Report "Distributed Robot Perception (AIS 2023/2024)"

1st Riccardo Bussola

2nd Nicola Zilio

***Abstract*—Accurate estimation and compensation of slippage are critical challenges for wheeled and tracked robots operating in unstructured terrain. This work presents a novel real-time method for estimating longitudinal and lateral slippage using the Iterative Reweighted Least Squares (IRLS) algorithm. By combining data from proprioceptive and exteroceptive sensors, the IRLS-based estimator is designed to adjust based on recorded data and predict slippage using only wheel speeds. We validate this approach in a simulated environment, comparing it to two other methods: a baseline with no slippage compensation and an exponential approximation technique. Our method demonstrates superior trajectory tracking performance, particularly as it adapts over time through data-driven updates.**

I. Introduction

The accurate estimation of slippage represents a significant challenge in robotics, particularly for wheeled robots operating in dynamic environments. The phenomenon of slippage has the potential to significantly impact a robot's ability to navigate autonomously, and perform tasks, thereby making it a pivotal issue in achieving the final objective of a task (e.g. planetary exploration [1]). When a robot moves through terrains with different characteristics, whether it's uneven ground, wet surfaces, or other unpredictable conditions, its motion can be influenced by numerous interaction factors that are often difficult to predict or measure. In such scenarios, accurately estimating slippage is essential for ensuring reliable operation and effective control. Slippage occurs when the robot wheels or tracks lose adhesion to the terrain, causing a discrepancy between the actual vehicle movement and the one predicted by the kinematic model.

Tracked Mobile Robots (**TMRs**) provide a large contact area with the ground, offering enhanced mobility in unstructured environments. However, the use of tracks can increase the occurrence of both longitudinal and lateral slippage, which poses challenges for precise control and path tracking. To address these challenges, kinematic models must be adapted to account for the effects of slippage.

J.Y. Wong's work [2]provides a comprehensive review of research in the field of vehicle-terrain interaction, commonly known as terramechanics, a subject essential for simulating vehicle behavior on diverse terrains. Building on this, Al-Milli et al. [3] introduced a model to predict the traversability of skid-steered vehicles on soft terrain, addressing excessive slippage during turns. Ahn et al. [4] used an Extended Kalman Filter (EKF) integrated with a soil model to estimate track-soil interactions in real time, identifying critical soil parameters like cohesion and friction angle for enhanced control. Zhao et al. [5] proposed a trajectory prediction method for tracked robots using EKF and an Improved Sliding Mode Observer (ISMO) to estimate slippage, improving navigation on varied terrains. Moosavian and Kalantari [6] experimentally modeled slip in tracked robots, using exponential functions based on path curvature and speed to estimate slip coefficients. Song et al. [7] employed the Newton-Raphson and Least Squares methods to estimate soil parameters like cohesion and internal friction angle in real-time for unmanned ground vehicles.

In this work, we propose a novel method for real-time estimation and compensation of both longitudinal and lateral slippage using the Iterative Reweighted Least Squares (**IRLS**) algorithm [8]. Our approach combines proprioceptive and exteroceptive sensors to estimate slippage by comparing the velocity predicted by a kinematic model with the actual measured velocity. Once sufficient data are recorded, the IRLS-based estimator is continuously fitted to the data. After the parameters are optimized, the estimator can predict slippage using only the wheel speeds as inputs. This method utilizes multiple robots to accelerate data collection and fit the estimator in a distributed manner. We will observe that the proposed approach effectively reduces tracking error, showing improvement and convergence over time. The terrain is divided into patches with distinct characteristics, with the exact location of each patch assumed to be known.

A. Problem Formulation

Given a composition of terrains with varying friction coefficients and a closed-loop desired trajectory, our objective is to minimize the tracking error in both position and orientation by performing real-time estimation and compensation of slippage. In this context, we consider a scenario with an arbitrary number of robots (five in our case) randomly positioned along the trajectory as shown in Fig 1.

The objective is to construct a distributed slippage map, where each terrain patch is associated with an estimator that utilizes the left and right wheel speeds ω_l and ω_r to predict the slippage components: β_l and β_r, representing longitudinal slippage, and α, representing lateral slippage. After the initial lap, during which each robot collects sufficient data, the IRLS algorithm is applied to generate an estimation based on the local data. This estimate is subsequently updated after every half-lap as new data is collected. The robots, connected via point-to-point communication, share their local estimates to collaboratively refine a global estimate. Each local estimator update results in a corresponding update to the global estimator. To validate the approach, we will compare the average tracking error across three methods: no slippage compensation, a compensation based on an exponential estimation developed

The authors have the following addresses: Italy {riccardo.bussola, nicola.zilio1}@studenti.unitn.it. This report is the final document for the course of "Distributed Systems for Measurement and Automation".

by Professors Focchi and Palopoli, and a compensation based on the IRLS- estimation introduced in this work.

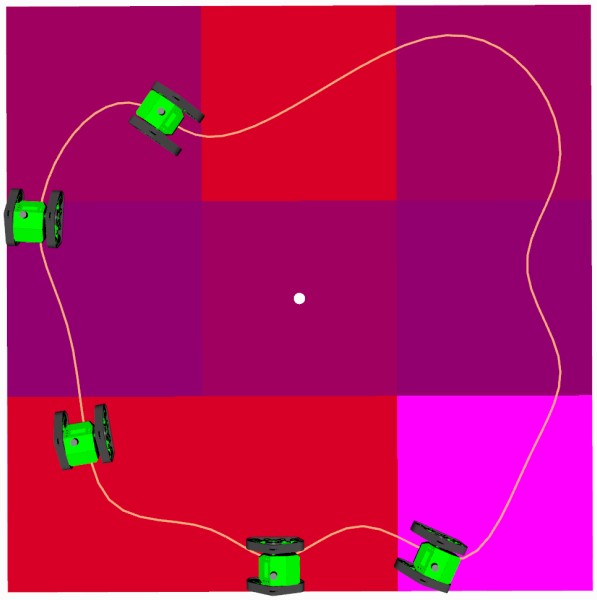

Fig. 1. RVIZ visualization of the tractor simulation, depicting the desired trajectory and the grid map representing terrain patches with distinct friction coefficients.

