Sensor-less Estimation of External Force acting on a Robot Motion Platform using LSTM Neural Network

Team members:
Yokesh Dhanabal (2011468),
Sihui Liu (2018004),
Tom Minten (1372300),
Nan Zhu (2044331)

Project owners:
Willem-jan Lamers,
Joep Selten

Project information
Document date: June 10, 2024
Project period: 6/2/24 - 17/6/24,
Course: 5ARIP10
Organisation: TU/e

***Abstract*—In hospitals, robots work together with humans who may eventually push the robot to move out of the way, which can be considered as an external force acting on the robot motion platform. Existing sensor-less disturbance observer based external force estimators use torque data from the motors to detect the external force. However, these estimators require a realistic robot model and are complex to implement. This paper proposes an alternative solution which uses LSTM neural network to estimate the external force using current and voltage data from the motors. This sensor-less solution is less complex and does not require a realistic model of the robot motion platform. A simplified proof of concept on a singular BLDC motor is made, which shows promising results. This concept now has to be applied on the robot motion platform itself. With the success of this concept, it can possibly be extended to other robot motion platforms with different dynamics without major changes. Furthermore, it will be a simple solution that enables robot motion platforms to manage and respond to pushes initiated by humans.**

***Index Terms*—sensor-less estimator, LSTM, RNN, external force estimator, BLDC motor**

I. Introduction

As the global trend of population aging intensifies, the healthcare system is facing unprecedented pressures. In hospital environments, the shortage of nursing staff and medical workers is becoming an increasingly severe issue. In the Netherlands, the expectation is that the part of the workforce which works in healthcare will increase, from 11% in 2019 to 18% in 2050 from the working population [6]. So, this shortage urgently demands technological innovation to improve the quality and efficiency of medical services. In this context, the application of automation technologies and intelligent robots is particularly important as they can take on repetitive and physically demanding tasks in hospitals, such as medication delivery, laboratory sample transport, and bed moving. This enables medical personnel to focus more on direct patient care and complex medical tasks. Ambee is designing a robot platform for two relevant use-cases: Medication & Lab samples, and Autonomous Bed Transportation [8]

The robot motion platform, shown in Figure 1, will operate in the corridors and rooms with several humans present nearby the robot, who eventually require pushing the robot aside. The environment requires a compliance controlled motion platform which enables the robot to be pushed away by humans. To perform compliance control, the externally applied forces should be estimated. Ambee asked us to explore options for designing and implementing a compliance controller for their robot motion platform.

For this project, some preliminary research related to classical compliance control and disturbance obsevers was done. J.J.E. Unkel[11] developed a disturbance observer to estimate external forces without additional sensors, using the position of the object as input. The estimated external force is passed through a low-pass filter to improve the performance of the observer and enhance the stability of the system. However, the ”position” based observer only works in the situation where the position of the robot changes.

R.M Beumer[2] proposes a more complex disturbance observer to estimate external forces. The state variables of the system are first estimated based on the Kalman filter. The state variables, wheel inputs, and the dynamic model are then used as inputs to the disturbance observer to estimate the user inputs. This observer provides accurate estimates of disturbances within 80 N. This concept enabled a certain robot motion platform to act on external forces by changing the trajectory, which is similar to the objective of this project. And this range of external forces is also used as a baseline for the project.

Apart from the robot's kinematic state (e.g., position), the actuator's mechanical signals can also be used to construct the observer. Hung, C.-W et al.[5] compare the feedback torque from motor with the estimated torque from the dynamic model and calculates the external torque, developing a torque-based observer. The result is then used to estimate external force. Shouhua Zhao et al.[13] propose a load torque observer to observe the load torque and estimate the perturbations caused by inaccurate model parameters. Identification of inertia and

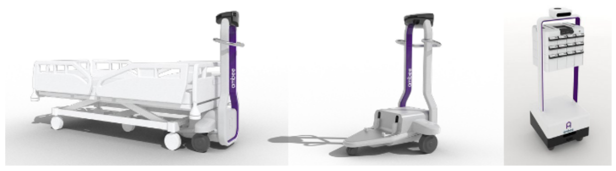


Fig. 1: Renders of the Ambee robot motion platform from [8]

friction coefficients is achieved by a gradient descent method.

These studies estimate external forces that are not directly observable by constructing different system models using different observable signals. However, these models are limited to specific systems, requiring the construction of complex system dynamics models. Furthermore, small changes in system model parameters can lead to large observation errors.

In addition to the classical control, there have been few studies conducted on the use of neural network-based models for external force estimation. But the problem of external force estimation can be translated into the problem of detecting abnormal data from normal data. Jonghwan Son, et al. [9] use a Convolutional Auto-encoder and one-class support vector to perform anomaly detection of electric motors. Sabtain Ahmad, et al.[1] propose a framework for rotating machine (RM) condition monitoring and anomaly detection based on Long short-term memory (LSTM). Their results show the effectiveness of LSTM in processing and estimating time-series data, which is particular interesting for this project.

In the initial project proposal, the implementation of a disturbance observer based solution was proposed. However, this approach was not successful. This report proposes a different solution to address the challenge, an external force estimator based on an LSTM machine learning model. The external force is estimated using the voltage and current data of the Brush Less DC (BLDC) motors of the robot motion platform. This data can be obtained from the drive used for motor control and no additional sensors are required.

This report will discuss the methodology and results for both the disturbance observer based solution and the LSTM model based solution to estimate external force. The main contribution of this paper is a proof of concept of the LSTM based solution in a simplified simulation. At the end, the conclusion and discussion are presented.

II. Methodology

A. Problem formulation and Objective(s)

Ambee asked to explore options for designing and implementing a compliance controller for the robot motion platform. For this, an estimation of the external force on the motion platform should be obtained. The estimated external force, caused by a push of a human, can be used when controlling the robot motion platform. The first and main objective was estimating the external force. Using the estimation in a controller was seen as secondary objective in case of time left.

To describe the required solution for the problem, the following requirements, preferences, and constraints were defined at the start. Objective 1 is defined in the requirements. Due to timing constraints, objective 2 is considered as an optional objective, which is to be implemented if there is time left after implementing objective 1.

Requirements: Based on the main objective, measurable requirements were defined on forehand. These requirements set the definition of the circumstances and enable assessment of the final product. A push is defined as an external force.

R1 The robot should be able to determine the magnitude of the external force with a maximum deviation of 20 Newton

R2 The robot should be able to determine the direction of the external force, 80 Newton or higher, is directed.

R3 The code should be written in python or C++

The above metrics of 80N and the deviation of 20N are based on previous research by R.M. Beumer [2] with some margin. Using these metrics enables comparisons between the result of this project and the results of the previous research. Also, these metrics are based on practical experiments of previous research.

Preferences: The preferences include the extra objective and some preferences stated by Ambee.

P1 The proposed solution does not require any additional sensors to estimate the external force.

P2 The robot should move in the direction where the external force, 80 Newton or higher, is directed.

P3 The robot should move smoothly, with a maximum acceleration and speed of 1 m/s^2 and 3 m/s respectively, when an external force between 80 and 200 Newtons acts upon it.

P4 The robot may not leave the defined location/ trajectory when there is no external force is applied.

P5 The software is delivered in the form of a package for the robot operating system (ROS2 Humble)

The forces and maximum acceleration are based on the previous research by R.M. Beumer [2].

Constraints: As the computational resources of the robot are also utilized for path planning, image processing and other inherent tasks, it introduces a constraint for our solution.

C1 The code should run on a Jetson Orin AGX module, utilizing minimum computational resources as possible.

The goal of the project is to fulfill the requirements and preferences, which also satisfies the constraints.

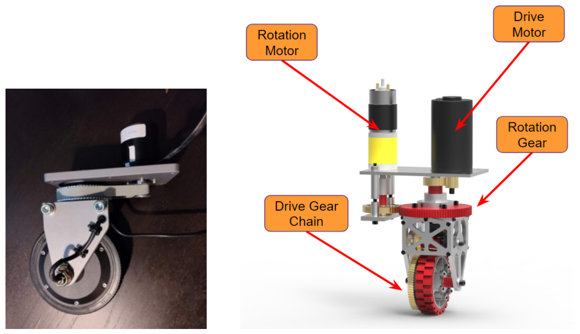
The report predominantly focuses on the methodologies and results related to objective 1. This is because, no work related to objective 2 was carried out due to timing constraints. The following sections delve into the detailed robot details, methodologies and results pertaining solely to objective 1.

B. System description

This section gives a short note on some of the components present in Ambee hospital robot motion platform (introduced in section I), which can be useful for this project. A Jetson Orin AGX, ZED X stereo cameras, and 2 swerve drives each consisting of 2 BLDC motors, one for controlling steer angle (rotation motor) and one as direct drive motor. The wheel hub of a robot used by Ambee can be seen in figure 2a and 2b shows the main components of a swerve drive.

C. Approach 1: Classical Disturbance Observer Method

In the Classical Disturbance Observer method, three main components are present, as shown in figure 3. The concept is taken from the research discussed in the section I.



(a) Picture of a wheel hub in Ambee robot

(b) Parts of Swerve Drive

Fig. 2: Wheel hub of Ambee robot and Swerve drive

- Dynamical Model: The dynamical model provided a foundation for the entire system by using mathematical equations to simulate the robot's physical behavior under various inputs. This model directly links key variables such as mass, inertia, and external forces, laying the groundwork for the controller's design. Additionally, the dynamical model made it feasible to use Disturbance Observer (DOB) to infer external forces based on the robot's movements. However, since it was challenging to incorporate friction into the existing model, it can result in significant errors.
- Disturbance Observer: The disturbance observer used the algorithm of Beumer and the dynamical model to detect and assess external forces acting on the robot without the need for direct sensors. By monitoring the robot's motion and comparing the resultant forces from the robot's velocity with the expected internal forces, it can estimate the external disturbances. This enhances the system's adaptability and rapid response capability in changing environments.
- Controller: The controller generated appropriate voltage and current outputs based on data from the disturbance observer, ensuring compliant control to achieve the robot's desired behaviors. It adjusted the platform's movements according to estimated external forces.

