WMR Control With UKF-Based Wheel Slippage Estimation

Autonomous and Mobile Robotics

Master's Degree in Artificial Intelligence and Robotics

Luigi Gallo, Marco Francomano, Alessia Tancredi

(Prof. Giuseppe Oriolo | Advisor Francesco D'Orazio)

Academic Year 2023/2024

SAPIENZA
UNIVERSITÀ DI ROMA

Problem addressed and proposed solution

- The problem addressed in our project regards the effective control of mobile robots in the presence of **slip phenomena**.
- The slipping parameters are estimated using the **Unscented Kalman Filter** (UKF).
- The estimates are integrated in the **control schema** to allow dynamic adaptation to slip changes.

Agenda

- **Theoretical Background**
  - Integrator Backstepping
  - Unscented Kalman Filter (UKF)
- **Model and Control**
  - Kinematic Model and Slipping Parameters Model
  - Dynamic Model
  - Control Design
  - Bezier Curve
- **Simulations**
  - ODE Simulation Results
  - ROS and GAZEBO Simulation Results
- **Conclusions**

Table of Contents

Integrator Backstepping

Consider a general nonlinear system in the following form:

$$\dot{\eta} = f(\eta) + g(\eta)\xi$$
$$\dot{\xi} = u$$

where (η, ξ) is the state and u is the control input.
Suppose $\dot{\eta}$ can be **stabilized** by a smooth state feedback control law $\xi = \alpha(\eta)$ with $\alpha(0) = 0$.

Integrator Backstepping

Adding and subtracting $g(\eta)\alpha(\eta)$ on the right-hand side of $\dot{\eta}$:

$$\dot{\eta} = [f(\eta) + g(\eta)\alpha(\eta)] + g(\eta)\left[\xi - \alpha(\eta)\right]$$

Change of **variables** $z = \xi - \alpha(\eta)$ results in:

$$\begin{aligned} \dot{\eta} &= [f(\eta) + g(\eta)\alpha(\eta)] + g(\eta)z \\ \dot{z} &= u - \dot{\alpha}(\eta) \end{aligned}$$

A Lyapunov function candidate of the form $V_e(\eta, z) = V(\eta) + \frac{1}{2}z^2$ shows that the origin of the system is asymptotically stable.

Unscented Transformation (UT)

- UT basic idea: approximate **probability distribution** is easier with respect to approximate a **nonlinear function**.
- UT is invertible and is used to approximate **mean** and **covariance** with a set of (redundant, i.e. not unique) **sigma points**.
- UKF use the true state and observation models.
- UKF models the state and observation as random variables adding noise.

Unscented Kalman Filter (UKF)

- Variant of the Extended Kalman filter (EKF).
  - EKF linearizes the dynamics and observation model, it needs Jacobians computation.
  - UKF uses UT to compute sigma points propagated through the models. Typically lead to a more accurate estimate.
- The steps of the algorithm are
  - **Initialization** of state and the covariance

$$\hat{x}_0 = \bar{x}_0, \quad P_0 = \bar{P}_0$$

  - **Predict** next state and covariance.
  - **Update** previous predictions.

Prediction

- **Process model** $f(x_k, u_k)$ and **measurement model** $h(x_k)$:

$$x_{k+1} = f(x_k, u_k) + \nu_k$$
$$y_k = h(x_k) + \delta_k$$

- Where ν_k and δ_k are the **process and measurement noises** respectively, that are independent zero-mean Gaussian random variables with covariance matrices respectively given by Q and R.

Prediction (Cont'd)

- The **Unscented Kalman Filter** (UKF) algorithm utilizes $2n + 1$ sigma points, where n is the dimension of the state vector.
- **Generate** and **propagate** sigma points through process model:

$$
\begin{aligned}
X_{k-1}^{i} &= \hat{x}_{k-1}, \quad i = 0 \\
X_{k-1}^{i} &= \hat{x}_{k-1} + \sqrt{(n+\lambda)P_{k-1}}, \quad i = 1, \dots, n \\
X_{k-1}^{i} &= \hat{x}_{k-1} - \sqrt{(n+\lambda)P_{k-1}}, \quad i = n+1, \dots, 2n
\end{aligned}
$$

- $\lambda = \alpha^2(n + \kappa) - n$ where α determines the **spread** of the sigma points around the mean and κ is a secondary **scaling** parameter.

