Received: 19 July 2025 | Revised: 29 September 2025 | Accepted: 26 February 2026 | Published online: 19 March 2026

RESEARCH ARTICLE

Journal of Climbing and Walking Robots
2026, Vol. 00(00) 1–9
DOI: 10.47852/bonviewJCWR62026852

BON VIEW PUBLISHING

Visual–Inertial–Wheel-Based Velocity Estimation for a Vertical Wall Climbing Mobile Robot

Stefano Mutti[1,*], Simone Sabbadini[1], Lorenzo Latini[1], Diego Gitardi[1] and Anna Valente[1]

[1] *Department of Innovative Technologies, University of Applied Science and Arts of Southern Switzerland, Switzerland*

Abstract: Wall-climbing mobile robots are emerging as pivotal tools in various industries, offering unique capabilities to navigate complex and hazardous environments and perform tasks traditionally undertaken by humans. In such environments, it is often difficult to navigate and localize due to adverse conditions, including poor lighting quality, the absence of features that external sensors can leverage, and the poor condition of the navigation surface, which causes wheel slips. To effectively perform odometry by means of robot velocity estimation, we propose a visual–inertial–wheel odometry estimation using a modular deep learning–based model that leverages insightful information collected by a down-facing camera, an inertial measurement unit mounted on the robot, and the wheel encoders. The proposed model enables modular plugin modules that focus on wheel slip estimation and perception, ensuring reliability when these models are unusable due to environmental conditions. We aim to correctly identify the undesired behavior of the robot, such as wheel slippage, to characterize the navigation surface and estimate the mobile robot's velocities in real time. The presented architecture is successfully evaluated on an internally designed vacuum-based climbing mobile robot, and the results are compared to state-of-the-art methods.

Keywords: vertical robotics, climbing robotics, robotic localization, visual odometry

1. Introduction

Climbing mobile robots enable tasks in various environments thanks to their dexterity, supported by diverse adhesion mechanisms [1–3], each suitable for different surfaces. This allows them to navigate scenarios often dangerous to humans and to transport equipment for maintenance and repair [4]. For tasks such as maintenance and inspection, localization capability is of utmost importance for consistently reporting detailed information regarding faults or anomaly locations. Inconsistent odometry and localization in such scenarios lead to misregistered defect maps and unreliable revisit points, complicating maintenance planning and putting the mobile robot at risk during autonomous navigation.

In these environments, traditional navigation algorithms and sensors are unsuitable due to the lack of external perceptual features for sensors such as LiDAR and cameras, the impracticability of GPS, unreliable lighting conditions, and wheel slippage when employing dead-reckoning odometry. Moreover, they are often feature-poor for LiDAR due to simple geometry, as in penstocks, tunnels, silos, and wind turbine towers. Wheel-based odometry also suffers due to friction-induced slippage and surface curvature, which reduce the wheel contact area.

Recently, multimodal sensor fusion has gained interest due to impressive results driven by deep learning–based models [5]. Approaches are commonly categorized into loosely coupled and tightly coupled fusion, based on how inputs are used and merged and on how models can be split and repurposed for single modalities.

To address this problem, we propose a modular architecture featuring a core deep learning model that estimates the robot's velocities from consecutive inertial measurement unit (IMU) and wheel-encoder readings. Two optional modules estimate wheel slippage and preprocess visual data from consecutive images captured by a surface-facing camera. These models incorporate the mechanical and visual properties of the surface, are pretrained on the same navigation surface, and serve as optional inputs to the core estimator. For perception, we leverage a common characteristic of climbing-robot scenarios: a static, reliable surface monitored at close range, which often also serves logging purposes during maintenance and surveillance operations.

The wheel slippage module is a deep learning–based architecture that processes continuous IMU and wheel data while accounting for vibration-rich operation (e.g., when vacuum suction cups are used to keep the robot attached to the surface). This also enables characterization of the navigation surface in terms of wheel friction, which can indicate surface quality for maintenance and inform a motion planner that can mitigate slipping. The modularity allows the vision component and slip detection to be optional, increasing estimation accuracy on trained surfaces while enabling the core to perform even when dynamic occlusions

***Corresponding author:** Stefano Mutti, Department of Innovative Technologies, University of Applied Science and Arts of Southern Switzerland, Switzerland. Email: stefano.mutti@supsi.ch

affect the camera, the surface lacks features, or the surface friction changes with aging. The model's modularity also supports preserving and reusing components and fine-tuning them on surfaces with characteristics similar to the training one.

This framework opens the possibility of closed-loop velocity control that acts on each wheel according to the current slippage, thereby correcting the robot's trajectory along a given path.

The main contributions of our work are as follows:

- Provide a deep learning–based classification model to detect wheel slip using wheel encoders, motor driver data, and IMU data trained with an external tracking device.
- Provide a visual pipeline that encodes successive images into geometric-related data that the core module can use as an optional component to refine odometry.
- Provide an odometry standalone correction core model that, utilizing the kinematic data from the robot and the optional inputs of visual data and slippage detection, estimates the corrected odometry.
- Provide a method to modularize the model, allowing for its usage in various environments, for example, where surface features are occluded or lacking or where wheel slippage plays an important role.

This paper is organized as follows: Section 2 reviews the related works on odometry correction and slip detection for generic and vertical mobile robots. Section 3 establishes the system components and the work focus. Section 4 introduces the methodology and the approach components, while Section 5 reports the results from the experimental campaign and comparisons to state-of-the-art models.