II. Adopted Models

A. Communication System

The communication system selected for this project is a peer-to-peer network between the robots, with no master-slave hierarchy. Each robot is considered equally important in relation to the others. The chosen network topology is a chain-ring configuration, where each robot can communicate with both its preceding and subsequent robots in the chain, ensuring full communication coverage within the network. A token-like protocol is simulated for message transmission, where at each timestep, a robot sends a message to the next robot in the ring. The first robot generates a JSON message containing its local estimation, identified by the robot's unique ID, and passes this message to the next robot. Upon receiving it, the next robot appends its own local estimate, similarly identified by its ID, and forwards the message. This process continues until the last robot receives the complete message containing all local estimates. A second pass is then initiated from the last robot, where the compiled message is transmitted back through the network, ensuring that all robots receive the full set of local estimates. After this second pass, each robot is able to compute the global estimate using average consensus, following the adjacency information of the network graph. The network topology and communication system are illustrated in Fig. 2.

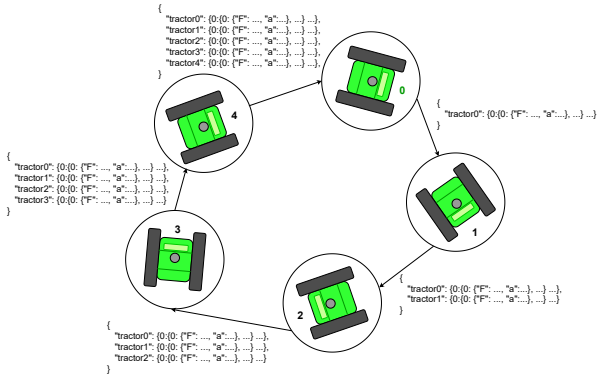

Fig. 2. Chain-ring network topology, robots exchange local estimates using a token-like protocol. The first pass aggregates estimates, and the second pass distributes the global estimate.

B. System Model

The robots employed in this experiment comprise five Robodyne MAXXII tracked robots. The MAXXII is a skid-steering unmanned ground vehicle (UGV) designed to ensure optimal traction across a range of terrains, including both soft and hard surfaces. The modular track system allows for customisation to meet specific operational requirements, thereby conferring high versatility.

Each robot is equipped with a number of advanced sensors, including a 9-degree-of-freedom (9DOF) inertial measurement unit (IMU), such as the XSens Mti-300, and a U-blox real-time kinematic (RTK) GPS. In this experiment, the GPS is used to continuously track the position of the robot at each time step. The IMU, through its gyroscope, provides the yaw angle, which is crucial for calculating the angular velocity of the robot. Additionally, the wheel encoders on the robot's tracks measure the actual velocities of the tracks. This data is instrumental in determining whether the robot is experiencing slippage with respect to the robot base, thereby allowing for the evaluation of traction performance under varying conditions.

As the present study is conducted within a simulated environment, the sensor data is inherently deterministic in nature. To more accurately replicate real-world conditions, where sensor data is typically subject to various uncertainties and noise, we introduced for each sensor a random Gaussian noise. This approach allows us to simulate the inherent imperfections of real-world sensors, thus creating a more realistic approximation of the conditions the robot would face in practical applications. The noise model for each sensor is characterised as follows:

$$\begin{aligned} X_{meas} &= X_{sim} + \varepsilon \\ \varepsilon &\sim \mathcal{N}(\mu, \sigma^2) \end{aligned} \tag{1}$$

The parameters for each sensors are reported in Tab. I

TABLE I
NOISE VALUES FOR EACH DATA PARAMETER

Parameter	Noise Value
$\dot{q}_L$	$\mathcal{N}(0, 0.01^2)$
$\dot{q}_R$	$\mathcal{N}(0, 0.01^2)$
v_x	$\mathcal{N}(0, 0.02^2)$
v_y	$\mathcal{N}(0, 0.02^2)$
ω	$\mathcal{N}(0, 0.01^2)$

III. SOLUTION

A. Lyapunov Based Control of tracked vehicle

To control the tracked robot along the desired trajectory, a Lyapunov-based controller, developed by Professors Focchi and Palopoli, has been adopted. Although the primary focus of this work is on the estimation process, understanding the controller is essential for comprehending the role of slippage values in affecting the trajectory tracking. The tracked vehicle's model is an adapted version of the unicycle model, with the following kinematic equations:

$$\begin{aligned} \dot{x} &= \frac{v}{\cos(\alpha)} \cos(\theta + \alpha) \\ \dot{y} &= \frac{v}{\sin(\alpha)} \sin(\theta + \alpha) \\ \dot{\theta} &= \omega \end{aligned} \tag{2}$$

where x and y are the Cartesian coordinates in the world frame, θ is the orientation with respect to the x-axis, v is the forward velocity, ω is the angular velocity, and α represents the lateral slippage angle, which depends on the terrain and robot characteristics.