In conclusion, the structured integration of dynamic models, disturbance observers, and controllers aimed to accurately identify external forces and achieve two objectives. This approach was based on the methodology of R.M. Beumer [2] and has been adapted for the Ambee robot motion platform to ensure reliable performance. It was designed to ensure the effective operation of hospital robot platforms in dynamic and complex environments. However, during our implementation, it was easy to find that the dynamical model could not include friction, and in the actual testing of the Ambee robot, the impact of friction was significant. This posed a challenge to achieving our objectives, so approach 2 is a more applicable way.

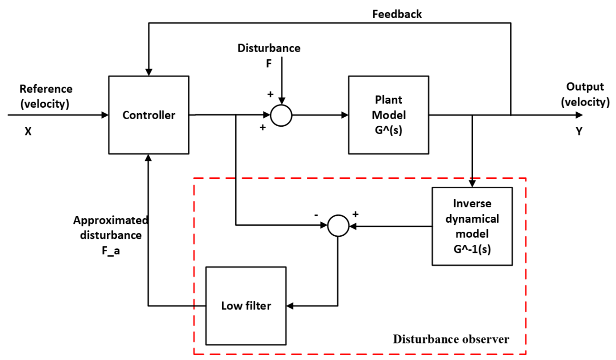


Fig. 3: Schematic Design of Disturbance Observer Method

1) Dynamical Model: The dynamic model described the robot's behavior, including speed, under various conditions such as different weights and external forces. This enables the Disturbance Observer to infer external forces by analyzing the robot's speed using the inverse dynamic model. Although the model for the Ambee robot was difficult to set up, R.M. Beumer[2] provides a modular dynamic model for the Ropod. Therefore, the following derivation was based on the Ropod model and can be adapted for future use with the Ambee robot motion platform.

This robot model employed different equations of motion that have been adapted to meet specific system requirements. Through the establishment of base robot model, robot kinematics, wheel dynamics, and wheel hub dynamics, the total platform dynamics model is ultimately derived, as represented by equation 1.

$$M_{\text{dir}}\ddot{q}_b = B_{F_b}F_b + B_\tau\tau + b_{\text{dir}} \tag{1}$$

In this equation, M_{dir} represents the direct mass matrix, incorporating the mass and inertia of the robot's components. $\ddot{q}_b$ is the second derivative of the robot's location, denoting acceleration. $B_{F_b}F_b$ and $B_\tau\tau$ represent the contributions of external forces and wheel inputs to the system's state, while b_{dir} includes all other relevant coefficients, such as damping and friction.

To enhance the accuracy and practicality of the model, dynamic parameter identification techniques were used to precisely measure and adjust key parameters within the model as shown in equation 2.

$$A_{\text{id}}m = B_{\text{id}} \tag{2}$$

Here, m represents a vector containing all dynamic parameters, including the robot system's mass, the inertia of its components, and $A_{\text{id}}$is a function of base position qb and its time derivatives, which is fixed. The detailed derivation formula will be provided in the appendix VI-C.

This dynamic model offerd significant advantages. Firstly, it converted the model into a solvable system of equations, enabling dynamic parameter identification, so it supports scalability, allowing easy adjustments and integration of robot components, such as attaching a bed, and altering the robot's mass and inertia. Moreover, the dynamical model has proven effective with compliance control in Beumer's real test[2], ensuring safe and reliable operation. Overall, it provided a robust and flexible foundation for diverse robotic applications, which was easy to easily adjustable to Ambee robot motion platform.

2) Control Strategy: Recognizing the limitations of traditional control methods, such as PID and state feedback control, in environments that required flexible human-robot interaction, and acknowledging that traditional PID control mainly eliminates errors through feedback mechanisms, while the robot controller in this project needed to adapt to external disturbances and compensate for system errors, the focus was shifted to compliance control strategies. Compliance control, which is not a traditional control method, includes impedance and admittance methods. These methods enable robot to intelligently respond to external forces by integrating them with internal force. Therefore, in this project, compliance control was preferred to achieve the second objective.

Admittance control was chosen over impedance control due to its flexibility[2]. Impedance control adjusts the robot's response through the specified mechanical impedance (inertia, damping, stiffness), making movements intuitive but less adaptable in dynamic environments. In contrast, admittance control generates reference trajectories based on measured external forces, allowing the robot to adjust its position and orientation accordingly[11]. The admittance control method is more suitable for varying conditions and load changes.

As shown in the red dotted box of figure 3, the proposed approach involved using the disturbance observer to estimate the user's input force from wheel inputs, state variables, and the dynamic model, filtering high-frequency noise for accuracy. This allowed the user to apply input anywhere on the motion platform. The estimated external force was then converted into a velocity set-point with safety constraints according to the dynamical model as shown in equation 3.

$$\ddot{q}_b = M_{\text{dir}}^{-1}\left(B_{F_b}F_b + B_\tau \tau + b_{\text{dir}}\right) \tag{3}$$

The velocity controller will be integrated to track the velocity set-point derived. This controller will ensure compliant control by adjusting the platform's speed appropriately, while also preventing overreaction to accumulated errors and maintaining stability through predefined safety constraints on maximum velocity and acceleration. Although the controller was not fully implemented, which also means there is no time for objective 2, J.J.E. Unkel[11] and R.M Beumer[2] have demonstrated the feasibility of compliance control.

3) Disturbance Observer: To accurately estimate the external force exerted on the robot motion platform, the project employed an approach based on a disturbance observer (DOB) in the frequency domain. This approach did not rely on specific sensors, but instead used the robot's dynamic model to estimate human user input force. The advantage of this approach was that the robot can measure forces from various directions and magnitudes without the need for specific sensors at those locations, and it was much easier to implement.

The design of the Disturbance Observer was based on calculating the actual torque experienced by the robot through the inverse plant model and using the system's actual output (location) to estimate external forces. The process began with rewriting the dynamic equations to isolate the user input F_b. By rearranging and parameterization, a new equation was obtained where the disturbance force appeared as an unknown variable. Utilizing the dynamic model and real-time measurements of wheel inputs and location variables, the observer calculates the disturbance force estimate $\hat{F}_b$ in real-time, filtering the results with a low-pass filter to suppress high-frequency noise, as shown in equation 4. The torque corresponding to external forces was then accurately measured by comparing the difference between the actual torque, calculated with the inverse plant model, and the internal torque provided by the motor.

$$\hat{F}_b = B_{F_b}^{-1}\left(M_{\text{dir}}\ddot{q}_b - B_\tau \tau - b_{\text{dir}}\right) \tag{4}$$

To implement this process, development began with creating pseudo code to outline the logic and functionality of the observer, mainly divided into the inverse dynamical model and the low-pass filter, as shown in VI-D. This pseudo-code served as the basis for developing the actual code. The developed code was then applied to a simplified model, and combined with the controller part for testing, as demonstrated in the simulator below, to validate the performance of the observer and the controller. And in the plan, if the initial validation is successful, further simulation tests are planned to adjust and optimize the DOB parameters to ensure accurate detection of external forces in both controlled and real-world environments.

4) Simulator: The goal of the simulator was to enable us to test solutions in a simple digital environment, as the advanced simulator of the Ambee robot motion platform was not available and possibly too complex to work with. The simulator can easily be connected to a disturbance observer and a controller.

The simulator was built using the Pygame library from scratch to meet the project's requirements, as shown in figure 4. The simulator is designed to be accessible by a disturbance observer and controllable by a controller, allowing external forces to be applied to the motors. It uses torque input to control two motors, facilitating easy calculation of applied torque for the disturbance observer. The velocity states of the motors in both the x and y directions are used to derive angles and resulting forces through simple calculations. External forces directly influence the speed of the wheels, with the robot's center of mass assumed to be centrally located between the wheels for simplicity.

The simulator also allows for manual control using arrow keys, to apply torque to the wheels and WASD keys to apply external forces. Simplifications include preventing the robot from turning on its axis, which is shown in figure 4. Nonlinear friction was added to the force balance on the wheels to enhance realism, and external noise disturbances can be introduced if necessary.

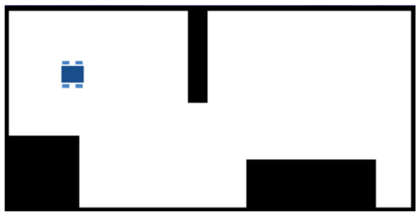


Fig. 4: Simulator designed using Pygame

D. Approach 2: Estimation of external force using machine learning

This method aims at real-time estimation of external force applied on the robot by employing machine learning techniques. The main idea is based on the fact that an increase in load attached to the motor causes variation in overall voltage and current of the motor, and that the external force can be found in this variation. The main objective of this approach can be categorized as a regression problem, where a ML model is trained to estimate the external force, by learning the varying patterns in voltage and current signals of the motor for different values of external. Another reason for choosing voltage and current signal for analysis is because, the motor drives usually have the capability to measure (voltage and current) and transmit those values back to control hardware of the robot. In this approach, the disturbance observer discussed in approach 1 is replaced by ML model.

Finally, during inference, the trained ML model can be used to estimate the magnitude of the external force applied to the robot by using the current and voltage data of the motor. Therefore, the machine learning model can be defined as a mapping function, performing a regression task.

$$f : \mathbb{R}^d \to \mathbb{R} \tag{5}$$

where d is the dimension of the input data.

The primary goal of this approach is to obtain a proof of concept, which can be expanded/implemented to Ambee robots or any other similar systems . For the purpose of simplifying the complex requirement (R1), the following assumptions are made.

- A1 The robot is assumed to move in a straight line path with a constant mass attached to it.
- A2 The external force is applied in a direction that is exactly opposite to the direction of motion of the robot.
- A3 Application of external force against the robot increases the load (torque) acting on the motor.
- A4 The wheels of the robot does not slip during the whole time frame of application of external force.
- A5 Only a single drive (BLDC) motor among all other motors of the robot is considered.
- A6 The transmission ratio (between the motor and wheel) is equal to 1 and radius of wheel is 5cm.
- A7 The (external) force applied on the robot platform can be measured, at-least in a single robot for data collection.
- A8 The total current and voltage data of the BLDC motor can be obtained (either continuous or sampled values) from the motor drive

Based on the the assumptions, A1 to A6, the drive motor attached to the wheel can be modelled as (BLDC) motor attached to a varying load (torque) as shown in figure 5.