2. Literature Review

2.1. Related work

Vertical climbing robots employ different mechanisms to ensure adhesion to the surface and navigation stability, depending on the surface characteristics and the robot parameters. Vacuum-based solutions, in contrast to electrical-based and magnetic [6, 7], or clamping-based [8], are able to adhere to many surfaces [9, 10], although suffering from the friction created by the vacuum seal contact, disrupting the wheel odometry computation by inducing slippage. Vertical climbing robots are often employed in scenarios that are hard to navigate and localize, making state-of-the-art localization algorithms inadequate for the task. Recent localization algorithms that leverage multi-sensor fusion [11], such as the studies of Lee et al. [12, 13] and Zhao et al. [14], proposed the joint usage of kinematic, IMU, and visual information to recover the robot's position in unstructured environments. These works don't take into account adversities commonly present in vertical robotics inspection scenarios, such as considerable wheel slippage due to poor surface friction and the absence of proper lighting for perception. Furthermore, these methods rely on a robust visual component that is strongly embedded into the model and cannot be detached, leveraging features in both color feed and depth, while in our common environments, the perceptive component cannot be taken for granted and might lack environment features to leverage. Confined spaces with few geometric features have been investigated [15], where the fusion of LiDAR-based SLAM and wheel–inertial data improved state-of-the-art localization in harsh environments. Relying on IMU-centric approaches yields suboptimal results in vertical robotics due to the vibrations produced by the friction of the adhesion mechanisms and the driving components [16]. Tavakoli et al. [17] address the problem of dead-reckoning computation for a lightweight omnidirectional climbing robot by integrating IMU data and using an optical flow sensor. These sensors are significantly affected by system vibrations, and in the case of the flow sensor, this can lead to incorrect results when the surface characteristics change during navigation. Moreover, the authors propose a model that embeds the perspective part, hence requiring good visual features during navigation. In some studies, vision-based pipelines employing a downward-facing camera are used to retrieve and match different kinds of features to reconstruct the odometry or map the environment [18–20]. These methods rely on visual input as their primary source of information to estimate the relative transformation between successive image frames. However, because they depend solely on visual data, without accounting for wheel slippage or incorporating IMU measurements, their accuracy is inherently limited by lighting conditions and the availability of distinguishable features in the environment. Filtering techniques have also been employed to estimate the robot slippage, such as the right invariant extended Kalman filter [21], and by filtering out wheel encoders' readings based on precomputed velocity thresholds [22]. These methods provide a whole robot slippage estimation, which effectively improves odometry accuracy along the navigation path, but do not provide a punctual wheel-based estimation, which can be used by the robot controller and path planner to enrich navigation data and overcome local negative behavior during the navigation [23].

Compared to the state-of-the-art methods, our proposal includes the following:

- Dynamic input model that can perform while leveraging different data sources and won't be disrupted by the absence of inputs, such as a lack of visual feed.
- Optional components focused on surface perception and wheel slip characterization, allowing pretraining of such modules, interoperability in akin environments, and fine-tuning in similar environments.
- Single wheel slip detection embedded into a classifier model in order to forward information to an upper-level kinematic controller or path planner. Giving real-time insight into the navigation surface.
- Beforehand model training allows tuning the model on selected surfaces and environments.

The provided model is simple and lightweight, assuring compatibility with edge devices that can perform estimation, detection, and fine-tuning directly on the mobile robot.

2.2. System overview

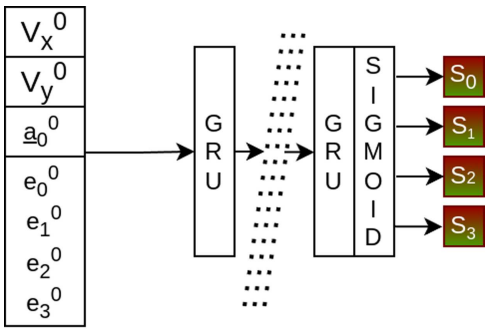
Our system comprises a vertical climbing mobile robot endowed with an IMU and a surface-facing camera. With the scope of producing a database, an external tracking device capable of tracking the real-time positions of the mobile robot through a fixed onboard visual marker is positioned in the environment during the data-acquiring phase. First, we define frames and notations that we will use throughout this work. We denote the world fixed frame as **W**, the IMU frame as **I**, the camera frame as **C**, and the mobile robot fixed frame as **M**. The external tracker measures the relative displacement of the mobile robot tracker between consecutive timestamps and then transforms this displacement with respect to the mobile robot frame; hence, the tracker positioning is irrelevant to the method. Each wheel's rotational velocity and torque are denoted as θ_n, e_n, where n is the wheel index

numbering, for N wheels, while the robot's velocities measured by the external device are denoted as $\tilde{v}_x$, $\widetilde{vm}_y$, $\tilde{\omega}$. Measured velocities are obtained by differentiating the measured position over time, while θ_n, e_n are measured by motor encoders and drivers. IMU data are computed as the norm of the three accelerations measured on the IMU axes as a, instead of the three components; this aims to lower the model's computational requirement and make data agnostic to orientation in order to possibly employ this method on variable steep surfaces. The mobile robot planar velocities are denoted as $[v_x,\ v_z,\ \omega]$, computed from the wheels' velocities and robot kinematic parameters through the forward kinematics as:

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = FK(.,R) \begin{bmatrix} \theta_. \\ \vdots \end{bmatrix}. \tag{1}$$

Each variable timestamp is denoted as $.^t$. We suppose the robot's intrinsic parameters computed through the kinematic calibration and the sensor's extrinsic parameters to be completely known.

Figure 1
Wheel slip detector based on GRU architecture



Note: Wheel slip detector, comprised of a gated recurrent unit (GRU) architecture and a buffer to store sequential data, taking as inputs the kinematic robot velocities, wheel efforts, and inertial readings, for a four-driving wheel mobile robot. The output consists of a discrete token for each well, depending on detected slippage or normal behavior, assuming the values [−1, 1].