Before discussing the implementation of the Lyapunov controller, we define the error vector between the actual and desired states:

$$\vec{e} = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} x - x_d \\ y - y_d \\ \theta - \theta_d \end{bmatrix}. \tag{3}$$

The angle ψ represents the orientation between the error vector $[e_x, e_y]$ and the robot's heading:

$$\psi = atan2(e_y, e_x) \tag{4}$$

We define Δ as the sum of the robot's current heading and the desired heading:

$$\Delta = \theta + \theta_d \tag{5}$$

Next, we can express e_x and e_y using ψ as follows:

$$\begin{aligned} e_{xy} &= \left\| \begin{bmatrix} e_x \\ e_y \end{bmatrix} \right\| = \sqrt{e_x^2 + e_y^2} \\ e_x &= e_{xy} \cos(\psi) \\ e_y &= e_{xy} \cos(\psi) \end{aligned} \tag{6}$$

The Lyapunov function is then defined as:

$$V = \frac{1}{2}(e_x^2 + e_y^2) + (1 - \cos(e_\theta + \alpha)) \tag{7}$$

To achieve convergence, the control inputs are select as follows:

$$\begin{aligned} v &= (v_d + \delta v)\cos(\alpha) \\ \omega &= \omega_d + \delta\omega \end{aligned} \tag{8}$$

where

$$\begin{aligned} \delta v &= -k_p e_{xy} \cos(\psi - (\theta + \alpha)) \\ \delta\omega &= -v_d e_{xy} \frac{1}{\cos(\frac{\alpha + e_\theta}{2})} \sin\left(\psi - \frac{\alpha + \Delta}{2}\right) \\ &\quad - k_\theta \sin(e_\theta + \alpha) - \dot{\alpha} \end{aligned} \tag{9}$$

Finally, the derivative of the Lyapunov function obtained by substituting the error dynamics and the above control low is the following:

$$\dot{V} = -k_p e_{xy}^2 \cos^2(\psi - (\alpha + \theta)) - k_\theta \sin^2(e_{theta} + \alpha) \tag{10}$$

This guarantees the stability of the control system. Hence becaue $\dot{V}$ is negative definite (except some configurations that are very unfrequent), the closed loop dynamics is guaranteed to be globally stable, with the error terms converging globally to zero over time.

B. Slippage components calculation

Both longitudinal and lateral slippage of the tracked vehicle are estimated using the base-frame linear and angular velocities, the robot's orientation, and the wheel speeds. While in simulation these values are directly provided by the simulator, we will describe the process as it would be implemented using real-world sensor data.

- **Wheel Velocities**: The left and right wheel velocities, ω_L and ω_R, are extracted from the encoder data $\dot{q}$.
- **Linear Velocity**: The linear velocity components of the base in the world frame, v_x and v_y, are obtained from GPS data.
- **Orientation**: The robot's orientation θ is measured using an IMU sensor after numeric differentiation.
- **Angular Velocity**: The angular velocity ω, around the z-axis, is provided by the gyroscope, also part of the IMU sensor.

The velocities from the world frame are transformed into the robot's body frame using the rotation matrix R_b^w, which is computed based on the robot's orientation θ:

$$R_b^w = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{11}$$

The body-frame velocity v_b is then given by:

$$v_b = {R_b^w}^T v_{xy} \tag{12}$$

where $v_{xy} = [v_x \ v_y]^T$ is the velocity vector in the world frame.

The longitudinal slippage is computed by comparing the actual track velocities, derived from the body-frame tangential velocity $v_{b,x}$ and ω, with the encoder-based velocities. Since we assume a unicycle model, only the tangential velocity is

used, as there is no radial component. The encoder-based velocities for the left and right tracks are calculated as:

$$\begin{aligned} v_{enc_L} &= R\omega_L \\ v_{enc_R} &= R\omega_R \end{aligned} \tag{13}$$

where R is the sprocket radius. The expected velocities for the left and right tracks are given by:

$$\begin{aligned} v_{trackL} &= v_{b,x} - \frac{\omega B}{2} \\ v_{trackR} &= v_{b,x} + \frac{\omega B}{2} \end{aligned} \tag{14}$$

where B is the track width. Finally, the longitudinal slippage values β_L and β_R for the left and right tracks are computed as the difference between the encoder-based velocities and the expected track velocities:

$$\begin{aligned} \beta_L &= v_{enc_L} - v_{track_L} \\ \beta_R &= v_{enc_R} - v_{track_R} \end{aligned} \tag{15}$$

The lateral (or side) slippage angle α is calculated using the body-frame velocity components:

$$\alpha = atan2(v_{b,y}, v_{b,x}) \tag{16}$$

These values are computed at each timestep and serve as ground truth data for fitting our estimator and subsequently evaluating the effectiveness of our proposed approach.

C. Distributed IRLS Slippage estimation

To estimate the slippage values for each robot, we employ an Iterative Reweighted Least Squares (IRLS) algorithm. The decision to use IRLS was driven by two key factors:

- **Near-Linearity of Slippage Values**: Through our analysis (see implementation section), we will demonstrate that the slippage values, particularly the longitudinal slippage components β_L and β_R, exhibit near-linear behavior with respect to ω_L and ω_R. The lateral slippage angle α shows more nonlinearity characteristics derived from the usage of $atan2$. To handle this problem, we employ a form of polynomial regression to keep the algorithm invariate.
- **Handling Sensor Uncertainty**: The second motivation for using IRLS is its ability to manage uncertainty, which arises from the fusion of data from multiple sensors such as encoders, GPS, and IMU in a nontrivial way. These sensors introduce varying levels of noise and potential outliers into the data. IRLS improves robustness by iteratively updating the weights of the least squares regression based on the residuals. This process reduces the impact of noisy or inaccurate data, ensuring more reliable estimates.

The implementation can be divided into two main phases: the local estimation phase and the distributed consensus phase. In the local estimation phase, each robot computes its individual estimate of the slippage parameters based on the data it collects. Once the local estimates are computed, the distributed consensus phase begins. In this phase, robots share their locally computed slippage estimates with one another. Through communication, the robots collaboratively refine their individual estimates to reach a consensus on the global slippage values, ensuring that the final estimates reflect the collective data across all robots [9].