1) Relation between current, voltage and load (torque) of BLDC motor: As discussed in section II-B, Ambee uses Swerve drive based on BLDC motors in their robot and hence, BLDC motor is considered for simulation. Brush-less DC (BLDC) motor is a type of electric motor that operates on direct current. One of the main differences between a normal (Brushed) DC motor and BLDC motor is that, the rotor of a BLDC motor is made up of permanent magnets and does not contain any coil/winding. A more detailed explanation, on the relation between current, voltage and load (torque) in a BLDC motor is provided in the appendix, section II-D1.

The variation (plots) in voltage, phase currents and measured speed of a BLDC motor for a step (increase) torque applied at t = 0.5 seconds is shown by figure 6. The simulation setup used to make the plots is explained in the next subsection.

2) Simulation setup: The simulation setup consists of a 3-phase BLDC motor coupled with a time-varying load (torque) as shown in figure 5. Since the transmission ratio is equal to one (assumption A6), both the rotor and load rotate at a same rate. The varying torque block accommodates the effects of external force acting on the robot. The detailed explanation of the simulation setup is provided in the appendix VI-F. Description on accommodating the external force in load and the process of data generation using this model are explained in the sections II-D3 and II-D4 respectively.

The system described above does not exactly align with the system used by Ambee. This is because, the specifications of BLDC motor and the drive are not exactly known. Also, these specifications may vary among different robots used within Ambee. However, this variation does not have any impact on the success of this approach. As the main focus is to develop a ML based solution, where the ML model can learn the patterns in the data acquired from robots with any kind of specification.

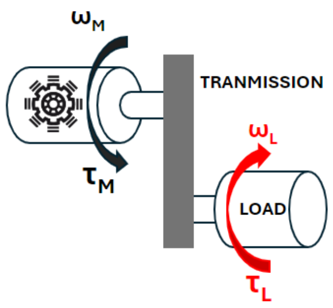
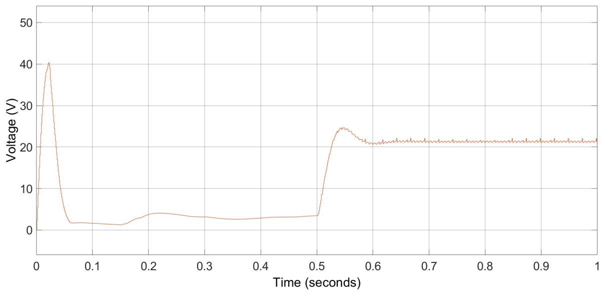
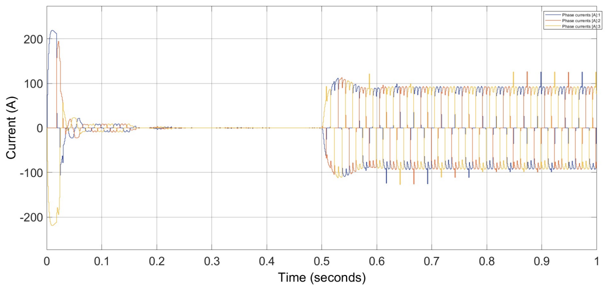


Fig. 5: BLDC motor and load


(a) Voltage (V)


(b) Phase Currents (A)

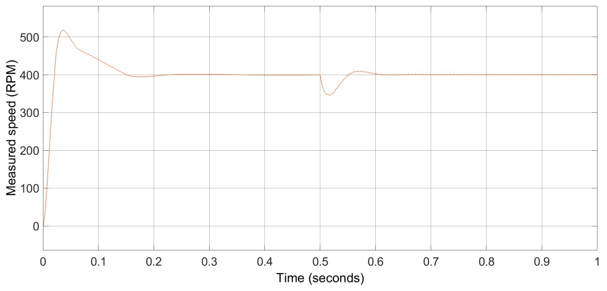

(c) Measured speed (RPM)

Fig. 6: Variation in voltage (input), phase currents and speed of BLDC motor for a step torque.

For this project, since it is not possible to acquire actual data from robots, the data generated using a similar simulation setup is used.

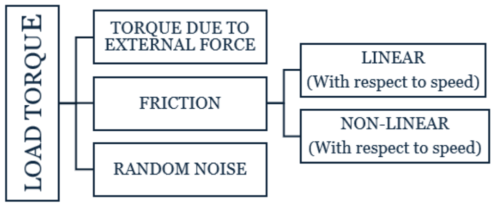

LOAD TORQUE
TORQUE DUE TO EXTERNAL FORCE
FRICTION
RANDOM NOISE
LINEAR (With respect to speed)
NON-LINEAR (With respect to speed)

Fig. 7: Main components of load attached to drive motor

3) Modelling external force as a load component: The load on the motor increases due to the application of external force acting against the robot, which is one of the assumptions of approach 2. However, the load is also influenced by various other components as shown in figure 7.

One of the main components that contributes to the increase in load is the external force applied against the robot. A simplified relation between the torque acting on the wheel and the tangential force exerted by the wheel on any surface is given by equation 6.

$$\tau = r \cdot F_T \tag{6}$$

Here, τ is the Torque (in Nm) produced by the motor, r is the radius of the wheel (in metre) and F_T is the tangential force (in Newton). Thus, the tangential force is directly proportional to the torque. Based on the figure 8, application of external force (F_E) increases the tangential force (F_T) which in-turn increases the load (torque) on the motor.

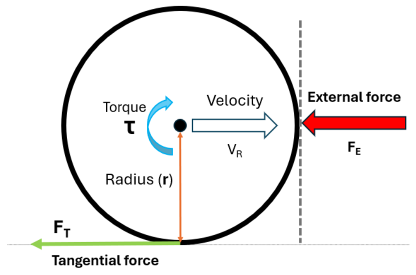


Fig. 8: Representation of tangential force and torque on a robot wheel

Therefore, it can be very well assumed that load torque increases proportionally to the applied external force. For the purpose of simulation, the load component due to the external force is modelled as a steeply increasing signal applied for a duration of 0.25 seconds as shown in figure 9.

The load of the motor is also influenced by frictional forces which are of two types: Linear friction (due to mass of the robot) and non-linear friction (due to the frictional between various moving components of the robot). Additionally, there is a noise component, that accounts for randomly occurring frictional forces, such as wheel slips, change in surface characteristics of the path of robot and so on.

Thus, the final load (torque) is modeled including all the components as discussed above and can be seen in figure 10a. From the plot, it can be seen that the external force is applied at t = 0.5 seconds for a duration of 0.25 seconds and the speed of motor varies t = 0.3 seconds (figure 10b). The variation of total current supplied to the motor due to varying speed and change in load due to application of external force can be seen in figure 10c.

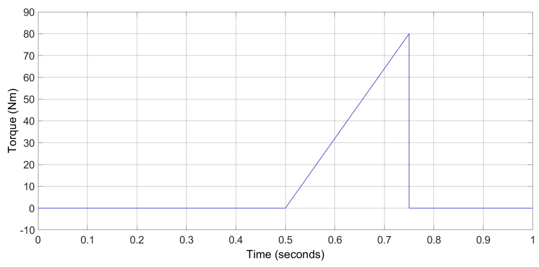
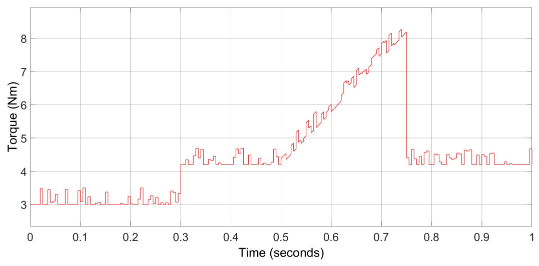
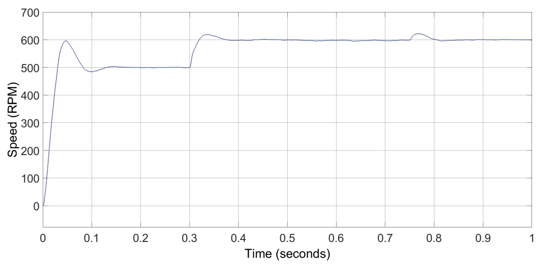


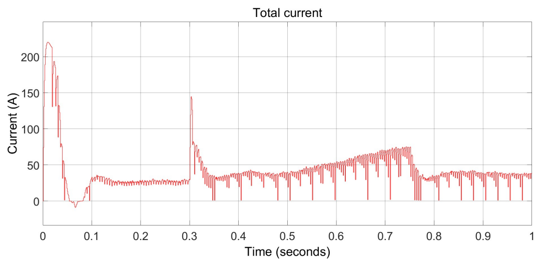
Fig. 9: Load torque (in Nm) due to external force



(a) Load torque



(b) Measured speed (RPM)



(c) Total current to motor

Fig. 10: Load torque, speed and total current of BLDC motor.

To generate the required training data, the whole simulation must be executed multiple times. Therefore, to reduce the overall time of data generation, the whole simulation is confined to a duration of one second.

4) Data generation: For a successful implementation of approach 2, data is required to train, validate and test the machine learning model. As the real robot nor any pre-built simulation environments are available to acquire the data, it is generated using the Simulink model (BLDC motor attached to varying load) described in section II-D2. Also, the generated data must mimic the actual behaviour of robot against the application of an external force, as closely as possible. Thus, the Simulink model is run for the scenarios (S1, S2 and S3) listed below.

S1 The robot is assumed to move at a constant speed and external force is applied at specific time, t = 0.5 second.

S2 The robot is accelerated (speed of motor is increased) at a specific point of time and external force is applied at time, t = 0.5 second.

S3 The robot is accelerated at a specific point of time and the external force is applied at a random time t, where t $\in$ [0.125, 0.875]

Initially, the model is trained and evaluated using the data generated for scenario 1 (S1), which is the simplest among the scenarios. Once the ML model is evaluated, to produce satisfactory results, then it is trained on more complex data generated by S2 followed by S3.

The motor current and voltage, not only increases with the load, but also due to increase in speed. The above characteristics is modelled in scenario 2, where the speed of motor is increased at t = 0.3 seconds and force is applied at t = 0.5 seconds.

In Scenario 3, the speed of the motor is increased at t = 0.3 seconds and the external force is applied at a random time. It is to be noted that, in this scenario, the external force is applied for a shorter duration of 0.125 seconds instead of 0.25 seconds in other two scenarios. This is to increase the time interval within which the external force can applied randomly. Else, the external force can only be applied within a time interval of 0.25 to 0.75 seconds.