Prediction (Cont'd)

- **Estimate** predicted state and covariance:

$$
\begin{aligned}
X^i_{k|k-1} &= f(X^i_{k-1}, u_{k-1}), \quad i = 0, \ldots, 2n \\
\hat{x}_k &= \sum_{i=0}^{2n} W_i^{(m)} X^i_{k|k-1} \\
P_k &= \sum_{i=0}^{2n} W_i^{(c)} \left[X^i_{k|k-1} - \hat{x}_k\right] \left[X^i_{k|k-1} - \hat{x}_k\right]^T + Q
\end{aligned}
$$

- $W_i^{(m)}$ and $W_i^{(c)}$ are the **weights** for computing the **mean** and **covariance**, respectively.

Prediction (Cont'd)

- The weights are obtained with the following computation:

$$W_0^{(m)} = \frac{\lambda}{n+\lambda}, \quad i = 0$$

$$W_0^{(c)} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta), \quad i = 0$$

$$W_i^{(m)} = W_i^{(c)} = \frac{1}{2(n+\lambda)}, \quad i = 1, 2, \ldots, n$$

- where β is a non-negative weighting term.

Update (Correction)

- **Propagate** sigma points through measurement model:

$$Y^i_{k|k-1} = h(X^i_{k-1}), \quad i = 0, \ldots, 2n$$

- Compute the new mean and covariance based on the estimate of measures:

$$\hat{y}_k = \sum_{i=0}^{2n} W_i^{(m)} Y^i_{k|k-1}$$

$$P_{\hat{y}_k\hat{y}_k} = \sum_{i=0}^{2n} W_i^{(c)} \left(Y^i_{k|k-1} - \hat{y}_k\right)\left(Y^i_{k|k-1} - \hat{y}_k\right)^T + R$$

Update (Correction)

- Compute **cross-covariance** and **Kalman gain**:

$$P_{\hat{x}_k \hat{y}_k} = \sum_{i=0}^{2n} W_i^{(c)} \left(X^i_{k|k-1} - \hat{x}_k \right) \left(Y^i_{k|k-1} - \hat{y}_k \right)^T$$

$$K_k = P_{\hat{x}_k \hat{y}_k} P^{-1}_{\hat{y}_k \hat{y}_k}$$

- **Update** the **state estimate** at time step k to obtain $\hat{x}_k$.
- **Update** the **state covariance matrix** at time step k to obtain P_k.

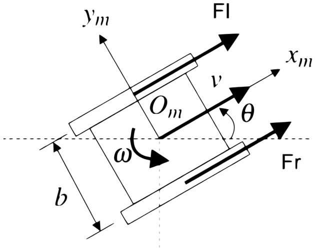
$$\hat{x}_k = \hat{x}_k + K_k \left(y_k - \hat{y}_k \right)$$

$$P_k = P_k - K_k P_{\hat{y}_k \hat{y}_k} K_k^T$$

Table of Contents

Kinematic Model

- **Differential drive** robot with slip model.
- $q = (x, y, \theta)$ is the robot **configuration**, $\xi = (\omega_L, \omega_R)$ are angular **velocities** of the left and right tracks, i_L and i_R the longitudinal **slip** ratio of the left and right wheels.



$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \frac{1}{2b} \begin{bmatrix} br(1-i_L)\cos\theta & br(1-i_R)\cos\theta \\ br(1-i_L)\sin\theta & br(1-i_R)\sin\theta \\ -2r(1-i_L) & 2r(1-i_R) \end{bmatrix} \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix} \quad \Leftrightarrow \quad \dot{q} = S(q)\xi$$

Slip Params Model

The unknown **slip parameters** can be defined as:

$$i_L = \frac{r\omega_L - v_L}{r\omega_L}, \quad i_R = \frac{r\omega_R - v_R}{r\omega_R}, \quad 0 \leq i_L, i_R < 1$$

where v_L and v_R are the **tangential** velocities of the left and right wheels with respect to the terrain.