The local IRLS estimator follows the formulation reported in the Algorithm 1 to iteratively compute and update the slippage parameters, ensuring accurate estimation despite noisy or inconsistent data. Through matrix manipulation, we have implemented a single regression model capable of simultaneously estimating three slippage values. This was achieved primarily by constructing a block diagonal matrix, which

Algorithm 1 Local Estimation using Iterative Reweighted Least Squares (IRLS)

Input: Pandas DataFrame $data$, degree of polynomial $degree$, number of iterations $iterations$=1
Output: Estimated coefficients $\hat{\theta}$

Extract wheel speeds and slippage values:
$\omega_l \leftarrow$ data.wheel_l
$\omega_r \leftarrow$ data.wheel_r
$\beta_l, \beta_r, \alpha \leftarrow$ data.beta_l, data.beta_r, data.alpha
$Y \leftarrow$ column stack of $(\beta_l, \beta_r, \alpha)$

Initialize matrix X:
$X \leftarrow$ ones$(N, 1)$ {N is the number of data points, bias}
for $d = 1$ to degree **do**
  $X \leftarrow$ column stack of $(X, \omega_l^d, \omega_r^d)$
end for

Initialize weights for OLS:
$W_l, W_r, W_\alpha \leftarrow$ ones(N) {Initial weights for β_l, β_r, α}
$W \leftarrow$ block diagonal matrix(W_l, W_r, W_α)

Expand X for the three output variables:
$X_{\text{expanded}} \leftarrow kron(I(3), X)$

for $i = 1$ to iteration **do**
  Weighted Least Squares:
  $F_0 \leftarrow X_{\text{expanded}}^T W X_{\text{expanded}}$
  $a_0 \leftarrow X_{\text{expanded}}^T W Y$
  $\hat{\theta} \leftarrow F_0^{-1} a_0$

  Compute residuals:
  $\hat{Y} \leftarrow X\hat{\theta}$
  residuals $\leftarrow Y - \hat{Y}$

  Update weights:
  $W_l \leftarrow$ diag$(1/|\text{residuals}[:, 0] + 1\text{e}{-10}|)$
  $W_r \leftarrow$ diag$(1/|\text{residuals}[:, 1] + 1\text{e}{-10}|)$
  $W_\alpha \leftarrow$ diag$(1/|\text{residuals}[:, 2] + 1\text{e}{-10}|)$
  $W \leftarrow$ block diagonal matrix(W_l, W_r, W_α)
end for

Return: F_0, a_0, θ_{hat}

integrates the three weighted matrices corresponding to the longitudinal slippage components, and the lateral slippage component. Additionally, we created an expanded input vector using the Kronecker product, allowing us to efficiently handle the three outputs in a unified regression framework. The initial regression step corresponds to an Ordinary Least Squares (OLS) approach, where all the diagonal elements of the weight matrix are set to one. In each subsequent iteration, the model recomputes the weights based on the residuals from the current estimated model parameters.

To develop our distributed algorithm, we avoided the use of a flooding approach, where each node broadcasts all stored data and sequentially computes its estimate. This distribution method is inefficient and unsuitable, particularly given our chain-ring network topology. Instead, as outlined in Algorithm 1, each robot computes its local estimate based on its own recorded data. Consequently, each i-th robot produces a local composite information matrix $F_i(k)$ and a local composite information state $a_i(k)$. To calculate the global estimate, as described in the communication section, each robot shares its local estimation with the network. To ensure average consensus, we adopted the maximum-degree weight scheme. The weighting factor $qw_{ij}(k)$ for each pair of robots i and j is defined as:

$$qw_{ij}(k) = \begin{cases} \frac{1}{n} & \text{if } (j,i) \in \epsilon \text{and} i \neq j, \\ 1 - \frac{d_i(k)}{n} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

where $d_i(k)$ represents the degree (number of adjacent robots) for the i-th robot, and n is the total number of robots in the network. The adjacency of robots is modeled using an adjacency matrix. Once the weights qw are computed, the update rule for achieving average consensus is given by:

$$\begin{aligned} F_i(k+1) &= F_i(k) + \sum_{j=1}^{n} q_{ij}(k)(F_j(k) - F_i(k)) \\ a_i(k+1) &= a_i(k) + \sum_{j=1}^{n} q_{ij}(k)(a_j(k) - a_i(k)) \end{aligned} \tag{18}$$

D. Slippage compensation

We have discussed the control strategy, the calculation of slippage components, and the distributed IRLS estimation strategy. Now, we will briefly address how to compensate for slippage based on the predicted values of β_L, β_R, and α from our estimator. First, the lateral slippage component α is directly applied in the control equation (8), influencing both the forward velocity v and the angular velocity ω.
After computing these two terms, the $\dot{q}^{des}$ is computed and a correction is applied as follows:

$$\begin{aligned} \dot{q}_L &= \frac{1}{R}(R\dot{q}_L^{des} + \beta_L) \\ \dot{q}_R &= \frac{1}{R}(R\dot{q}_R^{des} + \beta_R) \end{aligned} \tag{19}$$

IV. Implementation Details

The entire simulation is implemented using the Locosim robotic framework [10], developed by Professor Focchi and maintained with contributions from students. We began by utilizing the control scripts designed for the simulation of the MAXII tractor on various simulators, such as Gazebo and CoppeliaSim. However, due to the limitations in accurately simulating terrain interaction in these platforms, a standalone terramechanics simulator, developed by Davide Dorigoni under the supervision of Professor Biral, was employed to address these inaccuracies.