3. Research Methodology

3.1. Methodology overview

Our model comprises three main parts: a main model with two optional additional components. The first component is a fully connected classifier that detects the slippage of the driving wheels using wheel data and IMU data as inputs. The second component is an image feature extractor and preprocessing that serves as additional visual information for the model, working on consecutive camera images, and whose output embeds the displacement of tracked features from two consecutive images. The main model outputs corrected mobile robot velocities to reconstruct the odometry; it uses wheel data and IMU data as main inputs, and optionally, the wheel slip detection outputs from the first component and outputted embedded data from the second component. The aim is to have a standalone main model on which optional components might be added dynamically, depending on the environmental conditions. The main model has to correctly estimate the velocities while performing standalone, leveraging the optional models to increase its accuracy when possible. To this aim, the classifier module is trained by itself, while the main module is trained by using a high rate dropout on the optional inputs, as explained in the next sections.

3.2. Wheel slipping classification

The wheel slipping detector consists of a gated recurrent unit (GRU) model for binary classification, as shown in Figure 1. The slipping detector aims to detect instantaneous single-wheel slippage using kinematic and inertial data, providing insight into the surface each wheel is in contact with, and leveraging sequential data input. The model takes as input the wheel effort data consisting of $[e_n^t]$, robot linear kinematic data $[v_x^t]$, $[v_y^t]$, and the 9-axis IMU data $[a^t]$, forming a single flattened vector of length 15. To train a model that detects single wheel slippage effectively, we design a database with the aforementioned input and an externally computed slip token $[s^t]$ that assesses each wheel slippage at time t as a discrete variable; hence, the model's output is a length n vector of discrete values [−1, 1]. The slippage token is computed by measuring the robot movements externally and through the inverse kinematics, comparing externally measured wheel speed $\tilde{\theta}_.$ with the actual one measured by the encoders onboard $\theta_.$, giving a wheel slipping ratio $abs\left(\left(\theta_. - \tilde{\theta}_.\right)/\theta_.\right)$, which has to be compared to a threshold β to define the discrete token. The choice of having discrete token values equal to [−1, 1] instead of estimating a continuous slip coefficient is dictated by the aim to robustify this single model, generalize the model on different surface scenarios, and avoid computation redundancy due to the fact that the main model will employ similar data for localization purposes. Using robot velocities instead of the wheel's velocities comes as a result of the experimental phase, where it has been observed that it requires more computing capability from both the fully connected network estimating it and the main module using it, without accuracy improvements. This comes as a result of the wheel's data not being coherent with the IMU frame, while the robot velocities are expressed in a frame akin to the inertial one. This module output is zeroed when the wheel slip module is not functioning; in this way, the module has three possible outputs [−1, 0, 1] for slippage, module not in use, and no slippage.

In order to collect the ground truth labels, we reconstruct real wheel punctual velocities from the measured robot velocities through the inverse kinematics as:

$$\begin{bmatrix} \tilde{\theta}_. \\ \vdots \end{bmatrix} = FK^{-1}(.,R) \begin{bmatrix} \tilde{v}_x \\ \tilde{v}_y \\ \tilde{\omega} \end{bmatrix} \tag{2}$$

and then compare each externally measured wheel's velocity $\tilde{\theta}_.$ with the one read by the motor encoders θ, and set the token value as:

$$\begin{cases} \left|\left(\left(\theta_. - \tilde{\theta}_.\right)/\theta_.\right)\right| > \beta \rightarrow s_. = -1 \\ \left|\left(\left(\theta_. - \tilde{\theta}_.\right)/\theta_.\right)\right| < \beta \rightarrow s_. = 1 \end{cases} \tag{3}$$

where the absolute value is used to merge slipping and skidding wheel effects, and β is a velocity threshold that helps neglect numerical errors and defines the velocity difference at which the wheel is considered to be slipping.

3.3. Visual feature extraction and preprocessing

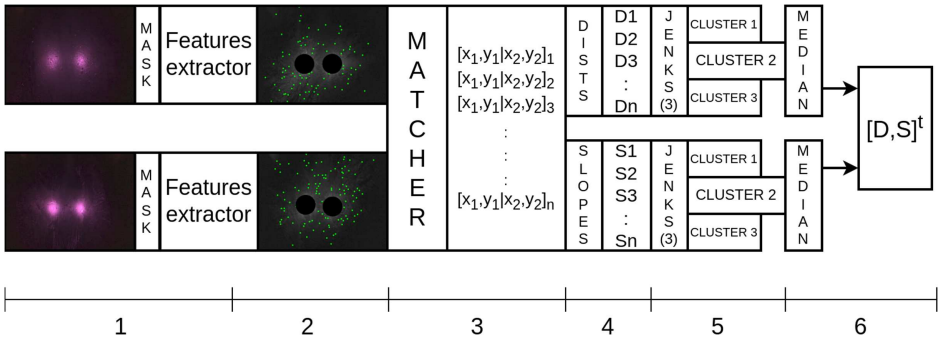
The image feature extractor and preprocessing component are responsible for extracting features from two consecutive images, preprocessing these features, and feeding the output to the main module. This aims to lower the computational load on the main module and detach the perceptive part from the other modules. This detachment allows the possibility of designing the model separately from the other modules, reusing the model in systems with a similar visual appearance, or switching between different models based on environmental conditions during the operation. The appropriate feature detection and matching algorithm depends on the images' characteristics and the relative transformation between them, which, in our case, consists of a two-dimensional linear transformation. The decision to preprocess the images instead of using a convolutional neural network (CNN) approach to process images directly is motivated by the fact that feature tracking can already significantly embed geometrical information related to image displacements without the computational complexity common in CNNs. Furthermore, using this approach lets us process full-resolution images, while CNN struggles to keep a good performance to computational complexity ratio in high-resolution images [24]. For the sake of generality, our proposal does not include a specific feature detection and matching algorithm but rather a method-agnostic pipeline, which will be tested in the experimental section using two mainstream algorithms: the Scale Invariant Feature Transform (SIFT) with FLANN matching implementation in OpenCV [25] and the SuperPoint [26] with SuperGlue [27] combination. This choice is made to have results based on an established and well-known method, such as the SIFT-FLANN, and on a new state-of-the-art method, such as SuperPoint with SuperGlue. Since the perception module is unaware of the data timings, the training and deployment are performed using data acquired at a constant rate. The processing pipeline shown in Figure 2 consists of the following steps:

