A Digital Twin Enabled Runtime Analysis and Mitigation for Autonomous Robots Under Uncertainties

Jalil Boudjadar and Mirgita Frasheri
Department of Electrical and Computer Engineering, Aarhus University, Denmark

Keywords: Autonomous Mobile Robots, Digital Twins, Runtime Monitoring, Performance Analysis, Uncertainty Mitigation, Validation.

Abstract: Autonomous mobile robots are increasingly deployed in various application domains, often operating in environments with uncertain conditions. Such robots rely on the state and performance assessments at runtime to autonomously control the robot functionality. However, uncertainty can significantly impact the robot sensors and actuators making it challenging to assess the robot state and quantify its performance reliably. This paper proposes a digital twin (DT) asset for the runtime estimation and validation of state and performance for a mobile autonomous robot "Turtlebot3" (TB3) operating under uncertainties, namely Lidar sensor obstruction and unknown floor friction and density. The proposed DT setup enables real-time state synthesis post-uncertainty, so that to estimate the performance and validate it using TeSSLa monitors, and compute mitigation actions. To maintain the robot autonomy, our DT intervenes only when an uncertainty is identified. The experimental results demonstrate that our DT enables to eliminate 70% of the related uncertainty while it mostly maintains the real-time synchronization with the physical TB3 robot operating a frequency of 0.2s.

1 INTRODUCTION

Autonomous mobile robots are increasingly being adopted in various application domains, including manufacturing, housekeeping services, and transportation (Abbadi and Matousek, 2018; Zhao and Chidambareswaran, 2023; Malik et al., 2023). These robots are designed to monitor their internal state, analyze performance metrics, reason about their surroundings, and execute actions autonomously to carry out a given mission (Lewis and Ge, 2018). A critical aspect of autonomous robots operation is the decision-making process, often referred to as “control loop”, which relies on essential runtime inputs such as state estimation and performance assessment.

Uncertainty in deployment environments presents a significant challenge to state estimation, performance assessment and overall functionality, such as sensors noise, localization errors, and unexpected floor density & friction (Ramesh et al., 2022; Zhang et al., 2024). In such dynamic and unpredictable environments, sensor obstruction can result in incomplete state information while environment conditions, such as muddy floor affecting traction, can lead to unreliable speed estimation. Addressing these uncertainties and mitigating their impact are crucial to ensure reliable autonomous operation, timely and efficient corrective actions (Fontanelli et al., 2021).

Various techniques have been proposed to mitigate the aforementioned uncertainty-related challenges (Filippone et al., 2024; Stephens et al., 2024; Kok and Soh, 2020; Lee et al., 2022; Fontanelli et al., 2021), among which digital twins (DTs) have emerged as a promising solution (Kaigom and Roßmann, 2020). Further elaboration on the state of the art for uncertianty mitigation in autonomous robots is provided in Section 2.

Recently, DTs are used in autonomous robots for different purposes such as runtime monitoring (Feng et al., 2021; Boudjadar and Tomko, 2022), uncertainty mitigation (Rivera et al., 2021; Betzer et al., 2024), performance assessment (Loquercio et al., 2020), runtime control (Andalibi et al., 2021), and self-adaptation (Allamaa et al., 2022), as DTs enable to synthesize a complete state of the robot and its environment from partial state data under different conditions.

However, deploying a DT to operate on-board the robot at runtime presents significant computational challenges. In fact, the limited computational resources available on autonomous robots, mostly deployed using resources-constrained platforms such as

Boudjadar, J. and Frasheri, M.
A Digital Twin Enabled Runtime Analysis and Mitigation for Autonomous Robots Under Uncertainties.
DOI: 10.5220/0013772600003982

In *Proceedings of the 22nd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2025) - Volume 2*, pages 96-104
ISBN: 978-989-758-770-2; ISSN: 2184-2809

micro-control and RaspBerry Pi boards, can hinder the real-time execution of a DT, potentially leading to desynchronization between the DT and the physical robot. Moreover, some of the performance indicators might not be computable simply by incorporating sensor data, for which additional models and simulations are needed.

In this paper, we develop a framework for leveraging DTs in autonomous robotics, operating under uncertainty, while addressing the challenges posed by computational constraints without compromising real-time decision-making. We use the MQTT protocol to livestream the data and commands between the TB3 robot and its cloud-located DT. By enabling real-time estimation of state and performance possible, our DT framework contributes to mitigate different uncertainty cases (Lidar obstruction, unexpected floor density-friction level, extreme path slope and steepness) via a set of runtime correction actions, mainly overriding the lidar readings and control actuations. To increase robot reliability, we use TeSSLa runtime monitors (Kallwies et al., 2022) to analyze and validate the functionality, performance deviation, and uncertainty mitigation in DT before integrating it into the decision making of TB3.