In our work, the dynamics are computed using this dedicated simulator, and the results are visualized in real time through RVIZ. We extended the basic simulation environment to handle multiple robots, with each robot instance managed independently. The dynamics and control computations are performed separately and sequentially for each robot at every timestep.

A. Cloosed loop trajectory generator

The generation of a continuous trajectory is a key component for simulating our task. The trajectory is initialized with an array of viapoints $S = \{(x_1, y_1), \ldots, (x_n, y_n), (x_1, y_1)\}$, representing the desired path. To ensure a closed-loop trajectory, the first point is repeated at the end of the array. The total time to complete the loop is denoted as t_{tot}. A cubic spline interpolation is applied to both the x- and y-coordinates. We chose cubic splines because they provide smooth second derivatives, which are required for computing both velocity and acceleration. The splines are constructed with periodic boundary conditions to ensure a smooth transition between the start and end of the loop:

$$\begin{aligned} CS_x(t) &= \text{CubicSpline}(t, S_x, \text{bc=periodic}) \\ CS_y(t) &= \text{CubicSpline}(t, S_y, \text{bc=periodic}) \\ t &\in [0, t_{\text{tot}}] \end{aligned} \tag{20}$$

At each time t, the trajectory generator computes the position, orientation, linear velocity, and angular velocity.

The position $(x(t), y(t))$ and the linear velocities $(\dot{x}(t), \dot{y}(t))$ are obtained by evaluating the splines and their first derivatives:

$$\begin{aligned} x(t) &= CS_x(t), \quad y(t) = CS_y(t) \\ \dot{x}(t) &= \dot{CS}_x(t), \quad \dot{y}(t) = \dot{CS}_y(t) \end{aligned} \tag{21}$$

The orientation $\theta(t)$ is computed as:

$$\theta(t) = \arctan 2(\dot{y}(t), \dot{x}(t)) \tag{22}$$

The body-frame velocity $v_b(t)$ is obtained by rotating the world-frame velocity using the robot's orientation. Only the tangential velocity $v_{b,x}(t)$ is considered for a unicycle model (same as 12):

$$\begin{aligned} v_b(t) &= R_b^w \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \end{bmatrix} \\ v(t) &= v_{b,x}(t) \end{aligned} \tag{23}$$

The angular velocity $\omega(t)$ is computed using the second derivatives of the cubic splines:

$$\ddot{x}(t) = \ddot{CS}_x(t), \quad \ddot{y}(t) = \ddot{CS}_y(t)$$
$$\omega(t) = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{\dot{x}(t)^2 + \dot{y}(t)^2} \tag{24}$$

The forward acceleration $\dot{v}(t)$ is calculated similarly to the forward velocity:

$$\dot{v}_b(t) = R_b^w \begin{bmatrix} \ddot{x}(t) \\ \ddot{y}(t) \end{bmatrix}$$
$$\dot{v}(t) = \dot{v}_{b,x}(t) \tag{25}$$

For simplicity, the angular acceleration $\dot{\omega}(t)$ is neglected and assumed to be zero:

$$\dot{\omega}(t) = 0 \tag{26}$$

B. Terrain Map and Slippage Estimation

As previously mentioned, the terrain is composed of multiple patches, each characterized by a different friction coefficient. This configuration simulates an unstructured terrain where the robot experiences varying slippage conditions, which it must estimate and compensate for. The class responsible for generating this terrain map is designed to handle arbitrary dimensions for the terrain's width, height, and patch size. In our case, the map is defined as a 9×9 meter area, and each patch is 3×3 meters in size (see Fig. 1). This patch size was chosen to ensure that the robot can traverse an entire patch with its full body, while allowing variations in orientation and velocities. The default friction coefficient in the original control script was set to 0.1. We use this value as a baseline and introduce slight random variations to construct the friction coefficient map, as shown in Table II. Since the locations of the patches are assumed to be known, the estimation problem involves creating a separate estimator for each patch. Each robot has its own slippage map, where each patch corresponds to a specific estimator. As the robot traverses the terrain, the process of local and global estimation is applied to each of these estimators. The terrain map class allows the robot to retrieve the correct patch based on its x, y coordinates, ensuring that the appropriate estimator is selected for each section of the terrain. This same process is also used to manage the coefficient parameters of the exponential estimation provided by Professor Focchi and Professor Palopoli.

TABLE II
FRICTION COEFFICIENT MAP

0.1	0.13349	0.1
0.09041	0.1	0.09041
0.13349	0.13349	0.15680

C. Analysis of Slippage Linearity and Motivation for IRLS

In the solution section, we introduced the motivation behind using the Iterative Reweighted Least Squares (IRLS) algorithm, emphasizing the near-linearity of the slippage components. We now delve deeper into this claim by analyzing the slippage behavior in relation to the left and right wheel speeds, ω_L and ω_R. Before this project, a preliminary analysis of the slippage surface shapes for β_L, β_R, and α, based on ω_L and ω_R, was conducted by Professor Focchi with the help of Bussola R. In their study, they approximated these values using a Neural Network. Their findings revealed that for a certain range of low, positive velocities, the longitudinal slippage components β_L and β_R exhibit nearly linear behavior. In contrast, the lateral slippage component α presents non-linear characteristics, particularly near the extremes $\{(\omega_{L,\min}, \omega_{R,\max}), (\omega_{L,\max}, \omega_{R,\min})\}$. In this project, the speed range is slightly higher than the one observed in previous experiments, showing that as the velocity increases pairwise, the slippage surfaces become more non-linear. However, due to the specific design of our trajectory and loop timing, the velocity trajectories remain within a range where the slippage components are approximately linear. Fig. 3 presents 3D plots of these slippage surfaces, the data was generated by Professor Focchi by collecting slippage values during the execution of an open-loop velocity trajectory on terrain with a default friction coefficient $\mu = 0.1$. To manage the low-level non-linearity observed, we opted for a third-order polynomial regression. While polynomial regression models non-linear relationships, it remains a linear statistical estimation problem, allowing us to apply the IRLS algorithm without modifications. This approach effectively balances the complexity of the model with the observed near-linearity of the slippage components. In the results section, we will compare the ground truth slippage surfaces with those produced by our estimation method to evaluate the accuracy and effectiveness of our approach.

