_i^{i-1}) \tag{9}$$

For every kinematic chain, angular and linear velocities are related as:

$$\begin{aligned} \boldsymbol{\omega}_{sp} &= \mathbf{R}_{sp}^{c}\boldsymbol{\omega}_c + \boldsymbol{\omega}_{rel_{sp}} \\ \boldsymbol{\omega}_{p} &= \mathbf{R}_{p}^{sp}\boldsymbol{\omega}_{sp} + \boldsymbol{\omega}_{rel_{p}} \\ \boldsymbol{\omega}_{st} &= \mathbf{R}_{st}^{p}\boldsymbol{\omega}_{p} + \boldsymbol{\omega}_{rel_{st}} \\ \boldsymbol{\omega}_{w} &= \mathbf{R}_{w}^{st}\boldsymbol{\omega}_{st} \end{aligned} \tag{10}$$

$$\begin{aligned} \mathbf{v}_{sp} &= \mathbf{R}_{sp}^{c}(\mathbf{v}_c + \boldsymbol{\omega}_c \times \mathbf{t}_{sp}^{c}) \\ \mathbf{v}_{p} &= \mathbf{R}_{p}^{sp}(\mathbf{v}_{sp} + \boldsymbol{\omega}_{sp} \times \mathbf{t}_{p}^{sp}) \\ \mathbf{v}_{st} &= \mathbf{R}_{st}^{p}(\mathbf{v}_{p} + \boldsymbol{\omega}_{p} \times \mathbf{t}_{st}^{p}) \\ \mathbf{v}_{w} &= \mathbf{R}_{w}^{st}(\mathbf{v}_{st} + \boldsymbol{\omega}_{st} \times \mathbf{t}_{w}^{st}) \end{aligned} \tag{11}$$

Where $\boldsymbol{\omega}_{rel_{sp}} = (\dot{\phi}, 0, 0)^\top$ and $\boldsymbol{\omega}_{rel_{st}} = (0, 0, \dot{\theta})^\top$ are vectors containing the angular velocities of suspension and steering joints, respectively, while $\boldsymbol{\omega}_{rel_p} = -\boldsymbol{\omega}_{rel_{sp}}$ due to the constraint imposed by the pantograph link.

Eq. (11) allow the wheel axis velocity in the wheel reference frame to be expressed as a function of the velocities and dimensions of the other robot links. By writing $\mathbf{v}_w$ with respect to the chassis frame, we can rearrange it to obtain the relationship shown in Eq. (12). To clarify that this refers to a single wheel, we reintroduced the subscript i. Matrices $\mathbf{J}_{free_i}$ and $\mathbf{J}_{fixed_i}$ are the wheel Jacobian matrices, which transform the chassis velocity into wheel axis velocity.

$$\mathbf{R}_{w_i}^{c}\mathbf{v}_{w_i} = \mathbf{J}_{free_i}\begin{pmatrix}\mathbf{v}_c \\ \boldsymbol{\omega}_c\end{pmatrix} + \mathbf{J}_{fixed_i}\begin{pmatrix}\dot{\theta}_i \\ \dot{\phi}_i\end{pmatrix} \tag{12}$$

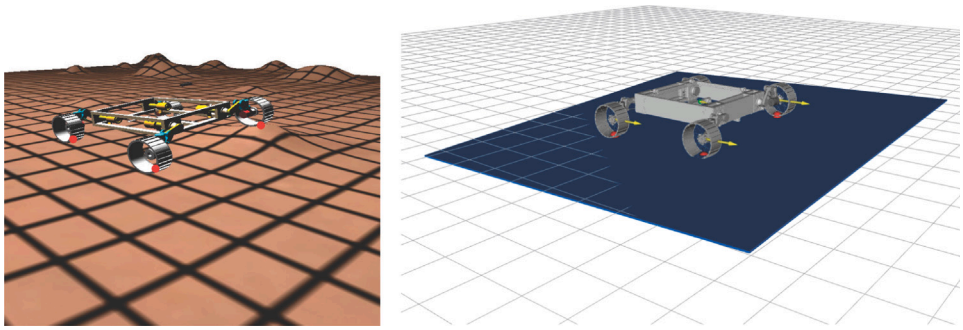


Fig. 6. The rover overcoming an obstacle in simulations. In the representation on the right-side, the grid represents the ideal horizontal plane, while the estimated local terrain plane is highlighted in blue. The red dots indicate the estimated wheel-terrain contact points, and the yellow arrows the wheel center velocities.

By repeating the same procedure for every kinematic chain and stacking the obtained equations row-wise, we obtain the Robot Composite Equation (RCE) (13), which represents our inverse velocity kinematics model. The system is overdetermined, and a best-fit solution for the unknowns $\mathbf{v}_c$ and $\boldsymbol{\omega}_c$ can be found in a least-squares fashion.

$$\mathbf{J}_{free}\begin{pmatrix}\mathbf{v}_c\\ \boldsymbol{\omega}_c\end{pmatrix} = \mathbf{R}_w^c\mathbf{v}_w - \mathbf{J}_{fixed}\begin{pmatrix}\dot{\theta}\\ \dot{\boldsymbol{\phi}}\end{pmatrix} \tag{13}$$

It is worth noting that terms in System (13) depend on joint variables measured from the rover motor encoders, which are subject to uncertainty. This is taken into account by scaling each joint variable inside the RCE to regulate its influence on the least squares solution. We introduce the diagonal covariance matrix $\mathbf{Cov}_{joint}$, whose elements represent the variances of the joint variables. In our kinematic chain, the measured joint variables are, respectively, the suspension angular position and velocity $(\phi, \dot{\phi})$, the steer angular position and velocity $(\theta, \dot{\theta})$, and the wheel angular velocity ω. Since all the kinematic chains in our system are equal, we consider the same covariance matrix for all of them.

$$\mathbf{Cov}_{joint} = \mathrm{diag}\big(\mathrm{Var}[\phi],\ \mathrm{Var}[\dot{\phi}],\ \mathrm{Var}[\theta],\ \mathrm{Var}[\dot{\theta}],\ \mathrm{Var}[\omega]\big) \tag{14}$$

Angular position variances are obtained by Eq. (15), where the angular position uncertainty u_q depends on the encoder pulses per revolution PPR and the motor gear ratio τ. Angular velocity variances are derived from position variances by considering the worst case where two successive motor positions are sampled by the encoder at the opposite side of the position uncertainty, as highlighted in Eq. (16), where $\Delta t_{sampling}$ represents the encoder sampling period. Position and velocity uncertainties are not independent of each other, so off-diagonal entries in $\mathbf{Cov}_{joint}$ should be considered. However, for simplicity and to reduce the computational time of matrix multiplication, we only account for diagonal elements.

$$Var[q] = u_q^2 = \left(\frac{2\pi}{\tau PPR}\right)^2 \tag{15}$$

$$Var[\dot{q}] = u_{\dot{q}}^2 = \left(\frac{2u_q}{\Delta t_{sampling}}\right)^2 \tag{16}$$

The relationship between the joint variables and the velocities stacked on the right side of the RCE is encoded by the Jacobian matrix $\mathbf{J}_{joint}$. This is used to transform $\mathbf{Cov}_{joint}$ to obtain the weight matrix:

$$\mathbf{W}_{joint} = (\mathbf{J}_{joint}\mathbf{Cov}_{joint}\mathbf{J}_{joint}^{\top})^{-1} \tag{17}$$

And to finally define the weighted least squares problem:

$$\begin{aligned}\mathbf{AJ}_{free}\begin{pmatrix}\mathbf{v}_c\\ \boldsymbol{\omega}_c\end{pmatrix} &= \mathbf{A}(\mathbf{R}_w^c\mathbf{v}_w - \mathbf{J}_{fixed}\begin{pmatrix}\dot{\theta}\\ \dot{\boldsymbol{\phi}}\end{pmatrix})\\ \mathbf{A} &= \mathbf{J}_{free}^{\top}\mathbf{W}_{joint}\end{aligned} \tag{18}$$

With best-fit solution:

$$\mathbf{v}_{opt} = \begin{pmatrix}\mathbf{v}_{c_{opt}}\\ \boldsymbol{\omega}_{c_{opt}}\end{pmatrix} = pinv[\mathbf{AJ}_{free}]\mathbf{A}(\mathbf{R}_w^c\mathbf{v}_w - \mathbf{J}_{fixed}\begin{pmatrix}\dot{\theta}\\ \dot{\boldsymbol{\phi}}\end{pmatrix}) \tag{19}$$

Where the symbol *pinv* indicates the Moore–Penrose pseudoinverse of the matrix $\mathbf{AJ}_{free}$.

