INTER IIT TECH MEET 13.0

Position and with Altitude Control Quadrotor Single Motor Failure.

Midterm Report

Authors

Aawish K Sudhish	Mechanical Engineering
Hitesh Kumar	Computer Science and Engineering
Kushagra Shahi	Mechanical Engineering
Malhar Karade	Mechanical Engineering
Parth Govale	Computer Science and Engineering
Sai Gawali	Mechanical Engineering
Shaury Patel	Mechanical Engineering

2024-11-10

Contents

1 Introduction

1.1 Problem Understanding

Quadrotors, with their symmetrical design and simplified control systems, are widely recognized for efficient maneuverability and ease of use. However, this design symmetry, while advantageous for normal operations, makes quadrotors highly vulnerable to single motor failures. When even one motor fails, the quadrotor loses its balance and may become uncontrollable, increasing the risk of a crash. Such failures can arise from various causes, including mechanical wear, electrical malfunctions, or challenging environmental conditions like strong winds or debris.
Addressing this problem requires a reliable method for detecting single motor failures in real-time and designing an algorithm capable of compensating for the loss by leveraging the remaining three motors. This algorithm should ensure the quadrotor can stabilize itself to hover momentarily and perform a safe, controlled landing. Developing this capability is crucial for preventing potential harm and ensuring the reliability of quadrotors in various applications.

1.2 Motivation

Quadrotors have become the preferred choice for small UAVs across numerous sectors, given their flexibility, agility, and ease of deployment. They are increasingly utilized in areas such as aerial surveillance, precision mapping, structural inspection, photography, search and rescue operations, agricultural monitoring, and logistical deliveries. With growing UAV deployments in highly populated urban environments, ensuring their safe and reliable operation is essential. Motor failure in these scenarios poses significant safety risks, especially where quadrotors operate over densely populated areas or near critical infrastructure. Developing a solution that addresses single motor failure recovery not only enhances the reliability of UAV systems but also aligns with the growing demand for secure UAV operations in civilian, commercial, and defense applications. A robust approach to handle motor failures will play a vital role in ensuring that quadrotors can operate safely and effectively, minimizing risks to people and property.

2 Approach Outline

2.1 Solution Overview

The solution to the problem is divided into three main components: failure detection, quadrotor stabilization, and safe landing.

For **failure detection**, we are utilizing a Long Short Term Memory (LSTM) model, which is well-suited for processing sequential or time-series data. To achieve this, we will build a model, train it and fine-tune it with simulation data of single motor failures, using real-time sensor data from the quadrotor, such as GPS, gyroscope, and accelerometer readings. The model will learn to map this sensor data to a binary output indicating whether or not a single motor failure has occurred. Once trained, the LSTM will serve as a detection mechanism, and its output will trigger the stabilization algorithm upon detecting a motor failure.

For **quadrotor stabilization** and **safe landing**, we will employ Reinforcement Learning (RL). An RL environment will be created, where the algorithm receives high rewards for stabilizing the quadrotor and penalties for actions that increase instability or risk of crash. In this setup: - The **state** represents the current data from all onboard sensors. - The RL model will learn the optimal **State Value function** and **State-Action Value function** to determine the best action based on the quadrotor's current state.

By optimizing the State-Action Value function, the RL model will be capable of stabilizing the quadrotor after a failure and guiding it safely to the ground for a controlled landing. This approach ensures real-time adaptability to various scenarios, enhancing both the stability and reliability of the quadrotor during a motor failure.

2.2 Background Study

This project is build and control using PX4-Autopilot, ROS2, Gazebo, MicroXRCE-DDS-Agent and QGroundControl. The ROS topics allows to control the quadrotor simulation in Gazebo using PX4- and MicroXRCE-DDS-Agent. ROS2 supports reliable communicaiton needed for these simulations, while Gazebo's physics engine enables realistic motor failure testing. QGoundControl integrates with these tools for efficient monitoring. Plotjuggler is used to effectively observe the data from the ROS topics in graphical format.
"Fault Detection in UAVs using Deep Learning and Machine Learning" by Niloy Chakraborty (GitHub Repository) provided comprehensive insights into leveraging machine learning algorithms for fault identification and prediction which helped us to create our own model.
The research paper on Autonomous quadrotor flight despite rotor failure [2] gave us useful insights on single motor failure problem and how to proceed with it. These combined references have laid a robust foundation for our project, enabling us to build a comprehensive failure detection system that incorporates machine learning algorithms and real-time data acquisition.