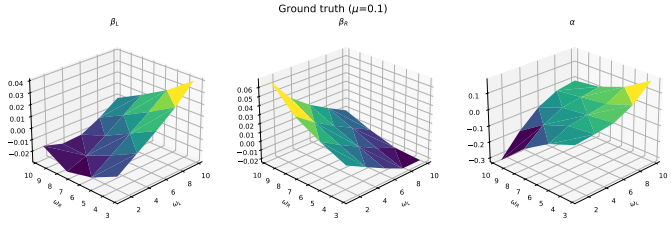

Fig. 3. 3D visualization of the ground truth slippage surfaces for β_L, β_R, and α as functions of the left and right wheel velocities ω_L and ω_R for $\mu = 0.1$

V. RESULTS

In this section, we present the results obtained from both the simulation and experimental evaluation. The primary objectives are to assess the accuracy of the slippage estimation and to validate the effectiveness of our approach in comparison to the method proposed by Professor Focchi and Professor Palopoli, particularly in terms of trajectory tracking improvement.

We conducted the same simulation under three different slippage compensation strategies: (1) the baseline, where no slippage estimation is applied, assuming the vehicle is

controlled as a unicycle, (2) our proposed approach, which employs a distributed IRLS algorithm, and (3) the approach based on the exponential approximation of slippage. By comparing these methods, we aim to demonstrate the advantages of our IRLS-based method in enhancing tracking performance, especially in varying terrain conditions. In Fig. 4, the actual

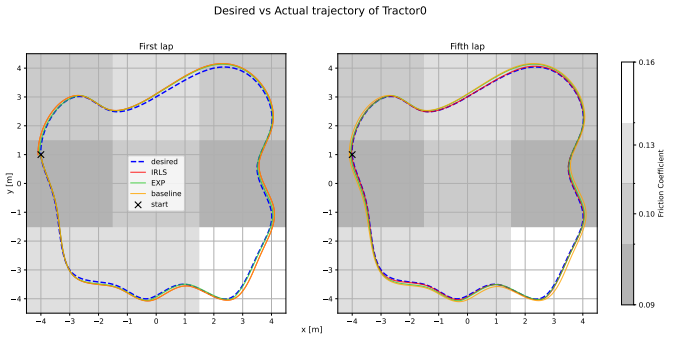

Fig. 4. Comparison of the desired trajectory and the actual trajectory realized during the first and fifth laps using three slippage compensation approaches. The comparison highlights the trajectory tracking performance and improvement over time for each method. In background, the friction map is reported

trajectories realized by the three slippage estimation and compensation approaches are presented for both the first and fifth laps. To fully analyze the first lap, it is important to consider that at time $t = 0$, the robot undergoes an initial acceleration phase to reach its full speed. As a result, part of the first lap is traversed at a velocity lower than that of subsequent laps.

The IRLS approach waits until the end of the first lap before executing the first update, after which updates are produced at intervals of every half lap. This delay allows the system to collect sufficient data before estimating the slippage components. A premature fitting would result in inaccurate estimates, leading to higher slippage due to incorrect correction behavior. In the first lap, the IRLS approach behaves similarly to the unicycle baseline, as no slippage is predicted (slippage is assumed to be zero). In contrast, the exponential approximation method does not change its tracking accuracy over time, as it relies on precomputed constants and does not adapt to new data like the IRLS approach. We can further analyze the evolution of tracking error by computing the average tracking error per lap as a function of the number of completed laps. The results are depicted in Fig. 5. As visually anticipated from Fig. 4 and as briefly discussed earlier, both the baseline and exponential approaches exhibit constant tracking error profiles throughout the laps, with the exception of lap 1, which is influenced by the initial acceleration phase. In contrast, the IRL method demonstrates an improvement over time. Specifically, after the initial lap, during which its first fitis is performed, the tracking error of the IRLS approach markedly decreases outperformin both the baseline and exponential methods. This substantial improvement underscores the effectiveness of IRLS in refining tracking accuracy as the system progresses through subsequent laps. Having analyzed the average tracking error, we now

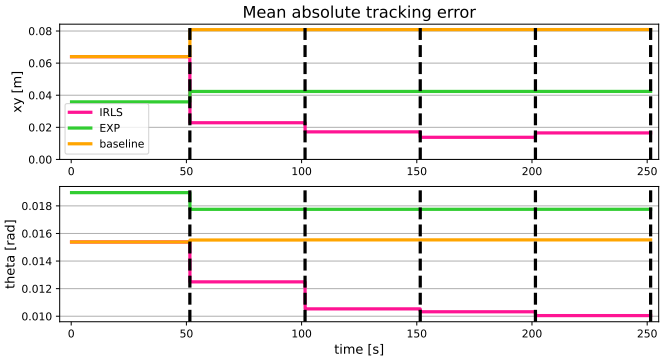

Fig. 5. Average Tracking Error Across Traveled Laps

delve deeper into the time series data for the slippage components estimated using the IRLS and exponential approximation methods. The plots containing these data are shown in Fig. 6. The signal obtained from the ground truth slippage component is notably noisy due to the integration of multiple sensor measurements. An initial observation is that, as previously explained, IRLS does not produce any estimates during the first lap, whereas the exponential approximation begins to correct the slippage immediately. It is evident that for the slippage components β_L and β_R, the exponential method tends to overestimate positive values while zeroing out negative values. Conversely, the exponential approach provides a more accurate prediction for the α component. This discrepancy suggests that the exponential approximation may not adequately capture the near-linear characteristics of the slippage components β_L and β_R. Regarding the IRLS estimates, the second lap exhibits some episodes of overestimation, which diminishes in the third lap thanks to the incorporation of new data and improved weighting accuracy. However, the IRLS method shows limitations in generalizing high spikes in slippage values, particularly for α. This issue might arise because such high spikes occur at the margins of the value surface, where the behavior becomes more nonlinear. This limitation highlights one drawback of using polynomial regression in our approach, as a fully nonlinear solution might better address these high-value spikes.