Existing proprioceptive-based velocity kinematics methods reported in the related work provide only point estimates of the chassis velocity, without an associated covariance matrix. Here, instead of assigning heuristic or manually tuned values, we compute the velocity covariance directly from the optimization outcome. Specifically, the covariance is derived from the residual statistics of the constrained optimization problem, while explicitly accounting for encoder resolution through appropriate weighting. This yields a data-driven uncertainty estimate that reflects both the quality of the kinematic fit and the underlying sensing precision. Since weighted least squares is used for estimation, the covariance of the solution can be computed as:

$$\mathbf{Cov}_{\mathbf{v}_{opt}} = \sigma_{err}^2(\mathbf{AJ}_{free})^{-1} \approx \frac{S}{12}(\mathbf{AJ}_{free})^{-1} \tag{20}$$

Where ${\sigma_{err}}^2$ is the true error variance, which is estimated with the reduced chi-squared index, by dividing the sum of squared residuals for the optimal solution S by the statistical degrees of freedom of the system.

4. Rover state estimation

Reliable state estimation is crucial for enabling autonomous behavior of rovers in lunar environments, where remote control is hindered by communication delays and bandwidth limitations. Autonomy enables the rover to make decisions locally, for navigation, obstacle avoidance, and scientific task execution, all of which depend on an accurate, real-time estimate of the system state [34]. In our case, the need for robust state estimation is further motivated by the multipurpose nature of the rover platform: to fully exploit this versatility, both payloads and GNC subsystem must be constantly updated about the rover state. In this work, the rover state at time i is defined as:

$$x_i = \left\{ {}^{I}\mathbf{P}_{c_i},\, {}^{c}(\mathbf{v}_{c_i}, \boldsymbol{\omega}_{c_i}) \right\} \tag{21}$$

Where ${}^{I}\mathbf{P}_{c_i} = {}^{I}(\mathbf{t}_{c_i}, \mathcal{Q}_{c_i})$ is the robot chassis pose with respect to a local fixed reference frame I, composed of a position vector ${}^{I}\mathbf{t}_{c_i}$ and an orientation quaternion ${}^{I}\mathcal{Q}_{c_i}$, while ${}^{c}(\mathbf{v}_{c_i}, \boldsymbol{\omega}_{c_i})$ are respectively the chassis linear and angular velocities, in the chassis reference frame. Since the robot state definition is fixed, from now on, all the frame-related subscripts and apexes will be omitted.

4.1. Factor graphs for robot state estimation

In robotics, the state of an agent is estimated by sampling the surrounding environment using the robot's on-board sensor suite. Typically, each sensor provides partial and possibly conflicting information about the robot's state. As a result, determining a unique and exact solution for the true state is infeasible. For this reason, in a sensor fusion approach, the optimal robot states are estimated as the MAP of

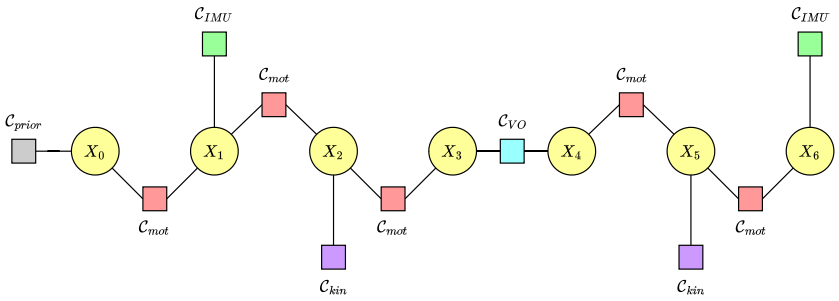


Fig. 7. The factor graph that models the rover state estimation, generated from sensor and motion factors.

the joint probability function $p(\mathbf{x}_k|\mathbf{z}_k)$, where $\mathbf{x}_k$ and $\mathbf{z}_k$ are respectively the sets of k robot states and sensor measurements at different time points, with the typical assumption for sensor measures to be random variables corrupted by zero-mean, Gaussian noise:

$$\mathbf{x}_{k_{opt}} = MAP\left[p(\mathbf{x}_k|\mathbf{z}_k)\right] = \arg\max_{\mathbf{x}_k} p(\mathbf{x}_k|\mathbf{z}_k) \tag{22}$$

The joint probability function can be factorized as:

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{z}_k) &\propto p(x_0)\prod_{t=1}^{T} p(x_t|u_t, x_{t-1})p(z_t|^j x_t) \\ {}^j x_t &\in \{x_t,\ (x_t, x_{t-1})\} \end{aligned} \tag{23}$$

That is, $p(\mathbf{x}_k|\mathbf{z}_k)$ can be written as a product of a prior probability $p(x_0)$ with a series of motion factors $p(x_t|u_t, x_{t-1})$ and sensor factors $p(z_t|^j x_t)$. This factorization makes factor graphs [35] a natural way of modeling sensor fusion problems. Factor graphs are bipartite graphs, meaning they contain two classes of variables: the robot states that need to be estimated and the factors connecting them, as shown in Eq. (23). In this way, multiple, asynchronous sensors where measurements are taken at different times and with different frequencies can be easily treated: a new measure is included in the estimation problem by simply adding a new factor to the graph that connects the involved robot states. Every factor encodes a non-linear error function f_i between the predicted and the actual robot state. Then, finding the optimal robot states by performing MAP of $p(\mathbf{x}_k|\mathbf{z}_k)$ turns into solving the non-linear least squares problem [13]:

$$\begin{aligned} \mathbf{x}_{k_{opt}} &= \arg\min_{\mathbf{x}_i} \sum_{i=0}^{k} C_i \\ C_i &= \|f_i\|^2_{\mathbf{Cov}_i} \end{aligned} \tag{24}$$

The factor cost function C_i is the squared Mahalanobis distance of the predicted state from its probability distribution, and $\mathbf{Cov}_i$ is the covariance matrix associated with the sensor measure or with the motion process model. An example of a factor graph for the application considered in this paper is shown in Fig. 7.

4.2. ROS 2 integration

Robot state estimation using factor graphs is implemented in the ROS 2 ecosystem by the *Fuse* package. The package follows a modular design: its components are developed as ROS 2 plugins, i.e., C++ classes that can be dynamically loaded at runtime. The framework offers sensor factors, motion factors, graph optimizers, and state publishers as plugins. In this way, the application can be customized by simply listing desired plugins inside a configuration file. For modeling and solving the graph, *Fuse* leverages the Google Ceres Solver library.

Fig. 8 illustrates the package architecture. Sensor model plugins are the entry points for observations from the ROS 2 environment: a subscriber listens for incoming messages and triggers the `process()` callback. Inside the function, the model generates the state variables and the constraints deriving from the sensor measurement. These are appended to a *Transaction* object, together with the message timestamp, and sent to the optimizer. *Transactions* waiting to be processed are queued, and a motion model plugin generates constraints between unconnected timestamps before optimization by calling the `predict()` method. Optimization occurs at fixed rate and, upon completion, the graph variables are updated with the optimized values. Publishers can query the graph and provide fresh robot states to other components. A complete overview of the *Fuse* framework and its design choices can be found in [36].

Fuse has been primarily employed for 2D navigation scenarios, with available classes tailored to the planar motion of mobile robots. In this work, we build upon this architecture to extend its capabilities to 3D state estimation, introducing new plugins for three-dimensional sensor and motion factors, as well as for rover odometry publishers. In addition to these extensions, we move beyond *Fuse* exclusive reliance on Google Ceres Solver automatic differentiation by providing closed-form analytical Jacobians for all implemented factors, thereby improving the computational efficiency of the optimization. The proposed implementation does not introduce new theoretical models, but delivers a complete and consistent 3D estimation pipeline that enables the 3D multi-sensor fusion under a factor-graph formulation. An open-source implementation is publicly available as a pull request to the Fuse repository.[1] The next subsections provide a detailed explanation of the implemented ROS 2 plugins.

4.3. Sensor factors

Sensor factors integrate the constraints imposed by the rover sensor measurements into the factor graph. Our framework includes both unary sensor factors $p(z_i|^j x_t)$ with ${}^j x_t = \{x_t\}$, which involve only the robot state at measurement time, and binary sensor factors $p(z_i|^j x_t)$ with ${}^j x_t = \{x_t, x_{t-1}\}$ which relate two consecutive robot states. Each employed sensor provides information about a subset of the full system state; accordingly, its corresponding sensor factor connects only the related state variable components in the graph. Three different sensor factors are considered in this work:

- *Visual odometry factors*: encode the change in rover pose between two measurement time points, as computed by a visual odometry pipeline. Developed as a `DeltaPose3D` plugin;
- *IMU factors*: encode the direct measurements of chassis angular velocity and linear acceleration, as provided by the on-board IMU. Developed as an `Imu3D` plugin;
- *Velocity kinematic factors*: encode the motion constraints imposed by the rover kinematic model. Developed as a `Twist3D` plugin.

The sensor models have been developed as `DeltaPose3D`, `Imu3D` and `Twist3D` plugins, derived from *Fuse* `SensorModel` base class.