The rest of the paper is structured as follows: Section 2 reviews the state of the art. In Section 3, we present the proposed methodology and requirements for runtime performance assessment and validation. Section 4 describes the DT asset, including the state estimation, performance assessment processes and validation. Experimental results regarding the performance assessment, validation and uncertainty mitigation are presented in Section 5. Finally, Section 6 concludes the paper.

2 UNCERTAINTY AND RUNTIME PERFORMANCE MONITORING

Different strategies on estimating and validating robots runtime performance under uncertainty have been explored, using, e.g., statistical models instead of deriving first principles dynamics models (Xu et al., 2022), probabilistic models and Bayesian filters (Kim et al., 2021), groups of robots to estimate their joint positions in a known map through cooperation (Schmitt et al., 2002), or extended Kalman filters for pose estimation (Hartley et al., 2020). In addition, reinforcement learning (RL) techniques (Singh et al., 2022), as well as DT-based approaches that enable run-time reconfiguration (Feng et al., 2021) have been investigated as well. RL techniques have been adopted in robotics to deal with tasks related to manipulators, trajectory tracking and path planning (Zhang and Mo, 2021). However, while simulation results show great promise, their real world application remains a challenge (Dulac-Arnold et al., 2019; Smyrnakis et al., 2020), mainly due to limited uncertainty samples needed for learning (Ahmadi and Fateh, 2016; Zhang and Mo, 2021). Furthermore, providing real-time inference is not trivial, as the system is simultaneously being subject to delays in sensory input and actuation (Dulac-Arnold et al., 2019), as well as limited memory and computational resources. Loquercio *et. al* on the other hand consider the uncertainty coming from the neural networks themselves, and use a combination of Bayesian belief networks and Monte Carlo simulations for uncertainty estimation (Loquercio et al., 2020).

On the other hand, DT-based approaches can be deployed in the cloud, where the restrictions on computational capacity do not apply. Dobaj *et al.* proposed a DevOps approach (Dobaj et al., 2022), where before replacing a service A with another service B, the latter is subjected to runtime verification to ensure B will generate reasonable output. Another approach consists in deploying control parameters estimated in simulation to the real system, such as an ECU, while considering noise and possible edge cases (Allamaa et al., 2022). DTs can also be used to monitor production systems and processes, thereby generating and deploying new strategies in order to optimize system performance (Kang et al., 2019). These strategies are to be verified and validated such that the correct actuation is applied to the system. Rivera *et al.* consider deriving an actuation from multiple outcomes of multiple DTs, while updating the reference signals used by the DTs, as the error with the physical robot increases (Rivera et al., 2021).

3 RUNTIME ANALYSIS AND VALIDATION OF PERFORMANCE

In order to deliver efficient control of the robot functionality, we need to enable reliable runtime performance assessment. This section defines the TB3 state, performance indicators, and requirements for runtime validation. It also presents the uncertainty features and how to mitigate the underlying impact.

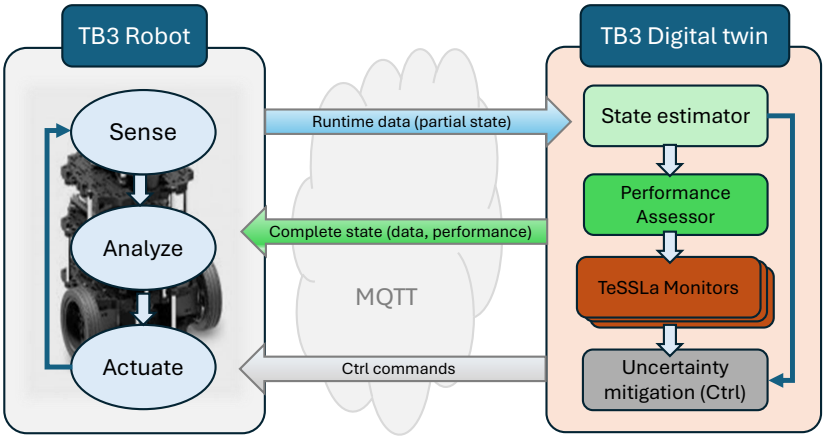

Figure 1: Proposed methodology for performance assessment and validation.

3.1 Overall Methodology

Figure 1 illustrates the methodology we propose to enable runtime quantification and validation of TB3 performance under uncertainties. TB3 publishes its state, including a set of ROS topics, to DT via the MQTT protocol. Upon receiving this data, DT synthesizes the missing or corrupted information using both the actual state and historical records. A performance assessment is then conducted on the reconstructed state.