Dynamic Model

The **dynamic model** is given by

$$\bar{M}\dot{\xi} = \bar{B}(q)\tau$$

where $\tau = (\tau_L, \tau_R)$ are the **generalized forces** on the left and right wheel, and $\bar{M}$ and $\bar{B}$ are, respectively

$$\bar{M} = S^T(q)MS(q), \quad \bar{B}(q) = S^T(q)B(q)$$

with the matrices M and $B(q)$ defined as

$$M = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix}, \quad B(q) = \begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ -b/2 & b/2 \end{bmatrix}$$

Control Design

The controller takes in input two **reference velocities** v_r and ω_r and produces **force control** τ. Defining an **auxiliary** control **input** u, thanks to the following **transformation**

$$\tau = \bar{B}(q)^{-1}\bar{M}u$$

replacing τ in the dynamic model, we **linearize** the dynamic model

$$\dot{q} = S(q)\xi$$
$$\dot{\xi} = u$$

with u designed to drive the error $e_d = (\xi_d - \xi)$ to zero, and defined as

$$u = \dot{\xi}_d + \begin{bmatrix} k_4 & 0 \\ 0 & k_5 \end{bmatrix} (\xi_d - \xi), \quad k_4, k_5 > 0$$

Obtaining Desired Velocity

- Starting from the relation between v, ω and the desired wheel velocity considering slipping estimation

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{r}{2b} \begin{bmatrix} b\omega_{Ld}(1 - i_{Le}) + \omega_{Rd}(1 - i_{Re}) \\ -2\omega_{Ld}(1 - i_{Le}) + 2\omega_{Rd}(1 - i_{Re}) \end{bmatrix} = T \begin{bmatrix} \omega_{Ld} \\ \omega_{Rd} \end{bmatrix}$$

- **Invert** this relation to obtain ξ_d:

$$\begin{bmatrix} \omega_{Ld} \\ \omega_{Rd} \end{bmatrix} = \frac{1}{2r} \begin{bmatrix} 2(1 - i_{Le})^{-1} & -b(1 - i_{Le})^{-1} \\ 2(1 - i_{Re})^{-1} & b(1 - i_{Re})^{-1} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

- With i_{Le}, i_{Re} being slipping parameters estimated using a **nonlinear filtering algorithm** (UKF).

Auxiliary Velocity for Tracking Problem

- Compute auxiliary velocity η to solve the tracking problem for the kinematic model:

$$\dot{q} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = S_a(q)\eta$$

- Find velocity $\eta = (v, \omega)$ such that $\lim_{t\to\infty}(q_r - q) = 0$
- Reference trajectory $q_r = (x_r, y_r, \theta_r)$, with **constant** reference $\eta_r = (v_r, \omega_r)$, generated by:

$$\dot{q}_r = S_a(q_r)\eta_r$$

Trajectory Tracking for Unicycle Model

- Control the linear and angular velocities to **minimize** the task **error** e_d.
- Define error $e = (e_1, e_2, e_3)$ in a rotated frame:

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix}$$

- With error dynamics:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} \omega e_2 + v_r \cos e_3 - v \\ -\omega e_1 + v_r \sin e_3 \\ \omega_r - \omega \end{bmatrix}$$

Input Transformation

- Use **invertible** input transformation to have $e = (0, 0, 0)$ as **unforced** equilibrium

$$u_1 = v_d \cos e_3 - v$$
$$u_2 = \omega_d - \omega$$

- **Substituting** u_1 and u_2 into the error dynamics:

$$\dot{e}_1 = \omega_d e_2 + u_1 - e_2 u_2$$
$$\dot{e}_2 = -\omega_d e_1 + v_d \sin e_3 + e_1 u_2$$
$$\dot{e}_3 = u_2$$

Nonlinear Feedback Controller for Local (Asymptotic) Stability

We finally employ the following **controller** to compute the velocity **inputs**

$$
\begin{aligned}
v &= v_r \cos e_3 - k_3 e_3 \omega + k_1 e_1 \\
\omega &= \omega_r + \frac{v_r}{2}\left[k_3(e_2 + k_3 e_3) + \frac{1}{k_2}\sin e_3\right], \quad k_i > 0
\end{aligned}
$$

Bézier Curves

Employed Bézier curves to obtain smoother velocity profiles at the beginning of the motion and, consequently, to avoid a large peak in the control input.