Scenario 3 closely resembles the real world, as the application of an external force on the robot can occur (randomly) at any time instant . By this way, the complexity of data generation is increased gradually along the scenarios.This step-by-step approach helps to analyse the performance of ML model and fine-tune it to handle complex patterns in data.

As the ML model is designed for a regression task (supervised learning), the actual value of external force is required to calculate the loss and train the model. Based on the discussion in previous section, the varying load also includes random (noise) friction component.

As the data generated is in time-series format, using an LSTM based RNN as the ML model is planned. Accoding to Kyongmin Yeo[7], as the training noise becomes larger, LSTM learns to depend more on its autonomous dynamics than the noisy input data. As a result, LSTM trained on noisy data becomes less susceptible to the perturbation in the data. On the other hand, when trained on noiseless data, LSTM becomes extremely sensitive to a small perturbation, but is able to adjusts to the changes in the input data.

Therefore, if there is a possibility to store the actual force along with this noise, it can leveraged, to improve robustness of the ML model. Finally, based on the test results, the performance of LSTM trained with and without noise can be

TABLE I: Cases for data generation

Scenario		**Actual label used for training**	
		Only external force	**External force and noise**
S1	The robot is moving in a constant speed and external force is applied at time, t = 0.5 second	Case 1	Case 2
S2	The robot is accelerated at a specific point of time and external force is applied at 0.5 second	Case 3	Case 4
S3	The robot is accelerated at a specific point of time and the external force is applied at random at a random time t, where t ∈ [0.125, 0.875]	Case 5	Case 6

analyzed.

Considering above possibility and different scenarios (S1, S2 and S3), table I summarizes different cases for which the data is to be generated. In according with the requirements defined in section II-A, an external force higher than 80N must be detected by the robot platform, with a deviation of 20N. Since, it is not possible to generate data for all possible of external force, the minimum force value (80N) to be detected, along with the deviation is considered for data generation. Thus, the force values ranges from 60N to 100 N, for all the six cases defined in table I.

The workflow involves multiple simulations and logging of large amounts of data. For this, the Multiple Simulations panel of the Simulink Editor is used, as described in appendix VI-G. The generated data is stored in time-series format as MAT-files.

5) Data pre-process pipeline: Data pre-processing is an important step in machine learning to yield highly accurate and insightful results. As the data is in time-series format, it must be sampled before use. To reduce the data overhead, the time-series data is sampled at a sampling frequency of 10^3 Hz before storing it as MAT-files. As the features in the raw data (current and voltage) have different scales, meaning that each feature cannot contribute equally to the model, the raw data is normalized by z-score. The z-score ($\hat{x}$) for a point x, is given by $\hat{x} = (x - \mu)/\sigma$. Here, μ and σ are the mean and the standard deviation of overall data, respectively. Z-score normalization gives each feature a mean of 0 and a standard deviation of 1, which facilitates the convergence of the gradient descent algorithm used in neural network training process. The normalized data is divided into a training set and a test set, where the training set accounts for 80% of all data. The training set is used for subsequent model training, while the test set is used to evaluate the generated model. In training, 10% of the training set will be used as a validation set to detect if the model is over-fitting.

The data pre-process pipeline implemented for this project is shown in figure 11.

6) Machine learning model: In this subsection, the general form of the required model is first discussed. The specific model chosen is then presented.

As seen in the results of simulation (figure 6), there is a non-linear relationship between the input data (i.e., current and voltage) and the output data (i.e., external force). Changes in current and voltage data caused by external forces are not limited to a single point in time, but continue to have an effect

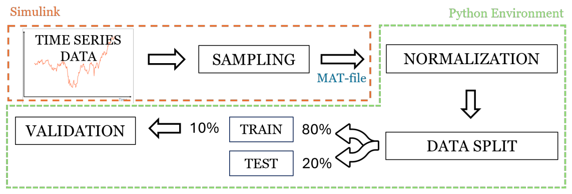


Fig. 11: Data pre-process pipeline

over a period of time. Therefore, when building the mapping model, it is not reasonable to map the current and voltage at a certain point in time to an external force; instead, it is appropriate to build a mapping of input data in a certain period of time to output data. The input data is time-series data, where each element is a two-dimensional variable representing the current and voltage data at that moment in time, respectively. The output data is a real number representing the average magnitude of the external force that was applied to the robot over this period of time. Following is the definition of mapping model.

$$f : \mathbb{R}^{m\times n} \to \mathbb{R} \tag{7}$$

Here, $m = 2$ is the dimension of an element, and n is the length of the time-series data. n needs to be determined according to the motor constant and sampling frequency. If n is too small, the motor cannot respond to the external force in such a short time period. The current and voltage data cannot represent the feature of external force, which makes the model accuracy decrease. On the other hand, if n is too large, the model will take into account data over a long time period, which will result in the model not being able to estimate the external force in a timely manner, and also the estimated average value will not accurately reflect the true external force data over the entire time period. Considering that the external force is expected to exhibit a low-frequency variation over time, it can be reasonably assumed that the nature of the external force can be adequately represented by a low-frequency estimate. In this project, n is set to 100, which means that a single piece of input data comprises 100 data points. Depending on the sampling frequency (10^3 Hz), such a piece of data represents a time interval of 0.1 seconds.

According to A. Tealab [10], classical machine learning algorithms such as feedforward neural networks process input data independently and are therefore not suitable for processing sequential data. But recurrent neural networks (RNNs), on the other hand, benefit from their special network structure that allows them to use the output of the previous step as part of the input to the current step. The basic structure in RNNs is shown in figure 12. The feedback loop (V) allows RNNs to store the output (o) of the previous step as “hidden states” (h) to affect subsequent input data (x). The ”hidden states” are also called ”memory”, which are crucial and powerful for processing time-series data.

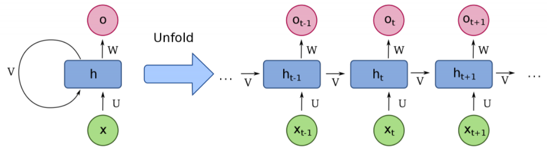


Fig. 12: The Basic structure of RNNs.

However, traditional RNNs suffer from the vanishing gradient problem[4], which make these models hard to train.

Hochreiter and Schmidhuber [4] proposed a new network structure, called Long short-term memory (LSTM), to solve this problem.

The basic network structure of LSTM is called "cell", and its structure is shown in figure 13. LSTM maintains "cell states" (C) to memory long-term data and "hidden state" to memory short-term data. There are three gates that are used to control the flow of information into and out of the cell. Forget gate determines what portion of the previous cell state C_{t-1} to forget. Input gate determines what portion of the new candidate cell state $\tilde{C}_t$ is added to the cell state C_t. The output gate determines the next hidden state h_t by applying the output O_t to the cell state C_t.

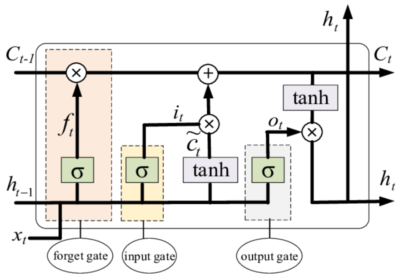


Fig. 13: Basic structure of LSTM

The structure of LSTM networks makes them particularly suitable for processing and estimating time series data. Meanwhile, the results of S. Ahmad et al.[1] showed that LSTM is very effective in situations similar to this project.

Furthermore, it must be acknowledged that in recent years, more advanced neural networks have been implemented, among which the "Transformer" proposed by A. Vaswani et al[12] has been widely used owing to its accuracy. However, transformers have a complex network structure, which results in high computational cost and the training process requires a large amount of data. For this project, data acquisition is difficult as it is manually generated. In addition, as mentioned in constraint C1, the model cannot take up too much computational resources. A simple model is preferred over a complex one.

Thus, the LSTM-based model, instead of transformer-based models, is adopted for this task, which is a trade-off between performance and complexity.

7) LSTM-based model: The model consists of multiple LSTM layers and one Fully Connected (FC) layer. Each LSTM layer contains a certain number of hidden states, and the output of the last LSTM layer is taken as the input to the fully connected layer. The number of neurons in LSTM layers corresponds to the dimension of each element of the input data, and the number of neurons in the last fully connected layer corresponds to the dimension of the output data. The schematic is shown in the figure 14. The FC layer maps the output of the LSTM layers to the target value's dimension. S. Haykin [3] demonstrated that, without nonlinear activation functions, multiple FC layers are equivalent to a single FC layer. Thus, to further simplify the model, the number of FC layers is set to one, meaning that the nonlinear relationships in the data are only learned by the LSTM layers.

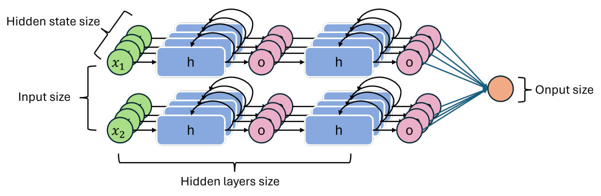


Fig. 14: Network Architecture

Tuning the hyper-parameters (e.g., the number of hidden states of LSTM layers, the number of hidden layers and the learning rate, etc.) is an important step in the training of the model and in this project the hyper-parameter search algorithm is introduced. A three-dimensional search grid is created before formal training begins. The three dimensions represent the hidden states of LSTM layers, the number of LSTM layers and learning rate. A point in the grid represents a combination of the three parameters. The model will be pre-trained on the same dataset using every combination in the grid. The loss function used for training is the Mean Squared Error (MSE) loss which can be calculated using equation 8.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{8}$$

Here, N is the number of data samples, y_i and $\hat{y}_i$ are the actual value and predicted value respectively, for the i-th data sample. MSE summarizes the prediction error of the model, which can indicate the accuracy of a regression model. A lower MSE value indicates a better fit of the model to the data, while a higher MSE value indicates poorer model performance.

After several epochs, the performance of each model will be evaluated the same loss funtion. The model parameters with the best performance (the smallest MSE) in the pre-training are selected for subsequent formal training. In formal training, the dataset is divided into three parts: training set, validation set and test set. The model is trained for multiple epochs on the training set, and after each epoch it is validated on the validation set to ensure that it does not over-fit. After training the model will be tested on a test set to evaluate the model performance.