Once Lidar data and velocity are fully synthesized, a validation check is performed using various TeSSLa monitors. The results of this runtime validation serve as a basis for uncertainty mitigation. If inconsistencies in performance or data are detected, a corrective action is issued to the robot through an MQTT packet. This mitigation may involve overriding Lidar data (correction action), allowing the robot to utilize more reliable and complete information to compute new actuations, or adjusting the robot's velocity through a direct control command (mitigation action).

3.2 State Specification

TB3 robot operates on battery power and utilizes a Light Detection and Ranging (*Lidar*) sensor to perceive its environment by performing a 360-degree scan ($a = \langle a_1, .., a_{360} \rangle$), measuring the distances at each angle a_i to detect obstacles. TB3 navigates using two wheel motors that can be actuated individually.

The Lidar data is communicated to the robot control system, deployed on a Raspberry Pi, to analyze its environment and estimate the robot state. As an output, the control loop actuates the wheels by updating the robot *expected velocity* v^e, given in terms of two attributes: linear velocity v^e_l and angular velocity v^e_a. The linear velocity is in fact the translational speed to move forward and backward, whereas the angular speed is the rotational velocity to steer left and right. Given unexpected environment conditions, the actual speed may differ from the expected one, e.g. wheels spinning due to low friction floor (mud), low density ground (sand) or highly steep paths. Similarly, we define the *actual velocity* $v^a = \langle v^a_l, v^a_a \rangle$ to be given in terms of the actual linear velocity v^a_l and angular velocity v^a_l. The distinction of actual and expected speeds enables to identify different uncertainty cases, at least those related to changes in the floor density, friction and extreme steepness. However, compared to expected speed computed by the TB3 control, estimating the actual speed requires further analysis using Lidar data and expected speed over a period of time, i.e. a sequence of data points.

Therefore, we define the robot *state* $S = \langle \langle a_1, .., a_{360} \rangle, \langle v^e_l, v^e_a \rangle, \langle v^a_l, v^a_a \rangle \rangle$ to comprise the Lidar readings, expected speed, and actual speed. The state is in fact time dependent, therefore we write S_t to be the robot runtime state at time t. In fact, the robot is not able to provide the entire set of state attributes at runtime depending on the uncertainties encountered.

The actual state is communicated between the robot and its digital twin replica mainly using the ROS topics `scan`, `odom` and `cmd_vel`.

3.3 Runtime Validation

TB3 functionality and performance are subject to different requirements to be validated for each runtime state. Namely, we consider the following requirements:

Safety Requirements. The distance to any obstacle must always be larger than a relative safety margin $\Omega()$ dependent on the actual speed.

$$\mathbf{P1} : \forall t\ i, S_t.a_i \geq \Omega(S_t.v^a)$$

Performance Requirements. This requirement aims to maintain energy-efficient navigation by reducing the difference between the actual and expected speeds, i.e the difference is always less than a threshold $\mathcal{E}$.

$$\mathbf{P2} : \forall t, S_t.v^a - S_t.v^e \leq \mathcal{E}$$

Consistency Requirements. A data consistency requirement is that the Lidar readings are always within the sensor range, i.e. 0.14m and 3.5m.

$$\mathbf{P3} : \forall t\ i,\ 0.14 \leq s_t.a_i \leq 3.5$$

Furthermore, the difference between two consecutive readings must be less than the maximum distance TB3 can travel for a scan sample duration δ.

$$\mathbf{P4} : \forall t\ i,\ S_t.a_i - S_{t+\delta}.a_i \leq |S_t.v^a * \delta|$$

In addition, consistency must be maintained between adjacent angle readings, ensuring that an obstacle is detected across multiple neighbor angles.

$$\mathbf{P5} : \forall t\ i,\ \forall i\ j,$$

$$(S_t.a_i - S_t.a_{i+1}) \leq \gamma \wedge (S_t.a_i - S_t.a_{i-1}) \leq \gamma$$

We implement each requirement as a runtime monitor in the TeSSLa tool (Kallwies et al., 2022). This will enable to ensure that TB3 behavior adheres to the specified properties.