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3 \quad 0 \le t \le 1$$

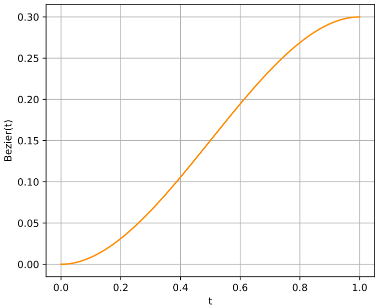


Table of Contents

Simulation settings

The **simulations** were performed both in a simpler environment using **ODE**, and in the robotics simulator **Gazebo** with **ROS** for simulation on **TIAGo**.

- Reference trajectories:
  - **Circular** ($v_r = 0.3$ m/s, $\omega_r = 0.1$ rad/s)
  - **Linear** ($v_r = 0.3$ m/s)
- The **initial state** for the filter is chosen as:
  $x_0 = (1, 1, \pi/4, 0, 0, 0, 0)^T + 0.01(1, 1, 1, 1, 1, 1, 1)^T\epsilon$
  - ϵ is a random value drawn from a normal distribution with mean zero and standard deviation one.
- **Bézier** profiles for both linear and angular velocities.

ODE simulation settings

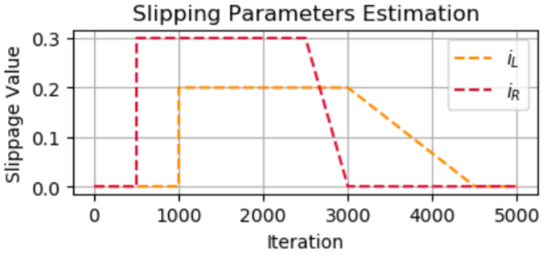
- To **simulate** the behavior of the real robot, a **fourth-order Runge-Kutta** method for numerical integration is used.
- Inclusion of the **real slipping parameters** as input to the process model.
  - Simulate the robot's motion with manually set slipping characteristics.
  - Evaluate the performance of the UKF in parameter estimation.
- **Controller** gains: $k_1 = 1$, $k_2 = 15$, $k_3 = 1$, $k_4 = 10$ and $k_5 = 10$.
- **Filter** parameters:
  - $\alpha = 0.001$, $\beta = 2$ and $\kappa = 0$
  - Covariance matrices:
    $Q_k = I_d \cdot 10^{-5}$ and $R_k = I_d \cdot 10^{-5}$.

ODE simulation settings

- The **slipping parameters** i_L and i_R are taken as:

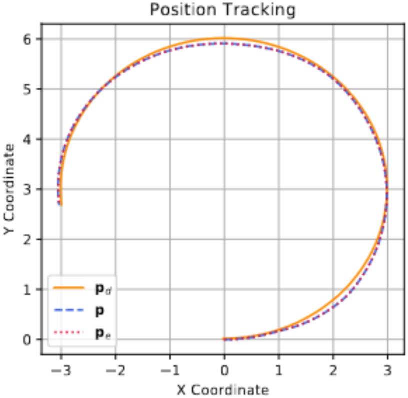
$$i_L(t) = \begin{cases} 0.0 & \text{if } 0 \le t < 20 \text{ s} \\ 0.2 & \text{if } 20 \le t < 60 \text{ s} \\ 0.2 - \frac{0.2}{30}(t-60) & \text{if } 60 \le t < 90 \text{ s} \\ 0.0 & \text{if } 90 \le t \le 100 \text{ s} \end{cases}$$