1) The vision pipeline takes as input the image at time $t-1$ and the one at time t, extracting the visual features and descriptors from both images.
2) Masking out features in portions that might be affected by the local illumination device, if present, so as not to include borders of the lighted area as features.
3) Match best features using a matching algorithm, outputting a list of paired points (pixel coordinates) of both images.
4) Compute the two arrays containing the Euclidean distance (DIST) of the pixel coordinates and the relative slope (SLOPES) of every pair in the matchings list. The values in DIST and SLOPE are related to the robot's velocities.
5) To reliably reject outsiders and select the most insightful pair, we cluster both distances and slope arrays using the Jenks natural breaks classification method to produce three clusters, knowing that the first and last clusters will contain lower-bound and upper-bound outsiders and the middle cluster will contain the most successful feature-matching values.
6) We take the median values of the middle clusters array to produce a $[D, S]^t$ pair, embedding a distance and slope value that the main network can leverage to produce a better velocity estimation.

3.4. Main module

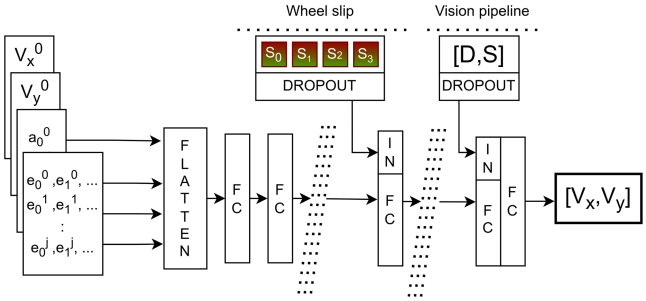
The main module is in charge of outputting an estimate of the translational velocities $\hat{v}_x$ and $\hat{v}_y$, taking into account the wheel efforts e., the computed nominal forward kinematics v_x and v_y, and the outputs of the wheel slip and visual modules. On the matter of generalization, while the kinematic structure of the robot takes part in the database creation, the model is itself general purpose since it is being fed by Cartesian velocities and the wheel's efforts. The model input is hence shaped as the input of the wheel slip classifier, a flattened vector of length $(3+N)\,j$, with j as the number of consecutive timestamped data. The main module needs to be trained with respect to having the optional wheel slip detection input and the optional visual pipeline input to retain the model's validity and performance even when performing alone. To do so, the inputted values from the visual and wheel slip modules are randomly zeroed during the training using the dropout regularization technique. While the dropout is mostly used on hidden layers to prevent the model from overfitting the data, in our case, it will be used to perform adequately, disregarding the presence of optional inputs, by using a dropout rate much higher than the conventional ones [28]. In contrast to common practice, the whole input vector is zeroed instead of the single input when dropout happens. This module architecture consists of multiple fully connected layers joined by rectifier units. Optional inputs are fed to the model in the mid-stage hidden layers. Feeding the optional inputs at the middle layers instead of at the first layer comes as a result of an experimental campaign of architecture optimization and is backed by the fact that a fine-tuning

Figure 2
Vision pipeline for visual displacement estimation from contiguous images



Note: Vision pipeline to process contiguous images, employing feature detection, matching, and clustering in order to provide the core module with visual displacement information to be leveraged for the robot velocities estimation. Numbered steps are defined in section IV-C. The example images shown are acquired in poor lighting conditions.

Figure 3
Core module architecture with fully connected layers for robot velocity estimation



Note: Core module architecture, comprised of fully connected (FC) layers. Optional inputs are fed into mid-stage hidden layers, separated by a dropout layer (on all the input vectors) used only in training in order to robustify the estimation when optional modules are not present. The model outputs $[v_x, v_y]$ robot velocities. This example is for a four-wheel mobile robot.

procedure has fewer parameters to adapt rather than the full network. The choice of not employing more complex neural architectures, such as the recurrent neural network, is mandated by the fact that, information-wise, the main module has to learn how to scale the translational velocities given the additional module inputs and the wheel torque recent values, while keeping simplicity to boost fine-tuning capabilities and the time required. The model architecture is shown in Figure 3.

