Autonomous Vehicle State Estimation and Mapping Using Takagi–Sugeno Modeling Approach

Shivam Chaubey and Vicenç Puig

Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens i Artigas, 4-6, 08028 Barcelona, Spain; shivam.chaubey1006@gmail.com
* Correspondence: vicenc.puig@upc.edu

Abstract: This paper proposes an optimal approach for state estimation based on the Takagi–Sugeno (TS) Kalman filter using measurement sensors and rough pose obtained from LIDAR scan endpoints matching. To obtain stable and optimal TS Kalman gain for estimator design, a linear matrix inequality (LMI) is optimized which is constructed from Lyapunov stability criteria and dual linear quadratic regulator (LQR). The technique utilizes a Takagi–Sugeno (TS) representation of the system, which allows modeling the complex nonlinear dynamics in such a way that linearization is not required for the estimator or controller design. In addition, the TS fuzzy representation is exploited to obtain a real-time Kalman gain, avoiding the expensive optimization of LMIs at every step. The estimation schema is integrated with a nonlinear model-predictive control (NMPC) that is in charge of controlling the vehicle. For the demonstration, the approach is tested in the simulation, and for practical validity, a small-scale autonomous car is used.

Keywords: simultaneous localization and mapping; linear matrix inequality; Takagi–Sugeno; linear quadratic regulator; nonlinear model-predictive control; Kalman filter

1. Introduction

To solve the issue of traffic accidents and congestion, autonomous vehicles provide a promising solution [1]. Research work on this technology has immensely progressed from perception algorithms to vehicle control algorithms. The perception stack exploits the sensor measurements to provide the vehicle state as well as environmental information. The correction of these measurements is crucial to the safety of the vehicle when navigating within the environment. On the other hand, the planner and the controller use this information to plan the motion and achieve the desired trajectory while maintaining some objectives such as speed, time to reach the destination, and comfort. For such reasons, the measurement filtering and the obtaining of unmeasured states are paramount (e.g., lateral velocity in many cases cannot be measured directly, which is important for lateral control). Advanced controllers, such as, e.g., NMPC, require a full state observation, which can be achieved using a state estimator in the loop for the full-body control.

The state estimation of an autonomous vehicle involves an algorithm that fuses the raw information provided by the sensors, which is affected by noise, and the system model, which is affected by some uncertainty represented as a disturbance. The estimation algorithm considers the measurement model, vehicle model, sensor uncertainty, and model perturbation to estimate the correct states. The standard approach is based on some form of Kalman filter, originally developed by R.E Kalman [2]. A Kalman filter provides an optimal estimate in the least-square sense, assuming that the model perfectly matches with the real system such that the noises/disturbances are Gaussian and their covariances are known. The first assumption is not always true, especially in nonlinear systems with complex dynamics. A variant named extended Kalman filter (EKF) for the nonlinear systems has been developed, which provides the solution by linearizing the system and
the measurement model around the current states. The linearization is based on a Taylor series expansion, rejecting higher-order terms. The linearization process does not preserve the random variable distributions of the states and the measurements as Gaussian, which means that the optimality condition is not guaranteed. For the systems with a complex nonlinear model, such as a vehicle, the assumption of a small range of operation for a linear model acquired by linearizing the nonlinear model is not valid. For all these reasons, new methods for extending Kalman filters to nonlinear systems are required.

The TS framework offers a systematic approach to obtain a multi-model system from the original mathematical model of the system, see, e.g., for more detail, [3]. The main feature of this approach is that it allows representing the model of the nonlinear system as a set of linear system models and the overall TS model of the system is achieved by the polytopic combination of these linear system models. This allows extending the LMI design procedures for control and estimation to the nonlinear systems represented in polytopic TS form. Moreover, there exists a systematic procedure for generating the TS model for the nonlinear model and approximating it in a polytopic way [4].

In this paper, the state estimation of an autonomous vehicle is developed using an approach alternative to the use of EKF. Moreover, this approach will be combined with a LIDAR scan matching algorithm that provides a rough pose estimation. For LIDAR scan matching, the real-time correlatable scan matching algorithm is being exploited instead of iterative closest point (ICP) [5], which has a drawback of expensive search for point correspondences in every iteration. As LIDAR scan matching is not within the scope of this paper, more details can be found in [6,7]. The methods such as Gmapping [8], which generally require a large number of particles to obtain good results, lead to higher computational complexity compared with our approach. In comparison with the methods such as LOAM [9], our method fuses the LIDAR and other sensor information with the accurate vehicle model prediction for precise localization. The proposed method can be extended to any degree of freedom (DoF) system if the model is known and additional sensors can be integrated. In our approach, dynamic states, such as longitudinal velocity and angular velocity, will be directly measured from the sensors. Then, the rough pose obtained from the LIDAR scan matching and direct measurements from sensors will be corrected using the TS Kalman filter design. A similar work [10] was carried out to improve the approximation error by utilizing the Takagi–Sugeno model; however, this approach considers a limited and known number of landmarks and neither ensures optimality. Another work [11] utilized a linear parameter varying (LPV) system, which is analogous to TS and has shown good performance in the simulation with different applications.

To summarize, the contributions of our work are as follows:

- A TS Kalman filter is developed which does not need linearization at each step, as in the case of EKF. In addition, the estimator provides a stable and optimal solution.
- The design of the TS Kalman filter is solved offline using LMI while online (in real-time), only the interpolation of the TS subsystems gains is required.
- A case study of a small-scale autonomous vehicle is presented to obtain the full state by correctly the raw measurement obtained from sensors and LIDAR scan end-point matching.
- The whole framework, including NMPC, is validated on a small computation board Odroid XU4. The low computational requirements of the framework, especially SLAM, make it easy to deploy on small robots.

The structure of the paper is the following: Section 2 describes the proposed approach and presents the autonomous vehicle considered as a case study. Section 4 presents the proposed TS Kalman filter for the state estimation. Section 5 presents the simulation and experimental results using the considered case study. Finally, Section 6 draws the main conclusions of the paper and proposes future research paths.
2. Proposed Approach

Figure 1 presents the overall architecture and integration of different modules for the autonomy. The proposed pipeline also includes an NMPC controller. Descriptions of the states $x = [v_x, v_y, \omega, X, Y, \theta]^T$ and control $u = [D, \delta]^T$ symbol are listed in Table 1. The NMPC controller uses the error model of the system so the estimated states $[\hat{X}, \hat{Y}, \hat{\theta}]^T$ are converted to the errors states $[\hat{\theta}_e, \hat{s}, \hat{y}_e]^T$ using the offline planner; details can be found in [12]. For the state measurements, an inertial measurement unit (IMU), a motor encoder, and scan end-point matching modules are utilized. The mapping module is dedicated to map the environment using the scanned points and estimated pose of the vehicle. The previous map (occupancy grid) is queried by the scan-matching module to roughly estimate the relative position of the current scan. Finally, the estimator module is responsible for correcting the uncertainty in the measurements and scan matching, as well as obtaining the hidden state $v_y$.