Listing 1: TeSSLa Specification for the runtime monitor of P4.

```
in act_speed : Events[Int]
in old_Lidar : Events[List[Int]]
in cur_Lidar : Events[List[Int]]

def consistency(old_Lidar:List[Int], cur_Lidar: List[Int],
      speed: Int):
Boolean = { all(range(0, 359), i -> abs(old_Lidar[i] –
      cur_Lidar[i]) <= speed * 0.2)}
out validation := lift(consistency)(old_Lidar, cur_Lidar,
      act_speed)
```

However, in case of uncertainty, the runtime monitors cannot validate the aforementioned properties given that some of the attributes are unreliable or not available. For example, in case of lidar obstruction, TB3 cannot estimate $S_t.a_i$ in **P1**-**P3**-**P4**. To tackle this challenge, Section 4 proposes a TB3 digital twin to synthesize the runtime state attributes, assess and validate the performance.

4 DIGITAL TWINS-BASED PERFORMANCE ESTIMATION

The proposed digital twin is composed of five core modules, coordinated by a central orchestrator, and communicates with TB3 via the MQTT protocol.

Orchestrator. The DT Orchestrator is responsible for managing the runtime execution of the DT modules based on the robot's current state and performance requirements (**P1**–**P5**). It is in fact a MQTT client process that is triggered every time a new packet is received from the actual robot via the MQTT listener.

Upon receiving new inputs from TB3, through the MQTT listener, the orchestrator initiates the execution of the State Estimator Module and collects its output. If the analysis detects an uncertainty, the orchestrator invokes the Performance Assessor, Safety and Performance Validation Modules, and, if necessary, the Mitigation Module too. To maintain the robot's autonomy, our DT is designed to be non-intrusive so that it does not override the robot's decisions unless a clear uncertainty or inconsistency is identified. Listing 2 depicts the orchestrator interface to acquire TB3 inputs and publish the results to the robot depending on the identified uncertainty (either Lidar data only, or velocity commands only, both or none).

Listing 2: Orchestrator interfaces.

```
# listening to TB3 inputs
def on_message(client, userdata, msg):
    global inputs, latest_velocity,
    latest_lidar
    inputs = json.loads(msg.payload.
    decode("utf-8"))
    twist.linear.x = inputs.
    get("linear_x", 0)
    twist.angular.z = inputs.
    get("angular_z", 0)
# publish the outcomes from DT to TB3
if (uncertain=1)
client.publish(LIDAR_TOPIC, command)
elif (uncertain=2)
client.publish(VELOCITY_TOPIC, command)
elif (uncertain=3)
payload = {  "lidar": new_lidar,
"velocity": new_velocity}
client.publish(OUTPUT_TOPIC,
json.dumps(payload))
```

State Estimator. This module reconstructs the Lidar data if found to be incomplete (missing angle readings) or inconsistent (a large number of zero-values or undefined values). Using historical data and

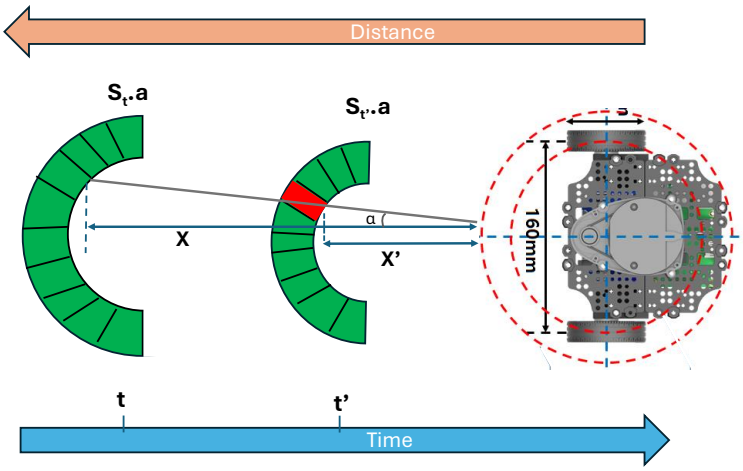

Figure 2: Estimation of a corrupted Lidar data point.

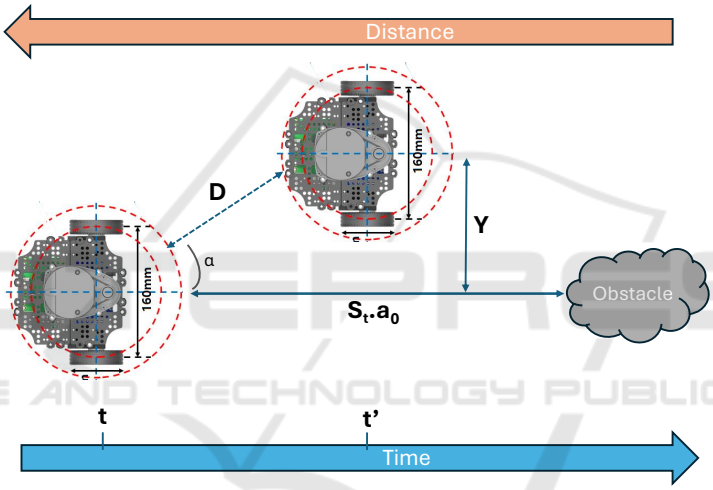

Figure 3: Assessment of the Robot linear velocity.

actual velocity, the state estimator computes how far TB3 would have moved during the sensor failure window and adjusts the corrupted angle readings accordingly. This estimation allows the module to infer plausible values for the corrupted parts of the Lidar scan, ensuring a complete and usable environmental representation.