8) Inference: Inference is the process of utilizing a trained neural network to infer a result. In this case, a new set of current and voltage data is input through the LSTM estimator and it outputs a estimate of force. The time taken for inference plays a crucial part for successful implementation of LSTM estimator on Ambee robots.

For every 0.1 seconds, a new set of current and voltage data is fed to the LSTM estimator. Hence, it must provide a force estimate within a duration of 0.1 seconds, within the arrival of next set of data. That is, the time taken for inference by the LSTM estimator must be less than 0.1 seconds. The training and inference model can be seen in figure 15.

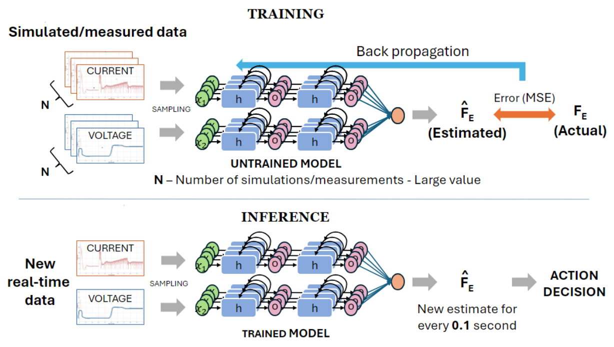


Fig. 15: Configuration: Training vs inference.

III. Results and Discussion

A. Approach 1: Classical Disturbance Observer Method

This approach was based on [2], which proved that the disturbance-observer concept is usable for compliance control. Hence, the first approach focused on implementing this concept to the robot motion platform of Ambee. However, the implementation of this concept was not successful due to the reasons listed below.

- The parameterization of variables is complex to work out for the Ambee robot motion system. The process of parameterization also introduces additional errors in estimation of external force.
- It seems difficult to include the nonlinear friction part in the model, hence possibly induce errors and difficulties in distinguishing the friction forces and the external force.
- The skill-set of the team is not aligned to develop a complex simulation environment/implement the solution in the available time-span.

There are some differences in dynamics between the Ropod platform from [2] and the Ambee Robot Motion platform. So, the derivation for the Ambee Robot platform is slightly different. An attempt to do the same derivation is made for the Ambee robot motion platform, but the parameterization of the variables was too complex for the team. The derivation until the parametrization part can be seen in appendix, section VI-C. The differences are discussed such that it enables others to work it out further. So, there are no results in sense of experiments, but some work is done which can be worked out further.

B. Approach 2: Estimation of External Force using Machine Learning

This section will discuss the results of approach 2, consisting of five subsections. The first three subsections show the training and testing process of the model within Case 6, which is the most complex case. It begins with hyper-parameter search, discussing the selection and optimization of model hyper-parameters. The training and validation subsection explains how to train and evaluate the model without over-fitting or under-fitting. The test subsection evaluates the model's performance using an independent test dataset. The fourth subsection compares and discusses the results for different cases. The section concludes with a discussion of the computational and temporal resources required for the estimation of external forces.

1) Hyper-parameter search: A search range is obtained experimentally. Through constant experimentation, it was eventually narrowed down from a very large range to the range listed below. Within this range, the network can fit the data. The search ranges for the parameters are as follows:

- For hidden states size, the search range varies from 80 to 152 with an interval of 24.
- For hidden layers size, the search range varies from 1 to 3 with an interval of 1.
- For learning rate, the search range varies from 0.001 to 0.003 with an interval of 0.001.

Based on the experiments, it was observed that the model begins to converge after approximately 10 epochs. Therefore, the results after 10 epochs is used as the evaluation metric for the hyper-parameter search. After 10 epochs of training, it was found that a network with a 80 hidden states, 2 hidden layers, and a learning rate of 0.003 achieved the best performance. The results of the hyper-parameter search with a learning rate of 0.003 are shown in figure 16.

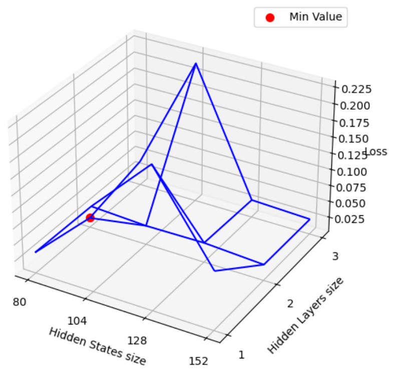


Fig. 16: Loss for different hidden states and layer (Learning rate $= 0.003$).

2) Training and validation: The relationships between the loss function values and the training epochs on training set and validation set are depicted in figure 17. As training epochs progress, the loss function value on the training set gradually decreases, while the loss function value on the validation set does not increase rapidly. This indicates that the model generalized well without over-fitting training data. And it can be observed that after 50 epochs, the model's improvement is not significant. Therefore, the result from the 50^{th} epoch is selected as the final model.

3) Test: On the test set, the model's loss function value, namely Mean Squared Error (MSE), is 2.54×10^{-3} (for case 6). Using the model to estimate the external force every 0.1s, the estimated external forces, along with the actual external forces, are illustrated in figure 18. It is evident that the model

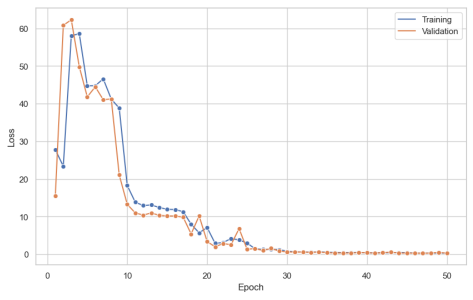


Fig. 17: Training and validation loss.

is able to approximate the mean value of the external force over 0.1s.

4) Results obtained for different cases: The results obtained for the six cases in table I is summarized in the table II.

By comparing the noise-free cases (1, 3, and 5) and the noisy cases (2, 4, and 6), it can be observed that as the complexity of the data increases, the corresponding MSE values also show an increasing trend. However, overall, the MSE remains quite low. This indicates that the model is able to effectively estimate the target data (external force) from the input data (current and voltage) in all scenarios. In addition when comparing the noise-free case and the noisy case of the same scenario (e.g. case 1 and case 2), it is observed that the addition of noise to training data makes the model more robust and generalized, leading to a decrease in MSE.

5) Inference: Currently, the LSTM estimator performs a total of 7,808,081 MAC (Multiply and accumulate) operations for a single inference. As explained in section II-D8, its latency must be less than 0.1 seconds.

The time of inference of the LSTM estimator was found to be 0.01 seconds, where it was run using tensor cores of NVIDIA GeForce RTX 4060 GPU, which has a peak computing capability of 242 TOPS (Tera operations per second).

As seen in section II-B Ambee uses Nvidia Jetson AGX Orin module in their robots. These modules can deliver up to 275 TOPS of AI performance with power configurable between 15W and 60W. The tensor computational capability of this module (275 TOPS) is comparatively higher than NVIDIA GeForce RTX 4060 GPU (242 TOPS).

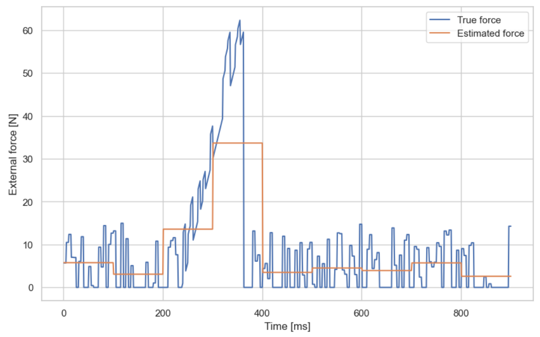


Fig. 18: Estimated and actual (external) force.

TABLE II: Loss (test) for different cases

Scenario		Case	MSE loss (approx.)	RMSE (approx.)
S1	The robot is moving in a constant speed and external force is applied at time, t = 0.5 second	1	8 x 10^{-3}	8 x 10^{-2}
		2	2.5 x 10^{-4}	1.58 x 10^{-2}
S2	The robot is accelerated at a specific point of time and external force is applied at 0.5 second	3	2.1 x 10^{-3}	4.58 x 10^{-2}
		4	1.01 x 10^{-2}	1 x 10^{-1}
S3	The robot is accelerated at a specific point of time and the external force is applied at random at a random time t, where t $\in$ [0.125, 0.875]	5	2.56 x 10^{-2}	1.6 x 10^{-1}
		6	2.54 x 10^{-3}	5.04x 10^{-2}

Thus, it can be concluded that the time taken for inference by the LSTM estimator must be less than 0.01 seconds, when run on Nvidia Jetson AGX Orin. Therefore, the LSTM estimator can be successfully used for inference in Ambee robots.

IV. Conclusion and Recommendations

The main aim of this project was to explore options for designing and implementing a compliance controller for the robot motion platform. This requires the estimation of the external force acting on the motion platform, which was considered as the main objective of this project. Three requirements (R1, R2 and R3) were defined to achieve this objective. An optional objective, to implement the controller, was defined among the preferences. Two different approaches (approach 1 and approach 2) as defined in the sections II-C and II-D, were explored to solve the main objective, among which approach 2 was implemented successfully in a proof of concept with convincing results.

From the results of approach 2, it can be inferred that requirement 1, to estimate the magnitude of external force is fulfilled and the LSTM based estimator shows compelling performance. This is evident from the test loss summarized in table II and figure 18. Since the ML model is implemented using Python based packages, requirement 3 is also satisfied. Furthermore, approach 2 does not require any additional sensors for implementation, thereby satisfying the first preference. Additionally, from section III-B5, it can be concluded that the LSTM estimator can be implemented on Nvidia Jetson AGX Orin and this satisfied the constraint.

Requirement 2 involves estimating the direction of external force. Currently, no work is done regarding this requirement. However, the current implementation can be extended to estimate the direction of force and the methodology is discussed in future work.

Apart from the promising results, approach 2 also has certain advantages which are listed below.

- One of the major drawbacks of approach 1 is the difficulty of including friction to the dynamical model. This, in-turn is expected to induce additional errors during force estimation. However, in approach 2, the data (either simulated or measured) used to train the model captures the non-linearity due to friction. The LSTM model is capable of learning those patterns and can provide an estimate with a very low error, compared to approach 1.