4. Experiments

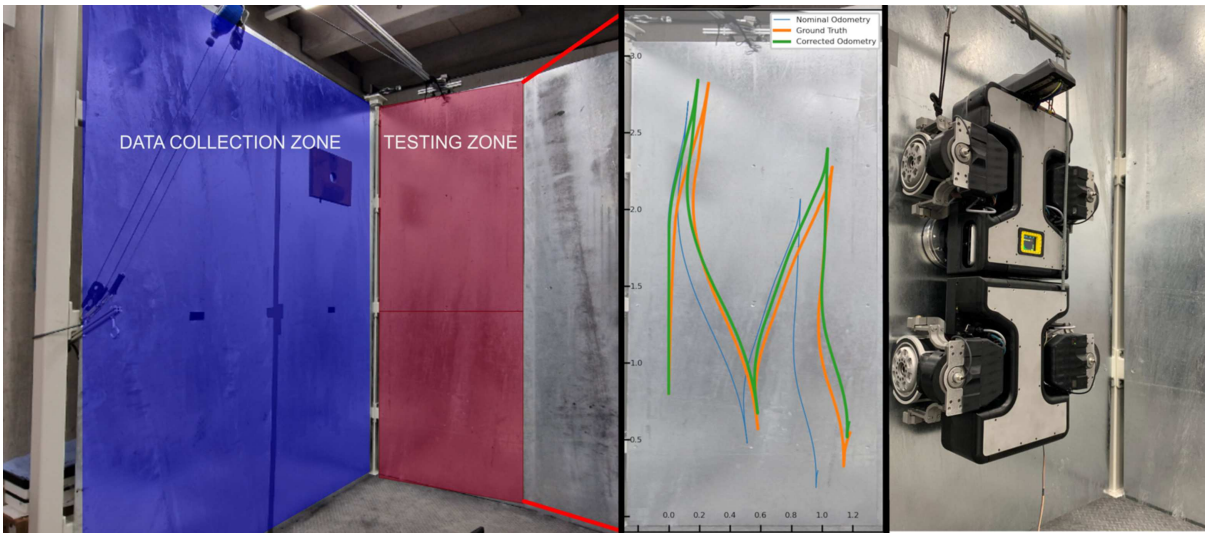
The mobile robot we take into account is a four-independent steering and four-independent driving wheels robot (4WIS4WID) internally designed [29], shown in the right panel of Figure 4, where all the wheels have a constant radius R and the steering angles are defined as σ and the driving angular speed as θ. Passive joints are installed between the steering and the body in order to adhere and navigate on curved surfaces. The kinematic model employed is described in [30], while the system is calibrated following [31]. The robot employs six vacuum suckers to adhere to the wall, each sucker driven by its own pump, in order to have redundancy on the adhesion mechanism. The robot's forward and inverse kinematics that will be used to compute robot velocities and generate the slipping database are the following:

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{cos(\sigma_1)}{4} & \frac{cos(\sigma_2)}{4} & \frac{cos(\sigma_3)}{4} & \frac{cos(\sigma_4)}{4} \\ \frac{sin(\sigma_1)}{4} & \frac{sin(\sigma_2)}{4} & \frac{sin(\sigma_3)}{4} & \frac{sin(\sigma_4)}{4} \\ G(\cdot)_1 & G(\cdot)_2 & G(\cdot)_3 & G(\cdot)_4 \end{bmatrix} \begin{bmatrix} \theta_1 R \\ \theta_2 R \\ \theta_3 R \\ \theta_4 R \end{bmatrix} \tag{4}$$

$$[\theta_\cdot] = FK^{-1}(., R) \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{5}$$

The data-acquiring site consists of a silo-like steel structure and a wall-like structure on which different panels (1.75 m × 3 m) can be mounted, providing different finishing textures, rugosity, and decline. For the silo-like structure, panels have a curvature radius of 3 m. Lighting conditions are also changed dynamically through the data collection phase. The data related to the training and validation of the models were acquired on the given panels. At the same time, the odometry reconstruction and test are performed on a separate panel with different superficial characteristics from the ones used in the training, as shown in the left panel of Figure 4.

Figure 4
Experimental setup, odometry reconstruction, and vertical mobile robot overview



Note: The left side shows the data-acquiring zone, comprised of the modular panels' wall, with the possibility of changing panels and simulating different navigation characteristics and surface conditions. This allows to perform many different tests in a confined space. The blue side is for model training and testing, while the red side is for odometry reconstruction. The central side shows an extraction of the reconstructed odometry comparison between wheel-computed, externally measured ground truth, and corrected odometry using the MM + WSC + Vision Module SIFT estimated velocities integration (Table 1). The right side shows the employed vertical mobile robot with the safety hanging mechanisms, whose kinematics are described in Section 5.

4.1. Data collection

Data collection is performed on the mobile robot using its ROS2-based controller to collect wheel speeds and torques at the rate of 1 KHz through the EtherCAT motor drivers. The robot is also equipped with an NVIDIA Jetson NX Orin, which is responsible for image collection through an IMX219 CSI camera at 30 Hz, positioned 0.15 m above the surface and equipped with a 1.8 mm wide lens, and will also be used to deploy deep learning–based models thanks to its embedded GPU. The inertial measurement unit employed is an LPMS-IG1, measuring at 500 Hz. Externally, the mobile robot's 6 degrees of freedom are tracked by a LEICA AT960 connected to an external computer, which tracks the position of a fixed marker on the mobile robot at 100 Hz. The transformation from the marker to the frame **M** is known by design and is used to transform ground truth readings into frame **M**. The acquired dataset comprises 120,000 images and 400,000 ground truth positions collected on the selected climbing surface for training. Being the camera with the slowest sensor in the system, the localization algorithm is run at the same frequency as the camera.

4.2. Model architecture and training

The models' parameters were selected after a sweep-based optimization campaign, where, first, the number of continuous inputs j was optimized and set to 10, and then the number of nodes in each fully connected layer, the learning rate, and the batch size were selected from a closed set of options in search of the validation loss minima. The GRU model for the wheel slippage classification comprises two layers with input and hidden size (15, 8) and a pre-output sigmoid for classification purposes. The main module comprises five fully connected layers with sizes (70, 60, 52, 34, 10), joined by a rectifier linear unit. In this setup, the wheel slip detection output is fed into the third layer, while the visual pipeline output is fed into the fifth. In the main module, the employed loss is the squared L2 norm loss, while in the wheel sleep classification, we employed the binary cross-entropy loss; each loss is averaged over the dataset length. Both outputs from the secondary models are passed through a dropout layer with a dropout rate of 0.6, before being fed to the main model. The parameter optimization was performed using WanDB, while the number of epochs was set to 3000 for all the training and the batch size to 128. The β threshold slipping parameter is set to 0.25; this has been computed by studying the slipping ratio $\left|\left(\theta_{\cdot} - \tilde{\theta}_{\cdot}\right)/\theta_{\cdot}\right|$ values across all the datasets and approximately selecting the discontinuity that divides normal behavior with measuring noise from slippage. The training was performed on an NVIDIA RTX 4090 using the PyTorch 2.2 framework, with a total training time of 5 h and 45 min for the main model, with the optional parts for each vision pipeline used. The dataset split ratio for training and validation is 80⁄20%, while the testing dataset is acquired on a different vertical surface. Apart from the wheel slip classifier, which is trained by itself, the other training includes the case with the additional wheel slip classifier module alone and an additional two with the different vision pipelines added. Additionally, a main module has been trained independently, without additional data from optional modules, to compare its performance with that of the others.