Figure 2 illustrates a scenario of a valid Lidar data snapshot ($S_t.a$) at time t and a corrupted Lidar data point (red cell) at time t' that is ulterior to t, and depicts the different variables used later on for the estimation of the corrupted data point.

The estimation of each corrupted data point a_i, in the front hemisphere of the Lidar, at time $t' = t + \delta$ is computed using its last valid value at time t as follows:

$$S_{t'}.a_i = \frac{X'}{cos(\alpha)}$$

where

$$X' = X - \int_t^{t'} S_t.v_l^a * \Delta t$$

is the x-axis distance at time t',

$$X = cos(i) * S_t.a_i$$

is the x-axis distance at time t', $\int_t^{t'} S_t.v_l^a * \Delta t$ is the linear distance traveled by TB3 between t and t', and $\alpha \in [-90, 90]$ corresponds to the Lidar index i within the range $[0, 90] \cup [270, 360]$.

Performance Assessor. This module evaluates the robot's actual velocity by calculating its real speed (actual linear velocity v_l^a) over the most recent interval. As stated earlier, we cannot rely on the wheels encoder (e.g., SLAM) to estimate the actual speed given the deployment environment uncertainties where the robot wheels can spin due to low floor friction or density.

Using consecutive state snapshots, the performance assessor determines the distance the robot has traveled with respect to a reference obstacle in the front hemisphere and divides it by the time elapsed (typically 0.2 seconds, matching the sensor sampling rate).

Figure 3 depicts a scenario where TB3 moves from a location (state at time t) to another location (state at time t') relatively to a static obstacle. Formally, the linear velocity of TB3 at time $t' = t + \delta$ can be computed using information from both the complete state S_t and its subsequent partial state $S_{t'}$ as follows:

$$S_{t'}.v_l^a = \begin{cases} \frac{S_t.a_0 - S_{t'}.a_0}{\delta} & \text{if } S_t.v_a^e = 0 \\ \frac{S_t.a_0 - A}{\delta} & \text{Otherwise} \end{cases}$$

where $A = \frac{Y}{sin(90-\alpha)}$ and Y is the y-axis angular displacement between time point t and t' computed as follows:

$$Y = (S_t.v_l^e + D/2) * sin(\alpha) * \delta$$

In fact, if the robot is heading towards the obstacle, we can just rely on the difference between the Lidar data point readings from angle 0 ($S_t.a_0$ and $S_{t'}.a_0$). Otherwise, one has to consider the angular deviation from the trajectory to the obstacle.

Both the estimated Lidar data and linear velocity will be commuinicated to the validation module for further analysis and approval to decide potential mitigation and intervention if the actual robot state does not satisfy the safety, performance and consistency requirements.

Performance and Safety Validator. This module ensures that the robot's behavior and performance remains within safe and expected boundaries. To do so, the validator acquires first the completed state robot thanks to the state estimator and performance assessor. The validator triggers the execution of the TeSSLa runtime monitoring, implementing properties **P1**–**P5**. If all monitors return positive evaluations, the DT does not emit updates to TB3. However, if any monitor flags a violation, the orchestrator activates the Mitigation Module, using the validator outcomes as parameters, to generate corrective actions and if necessary update the Lidar data and actuate the robot control.

Uncertainty Mitigator. This module computes the velocity updates and communicates the potentially adjusted Lidar data, as corrective actions to certain uncertainties, to the physical robot via MQTT interface. Furthermore, it leverages the fully reconstructed robot state from the DT and applies the control model to determine appropriate actuation commands. Depending on the nature of the identified uncertainty, the mitigation may consist of issuing updated velocity and steering commands to prevent unsafe motion or trajectory deviation, as well as overriding the Lidar data. By isolating this function, our DT minimizes unnecessary intervention to TB3.