[1] https://github.com/locusrobotics/fuse/pull/354

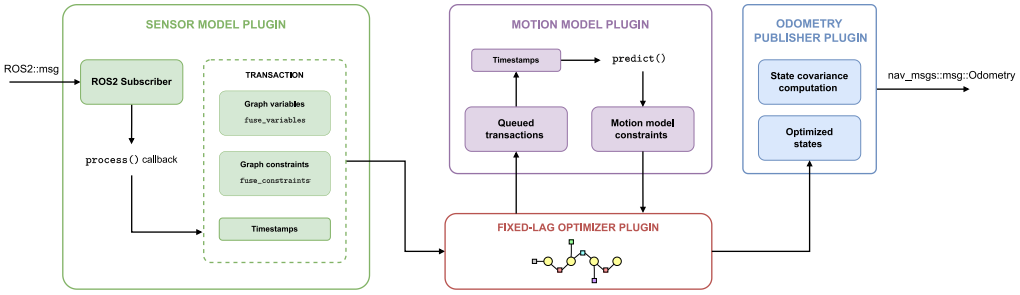


Fig. 8. Base architecture of the ROS 2 package *Fuse*.

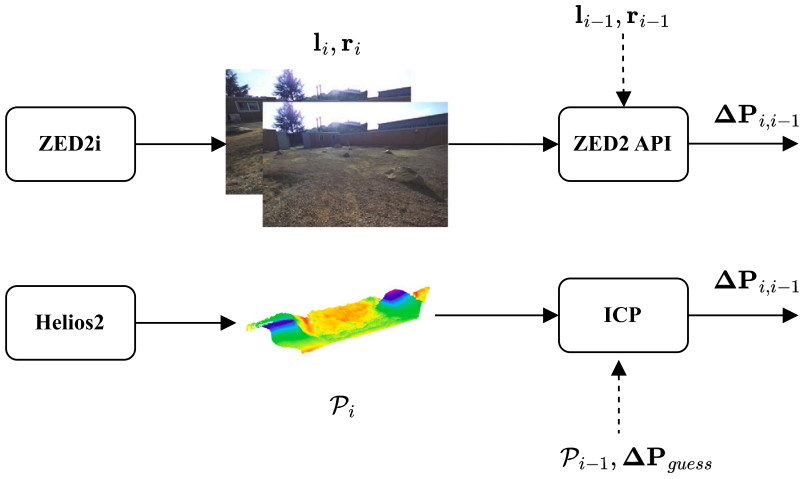


Fig. 9. The two visual odometry pipelines employed on the rover: matching of RGB stereo pairs is carried out during daylight (top section), while registration of time-of-flight's pointclouds is performed at night (bottom section).

4.3.1. Visual odometry factor

Two visual odometry pipelines are available in our system [37]. For daylight navigation, stereo matching is performed between two consecutive RGB image pairs (l_{i-1}, r_{i-1}) and (l_i, r_i) collected by a StereoLabs ZED2i stereo camera. The change in chassis pose $\Delta\mathbf{P}_{i,i-1}$, and its associated covariance matrix $\mathbf{Cov}_{VO}$ are estimated from the matching result using the proprietary ZED API. At night or in low-light environments, two LUCID Vision Labs Helios2 time-of-flight cameras provide point clouds $\mathcal{P}_i$ and $\mathcal{P}_{i-1}$ that are incrementally registered with an ICP approach [38]. To aid ICP convergence, a guess of the desired transformation $\Delta\mathbf{P}_{guess}$ can be provided. $\Delta\mathbf{P}_{i,i-1}$ and its covariance matrix are estimated from the cloud registration results.

The visual odometry factor introduces a relative constraint between the pose components of two robot states $\mathbf{P}_i$ and $\mathbf{P}_{i-1}$, at the time points when the measurements were collected. The associated cost function is composed of a position and an orientation residual:

$$C_{VO_i} = \left\| \begin{matrix} \mathbf{r}_{pos} \\ \mathbf{r}_{orient} \end{matrix} \right\|^2_{\mathbf{Cov}_{VO}} = \left\| \begin{matrix} (\mathbf{t}_i - \mathbf{t}_{i-1}) - \Delta\mathbf{t}_{VO} \\ \Delta\mathcal{Q}_{VO}(\mathcal{Q}_{i-1}^{-1}\mathcal{Q}_i) \end{matrix} \right\|^2_{\mathbf{Cov}_{VO}} \tag{25}$$

4.3.2. IMU factor

IMU measurements contribute to the factor graph with a unary factor corresponding to an absolute constraint on the instantaneous measured angular velocity and linear acceleration. Associated covariances are provided by the IMU driver. IMU measurements and covariances are rotated into the chassis frame before being integrated into the graph. The associated cost function is composed of an angular velocity residual and a linear acceleration residual:

$$C_{IMU_i} = \left\| \begin{matrix} \mathbf{r}_\omega \\ \mathbf{r}_a \end{matrix} \right\|^2_{\mathbf{Cov}_{IMU}} = \left\| \begin{matrix} \boldsymbol{\omega}_i - \boldsymbol{\omega}_{IMU} \\ \mathbf{a}_i - \mathbf{a}_{IMU} \end{matrix} \right\|^2_{\mathbf{Cov}_{IMU}} \tag{26}$$

4.3.3. Velocity kinematic factor

Velocity kinematic factors are employed in the optimization problem to penalize rover motions that are not physically achievable by the platform. In practice, this is obtained by treating the pipeline from Section 3 as a virtual velocity sensor: every $\mathbf{v}_{opt}$ sample received from the velocity kinematics is considered as an absolute reading of the chassis velocity at a specific time point. Virtual velocity measurements contribute to the factor graph with a unary factor corresponding to an absolute constraint on the instantaneous linear and angular rover velocities:

$$C_{kin_i} = \left\| \begin{matrix} \mathbf{r}_v \\ \mathbf{r}_\omega \end{matrix} \right\|^2_{\mathbf{Cov}_{kin}} = \left\| \begin{matrix} \mathbf{v}_i - \mathbf{v}_{opt_i} \\ \boldsymbol{\omega}_i - \boldsymbol{\omega}_{opt_i} \end{matrix} \right\|^2_{\mathbf{Cov}_{\mathbf{v}_{opt}}} \tag{27}$$

4.4. Rover motion factor

Due to the lack of synchronization among incoming measurements, and since every employed sensor produces data at different rates, the graph built from sensor factors is not fully connected. Missing constraints between unconnected state variables are generated using a kinematic motion model, which predicts the robot state x_i based on the previous state x_{i-1} and a motion command u_i. The state evolution is governed by the prediction function $\psi(u_i, x_{i-1})$ that encodes the kinematic equations of motion for the rover. Since constraints from robot kinematics are already incorporated into the graph via the virtual velocity sensor, we maintain generality by propagating the states using an omnidirectional model. The motion command is composed of the chassis angular velocity and linear acceleration at the initial state,

and is maintained constant during prediction. The motion process covariance matrix is filled with manually tuned parameters. Binary rover motion factors are represented by the cost function:

$$C_{mot_i} = \left\| \mathbf{r}_{mot} \right\|^2_{\mathbf{Cov}_{mot}} = \left\| x_i - \psi(u_i, x_{i-1}) \right\|^2_{\mathbf{Cov}_{mot}} \tag{28}$$

The motion model factor has been developed as a `Omnidirectional3D` plugin, derived from *Fuse* `MotionModel` base class.

4.5. Prior factor

The prior factor is the first constraint that is inserted into the graph, and consists of a unary factor representing the prior probability distribution of the initial rover state x_0:

$$C_{prior} = \left\| \mathbf{r}_{prior} \right\|^2_{\mathbf{Cov}_{init}} = \left\| x_0 - x_{init} \right\|^2_{\mathbf{Cov}_{init}} \tag{29}$$

Where x_{init} is the prior distribution mean and $\mathbf{Cov}_{init}$ its covariance matrix. Since we are not performing absolute localization, we consider the rover state to evolve in time starting from a local inertial reference frame. Thus, the only elements of x_{init} that are not zero-initialized are:

- The z component of the rover position vector, which equals the vertical component of the centroid of the contact points with the local terrain plane $\mathbf{c}_w$, computed from Eq. (5);
- The roll and pitch components of the rover orientation, which are initialized from the initial IMU orientation reading.

This enables the state estimation process to begin with a reasonable initial estimate of the chassis attitude and ground clearance. Again, the covariance matrix is manually tuned for our application.

4.6. Graph inference

To enable high-frequency state estimation while keeping computational cost bounded, this work leverages the fixed-lag smoother optimizer provided by the *Fuse* package. The optimization is performed over a sliding window of recent states, whose length is selected to balance estimation accuracy and real-time performance. The nonlinear least squares problem is solved using the Levenberg–Marquardt algorithm, and after each optimization cycle, variables that fall outside the lag window are marginalized. This approach preserves their informational contribution while maintaining a compact graph structure.