4.3. Results

In Table 1 are gathered the results of the models trained in terms of training and validation losses, testing error as the mean squared error on the estimated velocities, and accuracy for the wheel slip classification. The table also includes the "Test Error Main" voice, hence the error yielded by the main module used alone but trained with the optional modules. As shown by the results, the main module obtains the best performance in conjunction with the wheel slip classifier and the SIFT-based visual pipeline. The application of SuperPoint and SuperGlue did not yield meaningful results, which may be due to the fact that surface features are not well-defined on a plain navigation surface like the one used to train the mentioned models. All the modules but the one based on SuperPoint and SuperGlue performed better than the vanilla main module trained alone case (Test Error Table 1), while their respective main module without optional inputs also yielded better results than the main module alone (Test Error Main Table 1). To reconstruct the odometry, estimated mobile robot velocities are integrated at each time step. Due to the considerable role played

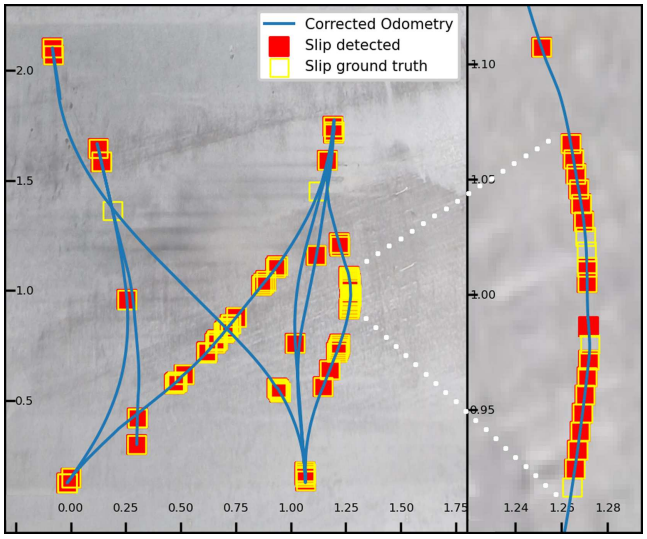
Table 1
Results of training, validation, and testing across models

Models training and testing results					
Model	Wheel Slip Classifier (WSC)	Main Module (MM) alone	MM + WSC	MM + WSC + Vision Module (SuperPoint and SuperGlue)	MM + WSC + Vision Module (SIFT)
Learning rate	0.002	0.002	0.0025	0.002	0.002
Training loss	0.17	0.051	0.047	0.053	0.043
Validation loss	0.22	0.055	0.049	0.061	0.046
Test accuracy	91.8%	·	·	·	·
Test error	·	6.12e–5	5.13e–5	6.12e–5	5.01e–5
Test error main	·	6.12e–5	5.82e–5	6.92e–5	5.24e–5
Inference time[s] (1)	3.08e–5	1.21e–4	1.17e–4	1.17e–4	1.17e–4
Inference time[s] (2)	1.17e–4	4.39e–4	4.31e–4	4.31e–4	4.31e–4

Note: Results relative to the training, validation, and testing of the various models. Losses refer to the average of the last ten epochs, the classification accuracy is averaged on the four wheels, and inference times refer to NVIDIA RTX 4090 (1) and NVIDIA Jetson NX Orin (2). Test errors are measured as the squared L2 norm of the testing portion of the dataset.

by friction in adhesion mechanisms, wheel slippage detection lacks a clear characterization and primarily depends on the threshold value β, making the problem more challenging to tackle and improving classification accuracy difficult. The odometry reconstruction from the estimated velocities for the MM + WSC + Vision Module (SIFT) architecture is shown in the middle panel of Figure 4. Figure 5 shows the wall characterization performed by the slippage classification algorithm during a wall vertical navigation; the red squares show the reconstructed odometry points where the right front wheel is slipping, tested on newly logged data on the test wall.

Figure 5
Slippage detection during vertical climbing



Note: Slippage detection on a vertical climbing wall, the blue line shows the odometry reconstructed by the MM + WSC + Vision Module SIFT method, while the red squares show estimated slippage detection triggered when the front right wheel is slipping. Yellow squares show positive slip ground truth, measured externally.

4.4. Fine-tuning

Fine-tuning enables the rapid deployment and optimization of the system in scenarios similar to those in which the training was performed. Each new model is fine-tuned, starting from the models trained using the initial, substantial amount of data, with a much smaller dataset. The aim is to retain the system knowledge embedded in the prior model while focusing on the new data. The slippage detection and the main modules are fine-tuned by training the model weights with the new data, using a smaller learning rate. Specifically, an amount of data equivalent to 15 s of logging was used for this purpose in order to test how a fast real-world deployment would perform. A value of 0.001 was used for the classifier learning rate, while the main module's learning rate was set to 0.0015. The training is performed directly on the edge using the capabilities of the Jetson Orin NX computer, with the same batch size as the main training and 15 epochs. Ten different trials have been performed, taking an average of 40 s for the fine-tuning, on vertical surfaces with different characteristics from one another. During runtime, the Jetson Orin NX, which was in charge of running the algorithms, averaged 35% CPU consumption and 70% GPU consumption. The fine-tuning procedure took an average of 72 s with 10 epochs. The results and comparison are shown in Table 2, where OpenVINS [32], OKVIS [33], and wheel odometry computed by integration are compared; IMU values are used to compute the robot's orientation. Selected comparison methods are the ones that bear meaningful results, while many methods fail due to inertial noise and a lack of visual features. Errors are measured as root mean squared error (RMSE) among all points in the trajectory, compared to ground truth. It is worth mentioning that the results of classical visual–inertial methods are obtained under suboptimal conditions, due to vibrations and light.