5 EXPERIMENTAL ANALYSIS

To analyze the effectiveness of our DT, we deploy TB3 robot in an environment where different uncertainties are synthesized artificially: low friction-density floor due to a spongy layer, and highly steep road slope. Figure 4 depicts the runtime analysis of safety property **P1**. The plot shows that whenever the distance to an obstacle is shorter than the actual safety (braking) distance $\Omega()$, i.e., possibility to collide, the runtime monitor identifies this case and a mitigation is triggered either to slow down (e.g., at t=44s) or stop the acceleration (e.g., at t=36s) of the robot linear speed. One can see that, our runtime monitors enable to timely mitigate 70% of the uncertainties occurred.

Figure 5 analyzes the response time of DT. Note that the communication time represents the major portion of the latency. We analyzed the latency using 1) Telegraf interface; 2) simple parameters parsing. One can see that our response time outperforms the one achieved using Telegraf, and is mostly within [0.143, 0.2] thus aligning with the robot sampling frequency of 0.2s. However, there are few cases where the response time can reach up to 0.25s leading TB3 to miss a timely synchronization of the DT outcomes, although we believe that an efficient scheduling of the DT tasks (Boudjadar et al., 2016) can lead to maintain a complete synchronization between the robot and its DT.

Figure 6 depicts the mitigation of uncertainties related to the actual speed either larger than the maximum expected speed or lower than the minimum expected speed. It is trivial to see that the actual speed is mostly within the acceptable range of [0, 0.22]. In fact, the mitigation is triggered whenever the actual speed is equal to the minimum or maximum expected speeds, where most of the corrections are efficient (e.g. 2.1s and 6.3s). However, due to larger mitigation latency, the corrective action impact might take place late (e.g., 12.1s and 13.8s).

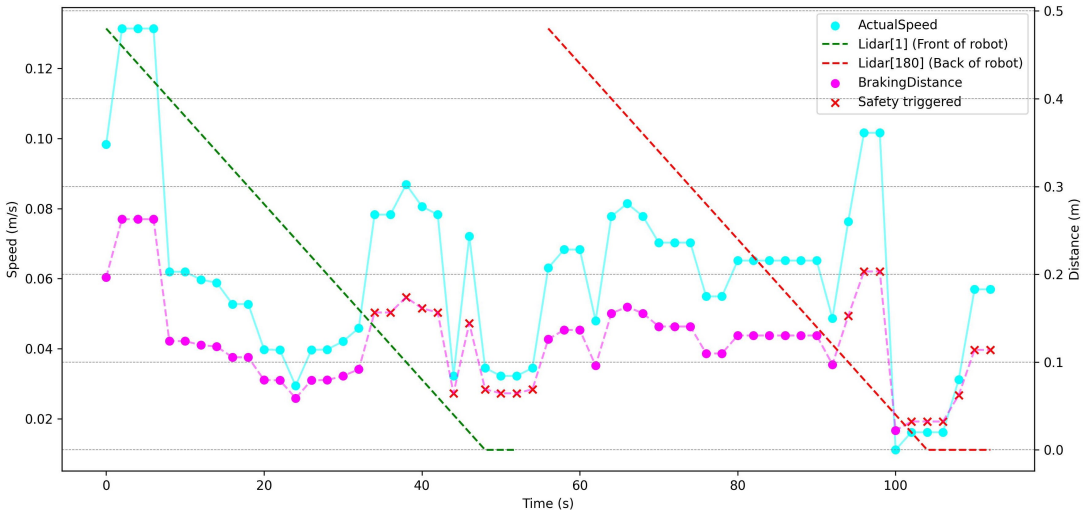

Figure 4: Runtime monitoring of the speed-related safety property.

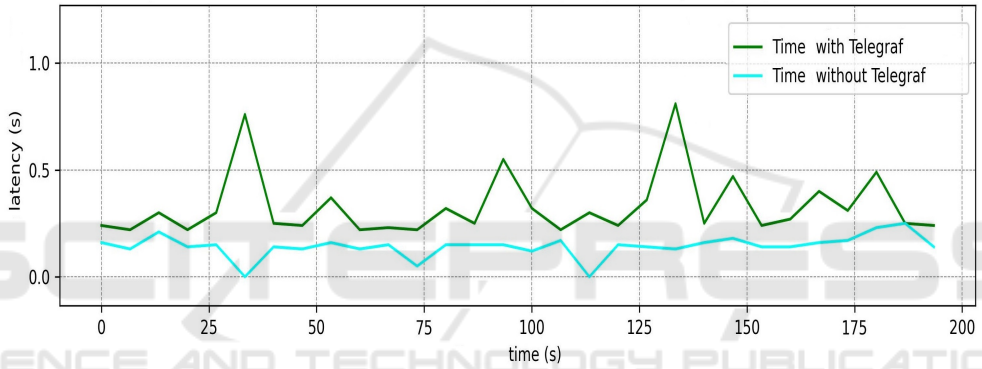

Figure 5: System latency and execution time.