- Approach 2 can be extended to any other similar robotic systems. This is because, LSTM model can learn the patterns of various electric motor parameters (time-series data) such as overall current, voltage, phase currents, measured torque, measured speed or any combinations of above parameters and can provide the estimated force.
- The LSTM based estimator is robust to sensor noise that may occur during data collection (measurement of external force). The LSTM estimator can be trained along with this noise, in addition to noise due to friction. As a comforting prospect, this makes the model more robust, as noted by Kyongmin Yeo [7].
- Compared to Transformers, which are computation heavy, the proposed LSTM estimator is a simpler model with comparatively fewer parameters. This leads to reduced MAC operations (both during training and inference), leading to less power consumption. This is critical for mobile robotic systems which are usually powered by batteries.

However, the LSTM based estimator also has some limitations listed below.

- For training the LSTM estimator, actual value of external force is required. Hence, it was assumed that the external force can be measured using a force sensor in any one of the robots (assumption 7). Thus, approach 2 can be successfully implemented only if the assumption A7 holds true. In certain cases, the overall torque coupled to the motor can be measured. This can be used to train the model instead of using external force. However, now the LSTM estimator estimates the output torque. Otherwise, a simulation setup is required, which mimics the actual system precisely. If above options are infeasible, only an LSTM based classifier which can predicts the presence of an external force can be implemented.
- Data plays a key role in approach 2. Also, the accuracy of the model improves with the quantity and diversity of data. This requires measuring (external force), recording (current and voltage) and storing large amounts of data for a wide range of external forces and it is a problem. This can be eliminated by utilizing simulated data. However, the simulation setup must exactly replicate the behavior of original system, which is not the case in most of the situations.

Following is the list of future work that can be carried out to extend/the existing solution.

- The current implementation can also be extended to estimate the direction of force as well (requirement 2), i.e. expanding the implementation from multiple dimensions. The external force applied on the robot not only acts (increases load) on the direct drive motor but also on the motor that controls direction/turn (rotation motor). By training and tuning the proposed LSTM estimator on the current and voltage data of both motors, the expectation is that this estimator is suitable for estimating the direction of the external force as well. This is a relevant option to work further on.
- Objective 2, designing the controller itself, is not done yet. A 1D controller can be designed using the presented solution
- At present, the LSTM model has approximately 1 million trainable parameters (Example: 934041 parameters, based on hyper-parameter tuning for case 6. This value is subjected to change based on data). This can be decreased by network pruning, thereby reducing overall MAC operations, computational cost, time for inference and power consumption without much loss in accuracy or performance.

As this project involves implementing AI-based solutions for robots working in the hospital environment, the ethical risks that may arise are carefully considered and discussed.

V. Ethical risk management

Key ethical risks identified for this project include privacy and data protection, decision-making transparency, and responsibility. The following measures have been implemented/proposed to mitigate these risks effectively.

1) Privacy and Data Protection: This project collects only motor parameters and external force values, with no user data involved. All data is stored and processed locally. The robot's stereo cameras are not used for this project, so ensure unauthorized access to visual data is prevented and conduct regular checks for data protection. Additionally, regular checks should be conducted during data collection to verify the effectiveness of data protection measures.

2) Transparency: The robot's decision-making process based on the results of LSTM estimator should be documented. Each decision step including data collection, modeling, training and justification should be recorded to maintain transparency. Also, since Neural Networks are opaque in their operation, care must be taken to make it more understandable and transparent. This helps in understanding and verifying the robot's actions for various real-world scenarios.

3) Responsibility: As the control action is based on the results from LSTM estimator, which is an AI tool, care should be taken to determine who is accountable when a robot causes harm or makes a wrong decision. Also, the degree of independence granted to robots and the ethical implications of their decision-making based on the output from AI tool should be regulated.

4) Trust and safety: A comprehensive classification of different dimensions of risk must be carefully analysed before implementing the AI based solution (LSTM estimator) in Ambee robots.

Through these measures, the project's success along with the safe, effective, and ethically compliant operation of the robot in hospital environment can be ensured.

References

[1] Sabtain Ahmad et al. “Autoencoder-based Condition Monitoring and Anomaly Detection Method for Rotating Machines”. In: *Proceedings - 2020 IEEE International Conference on Big Data, Big Data 2020*. 2020. DOI: 10.1109/BigData50022.2020.9378015.

[2] R.M Beumer. *MASTER Automated Support of Human-Induced Movements of a Swivel-Wheeled Mobile Robot*. Tech. rep. 2021.

[3] Simon Haykin. *Neural Networks and Learning Machines*. Vol. 3. 2008. DOI: 978-0131471399.

[4] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997). ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735.

[5] Chung-Wen Hung and Guan-Yu Jiang. “Application of External Torque Observer and Virtual Force Sensor for a 6-DOF Robot”. In: *Applied Sciences* 13.19 (2023). ISSN: 2076-3417. DOI: 10.3390/app131910917. URL: https://www.mdpi.com/2076-3417/13/19/10917.

[6] Nederlands Interdisciplinair Demografisch Instituut and Centraal Bureau voor de Statistiek. “Bevolking 2050 in beeld: opleiding, arbeid, zorg en wonen”. In: (2050).

[7] Kyongmin Yeo. “Short note on the behavior of recurrent neural network for noisy dynamical system”. In: *arXiv preprint arXiv:1904.05158* (2019).

[8] Willem-Jan Lamers and Joep Selten. “Initial project proposal”. 2023.

[9] Jonghwan Son, Chayoung Kim, and Minjoong Jeong. “Unsupervised Learning for Anomaly Detection of Electric Motors”. In: *International Journal of Precision Engineering and Manufacturing* 23.4 (2022). ISSN: 20054602. DOI: 10.1007/s12541-022-00635-0.

[10] Ahmed Tealab. “Time series forecasting using artificial neural networks methodologies: A systematic review”. In: *Future Computing and Informatics Journal* 3.2 (2018). ISSN: 23147288. DOI: 10.1016/j.fcij.2018.10.003.

[11] J J E Unkel. *ROPOD: FORCE SENSORLESS COMPLIANT CONTROL FOR MOBILE ROBOTS*. Tech. rep. 2018.

[12] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in Neural Information Processing Systems*. Vol. 2017-December. 2017.

[13] Shouhua Zhao et al. “An improved torque feed-forward control with observer-based inertia identification in PMSM drives”. In: *ICEMS 2012 - Proceedings: 15th International Conference on Electrical Machines and Systems*. 2012. ISBN: 9784886860774. DOI: 10.11142/jicems.2013.2.1.69.

VI. Appendix

A. Repository

Version control for this project is managed using Github. A repository, that holds the python code, simulation files and other required files used for this project. The structure and usage of this GitHub repository can be found in the repository’s *readme.md* file. Following is the link to GitHub repository.

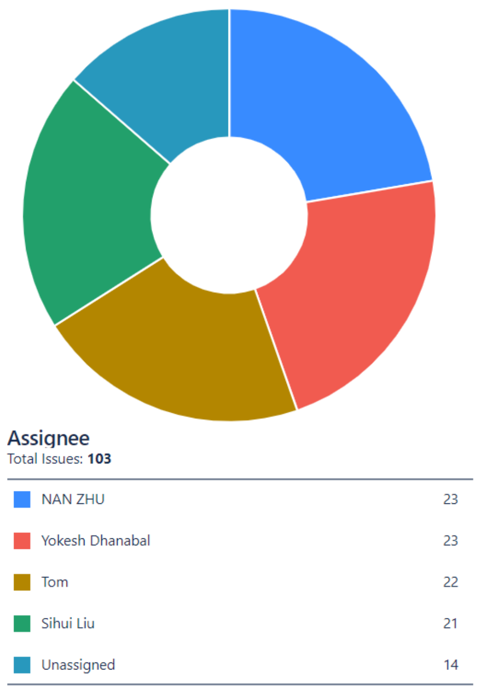
https://github.com/0error1000warning/Hospital-Robots.

B. Team work and individual contribution

1) Project management: Jira was used as the project management tool for the whole course of this project. During weekly team meetings, based on the discussion, the future tasks are finalized and are broken down as action points. Each action point is added as issues in Jira Kanban board with the responsible person added as an assignee. These issues may include both technical and non-technical tasks related to the project.

By this way, the issues were created, managed, tracked and categorized based on weekly progress of the team. Priority labels and deadlines are added to the issues of high importance. Total of 103 issues have been created during the project. Figure 19 shows the pie chart (generated using Jira) which depicts the distribution of issues (tasks) among the team members. The unassigned tasks are those for which the whole team is responsible.

2) Collaboration: In total, 25 team meetings were conducted in a weekly manner. This number also includes bi-weekly meetings with project owners, in the overall course of this project. The weekly reports, monthly time-sheets, presentations (used for mid-term and bi-weekly meets with Ambee), meeting agendas, team agreements and other common documents were maintained using a sharepoint folder. This folder can also be accessed by project owners whenever required.



Assignee
Total Issues: **103**

🟦	NAN ZHU	23
🟥	Yokesh Dhanabal	23
🟨	Tom	22
🟩	Sihui Liu	21
🟦	Unassigned	14

Fig. 19: Issue/Task distribution among the team

3) Skills training: The whole team took an active participation in every skills-training workshop. An online document to record key take-away/points of improvement attained by attending the training sessions, was created and maintained.

4) Individual contribution: In addition to common tasks such as presentations, workshops, meetings and report writing, the individual contribution of each team member to the success of the project is listed below.