Table 1. The list of symbols used in this article.

| Symbol | Description |
|--------|-------------|
| $v_x$  | Longitudinal velocity of vehicle in center of gravity (CoG) frame (C), see Figure 2. |
| $v_y$  | Lateral velocity of the vehicle in CoG frame (C). |
| $\omega$ | Angular velocity of the vehicle in CoG frame (C). |
| $X$    | Global position of the vehicle in $X$-axis frame (O). |
| $Y$    | Global position of the vehicle in $Y$-axis frame (O). |
| $\theta$ | Orientation of the vehicle with respect to the $x$-axis of the frame (O). |
| $D$    | Duty Cycle of motor, normalized between $[0,1]$. |
| $\delta$ | Steering angle. |

Figure 1. The architecture of the software modules, including information flow, and a conceptual overview of the interconnections. Software framework also includes low-level sensor measurement acquisition units.

To generalize the SLAM algorithm, i.e., free from the number of landmarks and environment, the rough pose estimation is based on LIDAR scan end-point matching by exploiting the Gauss–Newton optimization process. This matching approach allows estimating the pose without any prior information of the environment or landmarks. The scan end-points matching is similar to the work carried out in the paper [6], which is
presented in Section 3. In this development, LIDAR’s scan end-points are exploited for observing the environment. Instead, a camera can also be used for the rough localization using either feature matching technique [13] or deep learning technique [14]. Then, to correct the rough pose from the scan matching and acquired sensor measurements, a TS modeling approach is considered to design an optimal and stable Kalman estimator. The whole design procedure consists of the following steps:

1. Derive a TS model from the nonlinear model which embeds the nonlinearity term inside the system matrices.
2. Obtain the polytopic systems and derive the fuzzy representation of the TS model [4].
3. Formulate LMIs for Lyapunov stability and optimal dual LQR for all the obtained polytopes or system vertices of the fuzzy model to obtain the offline Kalman gain.
4. Exploit fuzzy interpolation technique to find the online gain for the TS Kalman filter.
5. Apply online TS Kalman gain for the state estimation.

Steps 3 to 5 are presented thoroughly in Section 4 of this paper, while the remaining steps are detailed in this section.

Figure 2. A representation of the bicycle model in 2D space for deriving the equations of motion. The vehicle schematic shows the inertial frame (O), the center of gravity (CoG) frame (C) attached to the center of gravity of the vehicle. The forces $F_{rx}$ and $F_{ry}$ on the rear wheel are the longitudinal and lateral force, respectively. The front wheel forces $F_{f_long}$ and $F_{f_lat}$ are the longitudinal and lateral forces, respectively, the forces $F_n$ and $l_r$ are the distance from CoG to the front wheel and rear wheel, respectively.

Formalizing the TS model for LMIs and fuzzy gain interpolation technique is based on the concept of fuzzy theory [4,15], which offers a systematic approach to obtain a multi-model system from the original mathematical model of the system. The main feature of this model is to express the local dynamics of each fuzzy implication by a set of linear system models, and the overall fuzzy model of the system is achieved by the fuzzy blending of these linear system models. It is proved that TS fuzzy models are universal approximators with a high degree of precision of any smooth nonlinear system [16].

2.1. Considered Autonomous Vehicle

To validate the proposed approach, a case based on a small-scale autonomous car is used. The states of the car have to be estimated using available sensors, and, further, this information will be exploited by the full-body controller.

The dynamic states, such as longitudinal velocity, can be roughly estimated using the radius and RPM of the rear wheel measured by the motor encoder. Angular velocity can be measured by an IMU sensor. The rough position of the vehicle can be obtained using the
scan end-point matching algorithm and orientation from IMU. Figure 1 shows the high-level scheme of the proposed pipeline which includes a model-predictive controller (MPC) in the loop. We do not provide further details about the MPC algorithm, as this is not the focus of this work. We assume that the control of the car and the track plan can be obtained.

As the estimator is based on a system model, the dynamics of the system are derived using the bicycle model \[17\] representation shown in Figure 2. The vehicle model includes kinematic as well as dynamic equations, represented by:

\[
\begin{align*}
\dot{v}_x &= \frac{1}{m} (F_{rx} - F_{flat \sin(\delta)} + m\dot{v}_y\omega) \\
\dot{v}_y &= \frac{1}{m} (F_{flat \cos(\delta)} + F_{ry} - m\dot{v}_x\omega) \\
\dot{\omega} &= \frac{1}{I_z} (l_f F_{flat \cos(\delta)} - l_r F_{ry}) \\
\dot{X} &= v_x \cos(\theta) - v_y \sin(\theta) \\
\dot{Y} &= v_x \sin(\theta) + v_y \cos(\theta) \\
\dot{\theta} &= \omega
\end{align*}
\]

where \(v_x, v_y, \omega\) are the longitudinal, lateral, and angular velocity of the vehicle, respectively. \(X, Y, \theta\) are the global pose of the vehicle in a fixed inertial frame. The available sensors are able to measure, directly or indirectly, \([v_x, \omega, X, Y, \theta]\) states, with some errors. The lateral velocity \(v_y\) cannot be measured and will be estimated. \(F_{rx}\) is the longitudinal force in the rear wheel which consists of force from the motor, drive-line resistance, and drag force. \(F_{flat}\) and \(F_{ry}\) are the lateral tire force in the front and rear wheel, respectively, which is obtained by simplifying the Pacejka tire model \([18]\). Longitudinal force on the front wheel is considered to be negligible since no brake nor torque is applied to it. The forces are given by

\[
\begin{align*}
F_{rx} &= (C_{m0} - C_{m1} v_x) D - C_0 v_x - C_1 - \frac{C_D A \rho}{2} v_x^2 \\
F_{flat} &= 2C_{af} \left( \delta - \arctan \left( \frac{v_y + l_f \dot{\theta}}{v_x} \right) \right) \\
F_{ry} &= -2C_{ar} \arctan \left( \frac{v_y - l_r \dot{\theta}}{v_x} \right)
\end{align*}
\]

where \(\delta\) and \(D\) are two control inputs, steering angle and duty cycle, respectively. Some of the system parameters are obtained by physical measurement, and the remaining ones by least-squares optimization. The obtained values and description of the parameters are listed in Table 2.

Table 2. Vehicle model parameters.

| Parameters | Values | Description |
|------------|--------|-------------|
| \(m\)      | 2.424 kg | Mass of the vehicle |
| \(l_f\)    | 0.1377 m | Distance from CoG to front wheel |
| \(l_r\)    | 0.1203 m | Distance from CoG to rear wheel |
| \(\rho\)    | 1.225 kg/m³ | Air density |
| \(C_{m0}\) | 9.4685 N | Motor parameter 1 |
| \(C_{m1}\) | 0.6672 kg/s | Motor parameter 2 |
| \(C_0\)    | 2.6104 kg/s | Resistive driveline parameter |
| \(C_1\)    | -0.00213 N | Static friction force |
| \(C_D A\)  | 0.466 m² | Coefficient of drag multiplied with area |
| \(C_{af}\) | 1.2354 N/rad | Front wheel cornering stiffness |
| \(C_{ar}\) | 1.4532 N/rad | Rear wheel cornering stiffness |
| \(I_z\)    | 0.02 kg·m² | Moment of inertia |
2.1.1. Construction of TS Model

The goal is to derive a TS model from the nonlinear vehicle dynamics (1) as if the response of the TS model exactly matches the response of a nonlinear system within a specified domain with the same input \( u \). To form the TS model, the varying nonlinear terms in the equations are set as fuzzy variables or scheduling variables \( (\theta_i(t)) \).