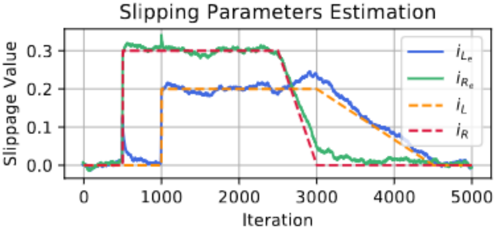
$$i_R(t) = \begin{cases} 0.0 & \text{if } 0 \le t < 10 \text{ s} \\ 0.3 & \text{if } 10 \le t < 50 \text{ s} \\ 0.3 - \frac{0.3}{10}(t-50) & \text{if } 50 \le t < 60 \text{ s} \\ 0.0 & \text{if } 60 \le t \le 100 \text{ s} \end{cases}$$



ODE simulation results

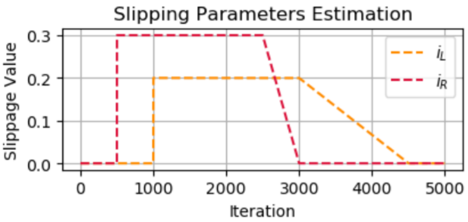
- Circular trajectory and trapezoidal profiles for i_L and i_R.

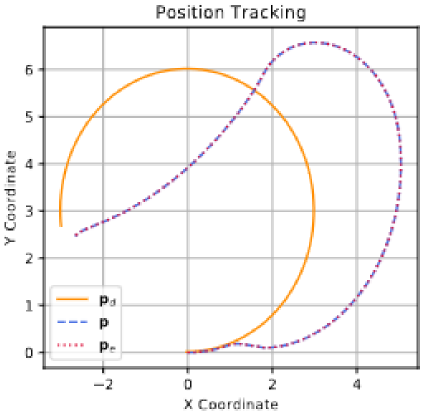




ODE simulation results: no slipping parameters

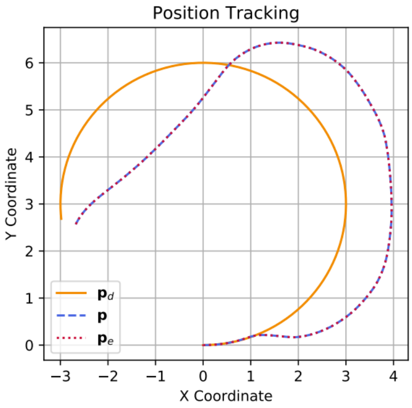
- Trajectory of the robot **without** estimating the slipping parameters.

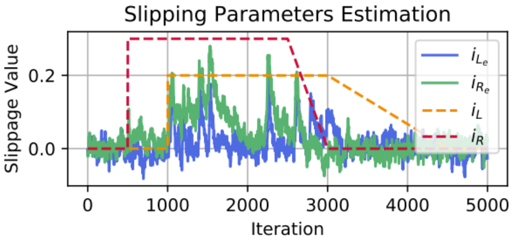




Impact of Q in Unscented Kalman Filter (UKF)

- Higher Q degrades trajectory tracking and slipping parameter accuracy.





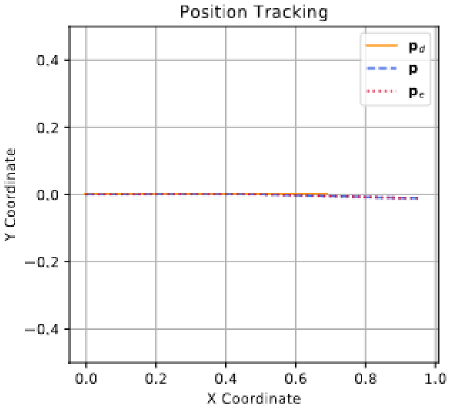
Gazebo simulation settings

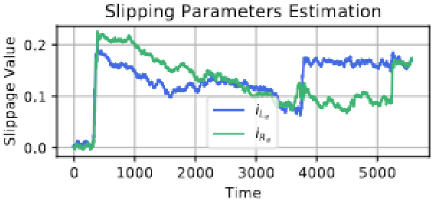
We also tried to test the implemented solution in combination with **Gazebo** and **ROS**.