3 Analysis and Initial Experiments

3.1 Simulation Setup

Our simulation environment integrates PX4-Autopilot, Gazebo-classic, and ROS 2 Humble on Ubuntu 22.04.5, with QGroundControl used for monitoring and control. To evaluate motor failure scenarios, we initially established intercommunication between these systems to verify proper integration and control functionality.

We configured the simulation using the IRIS quadrotor model in Gazebo to replicate a single motor failure. The setup includes:

- **Onboard Sensors**: We use the IRIS model's built-in sensors, including an accelerometer, gyroscope, and GPS, to track the quadrotor's position, orientation, and speed without any external equipment.
- **Motor Failure Simulation**: Through Gazebo's plugin functionality, we simulated a single motor failure to assess the quadrotor's response in a controlled and realistic virtual environment.
- **Custom Control Code**: After establishing communication between PX4 and ROS 2, we developed custom control algorithms to respond to motor failure conditions. These algorithms leverage ROS 2 topics and services to adjust the remaining motors, aiming to stabilize the quadrotor and, if necessary, initiate a controlled descent.

This simulation setup allows us to test and refine our control strategies under failure conditions before real-world testing.

3.2 Observations and Adjustments

3.2.1 On Ground Data

The data shows stability across x, y, and altitude with minimal fluctuations, suggesting a stationary state without significant movement or rotation.

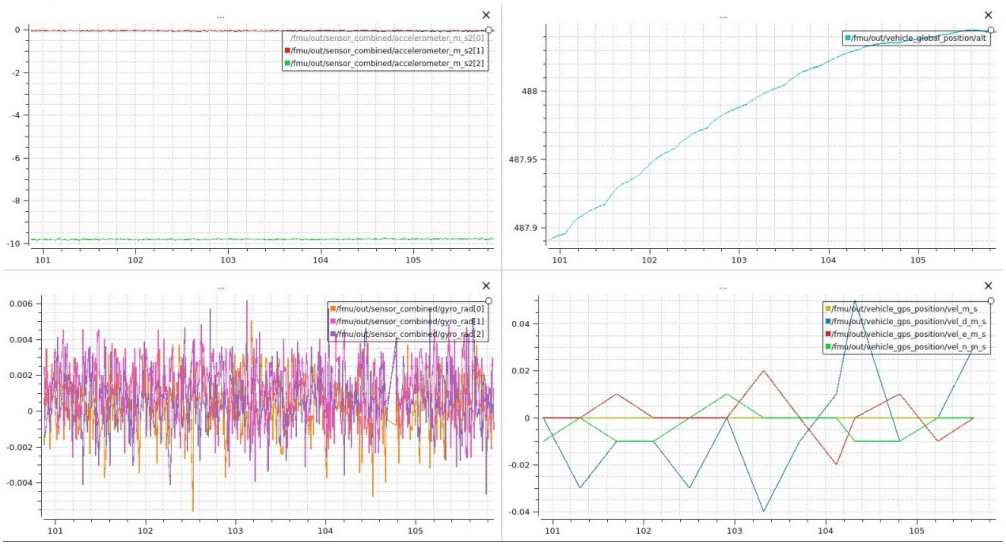

Figure 1: Sensors Data when the Quadrotor is on ground

3.2.2 Take Off Data

The data indicates a clear upward ascent with stable x and y components, rotational adjustments for stability, and initial spikes in downward velocity, confirming vertical takeoff with minor horizontal stabilization.