The set of fuzzy (or scheduling) variables is represented in Table 3. These are the state and control variables on which the nonlinear dynamics are dependent. These variables represent the physical limit of the vehicle, for example, the maximum and minimum longitudinal velocities or maximum and minimum steering angle that a vehicle can achieve.

| Variables \( (\theta_i(t)) \) | \( \min (\theta_i(t)) = \theta_i \) | \( \max (\theta_i(t)) = \bar{\theta}_i \) |
|--------------------------|------------------|------------------|
| \( v_x \)                | -5.0             | 5.0              |
| \( v_y \)                | -3.0             | 3.0              |
| \( \omega \)             | -1.5             | 1.5              |
| \( \theta \)             | -\pi             | \( \pi \)        |
| \( \delta \)             | -0.35            | 0.35             |

From the nonlinear model (1), the TS model (5) is obtained by embedding schedule variables inside the matrix:

\[
\begin{bmatrix}
v_x \\
v_y \\
\omega \\
x \\
y \\
\theta
\end{bmatrix}
= \begin{bmatrix}
A_{11} & A_{12} & 0 & 0 & 0 & 0 \\
A_{21} & A_{22} & 0 & 0 & 0 & 0 \\
A_{31} & 0 & 0 & 0 & 0 & 0 \\
A_{41} & A_{42} & 0 & 0 & 0 & 0 \\
A_{51} & A_{52} & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
v_x \\
v_y \\
\omega \\
x \\
y \\
\theta
\end{bmatrix}
+ \begin{bmatrix}
B_{11} & B_{12} \\
0 & B_{22} \\
0 & B_{32} \\
0 & 0 \\
0 & 0 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
\delta
\end{bmatrix}
\]

where \( A_{i,j} \) and \( B_{i,j} \) are the varying sub-elements of matrix \( A(\theta) \) and matrix \( B(\theta) \), respectively, and can be expressed as

\[
A_{11} = -\frac{1}{m}\left(C_0 + C_1 v_x + \frac{C_D Ap v_x}{2}\right), \quad A_{12} = \omega, \\
A_{21} = -\omega, \quad A_{22} = -\frac{2C_{ar}}{m \rho v_y} \arctan\left(\frac{v_y - l_\rho \dot{\theta}}{v_x}\right), \\
A_{31} = \frac{2C_{ar} l_r}{v_x I_z} \arctan\left(\frac{v_y - l_\rho \dot{\theta}}{v_x}\right), \\
A_{41} = \cos(\theta), \quad A_{42} = -\sin(\theta), \\
A_{51} = \sin(\theta), \quad A_{52} = \cos(\theta) \\
B_{11} = \frac{1}{m}(C_{m0} - C_m v_x), \\
B_{12} = \frac{2C_{af}}{m \delta} \left(\delta - \arctan\left(\frac{v_y + l_f \dot{\delta}}{v_x}\right)\right) \sin(\delta) \\
B_{22} = \frac{2C_{af}}{m \delta} \left(\delta - \arctan\left(\frac{v_y + l_f \dot{\delta}}{v_x}\right)\right) \cos(\delta), \\
B_{32} = \frac{2C_{af} l_f}{I_\delta} \left(\delta - \arctan\left(\frac{v_y + l_f \dot{\delta}}{v_x}\right)\right) \cos(\delta)
\]
The scheduling variables can be further represented as a set of membership function \( \vartheta_i \), as follows:

\[
\vartheta_i(t) = \mu_{i1}(\vartheta_i(t)) \cdot \vartheta_i(t) + \mu_{i2}(\vartheta_i(t)) \cdot \overline{\vartheta_i}, \quad i \in [1, 2, \ldots, 5]
\]  

(7)

\[\mu_{i1}(\vartheta_i(t)) + \mu_{i2}(\vartheta_i(t)) = 1,\]  

(8)

where \( \vartheta_i, \overline{\vartheta_i} \) are the minimum and maximum scheduling variables, respectively, and the \( \mu_{ij}(\vartheta_i(t)) \) are the fuzzy sets with two membership functions for each scheduling variable being in total 32 combinations. For each membership function, a system vertex or polytopic system represented by \( A_i x(t) + B_i u(t) \) can be obtained, and by applying the fuzzy rules, a nonlinear system can be modeled as:

\[
\dot{x}(t) = \sum_{i=1}^{32} h_i(\vartheta(t)) \{ A_i x(t) + B_i u(t) \}
\]  

(9)

where,

\[
\vartheta(t) = [\vartheta_1(t), \vartheta_2(t), \vartheta_3(t), \vartheta_4(t), \vartheta_5(t)]
\]  

(10)

and

\[
h_i(\vartheta(t)) = \mu_{i1}(\vartheta_1(t)) \cdot \mu_{i2}(\vartheta_2(t)) \cdot \mu_{i3}(\vartheta_3(t)) \cdot \mu_{i4}(\vartheta_4(t)) \cdot \mu_{i5}(\vartheta_5(t)), \quad j \in [1, 2]
\]  

(11)

The polytopic system representation will be used to formulate the LMIs to obtain offline gain in Section 4.1. Additionally, the fuzzy representation (9) of the TS model will also allow the computation of online gain using the interpolation technique without optimizing the LMIs at each time step.

2.1.2. Measurement Model

The rough position of the vehicle is obtained from the scan matching algorithm, and other information is directly obtained from various sensors listed in Table 4.

| States | Sensors     | Covariance \( (\sigma^2) \) \([m^2]\) |
|--------|-------------|-------------------------------------|
| \( v_x \) | Motor encoder | 0.04 |
| \( \omega \) | IMU | 0.0187 |
| \( X \) | Scan matching | 0.0225 |
| \( Y \) | Scan matching | 0.0225 |
| \( \theta \) | IMU | 0.01 |

The discretized state-space measurement model for the system is defined by \( y_k = C x_k + D u_k + E_v v_k \), where \( v_k \in \mathbb{R}^{n_v} \) is the measurement noise and \( n_v = 5 \), \( E_v \) is the distribution matrix with appropriate dimension. \( D = 0_{5 \times 2} \) is taken as there is no interaction from control input. The C matrix relates the system state to the output measurement \( y_k \).

\[
E_v = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}, \quad C = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]  

(12)
where this state variable attains this value, a bias $\epsilon$.

This approach allows to map and estimate the pose without any prior information about the
environment or landmarks. Later on, this pose and map will be cured using the TS estimator
due to corrupted matching as the LIDAR has an error in the measurement. For robust
matching is performed using Gauss–Newton optimization to obtain the relative pose.