4.7. Rover odometry publisher

The odometry publisher plugin is responsible for providing the estimated rover state to the GNC and payload systems at the desired frequency. The publisher queries the variables of the last optimized graph available and fills a standard ROS 2 message of type *nav_msgs/Odometry*. Again, Google Ceres Solver is employed for estimating optimized chassis pose and velocity covariances. In particular, we set sparse QR factorization as the algorithm for decomposing the residual Jacobian matrix at the optimization point, to compute the covariance of the solution.

5. Experimental results

This section presents the results obtained from the experiments performed to validate the proposed state estimation framework. First, tests have been conducted in a simulated environment to validate the framework's components and ensure their correct integration. Then, the state estimation pipeline was deployed on the real rover platform previously described, validating its accuracy and performance in a real-world scenario.

Field tests have been conducted at the Rover eXploration facilitY (RoXY) analogue terrain, located at TAS-I site in Turin, Italy. Initially conceived as a Martian yard, RoXY presents heterogeneous terrain characteristics, spanning from sandy dunes to gravel surfaces, with obstacles such as rocks and craters, and is conceived to challenge rover locomotion and navigation capabilities. A view of the facility is shown in Fig. 10(a). To replicate real-world test conditions, a 3D reconstruction of RoXY is loaded as the simulation environment. The model is illustrated in Fig. 10(b).

In the next subsections, results from the experimental campaigns are analyzed, and our approach is compared with two state-of-the-art state estimation methods in ROS 2.

5.1. Simulations in project chrono

Simulations are built on the multiphysics dynamics engine Project Chrono [39]. The engine choice has been guided by the desire to replicate system dynamics with high fidelity, while placing less emphasis on visual realism. Chrono features allow us to model our platform dynamics with a high level of completeness. In particular, we can simulate:

- The kinematic relations of the four wheel-steer-suspension chains as described in Section 3.2;
- The dynamic behavior of the SEA in suspension joints, with focus on the elastic response of the spring;
- The wheel–terrain contacts and compute the associated friction forces.

Project Chrono ships with plug-and-play sensor modules that can be integrated into the rover model. For our tests, we employed a simulated IMU sensor that outputs the chassis angular rates and linear accelerations with tunable levels of zero-mean random Gaussian noise on every axis, and a generic depth sensor, from which point clouds are generated, is also included in the simulations, to replicate the visual odometry pipeline described in Fig. 9. Ground truth values for chassis pose and velocity are provided by the simulator.

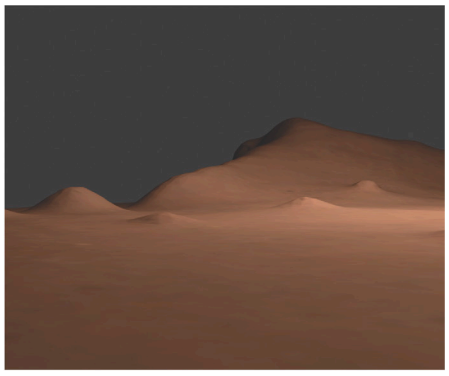
5.1.1. Velocity kinematics tests

Three test cases have been designed to test the estimation of chassis velocity and covariance using the method described in Section 3, to stimulate all 6 degrees of freedom of the chassis. In the first two, the rover overcomes a positive obstacle, such as a rock or a dune, with its left wheels. Obstacle heights are respectively ∼30% (Fig. 11(a)) and ∼60% (Fig. 11(b)) of wheel diameter. In the last one, the rover traverses the side of a negative obstacle, such as a crater, with its right wheels (Fig. 11(c)). Fig. 12 presents the time evolution of chassis velocity components in comparison with the ground truth values. For clarity, the rolling averages have been highlighted. Diagonal entries of the covariance matrix have also been reported.

Similar patterns emerge in every scenario. First, differences in trends are evident for v_x and v_y. This is expected, and mainly due to the fact that the model does not take into account wheel slippage. In fact, higher differences are reported in scenarios *b* and *c*, where the rover surpasses steeper obstacles and the slippage level is more influential. For the same reason, the model underestimates ω_z. Correct predictions are made for ω_x and ω_y, showing that the model is able to capture the variations in platform attitude. Variance trends follow what was just discussed; higher values are outputted in correspondence with higher velocity drifts: this again is expected, due to the proportionality relation of the covariance with the squared sum of residuals, as highlighted in Eq. (20).

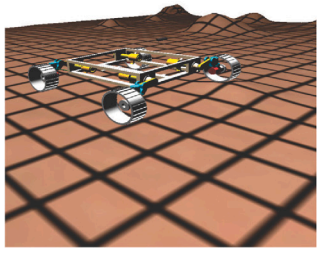
A quantitative evaluation is reported in Table 2, where Root Mean Square Error (RMSE) values of every chassis velocity component with respect to the ground truth are listed. Intuitively, higher errors emerge in cases *b* and *c* where the estimated local plane worse approximate the terrain surface, leading to higher residuals in least-squares optimization.

(a) RoXY at TAS-I Turin site, Italy.

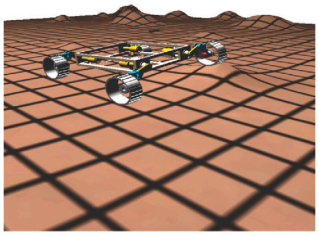


(b) RoXY 3D reconstruction.

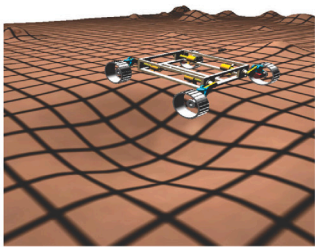
Fig. 10. The TAS-I Rover eXploration facilitY analogue terrain and its 3D reconstruction employed in the experiments.



(a) Small positive obstacle.



(b) Big positive obstacle.



(c) Negative obstacle.

Fig. 11. The three simulated test cases for chassis velocity and covariance estimation.

Table 2
RMSE of optimal chassis velocities in simulations.

Velocity component	Case a	Case b	Case c
v_x [m/s]	0.014	0.020	0.018
v_y [m/s]	0.013	0.022	0.013
v_z [m/s]	0.020	0.027	0.024
ω_x [rad/s]	0.018	0.032	0.021
ω_y [rad/s]	0.022	0.042	0.033
ω_z [rad/s]	0.006	0.011	0.007

5.1.2. State estimation tests

The state estimation pipeline has been tested by teleoperating the rover in the simulated environment. Two cases have been considered: the first path includes a plane section, a downhill section, and the crossing of a crater, while the second involves driving the rover up to a steep hill. In both cases, the rover switches between the available locomotion modalities.

For comparison we selected two state-of-the-art methods according to the following criteria: (i) the ability to fuse IMU, pose, and velocity measurements within a unified estimation pipeline, (ii) the estimation of 3D system state and (iii) ROS 2 support. The first method is *robot_localization* [40], which employs filter-based approaches and is the de-facto standard employed by the ROS community; the EKF was selected as the baseline for comparison. The second is WOLF [41] which, as our approach, performs multi sensor fusion with factor graph. To evaluate the frameworks under comparable test conditions, the same sensor inputs are provided to the three pipelines and the same motion model employed in our approach is used during the prediction step of the filter. Since WOLF does not natively support direct fusion of velocities, inputs from Section 3 have been pre-integrated with a constant velocity model and provided as pose measurements.

To quantitatively compare the predicted trajectories, we followed the approach proposed in [42]. First, the trajectories have been time-synchronized with the ground truth, allowing for the analysis of states at the same point in time. Then, the RMSE over the complete trajectory is computed for every component of the rover state. The orientation component of the pose is converted from quaternion to Euler angles representation before computing the RMSE. Results are reported in Table 3. We also indicate the trajectory length, which is approximated as the sum of 3D Euclidean distances of consecutive ground truth positions:

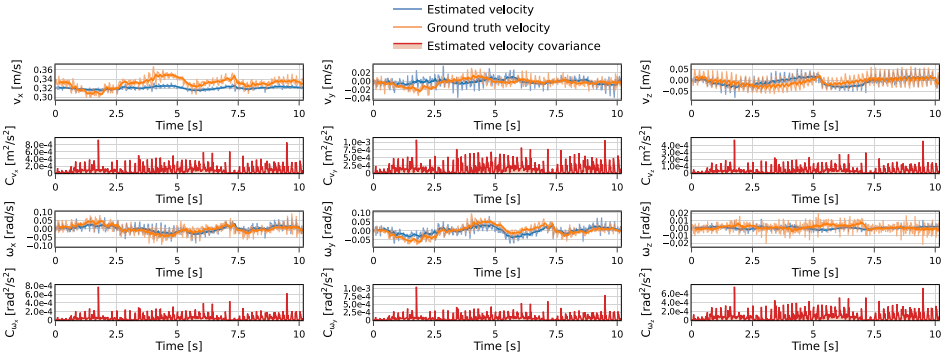
$$l = \sum_{i=0}^{t} \left\| \mathbf{t}_{i+1} - \mathbf{t}_i \right\| \tag{30}$$

Relative Trajectory Error (RE) statistics for the rover pose are provided by computing the RMSE over multiple sets of consecutive poses along the trajectories. By choosing a small set length, we can measure the local consistency of the two methods. RE statistics are reported in Fig. 15 as boxplots: the boxes are delimited by the upper and lower quartiles, while the whiskers indicate the maximum and minimum error value. The red line indicates the median value of the relative error.