To better evaluate the regression performance of our IRLS estimator, we recorded the slippage values from a simulation in which all terrain patches had a uniform friction coefficient of $\mu = 0.1$. With all patches sharing the same terrain characteristics, we aggregated the data to create a comprehensive dataset. After preprocessing, we applied the IRLS algorithm to fit the regressor on this dataset. The goal was to test the estimator against the ground truth values provided by Professor Focchi, which were computed without noise (as shown in Fig. 3).

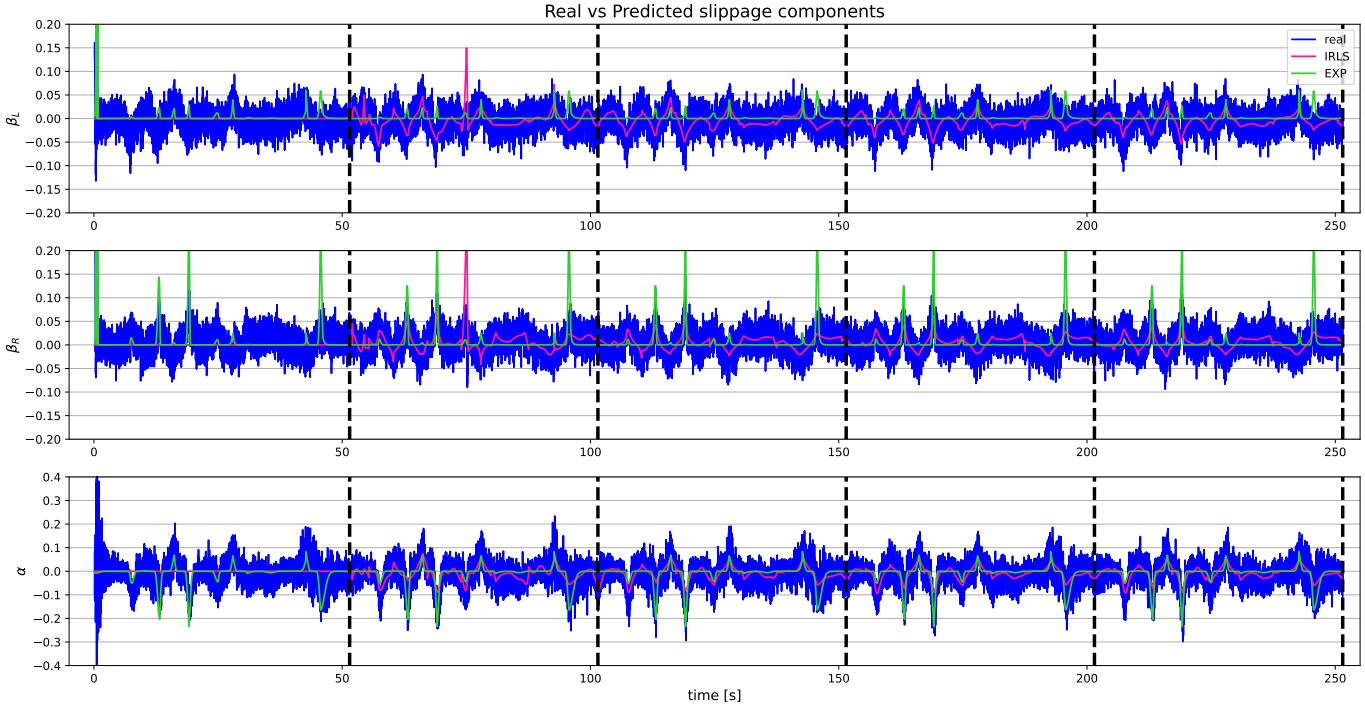

Fig. 6. Time series of calculated and predicted slippage components over laps

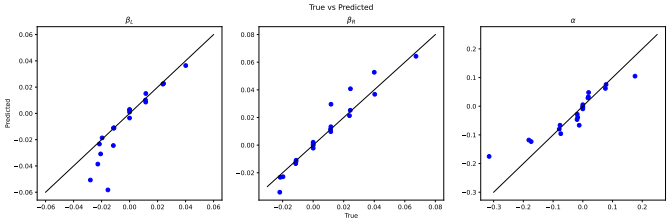

Fig. 7. True vs Predicted plot of the three slippage components obtained with IRLS

In Fig. 7, the True vs Predicted data obtain trough regression are plotted for the three slippage components. However, β_L displays an artifact on the left side of its value range. We believe this is due to the sparse availability of data in that interval, causing the limited data points to be treated as outliers by the regression model. In Table III, the r^2 values are reported for each slippage component. As observed, and as discussed earlier, β_L shows a lack of generalization.

TABLE III
REGRESSION PERFORMANCE ON $\mu = 0.1$

	β_L	β_R	α
r^2	0.54	0.90	0.82

Finally, to visually compare the surface shapes produced by our IRLS estimator and those computed using Professor Focchi's data, a 3D plot containing the three surfaces was generated and is presented in Fig. 8. As observed, the overlay for β_L in the lower range of ω_L is not accurate, showing some discrepancies. Regarding α, we can see, as previously discussed, a lack of generalization at both the upper and lower margins of the value range. Despite this characteristics, overall, the overlay is acceptable, and we consider our IRLS approach to provide a satisfactory approximation compared to the ground truth data.