- Additional modifiable factor to evaluate the effect of different terrains:
  - **friction** coefficients μ.
- The ground truth for the slipping parameters is **unknown**.
- **Controller** gains: $k_1 = 1$, $k_2 = 8$, $k_3 = 2$, $k_4 = 3$, and $k_5 = 3$.
- **Filter** parameters:
  - $\alpha = 0.001$, $\beta = 2$ and $\kappa = 0$.
  - Covariance matrices:
    $Q = I_d \cdot 10^{-5}$ and $R = I_d \cdot 10^{-4}$.

Gazebo simulation results: Linear Trajectory

- Linear trajectory in an **empty world**, $v_r = 0.3$ m/s.



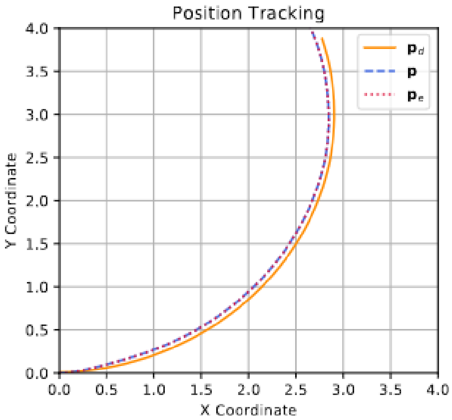


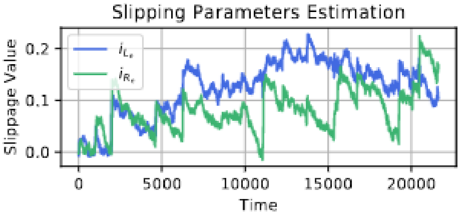
YouTube Links: Linear Trajectory in Gazebo

Linear Trajectory

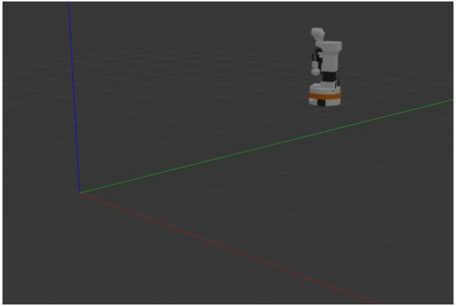
Gazebo simulation results: Circular Trajectory

- Circular trajectory in an **empty world**, $v_r = 0.3$ m/s, $\omega_r = 0.1$ rad/s.





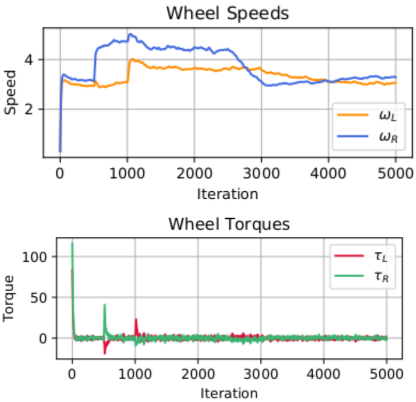
YouTube Links: Circular Trajectory in Gazebo

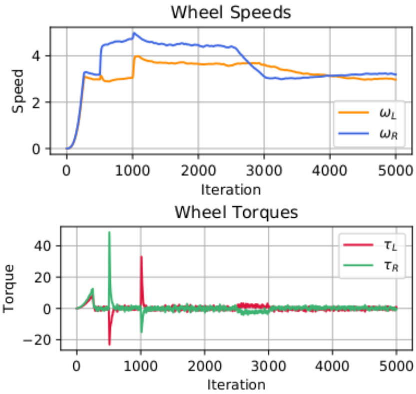


Circular Trajectory

Gazebo simulation results: effect of Bezier curve

- Impact of using Bezier curves for velocities profiles.