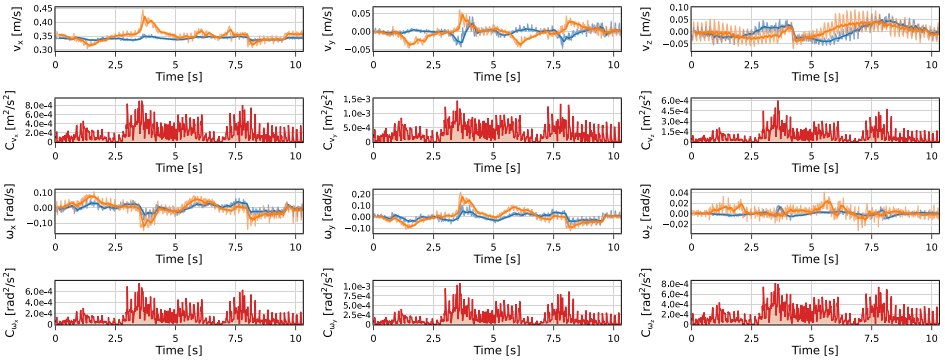
Figs. 13 and 14 report the time evolution of the robot state components as predicted by the three methods, compared with the ground truth values provided by the simulator, for the two simulated paths, respectively.

Results from simulations show that the 2D motion state components are largely consistent between the three approaches. Smaller RMSE and RE values are achieved by our method for the rover pose z component: in the first trajectory, it is able to detect that the rover is exiting the crater at $t = 25$ s, whereas the EKF continues to diverge in the negative z direction and WOLF fails to capture the variation of the pose vertical component. In the second test, our method exhibits less drift during uphill and downhill motion. High variations of chassis roll and pitch in the simulated trajectories further underline the improved robustness of our approach, showing markedly reduced drift compared to the other methods. Results consistency along the experiments is confirmed by relative error boxplots.

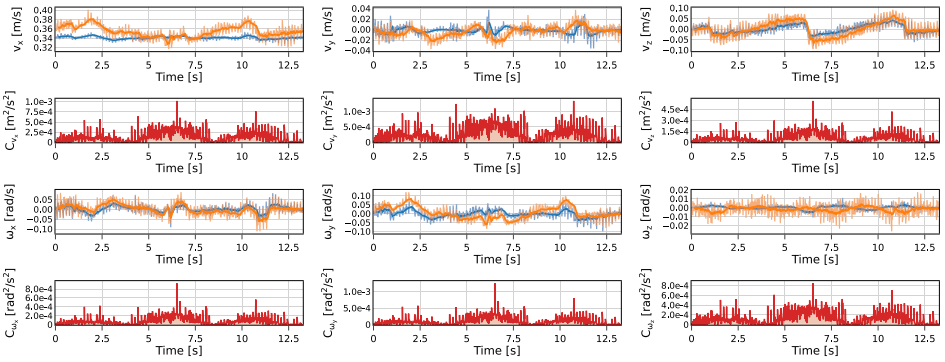
Both our approach and the EKF show minimal prediction errors for linear and angular velocity, while WOLF predictions for linear



(a) Case *a*: small positive obstacle.



(b) Case *b*: big positive obstacle.



(c) Case *c*: negative obstacle.

Fig. 12. Estimated chassis velocity and variances from velocity kinematics, in the three simulated scenarios.

velocities diverge. This is expected and mainly due to the different architecture of the latter: in WOLF, sensor measurements are converted to factors and associated to robot states inside the graph by creating keyframes, which are collections of state constraints. Multiple factors can be associated to a single keyframe: measurements from visual odometry and inverse velocity kinematics are inserted as constraints over the rover pose (position and orientation), while IMU measurements are pre-integrated and include factors on the linear velocity of the robot. This means that, in our setup, the latter is only constrained by simulated IMU measurements, which present high noise and biases, leading to diverging predicted linear velocity.

No angular velocities are reported for WOLF, since they are not provided by the framework at the time of writing.

5.2. Field tests

For field tests, ground truth rover pose is provided by an Xsens MTi-670 GNSS/INS sensor mounted on-board. The sensor utilizes GNSS-RTK and an integrated magnetometer to provide the rover's absolute position and heading angle. Moreover, roll and pitch angles are computed internally by estimating the direction of the gravity vector using sensor accelerometers and then corrected with GNSS data to compensate for the influence of external accelerations. This setup is suitable for our application because the system motion is relatively slow, and no high accelerations are present for an extended period. The ground truth pose is provided in the ENU reference system and is aligned with the local inertial system by measuring the initial heading of the rover. The ground truth value for rover velocity is not available; therefore, its prediction

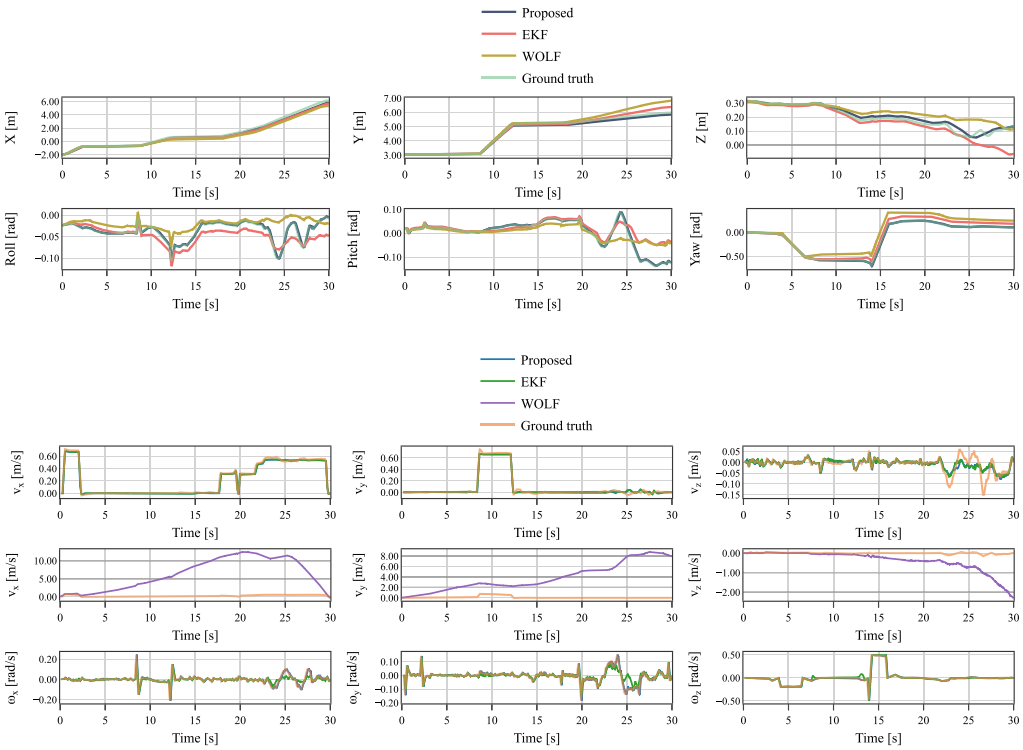


Fig. 13. Simulated trajectory 1: predicted pose and velocity.