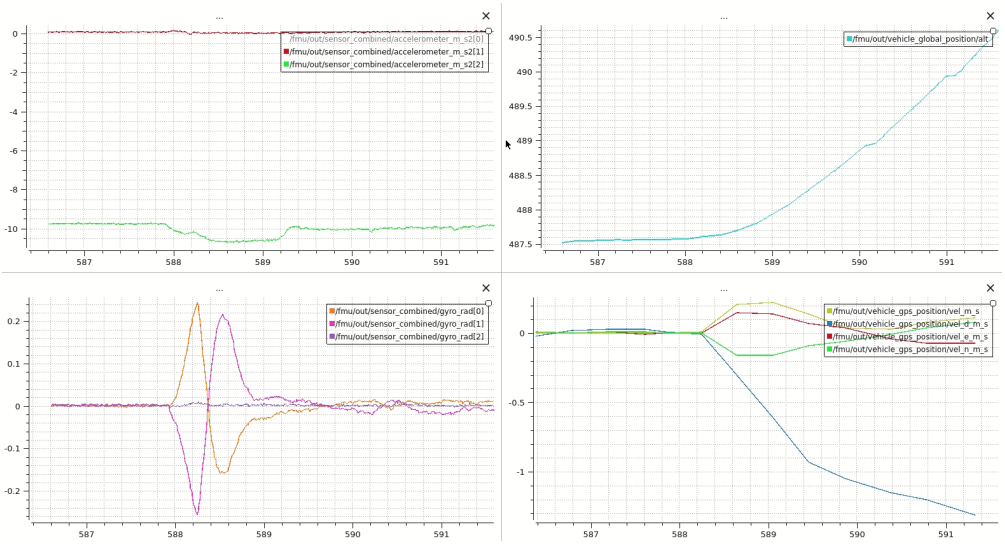

Figure 2: Sensors Data when the Quadrotor is Taking Off.

3.2.3 Hovering Data

The data shows stable altitude and minimal velocity or rotational fluctuations, indicating the drone effectively maintained a steady hover with slight drift adjustments.

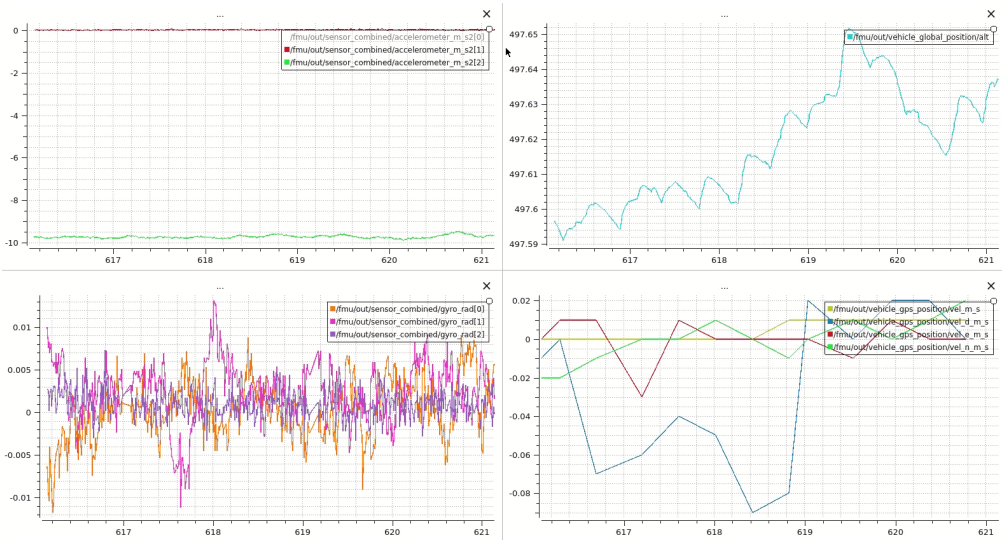

Figure 3: Sensors Data when the Quadrotor is hovering.

3.2.4 Landing Data

The landing data reflects a controlled descent, with a final adjustment in the z-axis acceleration and a gradual altitude decline. Minor fluctuations in orientation and a decrease in downward velocity indicate stability efforts as the drone prepares to touch down.