1) Yokesh Dhanabal (2011468)
   - Devised the overall concept for approach 2.
   - Pre-dominantly worked on the task of data generation in approach 2.
   - Worked on simulation setup and modelled time-varying load/torque for all six cases and setting up of multiple simulations.
   - Organized the task of data generation (for the six cases) among the team, utilizing their local devices to accelerate the process.
   - Worked in training and hyper-parameter tuning of LSTM estimator for case 6, in approach 2.
   - Worked on parameterization of the dynamical model, for approach 1.
   - Worked on the development of pygame based simulator for approach 1.
   - Worked on literature research for both approaches.
   - Worked on creating interactive presentations for bi-weekly meetings with Ambee.
   - Report work - Sections: II-D (intro and assumptions), II-D1, II-D2, II-D3, II-D4, II-D5, II-D8, III-B (Equal contribution along with Nan, Zhu), IV, V and appendices - VI-B1, VI-B2, VI-B3, VI-E, VI-F and VI-G. Provided feedback for possible improvements in other sections.
2) Tom Minten (1372300)
   - Team leader in Q3, organized weekly meetings and communication with Ambee
   - Learned other teammembers how to approach this kinds of projects with the proposal and Gantt charts, and in the meetings. Also tried to learn members to think in a more critical way by e.g. asking how/what/why
   - Did a major part in defining the project in terms of objectives and associated requirements, preferences and constraints, learned other groupmembers how to work on this.
   - Started with planning, which was later taken over by Sihui
   - Literature research
   - Made the derivations and discussions for approach 1
   - Worked on the simulation setup for approach 1
   - Did some work on the simulator of approach 2.
   - Did several heavy simulations and trainings
   - Github reposity manager in Q3
   - Report work, gave much feedback on others
3) Sihui Liu (2018004)
   - Project planner in Q3 and Q4, plan and manage the project timeline and update the Gantt chart.
   - Literature research
   - Worked on the design of disturbance observer in approach 1.
   - Worked on the design and implementation of control strategy in approach 1.
   - Worked on organizing the documentation and notes related to the report.
   - Improve the readability of the report based on the feedback provided by the writing support, meetings and align the report and other submissions according to rubrics.
4) Nan Zhu (2044331)
   - Created and managed the GitHub repository in Q3.
   - Team leader in Q4, organizing weekly meetings and communication with Ambee.
   - Literature research.
   - Participated in the design and implementation of the disturbance observer in approach 1.
   - Writing neural network-related code for approach 2, including hyper-parameter search, training, validation, testing and inference.

5) Project timeline: The Gantt chart (figure 20) provides an overview of the overall project timeline, which is divided into two primary periods, Q3 and Q4. The focus in the Q3 is approach 1 which involves a Disturbance Observer to detect external force. Based on feedback from the intermediate presentation and the fact that approach 1 cannot include friction, the project's focus shifted to Approach 2 in Q4, to provide an AI based solution.

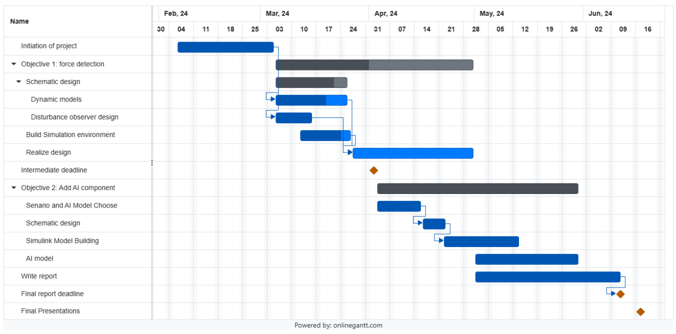


Fig. 20: Gantt chart

C. Detailed derivation of the dynamical model

In this Appendix, the dynamical model used for the first approach is discussed.

In the master thesis of R.M. Beumer [2], a modular dynamic model is given for of the Ropod, which has 8 actuated wheels in groups of 2. In figures 21 and 22, schematics of the Ropod and the wheel setup are shown. The modular dynamical model is rewritten in a solvable system of equations, which enables dynamic parameter identification. Although the setup of the Ambee robot is different, the same derivations should be possible. As stated by [2], the resulting direct equations of motion van be rewritten in difference forms to enable several applications. Hence, these equations give a highly flexible fundamental dynamical model to work with during this project. On top of that, this set of equations already has proven to be suitable for compliance motion problem with automatic parameter identification. Some important remarks are also made, which has to be kept in mind using this dynamical model.

1) The estimation of the dynamical parameters in the equations of motion has a low error. But, the identified mass values were less accurate, which indicates that some missing dynamical effects are still missing in the model, that are compensated with the less accurate mass values. A primary idea is using a machine learning to improve estimation of the parameters, which should make the mass estimation more robust. This is considered important due to the highly influencing masses attached to the Ambee robot.
2) [2] uses velocity based control, which is the same as used at the Ambee robot. As stated by [2], force based control is more intuitive, but requires a very accurate dynamical model to ensure good performance and stability.

In this report, a similar derivation for the Ambee will be obtained. The following notation is used for all coordinates in this section is $r^a_{A_1/A_2} = [x^a_{A_1/A_2}, y^a_{A_1/A_2}]^T$ to represent the coordinates of point P_{A1} with respect the origin, expressed in the directions $\overrightarrow{e}^a_x$ and $\overrightarrow{e}^a_y$ of frame a The same convention is used for their first and second time derivatives $\dot{r}^a_{A_1/A_2} = [\dot{x}^a_{A_1/A_2}, \dot{y}^a_{A_1/A_2}]^T$ and $\ddot{r}^a_{A_1/A_2} = [\ddot{x}^a_{A_1/A_2}, \ddot{y}^a_{A_1/A_2}]^T$ When using $r^a_{A_1/A_2}$, so with an additional point P_{A_2}, the coordinates of point A_1 are given with respect to this point A_2. Furthermore, the short notations s_{θ_b} and c_{θ_b} will be used for respectively $\sin(\theta_b)$ and $\cos(\theta_b)$ and similarly this will be done for the sine and cosine of θ_i and δ_i. [2]

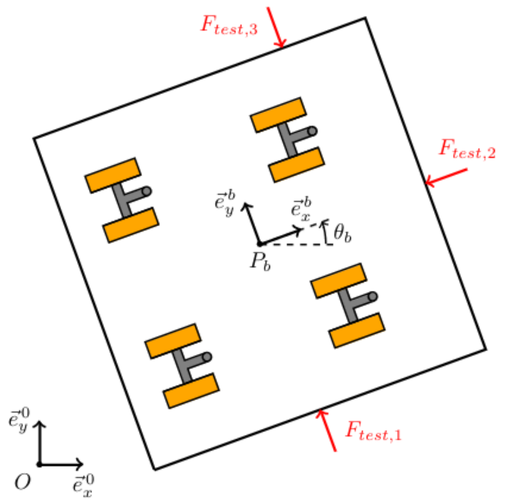


Fig. 21: Schematic of the Ropod wheel setup, from [2].

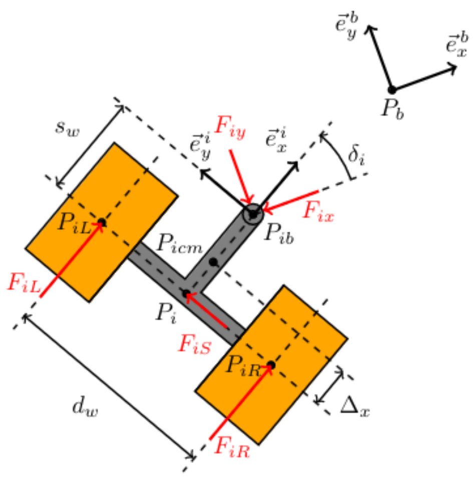


Fig. 22: Schematic of a Ropod wheel hub, from [2].

1) Base model: This part is similar to [2]. Note that the most of the derivations and explanations in this report are taken from [2]. A global frame base (b) of the Ambee robot is defined as $q^0_b = [x^0_b \quad y^0_b \quad \theta_b]$. The global coordinates are in fact not necessary for the control, but are used for practical reasons as simulations and debugging. All forces acting on the robot, including external as internal forces, is described by $F_b - [F_{bx} \quad F_{by} \quad T_{bz}]$, representing respectively the forces in the local x- and y-direction (through the center of mass) and the torque about the vertical axis. For the interaction forces with the Swerve Drive module Wheels (SDW) and their pivot points are denoted by $F_{ixy} = [F_{ix} \quad F_{iy}]^T$ for $i \in 1, 2$ for the two SDWs. Note that the third castor wheel is not taken into account, as it is not actuated and does not include sensors. The equations of motion of the robot are then given by:

$$M_B \ddot{q}^0_b = B_{F_b} F_b + B_{F_{1xy}} F_{1xy} + B_{F_{2xy}} F_{2xy} \tag{9}$$

in which M_b is the mass matrix of only the Ambee robot. The matrices B_{F_b} and $B_{F_{ixy}}$ convert the forces expressed in local coordinates to their contributions to the global directions.

$$\begin{aligned} M_b &= \begin{bmatrix} m_b & 0 & 0 \\ 0 & m_b & b \\ 0 & 0 & J_{zz,b} \end{bmatrix} \\ B_{F_b} &= \begin{bmatrix} c_{\theta_b} & -s_{\theta_b} & 0 \\ s_{\theta_b} & c_{\theta_b} & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ B_{F_{ixy}} &= \begin{bmatrix} c_{\theta_b} & -s_{\theta_b} \\ s_{\theta_b} & c_{\theta_b} \\ -y^b_{ib/b} & x^b_{ib/b} \end{bmatrix} \end{aligned} \tag{10}$$

Here, $x^b_{ib/b}$ and $y^b_{ib/b}$ represents the local coordinates of the point P_{ib} at which SDW i is attached to the base.