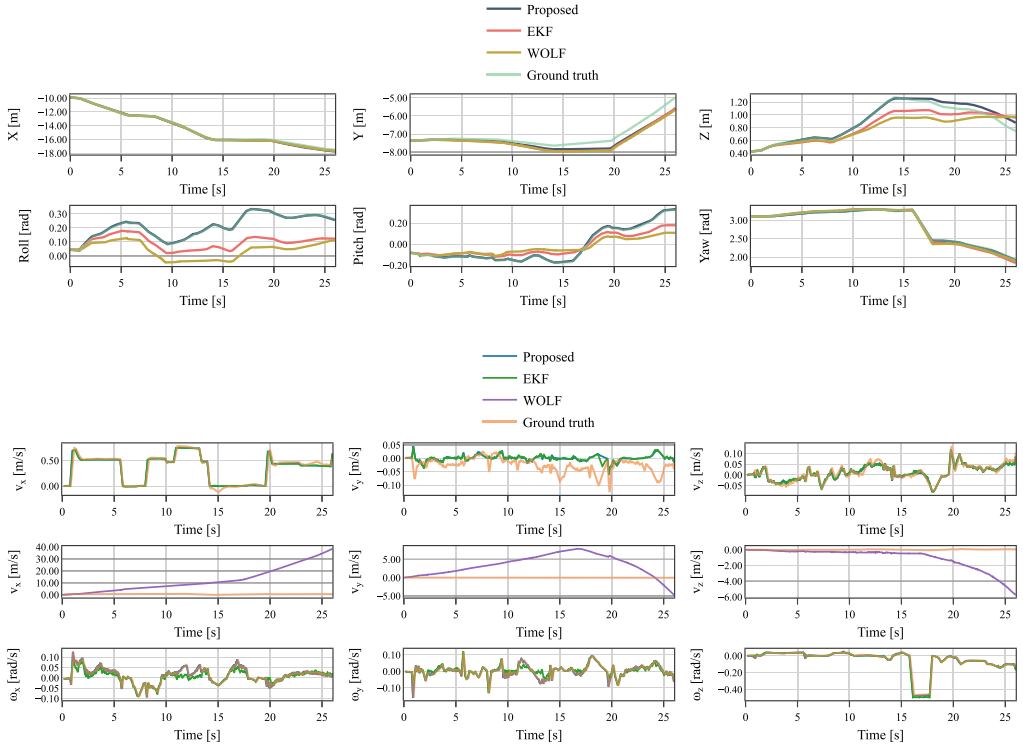
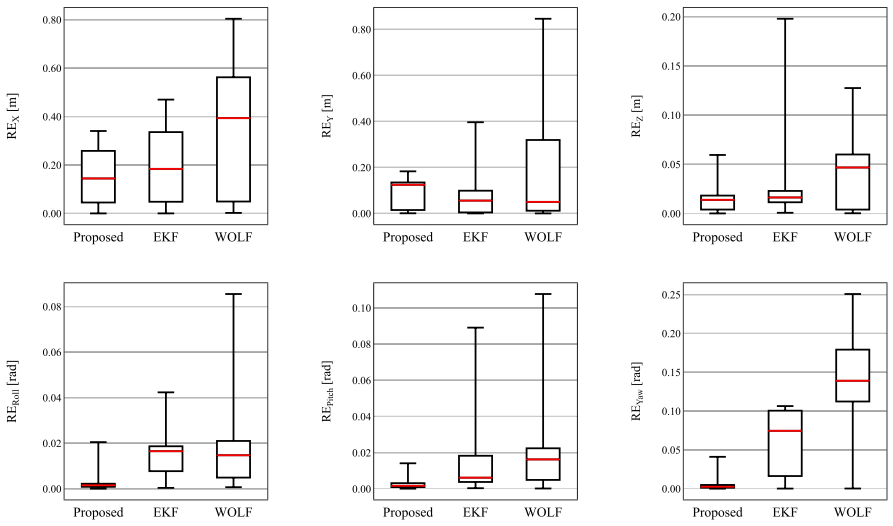
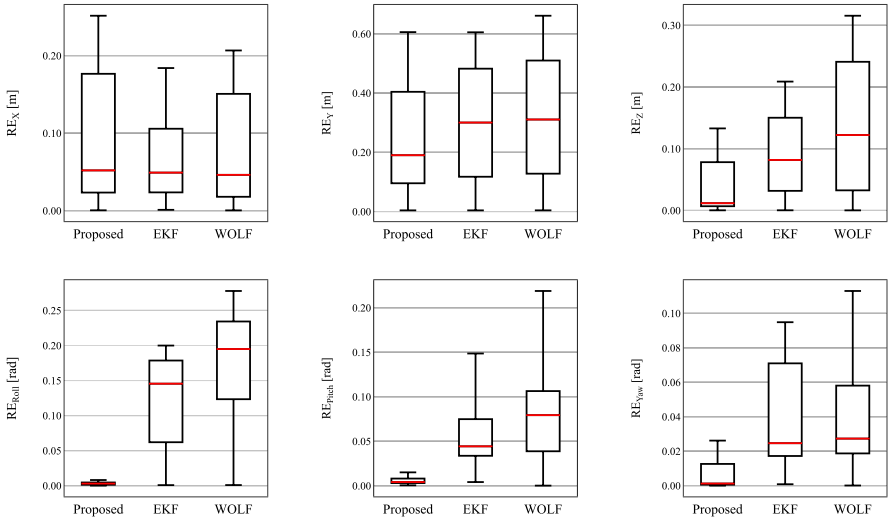


Fig. 14. Simulated trajectory 2: predicted pose and velocity.



(a) Simulated trajectory 1.



(b) Simulated trajectory 2.

Fig. 15. Statistics of predicted rover pose components RE in simulations. The boxes are delimited by the upper and lower quartiles. The whiskers indicate the maximum and minimum error value. The red line shows the median value for the relative error.

Table 3
RMSE of predicted rover state components. Lower error values are reported in bold. WOLF linear velocity predictions diverge in simulations. WOLF does not provide angular velocity estimates.

State component	Trajectory 1			Trajectory 2		
	Proposed	EKF	WOLF	Proposed	EKF	WOLF
X [m]	**0.186**	0.245	0.426	0.113	**0.074**	0.099
Y [m]	**0.109**	0.146	0.322	**0.319**	0.370	0.365
Z [m]	**0.018**	0.053	0.054	**0.059**	0.118	0.166
Roll [rad]	**0.003**	0.017	0.025	**0.003**	0.132	0.186
Pitch [rad]	**0.004**	0.033	0.037	**0.005**	0.067	0.091
Yaw [rad]	**0.010**	0.069	0.144	**0.006**	0.048	0.046
v_x [m/s]	0.030	**0.029**	*div*	0.063	**0.055**	*div*
v_y [m/s]	0.043	**0.035**	*div*	0.041	0.041	*div*
v_z [m/s]	0.026	0.026	*div*	**0.018**	0.019	*div*
ω_x [rad/s]	**0.006**	0.027	–	**0.008**	0.026	–
ω_y [rad/s]	**0.005**	0.030	–	**0.006**	0.025	–
ω_z [rad/s]	**0.004**	0.023	–	**0.004**	0.018	–
Trajectory length [m]	9.11			10.16		

will not be considered in this case. The two state estimation pipelines run on an AMD Ryzen 5 3500U CPU, providing rover odometry data at 100 Hz.

Two trajectories have been considered. In both, the vehicle is roving the RoXY test field, starting on a plane surface and traversing a downhill–uphill section. The rover primarily navigates in double axis Ackermann mode, utilizing spot-turn maneuvers around the middle sections. The trajectories are longer than the simulated ones, but they present only a moderate variation in the rover's chassis attitude.

Again, we compare our approach with EKF and WOLF results, as shown in Figs. 16 and 17, where the predicted pose components are reported against the ground truth value. RMSEs are computed over the entire trajectory length and reported in Table 4. Trajectory RE statistics are shown in Fig. 18. We also provide the average value of ROS 2 nodes' CPU usage, expressed as the total percentage of CPU time used by the single process sampled during the test duration.

Field experiments confirmed the results obtained in simulation: all the methods exhibit comparable behavior for 2D motion, with minor differences in planar pose components, even for longer trajectories. The larger gap is in the vertical pose component, where the EKF suffers from pronounced drift largely inherited from the visual odometry input,

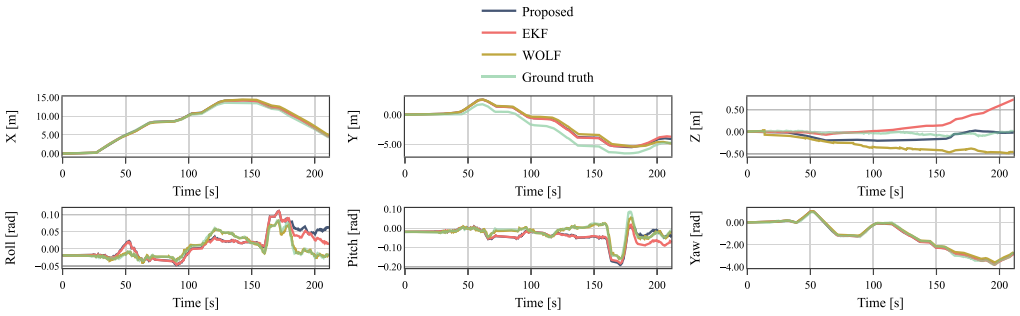


Fig. 16. Field test trajectory 1: predicted pose.

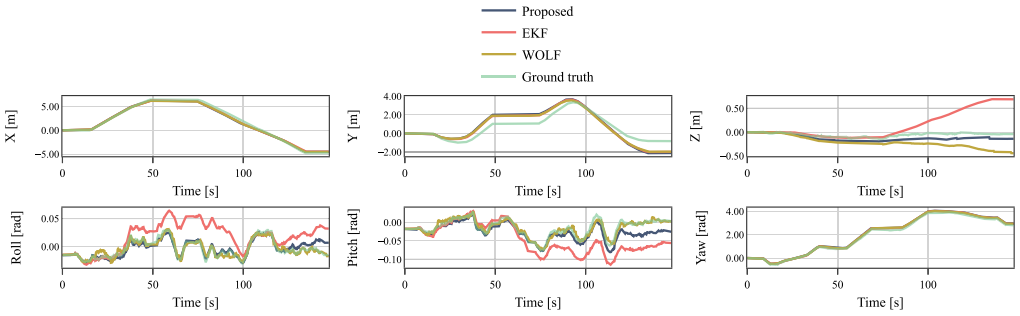


Fig. 17. Field test trajectory 2: predicted pose.