The measurement from the LIDAR provides the environment information through
building an occupancy grid probability map using Bayes inference theory which is later on
converted to occupancy grids for detecting obstacles and free zones within the map. Using
the occupancy probability map at time $t$ and previous probability maps $t-1:0$, a
robust matching is performed using Gauss–Newton optimization to obtain the relative pose.
This approach allows to map and estimate the pose without any prior information about the
environment or landmarks. Later on, this pose and map will be cured using the TS estimator
due to corrupted matching as the LIDAR has an error in the measurement. For robust
matching between occupancy probability maps, the sub-grid-level probability is accessed
for sub-grid-level accuracy $\sigma$.

The noise vector $v_k = [\sigma_{v_x}^2, \sigma_{v_y}^2, \sigma_{\theta}^2, \sigma_{\theta}^2, \sigma_{\delta}^2]^T$ has noise of each sensor with the covariance listed in Table 4.

2.1.3. Observability Analysis

Before designing the estimator, the observability of Equation (5) needs to be analyzed
for singularity. This equation consists of $n_x = 6$ state variables and the observability matrix
defined by:

$$O = [C, CA_d, CA_d^2, \ldots, CA_d^{n-1}]^T$$ (13)

If the rank of observability is equal to $n_x$, then the system is observable. To check
the observability, Equation (5) is discretized using Euler approximation $A_d = I + A \cdot \Delta t$s.

During the analysis, at certain states, the rank of the observability matrix $O$ reaches
singularity, precisely when any of the $v_x = 0, v_y = 0$ and $\delta = 0$. To resolve this issue
whenever this state variable attains this value, a bias $\epsilon = 0.0001$ is added to this variable.

3. SLAM Algorithm

The measurement from the LIDAR provides the environment information through
building an occupancy grid probability map using Bayes inference theory which is later on
converted to occupancy grids for detecting obstacles and free zones within the map. Using
the occupancy probability map at time $t$ and previous probability maps $t-1:0$, a
robust matching is performed using Gauss–Newton optimization to obtain the relative pose.
This approach allows to map and estimate the pose without any prior information about the
environment or landmarks. Later on, this pose and map will be cured using the TS estimator
due to corrupted matching as the LIDAR has an error in the measurement. For robust
matching between occupancy probability maps, the sub-grid-level probability is accessed
for sub-grid-level accuracy [6]. The algorithm has two main steps: (a) mapping the obtained
map into sub-grid level accuracy, and (b) localization using the optimization process.

3.1. Mapping

The map used for matching is developed using an interpolation scheme to overcome
the discrete nature of the map through bilinear filtering, which allows sub-grid-level cell
accuracy for estimating occupancy probabilities and derivatives. The occupancy value
$M(P_m)$ can be approximated by linear interpolation of a scan end-point $P_m$, similar to
Figure 3, in a continuous map coordinate given as:

$$M(P_m) \approx \frac{Y - Y_0}{Y_1 - Y_0} \left( \frac{X - X_0}{X_1 - X_0} M(P_{11}) + \frac{X_1 - X}{X_1 - X_0} M(P_{01}) \right)$$
$$+ \frac{Y_1 - Y}{Y_1 - Y_0} \left( \frac{X - X_0}{X_1 - X_0} M(P_{10}) + \frac{X_1 - X}{X_1 - X_0} M(P_{00}) \right)$$ (14)

where $P_{00}, P_{01}, P_{10},$ and $P_{11}$ are the closest integer point coordinates in a grid map. The
partial derivatives along the $X$ and $Y$ axis can be approximated by:

$$\frac{\partial M}{\partial X}(P_m) \approx \frac{Y - Y_0}{Y_1 - Y_0} (M(P_{11}) - M(P_{01}))$$
$$+ \frac{Y_1 - Y}{Y_1 - Y_0} (M(P_{10}) - M(P_{00}))$$

$$\frac{\partial M}{\partial Y}(P_m) \approx \frac{X - X_0}{X_1 - X_0} (M(P_{11}) - M(P_{10}))$$
$$+ \frac{X_1 - X}{X_1 - X_0} (M(P_{01}) - M(P_{00}))$$ (15)
Figure 3. Calculation of laser end-point probability at sub-grid level by using occupancy grid probability map.

3.2. Localization

The localization algorithm is based on the matching of scan end-points shown in Figure 4 with the existing maps to find the rigid transformation \( \xi = (X, Y, \theta)^T \) that minimizes

\[
\xi^* = \text{arg min}_{\xi} \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2
\]

where \( S_i(\xi) \) is the location of scan end-points in the world coordinate frame, given by:

\[
S_i(\xi) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} s_{i,x} \\ s_{i,y} \end{bmatrix} + \begin{bmatrix} X \\ Y \end{bmatrix}
\]

where \( s_i = (s_{i,x}, s_{i,y})^T \) is the location of end-points in the LIDAR coordinate frame. The \( X, Y, \) and \( \theta \) are the pose of the LIDAR in the world coordinate frame. The function \( M(S_i(\xi)) \) provides the map value at the coordinate given by \( S_i(\xi) \). Let us suppose the robot started at initial pose \( \xi \) and control input is given to move by \( \Delta \xi \). At position \( \xi \), the map is built from the measurement, and at \( \xi + \Delta \xi \), a new map is registered. To estimate the \( \Delta \xi \), the optimizer minimizes the error according to:

\[
\sum_{i=1}^{n} [1 - M(S_i(\xi + \Delta \xi))]^2 \rightarrow 0
\]

Solving for \( \Delta \xi \), using first-order Taylor expansion of \( M(S_i(\xi + \Delta \xi)) \) and setting the partial derivative with respect to \( \xi \), yields the Gauss–Newton minimization equation:

\[
\Delta \xi = H^{-1} \sum_{i=1}^{n} \left[ \Delta M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right]^T [1 - M(S_i(\xi))]
\]

where

\[
H = \left[ \Delta M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right]^T \left[ \Delta M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right]
\]

and from Equation (17), the gradient of \( \frac{\partial S_i(\xi)}{\partial \xi} \) can be represented as

\[
\frac{\partial S_i(\xi)}{\partial \xi} = \begin{bmatrix} 1 & 0 & -\sin(\theta)s_{i,x} - \cos(\theta)s_{i,y} \\ 0 & 1 & \cos(\theta)s_{i,x} - \sin(\theta)s_{i,y} \end{bmatrix}
\]
and the map gradient \( \Delta M(S_i(\xi)) \) can be approximated from Equation (15). The obtained pose might be inaccurate due to sensor noises and errors which need to be corrected before building the map. The estimator which is discussed in the next section will be used to correct this pose. The corrected pose is applied to Equation (17) to relocate the scan end-points \( S_i \), which eventually updates the occupancy probability grid map.

![Figure 4](image-url). Translation of scan information \([r, \theta]^T\) to scan end-points \([s_{iX}, s_{iY}]^T\) in LIDAR frame of reference.