5. Conclusion and Discussion

We propose a modular deep learning–based architecture comprising a central module responsible for estimating the robot's velocities from wheel and IMU data. This is paired with two optional modules that can aid the overall accuracy by estimating wheel slip during navigation and processing image pairs to extract useful features to be fed to the central module. The method is structured so that the central module can operate alone when the secondary modules are unusable in the given environmental conditions. Furthermore, the wheel slippage module can characterize the surface in terms of slippage and forward important information to the path planner. The results show that our framework accurately estimates velocities and slippage when working with all optional modules and maintains significant accuracy even when these modules are detached. This work can be inherently extended

Table 2
Test results on different wall panels before and after fine-tuning

Odometry error evaluation and comparison on the testing surface, before and after fine-tuning				
Model	Accuracy	Accuracy (FT)	Error	Error (FT)
Wheel Slip Classifier (WSC)	81.70%	89.10%	·	·
Main Module (MM) alone	·	·	0.0642	0.0541
MM + WSC	·	·	0.0592	0.048
MM + WSC + Vision Module (SuperPoint and SuperGlue)	·	·	0.0544	0.0501
MM + WSC + Vision Module (SIFT)	·	·	0.0513	0.0388
OpenVINS [32]	·	·	0.0672	0.0672
OKVIS [33]	·	·	0.0789	0.0789
Pure Wheel Odom	·	·	0.0936	0.0936

Note: Results relative to the test on different wall panels, prior to and after the fine-tuning (FT) procedure. Errors are measured as RMSE from the ground truth.

in two ways: first, by pursuing a self-supervised approach to make the system independent, and second, by attempting to unload part of the method or improve it through edge computing cameras directly on the mobile robot. This would help unload the computational strain on the main computer and access more advanced techniques (e.g., CNN) right on board. Most importantly, this work supports the investigation of a closed-loop-based scheme in order to control the robot's velocities and trajectory. Interesting directions for future research suggest employing this method to perform velocity control and wheel velocity compensation, thereby improving the maneuverability of vertical wall climbing robots and enabling them to reliably follow given trajectories.

Funding Support

This work was supported by the European Union WISE Project 101138718.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Author Contribution Statement

Stefano Mutti: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **Simone Sabbadini:** Methodology, Validation, Investigation, Resources, Data curation, Visualization. **Lorenzo Latini:** Methodology, Validation, Investigation, Resources, Data curation, Visualization. **Diego Gitardi:** Conceptualization, Software, Validation, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Supervision. **Anna Valente:** Writing – original draft, Writing – review & editing, Supervision, Project administration.

References

[1] Hajeer, A., Chen, L., & Hu, E. (2020). Review of classification for wall climbing robots for industrial inspection applications. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 1421–1426. https://doi.org/10.1109/CASE48305.2020.9216878

[2] Fang, Y., Wang, S., Bi, Q., Cui, D., & Yan, C. (2022). Design and technical development of wall-climbing robots: A review. *Journal of Bionic Engineering*, *19*(4), 877–901. https://doi.org/10.1007/s42235-022-00189-x

[3] Gitardi, D., Giardini, M., & Valente, A. (2021). Autonomous robotic platform for inspection and repairing operations in harsh environments. *International Journal of Computer Integrated Manufacturing*, *34*(6), 666–684. https://doi.org/10.1080/0951192X.2021.1925970

[4] Valente, A., Gitardi, D., & Carpanzano, E. (2020). Highly efficient compact cold spray system for in-situ repairing of stainless steel material components. *CIRP Annals*, *69*(1), 181–184. https://doi.org/10.1016/j.cirp.2020.04.095

[5] Tang, Q., Liang, J., & Zhu, F. (2023). A comparative review on multi-modal sensors fusion based on deep learning. *Signal Processing*, *213*, 109165. https://doi.org/10.1016/j.sigpro.2023.109165

[6] Khan, A., Ahmad, W., & Hussain, S. (2025). Design and fabrication of wall-climbing robot using magnetic adhesion. *Engineering Proceedings*, *111*(1), 8. https://doi.org/10.3390/engproc2025111008

[7] Leuthard, S., Eugster, T., Faesch, N., Feingold, R., Flynn, C., Fritsche, M., . . . , & Hutter, M. (2024). Magnecko: Design and control of a quadrupedal magnetic climbing robot. In *Climbing and Walking Robots Conference*, 55–67. Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-71301-9_5

[8] Li, R., Xiong, Y., Liu, Y., Shou, M., Qiu, B., Gong, X., & Lee, C. H. (2025). A two-wheel bite embracing modular climbing robot with high load. *International Journal of Mechanical Sciences*, *309*, 111068. https://doi.org/10.1016/j.ijmecsci.2025.111068

[9] Li, Zhengyang., Li, Zhenjing., Tam, L. M., & Xu, Q. (2023). Design and development of a versatile quadruped climbing robot with obstacle-overcoming and manipulation capabilities. *IEEE/ASME Transactions on Mechatronics*, *28*(3), 1649–1661. https://doi.org/10.1109/TMECH.2022.3221819

[10] Parween, R., Wen, T. Y., & Elara, M. R. (2021). Design and development of a vertical propagation robot for inspection of flat and curved surfaces. *IEEE Access*, *9*, 26168–26176. https://doi.org/10.1109/ACCESS.2020.3039014