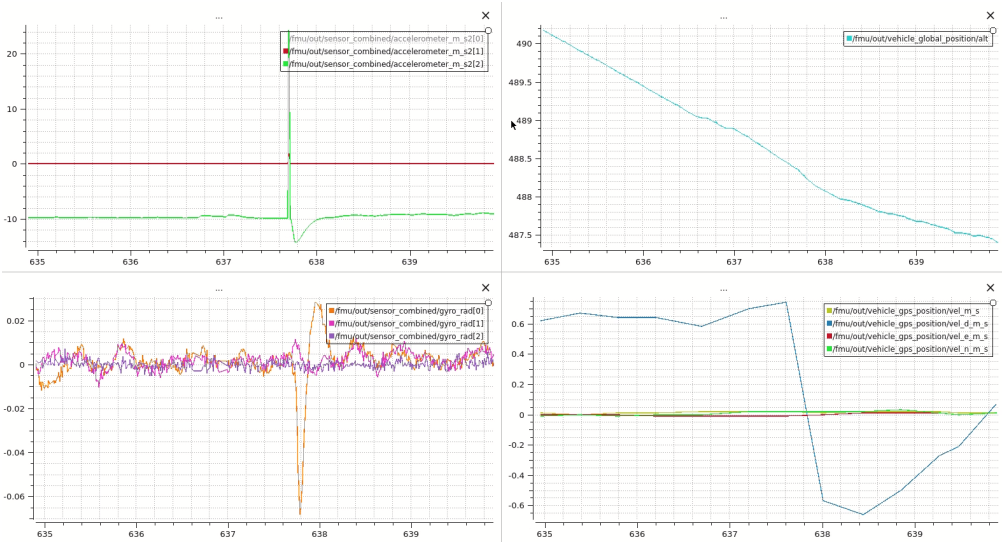

Figure 4: Sensors Data when the Quadrotor is on land after Landing.

3.2.5 Motor Failure Data

The motor failure data reveals clear signs of instability as indicated by significant accelerometer spikes in all axes, signaling abrupt movements that likely mark the onset of the failure. The altitude data shows a sharp drop, suggesting a rapid loss in lift and stabilization capacity. Gyroscope readings with larger oscillations confirm that the drone struggled to maintain its orientation. Correspondingly, the GPS velocity data displays abrupt shifts in

all components, highlighting an unsteady descent and fast horizontal adjustments as the drone copes with the sudden lack of thrust.

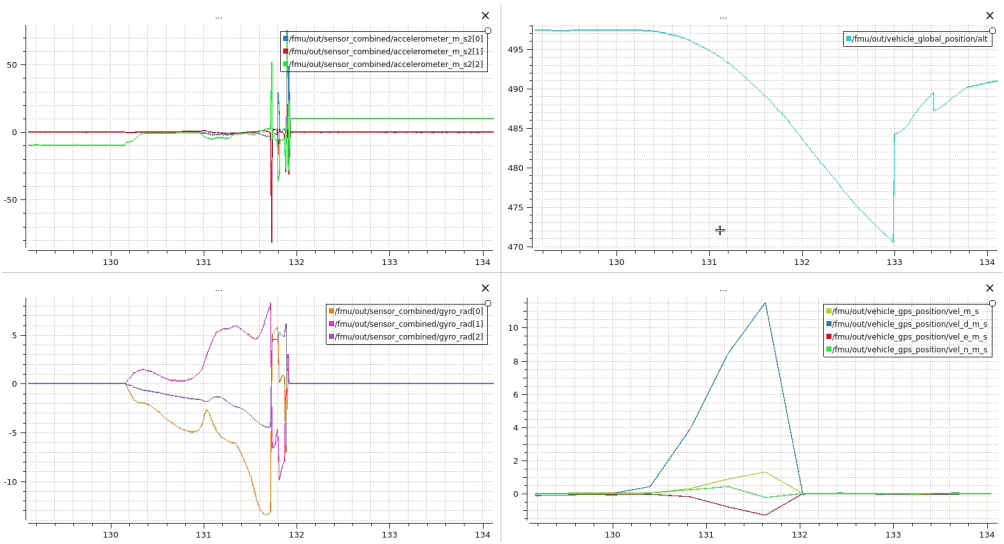

Figure 5: Sensors Data when the Quadrotor is in state of Motor Failure.