4. Takagi–Sugeno Kalman Filter Design

For the development of the proposed estimator design, first, in Section 4.1, LMIs for the TS Kalman filter design are formulated. Second, in Section 4.2, the measurement noise and system perturbation matrix are provided. Finally, the implementation of the proposed approach and its improvement are discussed in Section 4.3.

4.1. LMI Design Procedure

The following Kalman filter to obtain the \( \hat{x} \) estimation is required:

\[
\dot{\hat{x}}(t) = (A(\theta) - L(\theta)C)\hat{x} + (B(\theta) - L(\theta)D)u + L(\theta)y
\]  

(22)

where \( A(\theta), B(\theta) \) can be obtained by using TS model (5), \( L(\theta) \) is the online Kalman gain, and \( D = 0_{5 \times 2} \). The above continuous Kalman estimator is discretized for implementation on a real-time system. Now, the aim is to find the optimal \( L(\theta) \) which converges to the estimation ground truth in the presence of sensor noise and system disturbance. To obtain an optimal Kalman gain \( L(\theta) \), first, LMIs are offline optimized to obtain gain \( L^{\text{off}} \). Second, the obtained \( L^{\text{off}} \) is interpolated in real time to obtain \( L(\theta) \) by exploiting relation (9).

To formulate LMIs for the discretized system, the continuous fuzzy model Equation (9) can be discretized using Euler approximation with sampling time \( \Delta t_s \):

\[
x(k + 1) = x(k) + \sum_{i=1}^{32} h_i(\theta(t))\{A_i x(k) + B_i u(k)\} \Delta t_s
\]  

(23)

\[
= \sum_{i=1}^{32} h_i(\theta(t))\{(I + A_i \Delta t_s) x(k) + (B_i \Delta t_s) u(k)\}
\]  

(24)
The LQR optimal control objective is to find a state feedback control \( u(k) = -Kx(k) \), where \( K \) is the gain for the system matrix, that minimizes the following objective function:

\[
J = \sum_{k=0}^{\infty} [x^T(k)Qx(k) + u^T(k)Ru(k)]
\]  

(25)

Introducing a Lyapunov function for each vertex of the polytopic model:

\[
V(x(k)) = x^T(k)Px(k)
\]

(26)

that satisfies the following conditions:

\[
V(x_0) < \gamma
\]

(27)

\[
V(x(k + 1)) - V(x(k)) + x^T(k)Qx(k) + u^T(k)Ru(k) < 0
\]

(28)

Integrating the Equation (28) and substituting \( u(k) = -Kx(k) \) yields

\[
\sum_{k=0}^{\infty} [x^T(k)Qx(k) + u^T(k)Ru(k)] < V(x_0) = x_0^T P x_0 < \gamma
\]

(29)

The above equation ensures that LQR objective function is bounded by the optimal value \( \gamma \) for all the polytopes. The increment of Lyapunov function (26):

\[
\Delta V(k + 1) = x^T(k)A_d(\theta)^T P A_d(\theta) x(k) - x^T(k)Px(k)
\]

(30)

For the closed-loop system the Lyapunov function becomes

\[
\Delta V(k + 1) = x^T(k)(A_d - B_dK)^T P (A_d - B_dK) x(k) - x^T(k)Px(k)
\]

(31)

The previous inequality Equations (29) and (31) can be rewritten as follows:

\[
x_0^T P x_0 < \gamma \]

(32)

\[
(A_d - B_dK)^T P (A_d - B_dK) - P + Q + K^T R K < 0
\]

(33)

Applying duality principle \( A_d \rightarrow A_d^+ , B_d \rightarrow C_d^+ , K \rightarrow L^T \) [19] to LQR inequality Equation (33) results:

\[
(A_d^+ - C_d^+ L^T)^T P (A_d^+ - C_d^+ L^T) - P + Q + L R L^T < 0
\]

(34)

By applying some algebraic manipulations to Equations (32) and (34), some changes of variables \( Q = H^T H \), \( Y = P^{-1} \) and Schur complement lead to the following LMIs optimization:

\[
\begin{bmatrix}
\gamma I & I \\
I & Y
\end{bmatrix} > 0
\]

\[
\begin{bmatrix}
-A_d^+ Y - Y A_d - W^T C_d & Y H^T & W^T \\
-Y & 0 & 0 \\
HY & 0 & -I \\
W & 0 & 0 & -R^{-1}
\end{bmatrix} < 0
\]

(35)

The optimal gain can be found by \( L = (W Y^{-1})^T \). The solution involves optimization at each time step, due to the varying system matrices \( (A_d,B_d) \), which are computationally expensive and slow. Instead, we exploit the method developed in Section 2.1.1. Any nonlinear system can be represented in the form (24), which means the offline gain will be found at the system vertices (polytopes) of these systems and later on interpolated online using fuzzy membership grade function (11) for the TS model (5). Therefore, for the 5 scheduling variables, 32 system matrices can be obtained at the polytopes, result-
ing in 32 LMIs and offline gain. The gain obtained from solving LMIs will be called $L_i^{off} \in [1, \cdots, 32]$. All the steps for obtaining the LMIs are mentioned in Algorithm 1.

4.2. Disturbance and Noise Matrix

The disturbance of the vehicle model and sensor noise is modeled by Gaussian distribution whose mean is set to zero, and covariance is obtained experimentally. The disturbance and the noise of the model are found experimentally, and some of the sensor noise from the manufacturer’s data sheet. The scan matching covariance is obtained via a sampling-based covariance estimate. However, a second method based on the Hessian matrix can also be used [20]. The disturbance in the model is considered to be higher than the measurement noise to compensate for uncaptured dynamics during parameter estimation. The following covariances ($m^2$) for the experiment were set for the disturbance ($Q$) and noise ($R$) matrices, respectively:

$$Q = \text{diag}(0.15, 0.05, 0.15, 0.25, 0.25, 0.1)$$

$$R = \text{diag}(0.04, 0.0187, 0.0225, 0.0225, 0.01)$$

4.3. Switching Estimator Design

During the experiment phase, the scheduling variable $\theta : [-\pi, 0]$ does not yield a better approximation of the nonlinear function. This can be seen by substituting the values in element $A_{41}, A_{42}, A_{51}$, and $A_{52}$ in Equation (5) and taking only these subelements:

$$A_{4,5,1,2}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$A_{4,5,1,2}(-\pi) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad A_{4,5,1,2}(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The above polytopic systems are true considering the minimum and maximum vertices in $\cos$ function, but it is not valid for $\sin$ function. To accurately model this effect, the $\theta$ scheduling variable is chosen for each quadrant $\theta_{ij} = [0, \frac{\pi}{2}, \frac{\pi}{2}, \frac{3\pi}{2}], \quad j \in [1, \cdots, 4]$. This improvement provides the system vertex to reach all the possible values:

$$A_{4,5,1,2}(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_{4,5,1,2}\left(\frac{\pi}{2}\right) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$A_{4,5,1,2}(-\pi) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad A_{4,5,1,2}\left(-\frac{\pi}{2}\right) = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

For the implementation of this system, every four sections of the quadrant are considered and the LMIs for all the quadrants are computed offline, following the steps mentioned in Algorithm 1.