YouTube Links: Effect of Bezier curves

No use of Bezier curve

Effect of Bezier curve

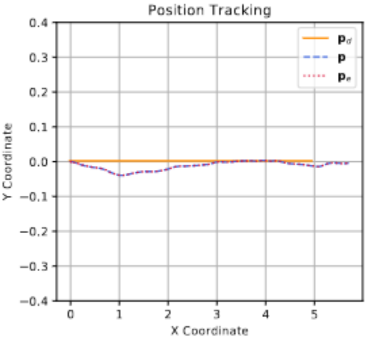
Reduced friction world

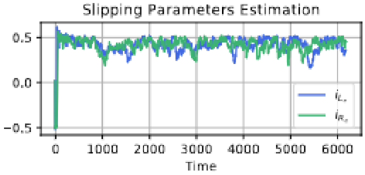
With the aim of evaluating the effect of different terrains on the slipping parameters, we performed the **simulation** in a **reduced friction** world.

- Friction parameters **reduced** from $\mu = 100$ and $\mu_2 = 50$ to $\mu = 2$ and $\mu_2 = 1$.
- **Linear** trajectory
  - Linear velocity increased from $v_r = 0.3$ m/s in the empty world to $v_r = 0.8$ m/s .
- **Circular** trajectory
  - Linear velocity increased from $v_r = 0.3$ m/s in the empty world to $v_r = 0.8$ m/s .
  - Angular velocity increased from $\omega_r = 0.1$ rad/s in the empty world to $\omega_r = 0.2$ rad/s.
- The initial Bézier curve on the velocity profiles was removed to further accentuate slipping from a standstill.

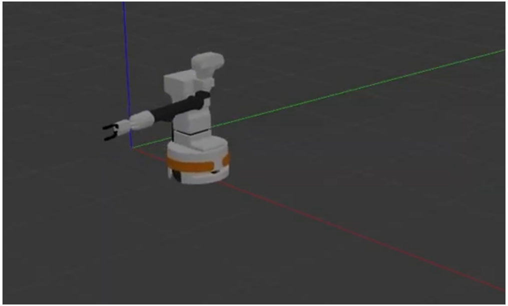
Reduced friction world: Linear Trajectory

Linear trajectory in a reduced friction world with higher linear velocity:





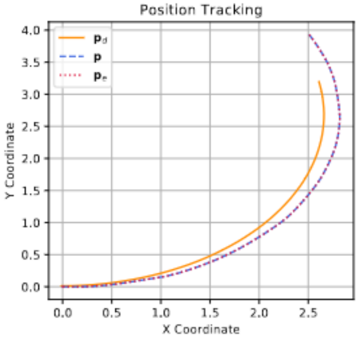
YouTube Links: Linear Trajectory in modified World

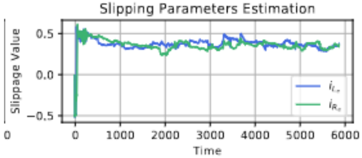


Linear Trajectory

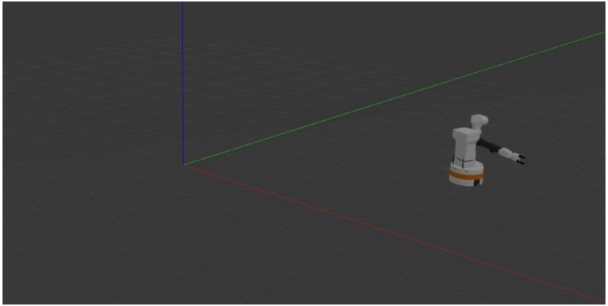
Reduced friction world: Circular Trajectory

Circular trajectory in a reduced friction world with higher linear and angular velocities:





YouTube Links: Circular Trajectory in modified World



Circular Trajectory

Table of Contents

Conclusions

In this project, we have **demonstrated** the use of the **Unscented Kalman Filter** (UKF) to **enhance** the control and navigation of mobile robots dealing with **wheel slippage**.

- By incorporating the UKF into our control strategy we conducted the main simulations in **ODE**, which gave satisfactory results both on trajectory tracking and on the estimation of the slipping parameters.
- To gain a more comprehensive view, we also tested the approach in **ROS** with **Gazebo** to try different terrains. These additional tests showed that while our approach is promising, the results can be improved by future works.

References

Juliano G. Iossaqui.
Slip estimation using the unscented kalman filter for the tracking control of mobile robots.
2011.

Hassan K Khalil.
Nonlinear Systems.
Prentice Hall, 2002.