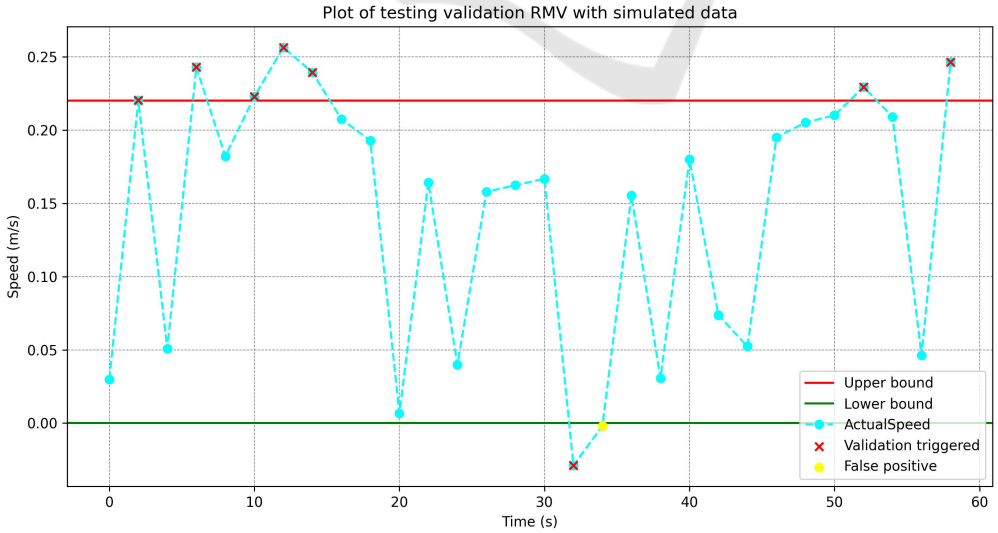

Figure 6: Runtime mitigation of the speed performance.

6 CONCLUSIONS

This paper presented a digital twin-assisted methodology to estimate runtime state and performance, identify, and mitigate uncertainties for an autonomous mobile TB3 robot operating under unknown conditions.

To mitigate the limitations of on-board processing, we used the MQTT protocol to offload computations to the cloud-hosted DT. In fact, DT reconstructs incomplete or unreliable state data, while also performing runtime performance assessment and validation. When inconsistencies are identified, thanks to a set of TeSSLa runtime monitors, corrective actions including data overrides and actuation adjustments are issued to the robot. The experimental results show that our DT can identify and mitigate 70% of relative uncertainties, while it maintains synchronization with TB3 in most of the cases.

As a future work, we plan to integrate extra safeguard monitors, optimize the DT latency and consider intelligent strategies to mitigate the uncertainties. Furthermore, additional real-world scenarios will be considered for the experimental validation.

ACKNOWLEDGEMENTS

This research was supported by the **RoboSAPIENS** project (Robotic Safe Adaptation In Unprecedented Situations), funded by the Horizon Europe 2021-2027 research and innovation programme under grant agreement No 101133807.

REFERENCES

Abbadi, A. and Matousek, R. (2018). Hybrid rule-based motion planner for mobile robot in cluttered workspace. *Soft Computing*, 22:1815–1831.

Ahmadi, S. and Fateh, M. (2016). Robust control of electrically driven robots using adaptive uncertainty estimation. *Computers & Electrical Engineering*, 56:674.

Allamaa, J., Patrinos, P., Auweraer, H., and Son, T. (2022). Sim2real for autonomous vehicle control using executable digital twin. *IFAC-Papers On Line*, 55:385–391.

Andalibi, M., Hajihosseini, M., Gheisarnejad, M., Khooban, M.-H., and Boudjadar, J. (2021). A novel method for stabilizing buck-boost converters with cpl using model prediction control. In *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*.

Betzer, J. S., Boudjadar, J., Frasheri, M., and Talasila, P. (2024). Digital twin enabled runtime verification for autonomous mobile robots under uncertainty. In *2024 28th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*.

Boudjadar, A., David, A., Kim, J. H., Larsen, K. G., Mikučionis, M., Nyman, U., and Skou, A. (2016). Statistical and exact schedulability analysis of hierarchical scheduling systems. *Science of Computer Programming*, 127:103–130.

Boudjadar, J. and Tomko, M. (2022). A digital twin setup for safety-aware optimization of a cyber-physical system. In *ICINCO*, pages 161–168.