[11] Yin, J., Yan, F., Liu, Y., He, G., & Zhuang, Y. (2024). An overview of simultaneous localisation and mapping: Towards multi-sensor fusion. *International Journal of Systems Science*, *55*(3), 550–568. https://doi.org/10.1080/00207721.2023.2282409

[12] Lee, W., Eckenhoff, K., Yang, Y., Geneva, P., & Huang, G. (2020). Visual-inertial-wheel odometry with online calibration. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4559–4566. https://doi.org/10.1109/IROS45743.2020.9341161

[13] Lee, W., Yang, Y., & Huang, G. (2021). Efficient multi-sensor aided inertial navigation with online calibration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 5706–5712. https://doi.org/10.1109/ICRA48506.2021.9561254

[14] Zhao, X., Li, Q., Wang, C., Dou, H., & Liu, B. (2024). Robust depth-aided visual-inertial-wheel odometry for mobile robots. *IEEE Transactions on Industrial Electronics*, *71*(8), 9161–9171. https://doi.org/10.1109/TIE.2023.3323731

[15] Junior, G. P. C., Rezende, A. M. C., Miranda, V. R. F., Fernandes, R., Azpurua, H., Neto, A. A., . . . , & Pessin, G. (2022). Ekf-loam: An adaptive fusion of lidar SLAM with wheel odometry and inertial data for confined spaces with few geometric features. *IEEE Transactions on Automation Science and Engineering*, *19*(3), 1458–1471. https://doi.org/10.1109/TASE.2022.3169442

[16] Zhao, S., Zhang, H., Wang, P., Nogueira, L., & Scherer, S. (2021). Super odometry: IMU-centric lidar-visual-inertial estimator for challenging environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8729–8736. https://doi.org/10.1109/IROS51168.2021.9635862

[17] Tavakoli, M., Lopes, P., Sgrigna, L., & Viegas, C. (2015). Motion control of an omnidirectional climbing robot based on dead reckoning method. *Mechatronics*, *30*, 94–106. https://doi.org/10.1016/j.mechatronics.2015.06.003

[18] Wilhelm, A., & Napp, N. (2024). Lightweight ground texture localization. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 10223–10229. https://doi.org/10.1109/ICRA57147.2024.10611712

[19] Hart, K. M., Englot, B., O'Shea, R. P., Kelly, J. D., & Martinez, D. (2023). Monocular simultaneous localization and mapping using ground textures. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2032–2038. https://doi.org/10.1109/ICRA48891.2023.10161558

[20] Gu, Z., Gong, Z., Tao, B., Yin, Z., & Ding, H. (2023). Global localization based on tether and visual-inertial odometry with adsorption constraints for climbing robots. *IEEE Transactions on Industrial Informatics*, *19*(5), 6762–6772. https://doi.org/10.1109/TII.2022.3205952

[21] Yu, X., Teng, S., Chakhachiro, T., Tong, W., Li, T., Lin, T.-Y., . . . , & Ghaffari, M. (2023). 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)*, 8096–8103. https://doi.org/10.1109/IROS55552.2023.10342519

[22] Qiao, C., Zhao, S., Zhang, Y., Wang, Y., & Zhang, D. (2023). Viw-fusion: Extrinsic calibration and pose estimation for visual-imu-wheel encoder system. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1489–1496. https://doi.org/10.1109/IROS55552.2023.10341453

[23] Gitardi, D., Sabbadini, S., & Valente, A. (2023). Trajectory error compensation for optimal control of UMA-2 – a climbing robot executing maintenance operation in harsh environment. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 10090–10096. https://doi.org/10.1109/ICRA48891.2023.10161025

[24] Shah, B., & Bhavsar, H. (2022). Time complexity in deep learning models. *Procedia Computer Science*, *215*, 202–210. https://doi.org/10.1016/j.procs.2022.12.023

[25] Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, *25*(11), 120–123.

[26] DeTone, D., Malisiewicz, T., & Rabinovich, A. (2018). Superpoint: Self-supervised interest point detection and description. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 337–33712. https://doi.org/10.1109/CVPRW.2018.00060

[27] Sarlin, P.-E., DeTone, D., Malisiewicz, T., & Rabinovich, A. (2020). SuperGlue: Learning feature matching with graph neural networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4937–4946. https://doi.org/10.1109/CVPR42600.2020.00499

[28] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

[29] Gitardi, D., Sabbadini, S., & Valente, A. (2022). UMA Universal Maintenance Automata – an adaptable robotic platform designed to run maintenance operations in harsh environment. *Procedia CIRP*, *107*, 1473–1478. https://doi.org/10.1016/j.procir.2022.05.177

[30] Lee, M., & Li, T. S. (2015). Kinematics, dynamics and control design of 4WIS4WID mobile robots. *The Journal of Engineering*, *2015*(1), 6–16. https://doi.org/10.1049/joe.2014.0241

[31] Mutti, S., & Pedrocchi, N. (2021). Improved tracking and docking of industrial mobile robots through ukf vision-based kinematics calibration. *IEEE Access*, *9*, 127664–127671. https://doi.org/10.1109/ACCESS.2021.3111004

[32] Geneva, P., Eckenhoff, K., Lee, W., Yang, Y., & Huang, G. (2020). OpenVINS: A research platform for visual-inertial estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 4666–4672. https://doi.org/10.1109/ICRA40945.2020.9196524

[33] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., & Furgale, P. (2015). Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, *34*(3), 314–334. https://doi.org/10.1177/0278364914554813

How to Cite: Mutti, S., Sabbadini, S., Latini, L., Gitardi, D., & Valente, A. (2026). Visual–Inertial–Wheel-Based Velocity Estimation for a Vertical Wall Climbing Mobile Robot. *Journal of Climbing and Walking Robots*. https://doi.org/10.47852/bonviewJCWR62026852