Algorithm 1: Offline Gain Optimization Algorithm

1. Define maximum and minimum scheduling variables for four sectors:
   (a) $[\theta_1, \bar{\theta}_1], [\theta_2, \bar{\theta}_2], [\theta_3, \bar{\theta}_3], [\theta_{41}, \bar{\theta}_{41}], [\bar{\theta}_5, \bar{\theta}_5]$
   (b) $[\theta_1, \bar{\theta}_1], [\theta_2, \bar{\theta}_2], [\theta_3, \bar{\theta}_3], [\theta_{42}, \bar{\theta}_{42}], [\bar{\theta}_5, \bar{\theta}_5]$
   (c) $[\theta_1, \bar{\theta}_1], [\theta_2, \bar{\theta}_2], [\theta_3, \bar{\theta}_3], [\theta_{43}, \bar{\theta}_{43}], [\bar{\theta}_5, \bar{\theta}_5]$
   (d) $[\theta_1, \bar{\theta}_1], [\theta_2, \bar{\theta}_2], [\theta_3, \bar{\theta}_3], [\theta_{44}, \bar{\theta}_{44}], [\bar{\theta}_5, \bar{\theta}_5]$
2. $A_{ij}, B_{ij} \quad i \in [1, 2, \cdots, 32], j \in [1, 2, \cdots, 4] \leftarrow$ Using Equation (5) obtain 32 polytopic systems for each scheduling variables (a)–(d);
3. $A_{dij}, B_{dij} \leftarrow$ Discretize using sampling time ($\Delta t$);
4. $L_{ij}^{off} \leftarrow$ Optimize formulated LMIs Equation (35);
The estimator gains $L_{ij}^{off}$, $j \in \{1, ..., 4\}$ are obtained for each quadrant, and then particular gains are applied according to the region in which the previous yaw estimate ($\hat{\theta}_{\text{wrap}}$) lies. For example, in Figure 5, $\theta$ lies in the first quadrant, then $L_{i1}^{off}$ gain will be weighted using the TS fuzzy interpolation function depending on the scheduling variable $[0, \pi/2]$ to obtain the current online gain. Likewise, for each quadrant, the $L_{ij}^{off}$ is interpolated for the current states using the membership function. Algorithm 2 presents the method to obtain the fuzzy gain interpolation scheme.

![Figure 5. A defined sector that is designed to capture kinematic system matrices to obtain offline gain. Later on, using the same sector approach, online interpolated gains are obtained.](image)

**Algorithm 2: Gain Interpolation Algorithm**

\[
L_{\text{interpolation}}(\theta(k), L_{ij}^{off});
\]

**Input**: Scheduling variable $\theta(k)$

**Output**: Interpolated gain $L$

1. $\bar{\theta}, \theta \leftarrow$ Load scheduling variables;
2. $\mu_{1}(k), \mu_{2}(k) \leftarrow$ Compute fuzzy sets for all the $\theta(k)$ using Equations (7) and (8);
3. $h_{i}(\theta(k)) \leftarrow$ Compute fuzzy interpolation function using Equation (11);
4. $L \leftarrow \sum_{i=1}^{32} h_{i}(\theta(k))L_{ij}^{off}$;

The algorithm for the proposed state estimation technique can be referred from Algorithm 3. The $\text{wrap}()$ function is used to discretize the $\theta$ value between $[-\pi, \pi]$, and the $\text{unwrap}()$ function is used to change the $\theta$ value to continuous value. This is required as the yaw measurement from the sensor provides a measurement between $[-\pi, \pi]$, which is discontinuous. The measurement discontinuity results in wrong estimates due to the lagging or leading effect of the measurement. In Figure 6, the wrong estimation of states is depicted. Due to the abrupt change in yaw measurement in Figure 6b, the robot position is wrongly estimated in Figure 6a. It can be seen in Figure 6b that at around 12.5 s, the estimated yaw slightly leads the measurement, due to which the estimator tries to track the measurement data, which results in the full turn of the robot at location $[-1.8, 4]$. The same phenomenon at 19 s also occurs when the estimated yaw lags behind the measurement data. Changing the discontinuity using the $\text{unwrap}()$ function provides an accurate estimation of all the states, which is shown in Figure 6c,d.
Algorithm 3: State Estimation Algorithm

\begin{verbatim}
initialization;
\[ L_{ij}^{\text{off}} \leftarrow \text{Load computed gain using Algorithm 1}; \]
y \leftarrow \text{SLAM and Sensor Measurement};
\[ \hat{x} \leftarrow [y[0], 0, y[2], y[3], y[4], y[5]]; \]
u \leftarrow [0, 0];
\end{verbatim}

\begin{verbatim}
while \text{Robot is moving} do
\end{verbatim}

\begin{verbatim}
y \leftarrow \text{Measurement};
y[5] \leftarrow \text{unwrap}(y[5]) // Transform to continuous measurement;
\[ \hat{\theta}(k) \leftarrow [\hat{x}[0], \hat{x}[1], \hat{x}[2], \hat{x}[3], \hat{x}[4], \hat{\theta}_{\text{wrap}}]; \]
\end{verbatim}

\begin{verbatim}
if \[ 0 \leq \hat{\theta}_{\text{wrap}} \leq \frac{\pi}{2} \] then
\end{verbatim}

\begin{verbatim}
// Interpolate gain lie in the region\([0, \frac{\pi}{2}]\)
\[ L \leftarrow L_{\text{interpolation}}(\hat{\theta}(k), L_{ij}^{\text{off}}) \]
\end{verbatim}

\begin{verbatim}
end
\end{verbatim}

\begin{verbatim}
if \[ \frac{\pi}{2} < \hat{\theta}_{\text{wrap}} \leq \pi \] then
\end{verbatim}

\begin{verbatim}
// Interpolate gain lie in the region\((\frac{\pi}{2}, \pi]\)
\[ L \leftarrow L_{\text{interpolation}}(\hat{\theta}(k), L_{i2}^{\text{off}}) \]
\end{verbatim}

\begin{verbatim}
end
\end{verbatim}

\begin{verbatim}
if \[ -\pi \leq \hat{\theta}_{\text{wrap}} \leq \frac{-\pi}{2} \] then
\end{verbatim}

\begin{verbatim}
// Interpolate gain lie in the region\((\pi, \frac{-\pi}{2}]\)
\[ L \leftarrow L_{\text{interpolation}}(\hat{\theta}(k), L_{i3}^{\text{off}}) \]
\end{verbatim}

\begin{verbatim}
end
\end{verbatim}

\begin{verbatim}
if \[ \frac{-\pi}{2} < \hat{\theta}_{\text{wrap}} < 0 \] then
\end{verbatim}

\begin{verbatim}
// Interpolate gain lie in the region\((\frac{-\pi}{2}, 0)\)
\[ L \leftarrow L_{\text{interpolation}}(\hat{\theta}(k), L_{i4}^{\text{off}}) \]
\end{verbatim}