Dobaj, J., Riel, A., Krug, T., Seidl, M., Macher, G., and Egretzberger, M. (2022). Towards digital twin-enabled devops for cps providing architecture-based service adaptation & verification at runtime. In *17th Symposium On Software Engineering For Adaptive And Self-Managing Systems*.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. (2019). Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*.

Feng, H., Gomes, C., Thule, C., Lausdahl, K., Iosifidis, A., and Larsen, P. (2021). Introduction to digital twin engineering. In *2021 Annual Modeling And Simulation Conference (ANNSIM)*, pages 1–12.

Filippone, G., García, J. A. P., Autili, M., and Pelliccione, P. (2024). Handling uncertainty in the specification of autonomous multi-robot systems through mission adaptation. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*.

Fontanelli, D., Shamsfakhr, F., Macii, D., and Palopoli, L. (2021). An uncertainty-driven and observability-based state estimator for nonholonomic robots. *IEEE Transactions on Instrumentation and Measurement*, 70:1–12.

Hartley, R., Ghaffari, M., Eustice, R., and Grizzle, J. (2020). Contact-aided invariant extended kalman filtering for robot state estimation. *The International Journal of Robotics Research*, 39:402–430.

Kaigom, E. G. and Roßmann, J. (2020). Value-driven robotic digital twins in cyber–physical applications. *IEEE Transactions on Industrial Informatics*, 17(5).

Kallwies, H., Leucker, M., Schmitz, M., Schulz, A., Thoma, D., and Weiss, A. (2022). TeSSLa – an ecosystem for runtime verification. In *Runtime Verification Conference*.

Kang, S., Chun, I., and Kim, H. (2019). Design and implementation of runtime verification framework for cyber-physical production systems. *Journal of Engineering*, 2019:2875236.

Kim, D., Park, M., and Park, Y. (2021). Probabilistic modeling and bayesian filtering for improved state estimation for soft robots. *IEEE Transactions on Robotics*, 37:1728–1741.

Kok, B. C. and Soh, H. (2020). Trust in robots: Challenges and opportunities. *Current Robotics Reports*, 1(4):297–309.

Lee, J., Feng, J., Humt, M., Müller, M. G., and Triebel, R. (2022). Trust your robots! predictive uncertainty estimation of neural networks with sparse gaussian

processes. In *Conference on Robot Learning*, pages 1168–1179. PMLR.

Lewis, F. L. and Ge, S. S. (2018). *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*. CRC Press.

Loquercio, A., Segu, M., and Scaramuzza, D. (2020). A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2).

Malik, M. F., Shahid, H., Saleem, M. H., Nazir, H., and Nouman, A. (2023). Housekeeping using multi-agent service robotics. In *International Conference on Robotics and Automation in Industry (ICRAI)*.

Ramesh, A., Stolkin, R., and Chiou, M. (2022). Robot vitals and robot health: Towards systematically quantifying runtime performance degradation in robots under adverse conditions. *IEEE Robotics and Automation Letters*, 7(4):10729–10736.

Rivera, L., Jiménez, M., Tamura, G., Villegas, N., and Müller, H. (2021). Designing run-time evolution for dependable and resilient cyber-physical systems using digital twins. *Journal of Integrated Design and Process Science*, 25:48–79.

Schmitt, T., Hanek, R., Beetz, M., Buck, S., and Radig, B. (2002). Cooperative probabilistic state estimation for vision-based autonomous mobile robots. *IEEE Transactions on Robotics and Automation*, 18:670–684.

Singh, B., Kumar, R., and Singh, V. (2022). Reinforcement learning in robotic applications: A comprehensive survey. *Artificial Intelligence Review*, 55:945–990.

Smyrnakis, M., Qu, H., Bauso, D., and Veres, S. (2020). Multi-model adaptive learning for robots under uncertainty. In *ICAART (1)*, pages 50–61.

Stephens, A., Budd, M., Staniaszek, M., et al. (2024). Planning under uncertainty for safe robot exploration using gaussian process prediction. *Autonomous Robots*, 48:18.

Xu, W., He, D., Cai, Y., and Zhang, F. (2022). Robots' state estimation and observability analysis based on statistical motion models. *IEEE Transactions on Control Systems Technology*, 30:2030–2045.

Zhang, T. and Mo, H. (2021). Reinforcement learning for robot research: A comprehensive review and open issues. *International Journal of Advanced Robotic Systems*, 18.

Zhang, T., Peng, F., Yan, R., Tang, X., Deng, R., and Yuan, J. (2024). Quantification of uncertainty in robot pose errors and calibration of reliable compensation values. *Robotics and Computer-Integrated Manufacturing*, 89.

Zhao, X. and Chidambareswaran, T. (2023). Autonomous mobile robots in manufacturing operations. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*.