Table 4
RMSE of predicted rover pose components. Lower error values are reported in bold.

State component	Trajectory 1			Trajectory 2		
	Proposed	EKF	WOLF	Proposed	EKF	WOLF
X [m]	0.403	**0.313**	0.580	0.310	**0.279**	0.316
Y [m]	0.993	**0.986**	1.189	0.768	**0.680**	0.684
Z [m]	**0.115**	0.265	0.297	**0.079**	0.335	0.198
Roll [rad]	0.030	0.023	**0.004**	0.006	0.028	**0.006**
Pitch [rad]	0.031	0.033	**0.010**	**0.007**	0.041	0.015
Yaw [rad]	**0.072**	0.077	0.130	0.110	**0.094**	0.098
Average CPU usage [%]	55.7 ± 9.3	18.3 ± 3.1	41.4 ± 6.9	42.5 ± 15.2	14.9 ± 5.5	29.1 ± 9.9
Trajectory length [m]	30.06			22.73		

while our method effectively smooths and attenuates its influence. This effect is amplified by additional drift in pitch angle in the EKF, which further biases the vertical estimate. WOLF drift in z is less pronounced but still non-negligible, reaching a maximum of ∼0.45 m. Our approach outperforms other methods by limiting vertical drift. This is obtained by accurate graph constraints balancing: the employed scaled loss functions allow for decreasing the influence of visual odometry drift while maintaining the correct trend at the same time. The two graph-based methods outperform the EKF in roll and pitch angles estimation, with WOLF showing marginally better results compared to the proposed approach. However, the rover attitude variation along both trajectories is very limited.

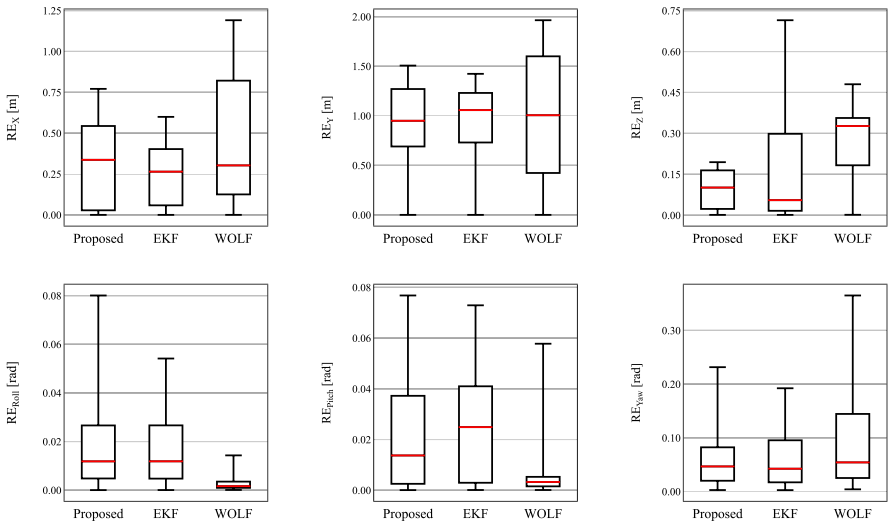
The influence of wheel slippage on RoXY sandy-gravelly terrain is evident for both methods, and primarily reflected by the drift accumulated along the Y axis.

6. Conclusions

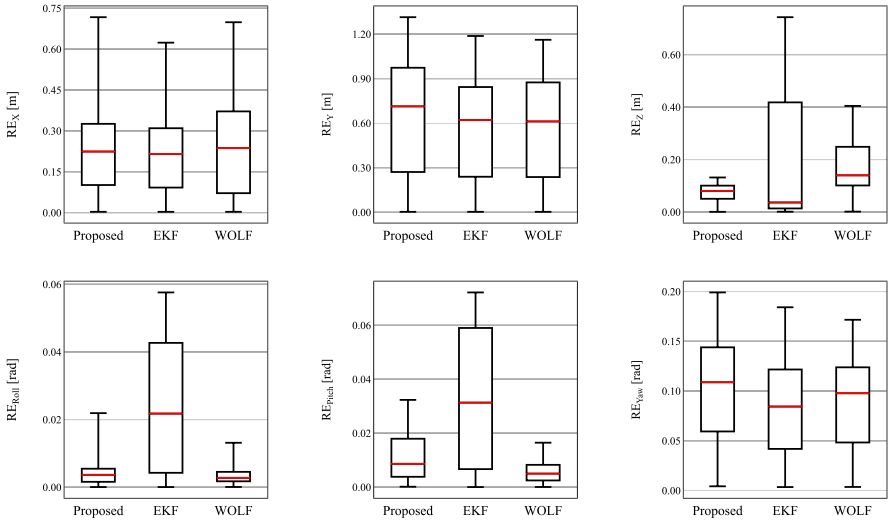
In this work, we presented a state estimation framework based on factor graphs deployed on a multipurpose lunar rover prototype. The approach extends the ROS 2 *Fuse* package by introducing dedicated 3D sensor and motion factors, as well as a novel module that exploits the rover kinematic model and wheel encoders measurements to estimate the instantaneous ground plane of the vehicle and its chassis velocities. These are used as additional graph constraints to strengthen the estimation process and enable a more accurate and robust reconstruction of the rover's motion.

Validation was conducted both in simulations and in a field test campaign conducted in a representative environment. Results consistently show that the proposed method achieves performances comparable to the state-of-the-art filter-based state estimation implementation in ROS 2 in terms of planar motion, while, at the same time, clear improvements are observed in the estimation of the out-of-plane pose components: qualitative and quantitative results are reported together with an analysis of the trajectory error statistics.

A relevant future improvement will be the integration of a wheel slip estimation methodology into the inverse velocity kinematic module. Currently, the chassis linear and angular velocities are estimated under the assumption of no wheel slippage: by introducing slip ratios along the longitudinal and lateral directions of each wheel, the module can be adapted to account for non-ideal ground contact conditions. These slip-aware velocity estimates will then be provided, together with their covariance, as refined constraints in the factor graph. Such an extension is expected to significantly reduce drift in challenging terrain, where wheel–soil interactions strongly influence the rover mobility performance.



(a) Field test trajectory 1.



(b) Field test trajectory 2.

Fig. 18. Statistics of predicted rover pose components RE in field tests. The boxes are delimited by the upper and lower quartiles. The whiskers indicate the maximum and minimum error value. The red line shows the median value for the relative error.

CRediT authorship contribution statement

Giacomo Franchini: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Patrick Roncagliolo:** Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Davide Grazi-ato:** Software, Investigation, Conceptualization. **Alessandro Ruggiero Chiminelli:** Software, Investigation, Conceptualization. **Andrea Merlo:** Supervision, Project administration. **Marcello Chiaberge:** Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is part of a Piano Nazionale di Ripresa e Resilienza (PNRR) PhD scholarship founded by the Italian Ministry of Education, University and Research – DM 352/2022.

References

[1] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, Robot Operating System 2: Design, architecture, and uses in the wild, Sci. Robot. 7 (66) (2022) http://dx.doi.org/10.1126/scirobotics.abm6074.

[2] S. Williams, Fuse: A graph-based sensor fusion framework, 2021, Vimeo, [Online]. Available: https://vimeo.com/649648867/c524ae57fd.

[3] S. Williams, Fuse, 2022, [Online]. Available: https://github.com/locusrobotics/fuse.

[4] S. Agarwal, K. Mierle, T.C.S. Team, Ceres Solver, 2023, URL https://github.com/ceres-solver/ceres-solver.

[5] M. Kam, X. Zhu, P. Kalata, Sensor fusion for mobile robot navigation, Proc. IEEE 85 (1) (1997) 108–119, http://dx.doi.org/10.1109/JPROC.1997.554212.

[6] J.M. Pak, C.K. Ahn, State estimation algorithms for localization: A survey, Int. J. Control. Autom. Syst. 21 (9) (2023) 2771–2781, http://dx.doi.org/10.1007/s12555-023-9902-z, Cited by: 10.

[7] R.E. Kalman, A new approach to linear filtering and prediction problems, J. Basic Eng. 82 (1) (1960) 35–45, http://dx.doi.org/10.1115/1.3662552.

[8] S. Julier, J. Uhlmann, Unscented filtering and nonlinear estimation, Proc. IEEE 92 (3) (2004) 401–422, http://dx.doi.org/10.1109/JPROC.2003.823141.