\begin{verbatim}
end
\end{verbatim}

\begin{verbatim}
A_{c}, B_{c} \leftarrow \text{Compute TS continuous system matrix}
\end{verbatim}

\begin{verbatim}
\hat{x} \leftarrow \hat{x} + (A_{c} - LC)\hat{x} \cdot \Delta t_{s} + B_{c} \cdot u \cdot \Delta t_{s} + L \cdot y \cdot \Delta t_{s}
\end{verbatim}

\begin{verbatim}
u \leftarrow \text{MPC(Desired path, \hat{x}) Update robot position}
\end{verbatim}

end

\begin{figure}[h]
\centering
\begin{subfigure}[b]{0.5\linewidth}
\centering
\includegraphics[width=\textwidth]{figure6a.png}
\caption{(a)}
\end{subfigure}\hspace{1cm}
\begin{subfigure}[b]{0.5\linewidth}
\centering
\includegraphics[width=\textwidth]{figure6b.png}
\caption{(b)}
\end{subfigure}
\caption{Figure 6. Cont.}
\end{figure}
Figure 6. Estimation comparison between wrapped yaw measurement and unwrap measurement. Panels (a, b) represent the position of vehicle and yaw measurement in the global frame, respectively, both of these states are estimated using yaw measurement between \([-\pi, \pi]\). Panels (c, d) represent the position of vehicle and yaw measurement in the global frame, respectively, both of these states are estimated using continuous yaw measurement. (a) X–Y global position on yaw measurement between \([-\pi, \pi]\). (b) Yaw sensor measurement. (c) X–Y global position on yaw measurement between \([-\infty, \infty]\). (d) Yaw transformed to continuous measurement.

5. Results

To validate the estimator performance, experiments were performed in the simulator as well in a real environment. First, the estimator is tested using manual control input, and once it is proved to be working, an NMPC controller is utilized to follow a track. The vehicle is 41 cm long and 21 cm wide, and the track is 50 cm in width. The setting of the controller is tuned to complete two laps while achieving a longitudinal velocity of 0.8 m/s, keeping the vehicle inside the track and closer to the center track, shown as a dotted line in Figure 7. The objective of the validation is to (i) estimate full states correctly, including unmeasured state \(v_y\), in the simulator as well as real vehicle, and (ii) validate real-time performance. For the simulator, the NRMSE evaluation metric is used to compare the error between the simulated and estimated state, which can be computed by:

\[
\text{NRMSE} = \sqrt{\frac{\sum_{i=1}^{N} (x - \hat{x})^2}{x_{\text{max}} - x_{\text{min}}}}
\]

(42)

Figure 7. Vehicle estimated trajectory in the presence of sensor disturbance during the simulation. As per the controller policy, the vehicle tried to follow two laps of the center-line track.
5.1. Simulation

The vehicle simulator is developed using the vehicle full state dynamics obtained in Section 2.1. The perturbations and noises are injected into the vehicle model and sensor model, respectively, to simulate the real world. The perturbation and noises are kept a little higher than the actual one to ensure the estimator works even in the worst cases. By analyzing Figure 7, the estimated position is very close to the simulator state of the vehicle. The NRMSE error for all the estimated states is presented in Table 5.

Table 5. NRMSE evaluation of estimated states.

| States | \( v_x \) | \( v_y \) | \( \omega \) | \( X \) | \( Y \) | \( \theta \) |
|--------|-----------|-----------|-----------|-----|-----|-----|
| NRMSE  | 0.0781    | 0.0323    | 0.0913    | 0.0157 | 0.0175 | 0.0198 |

The rest of the estimated states are compared in Figure 8. The lateral velocity \( (v_y) \), which cannot be measured directly, has an NRMSE value of 0.0323 (Figure 8b). The errors of all the dynamic estimated states are within a certain range, which indicates that the estimator is fully working and ready to be tested on the real vehicle.

![Figure 8. Estimated states on vehicle simulator for two laps.](attachment://diagram.png)

5.2. Real Experiment

Note that the RMSE error is not used here due to a lack of ground truth measurement. The validation for this experiment is performed visually.

The estimated X–Y position for the real experiment is shown in Figure 9. Some snapshots for visual validation are shown in Figure 10 and compared to Figure 9; it can be noticed that the estimated position of the vehicle matches the real position of the vehicle in the snapshots. For properly substantiated illustration, the media attached ([https://youtu.be/Oey2ZsxlnY](https://youtu.be/Oey2ZsxlnY) (accessed on 29 March 2022)) shows the estimated states and NMPC controller performance. The media validates the performance of the estimator design.
Figure 9. Vehicle estimated trajectory in the presence of sensor disturbance during the real experiment. As per the controller policy, the vehicle tried to follow two laps of the center-line track.

Figure 10. Snapshots of the vehicle following the center line of the oval track. The vehicle motion is represented by a sequence of images clockwise from the upper-left to the bottom-left.

Additional estimated states are shown in Figure 11. The velocity obtained from the motor encoder is very noisy and inaccurate (see Figure 11a), but the TS Kalman filter can provide quite a clean estimation. Similarly, this happens with the angular velocity. The lateral velocity was not measured and is successfully estimated.

(a) Longitudinal velocity  (b) Lateral velocity
The final corrected map after completing the laps is shown in Figure 12b. Figure 12a represents the environment which includes fixtures, obstacles, and free space. There are some unoccupied cells, particularly when the LIDAR scan is hindered by the fixtures present in the environment. For this reason, the error between ground truth and map is only calculated for properly scanned areas, i.e., the fixture occlusion region is excluded (see Figure 12a). The intersection-over-union (IOU) score of 0.9388 is obtained between the ground truth and the corrected map.

5.3. Simulated vs. Real Experiment

It is worth noting that the time taken for completing two laps in the simulator and real experiments are 38 s and 32 s, respectively.

The real experiment completed the lap faster than the simulation due to strictly maintaining a longitudinal velocity of 0.8 m/s. There are discrepancies between some estimated states which are due to different control input in simulation and real experiment phase or mismatch of the simulator model from the real vehicle model. Table 6 shows the minimum and the maximum absolute differences between estimated and measured states, as well as their standard deviations for both the simulator and real platform experiment. The minimum deviations for the real experiment are similar to those of the simulator, while there is a deviation of about 10 cm for real experiment values compared with simulator values; this is because the standard deviation of simulation measurements is set higher than the real standard deviation. This can also be validated from the standard deviation values in the simulator and real experiments. Even though the performance of the estimator in the real experiment cannot be verified due to the lack of ground truth, the values in the real experiment appear to be similar to those in the simulator, suggesting that the estimator performs similarly.
Table 6. Evaluation between estimated states and measured states for the experiment in the simulator and real platform. The min and max header represent the minimum and maximum absolute difference between estimated and measured states, respectively. The std header represents the standard deviation between estimated and measured states.

| States | min [m] | max [m] | std [m] |
|--------|---------|---------|---------|
|        | Sim     | Real    | Sim     | Real    | Sim     | Real    |
| X      | 0.124   | 0.094   | 0.405   | 0.317   | 0.159   | 0.116   |
| Y      | 0.137   | 0.119   | 0.344   | 0.241   | 0.138   | 0.126   |

6. Conclusions

This paper has presented an approach for full-state estimation of an autonomous vehicle using a TS Kalman filter as well as imprecise scan matching and measurements. Further, this technique can also correct the map obtained from the noisy LIDAR scan end-points matching. The proposed approach provides a stable and optimal solution in the presence of model disturbance and measurement noises at a high update rate of 100 Hz for state estimation, and simultaneous correction of the obtained map. The update rate can go much faster than 100 Hz; the only limitation is the update rate of the sensor measurements. The computational cost of the proposed approach is fairly low if compared with EKF SLAM, which depends on the number of landmarks. The result produced motivates the usage of the TS-based state estimation and mapping in the field of autonomous vehicles and robotics.