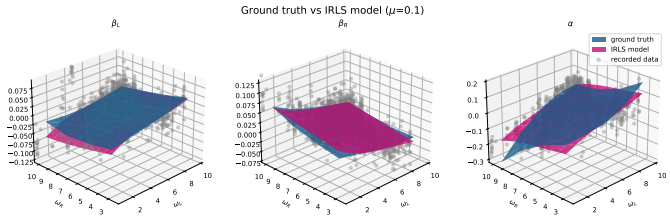

Fig. 8. True vs Predicted plot of the three slippage components obtained with IRLS

VI. CONCLUSIONS

In this work, we proposed and evaluated a novel approach for estimating and compensating slippage in real-time using a distributed Iterative Reweighted Least Squares (IRLS) algorithm. The goal was to enhance the accuracy of slippage estimation and improve trajectory tracking in unstructured terrain, characterized by varying friction coefficients. Through simulations, we compared the performance of our IRLS-based approach with two other methods: the baseline approach without slippage compensation and the exponential approximation method proposed by Professor Focchi and Professor Palopoli.

Our IRLS-based estimator demonstrated significant improvements in trajectory tracking accuracy, particularly over time, as it collected sufficient data to refine the slippage estimates. The baseline approach, which assumes the vehicle behaves like a unicycle, performed poorly due to its inability to account for terrain-induced slippage. While the exponential approximation method offered consistent performance, it lacked adaptability, as its estimation relied on precomputed constants that did not adjust based on new data.

Through our analysis, we observed that β_L and β_R exhibit near-linear behavior in the range of velocities tested, validating the motivation for using IRLS. However, the lateral slippage component α displayed non-linear characteristics, especially near the velocity extremes, where generalization proved more challenging. We addressed this by using a third-order polynomial regression, which fit the data well while remaining compatible with the IRLS framework.

The regression analysis showed that our approach achieves satisfactory accuracy when compared with the ground truth data. Although some artifacts and lack of generalization were observed in specific regions, particularly for β_L in the lower range of ω_L, the overall results were promising. The 3D surface comparisons illustrated acceptable overlays between the surfaces generated by our estimator and the ground truth.

A range of potential improvements can be explored for future iterations of this work. One significant enhancement would involve incorporating the robot's positional data into the input provided to the model. This approach could eliminate the current need for separate regressors for each distinct "patch" of the terrain map. By utilizing the robot's position, the model could infer the specific region in which the robot operates, streamlining the slippage estimation process and improving computational efficiency.

Additionally, future work could investigate more sophisticated non-linear approaches to better handle the complexities of slippage, particularly in regions where the slippage components, especially α, exhibit non-linear behavior. Potential solutions could include non-linear regression techniques, which are capable of capturing complex relationships in the data that linear models may struggle to approximate.

In conclusion, the IRLS-based estimator offers a robust solution for slippage estimation and compensation in environments with varying terrain conditions. Its ability to dynamically adapt based on real-time data, coupled with its strong performance in tracking accuracy, makes it a valuable tool for improving the control of vehicles in unstructured environments. Future research should focus on incorporating positional data into the model, exploring non-linear regression techniques for greater flexibility, and refining the estimator for improved generalization in extreme velocity ranges.

REFERENCES

[1] C. Kilic, Y. Gu, and J. N. Gross, "Proprioceptive slip detection for planetary rovers in perceptually degraded extraterrestrial environments," *arXiv preprint arXiv:2207.13629*, 2022.

[2] J. Y. Wong, *Theory of ground vehicles*. John Wiley & Sons, 2022.

[3] S. Al-Milli, L. D. Seneviratne, and K. Althoefer, "Track–terrain modelling and traversability prediction for tracked vehicles on soft terrain," *Journal of Terramechanics*, vol. 47, no. 3, pp. 151–160, 2010.

[4] A. T. Le, D. C. Rye, and H. F. Durrant-Whyte, "Estimation of track-soil interactions for autonomous tracked vehicles," in *Proceedings of International conference on robotics and automation*, vol. 2. IEEE, 1997, pp. 1388–1393.

[5] X. Zhao, E. Lu, Z. Tang, C. Luo, L. Xu, and H. Wang, "Trajectory prediction method for agricultural tracked robots based on slip parameter estimation," *Computers and Electronics in Agriculture*, vol. 222, p. 109057, 2024.

[6] S. A. A. Moosavian and A. Kalantari, "Experimental slip estimation for exact kinematics modeling and control of a tracked mobile robot," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 95–100.

[7] Z. Song, S. Hutangkabodee, Y. H. Zweiri, L. D. Seneviratne, and K. Althoefer, "Identification of soil parameters for unmanned ground vehicles track-terrain interaction dynamics," in *SICE 2004 Annual Conference*, vol. 3. IEEE, 2004, pp. 2255–2260.

[8] C. S. Burrus, "Iterative reweighted least squares," *OpenStax CNX. Available online: http://cnx. org/contents/92b90377-2b34-49e4-b26f-7fe572db78a1*, vol. 12, p. 6, 2012.

[9] Q. Yang, Z. Zhang, and M. Fu, "Distributed weighted least-squares estimation for networked systems with edge measurements," *Automatica*, vol. 120, p. 109091, 2020.

[10] M. Focchi, F. Roscia, and C. Semini, "Locosim: an open-source cross-platform robotics framework," in *Climbing and Walking Robots Conference*. Springer, 2023, pp. 395–406.