NOTE:
The video recording of the varying sensor data during the takeoff, and landing: Video
The video recording of the sensor data during motor failure: Video

3.3 Failure Detection

The `model.ipynb` notebook in the failure detection folder of the workspace is designed for training and evaluating a motor failure detection model using LSTM (Long Short-Term Memory) neural networks. This section provides an analysis of the notebook, including its structure and key components.

3.3.1 Structure and Key Components

1. **Dataset Processing:** The notebook reads multiple CSV files from the dataset folder, combines them into a single DataFrame, and processes the data.
2. **Model Definition:** The `MotorFailureDetectionModel` class is defined, which is an LSTM-based neural network model. The model includes LSTM layers, a fully connected layer, dropout, and a sigmoid activation function.
3. **Training the Model:** The model is trained using the `train_model` method, which includes defining the loss function, optimizer, and training loop.
4. **Evaluating the Model:** The `evaluate_model` function is used to evaluate the model's performance on the validation set. It calculates the loss and various metrics such as accuracy, precision, recall, and F1 score.

3.3.2 Model Training

We trained our model on the time-series data of all the given topics:

1. **/fmt/out/sensor_combined** : This topic provides combined sensor data, including gyroscope and accelerometer measurements.

2. **/fmt/out/vehicle_global_position** : This topic provides the global position of the vehicle, including latitude, longitude, and altitude.

3. **/fmt/out/vehicle_gps_position** : This topic provides GPS position data, including latitude, longitude, and altitude.

3.3.3 Model Testing

We trained our model for 20 epochs with a learning rate of 0.001 on 80% training data and 20% testing data, and got the following results:

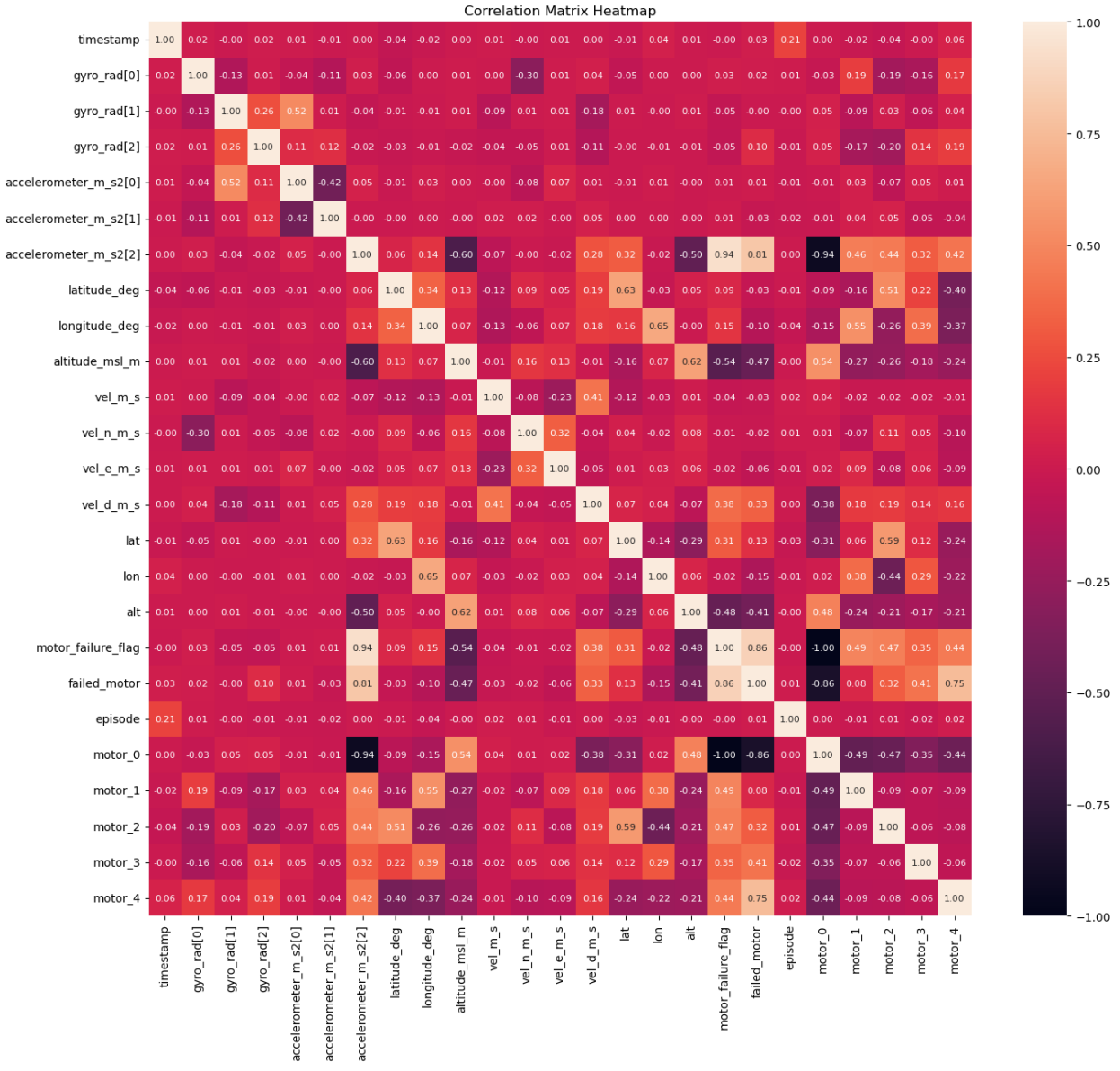

Figure 6: Correlation Matrix of Collected Sensor Data.

The model can be used for motor failure detection in a real-time simulation as per the validation results that we got from the data. For real-time simulation, we can collect the data from the simulation and continuously feed

Description	Accuracy	Precision	Recall	F1 Score
Is Motor Failure	0.9953	0.9920	0.9920	0.9920
Correct Motor No.	0.9968	0.9922	0.9464	0.9675

Table 1: Performance Metrics for Motor Failure Detection Model

it into the model. The model processes this data and provides outputs continuously. This approach is similar to what is happening in validation. The difference is that during validation we already have the data and we are passing that in the model to get the output and in case of real-time detection we will simultaneously read and pass the data into the model. Upon detecting a motor failure, the model sets the `motor_failure_flag` to '1' and identifies the failed motor by providing its corresponding motor number. This flag is then used to trigger the safe landing protocol.

4 Challenges and Next Steps

4.1 Technical Challenges

- **Environment Setup on macOS:** Setting up the PX4 environment on macOS presented initial challenges, particularly related to compatibility issues with dependencies and specific binaries. The lack of native support for certain PX4 components on macOS required additional configuration steps, which led to delays in establishing a functional simulation environment.
- **Automation in Data Collection:** Automating the data collection process proved to be challenging and error-prone, as it required running two parallel processes: one for controlling the quadrotor and the other for collecting data without interference. Additionally, there were instances where the quadrotor failed to arm, halting the entire automation process. This failure prevented the continuation of data collection, as any data captured after an arm failure would be corrupted, necessitating a restart of the simulation.

4.2 Next Steps

- **Algorithm Optimization:** Continue refining the LSTM-based failure detection model using more simulation data to make our model more robust across all possible scenarios, and reducing the parameters to reduce computation time for real-time outputs from the model with high-frequency compatibility.
- **Reinforcement Learning Tuning:** Fine-tune the RL-based stabilization and landing system, focusing on rewarding smooth, stable adjustments and penalizing fast movements that could worsen instability.
- **Real-World Testing:** Transition from simulation to real-world testing under controlled conditions to validate algorithm performance and adjust for any inconsistencies.

References

[1] P.-F. Liu, Y. Ding, and W.-Z. Zhang, "Analysis on the Attitude Kinematic Equation of a Quadcopter with Rotor Failure," in *Proceedings of the 2024 36th Chinese Control and Decision Conference (CCDC)*, Xi'an, China, 2024, pp. 5781–5786, doi: 10.1109/CCDC62350.2024.10587843.

2

[2] S. Sun, G. Cioffi, C. de Visser, and D. Scaramuzza, "Autonomous quadrotor flight despite rotor failure with onboard vision sensors: Frames vs. events," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 580–587, April 2021, doi: 10.1109/LRA.2020.3048875.