The proposed estimation technique depends on the model of the system and the measurement model with certain disturbance and noise, respectively. This requirement is sometimes hard to fulfill, and also, for some systems, the parameters change within a certain range, for example, the tire coefficient of the vehicle. For such a kind of system, online model learning techniques need to be incorporated [21,22]. The proposed method can also be scaled to a high-DoF system if the system model is known, and extra sensors can be installed, for example, a height sensor, such as a barometer, or range sensor can be used to detect z-pose. The proposed scan matching technique can be applied to an unknown environment but the accuracy may degrade in a highly dynamic environment. For such cases, robust scan matching techniques need to be applied, such as in [23].

As future work, the proposed approach will be benchmarked against competing approaches.

Author Contributions: Conceptualization, S.C. and V.P.; methodology, S.C. and V.P.; software, S.C.; writing—original draft preparation, S.C.; writing—review and editing, V.P.; supervision, V.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been co-financed by the Spanish State Research Agency (AEI) and the European Regional Development Fund (ERFD) through the project SaCoAV (ref. MINECO PID2020-114244RB-I00), by the European Regional Development Fund of the European Union in the framework of the ERDF Operational Program of Catalonia 2014-2020 (ref. 001-P-001643 Looming Factory) and by the DGR of Generalitat de Catalunya (SAC group ref. 2017/SGR/482). The author is also supported by an Industrial PhD AGAUR grant (Ref. 2019 DI 040).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the required data is included in the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Litman, T. Autonomous Vehicle Implementation Predictions: Implications for Transport Planning. 2020. Available online: https://www.vtpi.org/avip.pdf (accessed on 29 March 2022).
2. Kalman, R.E. A New Approach to Linear Filtering and Prediction Problems. J. Basic Eng. 1960, 82, 35–45. [CrossRef]
3. Rotondo, D. *Advances in Gain-Scheduling and Fault Tolerant Control Techniques*; Springer: Berlin/Heidelberg, Germany, 2017.
4. Takagi, T.; Sugeno, M. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Syst. Man Cybern.* 1985, SMC-15, 116–132. [CrossRef]
5. Zhang, Z. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vis.* 2005, 13, 119–152. [CrossRef]
6. Kohlbrecher, S.; Meyer, J.; von Stryk, O.; Klingauf, U. A Flexible and Scalable SLAM System with Full 3D Motion Estimation. In Proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), Tokyo, Japan, 9–12 November 2011.
7. Olson, E.B. Real-time correlative scan matching. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 4387–4393. [CrossRef]
8. Grisetti, G.; Stachniss, C.; Burgard, W. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Trans. Robot.* 2007, 23, 34–46. [CrossRef]
9. Zhang, J.; Singh, S. LOAM: Lidar Odometry and Mapping in Real-time. In Proceedings of the 2014 Robotics: Science and Systems, Berkeley, CA, USA, 12–16 July 2014.
10. Pathirana, C.D. Fuzzy Kalman Filtering of the Slam Problem Using Pseudo-Linear Models with Two-Sensor Data Association. *Int. J. Syst. Signal Control. Eng. Appl.* 2008, 1, 89–100.
11. Rotondo, D.; Reppa, V.; Puig, V.; Nejjari, F. Adaptive Observer for Switching Linear Parameter-Varying (LPV) Systems. *IFAC Proc. Vol.* 2014, 47, 1471–1476. [CrossRef]
12. Alcalá, E.; Puig, V.; Quevedo, J.; Rosolia, U. Autonomous racing using Linear Parameter Varying-Model Predictive Control (LPV-MPC). *Control. Eng. Pract.* 2020, 95, 104270. doi: 10.1016/j.conengprac.2019.104270. [CrossRef]
13. Hartmann, J.; Klüssendorff, J.H.; Maechle, E. A comparison of feature descriptors for visual SLAM. In Proceedings of the 2013 European Conference on Mobile Robots, Barcelona, Spain, 25–27 September 2013; pp. 56–61. [CrossRef]
14. Chen, C.; Wang, B.; Lu, C.X.; Trigoni, N.; Markham, A. A Survey on Deep Learning for Localization and Mapping: Towards the Age of Spatial Machine Intelligence. *arXiv* 2020, arXiv:2006.12567
15. Zadeh, L.; Klir, G.; Yuan, B. Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems—Selected Papers by Lotfi A Zadeh. In *Advances in Fuzzy Systems—Applications and Theory*; World Scientific Publishing: Singapore, 1996.
16. Fantuzzi, C.; Rovatti, R. On the approximation capabilities of the homogeneous Takagi–Sugeno model. In Proceedings of the IEEE 5th International Fuzzy Systems, New Orleans, LA, USA, 11 September 1996; Volume 2, pp. 1067–1072 [CrossRef]
17. Rajmani, R. *Vehicle Dynamics and Control*; Springer: New York, NY, USA, 2012; Volume 2. doi: 10.1007/978-1-4614-1433-9. [CrossRef]
18. Pacejka, H.B.; Bakker, E. The magic formula tyre model. *Veh. Syst. Dyn.* 1992, 21, 1–18. [CrossRef]
19. Ostertag, E. *Mono- and Multivariable Control and Estimation: Linear, Quadratic and LMI Methods*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 2, pp. 239–241. [CrossRef]
20. Burguera, A.B.; Cid, Y.G.; Oliver, G. On the use of likelihood fields to perform sonar scan matching localization. *Auton. Robot.* 2009, 26, 203–222. [CrossRef]
21. Masti, D.; Bemporad, A. Learning nonlinear state–space models using autoencoders. *Automatica* 2021, 129, 109666. doi: 10.1016/j.automatica.2021.109666. [CrossRef]
22. Yu, H.S.A.; Yao, D.; Zimmer, C.; Toussaint, M.; Nguyen-Tuong, D. Active Learning in Gaussian Process State Space Model. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*; Springer: Cham, Switzerland, 2021; pp. 346–361.
23. Li, X.; Du, S.; Li, G.; Li, H. Integrate Point-Cloud Segmentation with 3D LiDAR Scan-Matching for Mobile Robot Localization and Mapping. *Sensors* 2020, 20, 237. [CrossRef] [PubMed]