[9] X. Yu, S. Teng, T. Chakhachiro, W. Tong, T. Li, T.-Y. Lin, S. Koehler, M. Ahumada, J.M. Walls, M. Ghaffari, Fully proprioceptive slip-velocity-aware state estimation for mobile robots via invariant Kalman filtering and disturbance observer, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and

Systems, IROS, 2023, pp. 8096–8103, http://dx.doi.org/10.1109/IROS55552.2023.10342519.
[10] T.-Y. Lin, T. Li, W. Tong, M. Ghaffari, Proprioceptive invariant robot state estimation, 2024, arXiv:2311.04320.
[11] N. Gordon, D. Salmond, A. Smith, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, IEE Proc. F 140 (1993) 107–113, http://dx.doi.org/10.1049/ip-f-2.1993.0015.
[12] F. Dellaert, Factor graphs: Exploiting structure in robotics, Annu. Rev. Control. Robot. Auton. Syst. 4 (Volume 4, 2021) (2021) 141–166, http://dx.doi.org/10.1146/annurev-control-061520-010504.
[13] V. Indelman, S. Williams, M. Kaess, F. Dellaert, Information fusion in navigation systems via factor graph based incremental smoothing, Robot. Auton. Syst. 61 (8) (2013) 721–738, http://dx.doi.org/10.1016/j.robot.2013.05.001.
[14] X. Liu, G. Clark, J. Campbell, Y. Zhou, H.B. Amor, Enhancing state estimation in robots: A data-driven approach with differentiable ensemble Kalman filters, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2023, pp. 1947–1954, http://dx.doi.org/10.1109/IROS55552.2023.10341617.
[15] D. Van Nam, K. Gon-Woo, Learning observation model for factor graph based-state estimation using intrinsic sensors, IEEE Trans. Autom. Sci. Eng. 20 (3) (2023) 2049–2062, http://dx.doi.org/10.1109/TASE.2022.3193411.
[16] W. Talbot, J. Nubert, T. Tuna, C. Cadena, F. Dümbgen, J. Tordesillas, T.D. Barfoot, M. Hutter, Continuous-time state estimation methods in robotics: A survey, 2024, arXiv:2411.03951.
[17] B. Hoffman, E. Baumgartner, T. Huntsberger, P. Schenker, Improved rover state estimation in challenging terrain, Auton. Robots 6 (1999) 113–130, http://dx.doi.org/10.1023/A:1008879310128.
[18] A. Sakai, Y. Tamura, Y. Kuroda, An efficient solution to 6DOF localization using unscented Kalman filter for planetary rovers, in: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 4154–4159, http://dx.doi.org/10.1109/IROS.2009.5354677.
[19] C. Kilic, J.N. Gross, N. Ohi, R. Watson, J. Strader, T. Swiger, S. Harper, Y. Gu, Improved planetary rover inertial navigation and wheel odometry performance through periodic use of zero-type constraints, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2019, pp. 552–559, http://dx.doi.org/10.1109/IROS40897.2019.8967634.
[20] S. Das, C. Kilic, R. Watson, J. Gross, A comparison of robust Kalman filters for improving wheel-inertial odometry in planetary rovers, 2021, pp. 2621–2632, http://dx.doi.org/10.33012/2021.17938.
[21] D. Geromichalos, M. Azkarate, E. Tsardoulias, L. Gerdes, L. Petrou, C. Perez Del Pulgar, SLAM for autonomous planetary rovers with global localization, J. Field Robot. 37 (5) (2020) 830–847, http://dx.doi.org/10.1002/rob.21943.
[22] B. Zheng, T. Cao, T. Hu, Z. Qian, S. Wang, F. Han, An autonomous navigation method for planetary rover based on multi-modal fusion and multi-factor graph optimization, J. Phys.: Conf. Ser. 2762 (1) (2024) 012002, http://dx.doi.org/10.1088/1742-6596/2762/1/012002.
[23] W. Deng, B. Hou, X. Zhou, J. Wang, X. Zhao, H. Zhou, Robust factor graph-based state estimation for mars rovers under sparse and low-quality observations, Aerosp. Sci. Technol. (2025) 111221, http://dx.doi.org/10.1016/j.ast.2025.111221.
[24] P.F. Muir, C.P. Neuman, Kinematic modeling of wheeled mobile robots, J. Robot. Syst. 4 (2) (1987) 281–340, http://dx.doi.org/10.1002/rob.4620040209.
[25] M. Tarokh, G. McDermott, Kinematics modeling and analyses of articulated rovers, IEEE Trans. Robot. 21 (4) (2005) 539–553, http://dx.doi.org/10.1109/TRO.2005.847602.
[26] A. Kelly, N. Seegmiller, Recursive kinematic propagation for wheeled mobile robots, Int. J. Robot. Res. 34 (3) (2015) 288–313, http://dx.doi.org/10.1177/0278364914551773.
[27] N. Seegmiller, A. Kelly, Enhanced 3D kinematic modeling of wheeled mobile robots, in: Robotics: Science and Systems, 2014, http://dx.doi.org/10.15607/RSS.2014.X.020.
[28] T. Okawara, K. Koide, S. Oishi, M. Yokozuka, A. Banno, K. Uno, K. Yoshida, Tightly-coupled LiDAR-IMU-wheel odometry with an online neural kinematic model learning via factor graph optimization, Robot. Auton. Syst. 187 (2025) 104929, http://dx.doi.org/10.1016/j.robot.2025.104929.
[29] A. Chiminelli, P. Roncagliolo, A. Ferri, A. Merlo, S. Bologna, A. Cernusco, M. Deremetz, M. Debroise, R. Boitte, J. Reynolds, M. Landgraf, European moon rover system (EMRS), 2023.
[30] S. Ebrahimi, A. Mardani, A new contact angle detection method for dynamics estimation of a UGV subject to slipping in rough-terrain, J. Intell. Robot. Syst. 95 (3) (2019) 999–1019, http://dx.doi.org/10.1007/s10846-018-0932-3.
[31] J. Guo, W. Li, L. Ding, T. Guo, H. Gao, B. Huang, Z. Deng, High–slip wheel–terrain contact modelling for grouser–wheeled planetary rovers traversing on sandy terrains, Mech. Mach. Theory 153 (2020) 104032, http://dx.doi.org/10.1016/j.mechmachtheory.2020.104032.
[32] F. Xue, C. Yao, Y. Yuan, Y. Ge, W. Shi, Z. Zhu, L. Ding, Z. Jia, Wheel-terrain contact geometry estimation and interaction analysis using aside-wheel camera over deformable terrains, IEEE Robot. Autom. Lett. 8 (11) (2023) 7639–7646, http://dx.doi.org/10.1109/LRA.2023.3320614.
[33] C. Yao, F. Xue, Z. Wang, Y. Yuan, Z. Zhu, L. Ding, Z. Jia, Wheel vision: Wheel-terrain interaction measurement and analysis using a sensorized transparent wheel on deformable terrains, IEEE Robot. Autom. Lett. 8 (12) (2023) 7938–7945, http://dx.doi.org/10.1109/LRA.2023.3324291.
[34] S. Fortuna, P. Roncagliolo, D. Graziato, A. Merlo, S. Chiodini, A. Valmorbida, M. Pertile, ROS 2-based autonomous navigation strategy for a lunar rover featuring multiple locomotion modes, Acta Astronaut. (2025) http://dx.doi.org/10.1016/j.actaastro.2025.09.040.
[35] F. Kschischang, B. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, IEEE Trans. Inform. Theory 47 (2) (2001) 498–519, http://dx.doi.org/10.1109/18.910572.
[36] S. Macenski, T. Moore, D.V. Lu, A. Merzlyakov, M. Ferguson, From the desks of ROS maintainers: A survey of modern & capable mobile robotics algorithms in the robot operating system 2, Robot. Auton. Syst. 168 (2023) 104493, http://dx.doi.org/10.1016/j.robot.2023.104493.
[37] P. Roncagliolo, G. Berardi, P. Lanza, A. Merlo, D. Graziato, G. D'amore, Innovative solutions for fast autonomous navigation (SINAV), 2023.
[38] K. Koide, small_gicp: Efficient and parallel algorithms for point cloud registration, J. Open Source Softw. 9 (100) (2024) 6948, http://dx.doi.org/10.21105/joss.06948.
[39] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, D. Negrut, Chrono: An open source multi-physics dynamics engine, 2016, pp. 19–49.
[40] T. Moore, D. Stouch, A generalized extended Kalman filter implementation for the robot operating system, in: Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13), Springer, 2014.
[41] J. Solà, J. Vallvé, J. Casals, J. Deray, M. Fourmy, D. Atchuthan, A. Corominas-Murtra, J. Andrade-Cetto, WOLF: a modular estimation framework for robotics based on factor graphs, IEEE Robot. Autom. Lett. 7 (2) (2022) 4710–4717, http://dx.doi.org/10.1109/LRA.2022.3151404.
[42] Z. Zhang, D. Scaramuzza, A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2018, pp. 7244–7251, http://dx.doi.org/10.1109/IROS.2018.8593941.