2) Kinematics: In this section, the ambee robot differs from the Ropod. In figure 22, the ropod wheel hubs are different due to their dual wheel layout, compared to the one-wheel layout of the Ambee robot. There is still a pivot point P_{ib}, but the wheel of Ambee is present at the kinematic center P_i

instead of $P_{iL/R}$. The coordinates of the picot point location of P_{ib} are:

$$\underbrace{\begin{bmatrix} x_{ib}^0 \\ y_{ib}^0 \end{bmatrix}}_{r_i^0 b} = \underbrace{\begin{bmatrix} x_b^0 \\ y_b^0 \end{bmatrix}}_{r_b^0} + \underbrace{\begin{bmatrix} c_{\theta_b} & -s_{\theta_b} \\ s_{\theta_b} & c_{\theta_b} \end{bmatrix}}_{R_b^0} \underbrace{\begin{bmatrix} x_{ib/b}^b \\ y_{ib/b}^b \end{bmatrix}}_{r_i^b b/b} \tag{11}$$

With their first and second derivative in [2]. The kinematic center P_i is defined as the centre of the wheel of the Ambee robot. Its coordinates are given by:

$$\underbrace{\begin{bmatrix} x_i^0 \\ y_i^0 \end{bmatrix}}_{r_i^0} = \underbrace{\begin{bmatrix} x_{ib}^0 \\ y_{ib}^0 \end{bmatrix}}_{r_i^0} + \underbrace{\begin{bmatrix} c_{\theta_i} & -s_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} \end{bmatrix}}_{R_i^0} \underbrace{\begin{bmatrix} -s_w \\ 0 \end{bmatrix}}_{r_i^i/ib} \tag{12}$$

In which $\theta_i = \theta_b + \delta_i$ and s_w is the caster offset. Taking into consideration that $r_{i/ib}^i$ is constant, the velocity and acceleration are calculated by taking the time derivatives, which are also available in [2]. The assumption of no slip is taken, meaning P_i can not move sideways. Mathematically, this non-holonomic constraint means that the projection of the velocity $\dot{r}_i^0$ of this point on the local side ward direction is zero.

$$-s_{\theta_i}\dot{x}_i^0 + c_{\theta_i}\dot{y}_i^0 \tag{13}$$

After substituting the derivatives of 11 and 12 in 13, the resulting equation can be rewritten to:

$$\dot{\theta} = \frac{1}{s_w}\Big(- s_{\theta_i}\dot{x}_b^0 + c_{\theta_i}\dot{y}_b^0 + \dot{\theta}_b\big(x_{ib/b}^b c_{\delta_i} + y_{ib/b}^b s_{\delta_i}\big)\Big) \tag{14}$$

The time derivative of 14 yields the angular acceleration of P_i around Pib, influenced by the movements of the base

$$\begin{aligned} \ddot{\theta}_i = \frac{1}{s_w}\Big(& - c_{\theta_i}\dot{\theta}_i\dot{x}_b^0 - s_{\theta_i}\ddot{x}_b^0 - s_{\theta_i}\dot{\theta}_i\dot{y}_b^0 + c_{\theta_i}\ddot{(y)}_b^0 \\ & + \ddot{\theta}_b\big(x_{ib/b}^b c_{\delta_i}{}^b_y{}_{ib/b} s_{\delta_i}\big) \\ & + \dot{\theta}_b\big(- x_{ib/b}^b s_\delta\dot{\delta}_i + y_{ib/b}^b c_{\delta_i}\dot{\delta}_i\big)\Big) \end{aligned} \tag{15}$$

Derivations of r_{icm}^0 and r_{iw}^0 are comparable to r_i^0.

$$r_{icm}^0 = r_{ib}^0 + R_i^0 r_{icm/ib}^i \tag{16}$$

$$r_{iw}^0 = r_{ib}^0 + R_i^0 r_{iw/ib}^i \tag{17}$$

3) Wheel dynamics: From this section onwards, the derivation will slightly differ compared to [2] due to the one wheel case. In [2], $w \in L, R$ represents the left or right wheel. The location of the wheel is given by $r_{iw}^0 = r_{ib}^0 + R_i^0 r_{iw/ib}^i$. In the Ambee robot case of one wheel, one gets $w \in 1$ and $R_i^0 r_{iw/ib}^i = 0$.

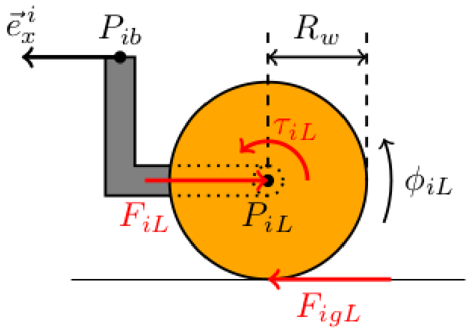


Fig. 23: The schematic of the wheel, from [2]

In figure 23, the schematics of the left (L) wheel from the ropod is given. No slip in longitudinal direction is assumed, resulting that the local forward velocity of point P_{iw} is directly coupled to the rotational velocity $\dot{\phi}_{iw}$ of the wheel:

$$R_w\dot{\phi}_{iw} = c_{\theta_i}\dot{x}_{iw}^0 + s_{\theta_i}\dot{y}_{iw}^0 \tag{18}$$

in which R_w is the wheel radius. Hence, dividing by R_w and taking the the derivative with respect to time yields the angular acceleration of the wheel:

$$\ddot{\phi}_{iw} = \frac{1}{R_w}(-s_{\theta_i}\dot{\theta}_i\dot{x}_{iw}^0 + c_{\theta_i}\dot{\theta}_i\dot{y}_{iw}^0) \tag{19}$$

The most intuitive way to derive the equation is to first include the mass m_w of the individual wheel in this model and then set it equal to 0. In the local longitudinal direction $\overrightarrow{e}_{xi}$, the acceleration is determined by the interaction force with the wheel hub and the reaction force from the ground:

$$m_w\ddot{x}_{iw}^i = -F_{iw} + F_igw \tag{20}$$

Since the interaction force with the wheel hub has no arm with respect to the wheel axle, the angular acceleration is determined by the wheel input torque and the reaction force from the ground:

$$J_{rot,w}\ddot{\phi}_{iw} = \tau_{iw} - F_{igw}R_w \tag{21}$$

However, the rolling and translational accelerations are moment of inertia about the vertical axis trough the coupled when no slip is assumed: $\ddot{x}_{iw}^i = R_w\phi_{iw}$. Substituting this into 20 and solving F_igw yields:

$$F_{igw} = m_wR_w\ddot{\phi}_{iw} + F_{iw} \tag{22}$$

Substituting 22 in 21 and setting the individual wheel mass for this part of the model equal to zero, results in

$$\ddot{\phi}_{iw} = \frac{1}{J_{rot,w}}(\tau_{iw} - F_{iw}R_w) \tag{23}$$

Note that the interaction force computed in this way is different then the interaction force that would be computed if the mass of the individual wheel was not taken as zero for this part. As the mass will be contained in the dynamics of the wheel hub, this does not have an influence on the dynamics described by the model as a whole, Therefore, this difference is only relevant when the value of this specific interaction should be known.

4) Wheel hub dynamics: The forces considered are the sum F_{iS} of the lateral interaction forces with the wheels (as they work along the same axis) and the interaction forces F_{ix} and F_{iy} with the platform base in the two horizontal directions. Then, the equations of motion for the wheel hub of the ambee robot are:

$$\underbrace{M_{wh}\begin{bmatrix}\ddot{x}_{icm}^{0}\\ \ddot{y}_{icm}^{0}\\ \ddot{\theta}_i\end{bmatrix}}_{\ddot{q}_i^0} = \underbrace{A_{iwh}\begin{bmatrix}F_{ix}\\ F_{iy}\\ F_{iS}\end{bmatrix}}_{\text{Interactionforces}} + A_{iwh2}F_{iw} \tag{24}$$

in which

$$M_{wh} = \begin{bmatrix} m_{wh} & 0 & 0\\ 0 & m_{wh} & 0\\ 0 & 0 & J_{wh,zz}\end{bmatrix} \tag{25}$$

$$A_{iwh} = \begin{bmatrix} -c_{\theta_b} & s_{\theta_b} & -s_{\theta_i}\\ -s_{\theta_b} & -c_{\theta_b} & c_{\theta_i}\\ s_w s_{\delta_i} & s_w c_{\delta_i} & 0\end{bmatrix} \tag{26}$$

$$A_{iwh2} = \begin{bmatrix} c_{\theta_i}\\ s_{\theta_i}\\ 0\end{bmatrix} \tag{27}$$

in which m_{wh} is the mass of the wheel hub including the masses of the individual wheels attached to it. Similarly, $J_{wh,zz}$ is the moment of inertia about the vertical axis through the center of mass of the wheel hub, which is the same as the kinematic point, including the individual wheels/

5) Total platform dynamics: [2] proposes a method which reduce the system to a form in which the relation between wheel inputs an dplatform base accelerations can be described directly, by means of eliminating all additional variables with a step-by-step approach. These additional variables are the interaction forces and intermediate coordinates, velocities and accelerations. Besides (and due to) the fact that this omits the need of a solver, it complies with the desired modularity and scalability of the model.

The derivation of the reduced set of equation of motions using this method is quite extensive and is left out of this report. One can find it in [2]. The only difference is that there is only one wheel instead of two. The results of the parameters are substituted in (9)

$$M_{dir}\ddot{q}_b^0 = B_{F_b}F_b + B_\tau\tau + b_{dir} \tag{28}$$

$$\underbrace{M_{dir}\ddot{q}_b^0}_{\text{Mass/Inertia}} = \underbrace{B_{F_b}F_b}_{\text{External forces}} + \underbrace{B_\tau\tau}_{\text{Wheel input}} + \underbrace{b_{dir}}_{\text{Coefficient}} \tag{29}$$

The authors were not able to work out this parameterisation for the Ambee robot motion platform. This is essential due to some small differences between the Ropod and the Ambee robot motion platform. One has to follow the parameterization rules to write the slightly different motion platform dynamics in a function of q_b and its time derivatives, δ_i for $i \in \{1,2\}$ and $\tau = [\tau_1, \tau_2]$.

6) Dynamical parameter identification: The obtained equation in (29 can be rewritten in a form in which the dynamic parameters m_b , $J_{zz,b}$, m_{wh} , $J_{zz,wh}$ and $J_{rot,w}$ are isolated:

$$M_{dir}\ddot{q}_b^0 = \underbrace{B_{F_p}F_p + B_\tau\tau}_{b_{ID}} + b_{dir} \tag{30}$$

This gives a model

$$M_{dir}\ddot{q}_b^0 - b_{dir} = b_{ID} \tag{31}$$

$$A_{ID}\underline{m} = B_{F_p}F_p + B_\tau\tau = b_{ID} \tag{32}$$

in which $\underline{m} = \begin{bmatrix} m_b & J_{zz,b} & m_{wh} & J_{zz,wh} & J_{rot,w}\end{bmatrix}^T$

Due to parameterization, A_{ID} only a function of base position q_b and its time derivatives, wheel hub angles δ_i for $i \in \{1,2\}$ wheel hubs and torques $\tau = [\tau_1, \tau_2]$ on both drive motors.

This is a quite interesting way to obtain a model with only some relevant parameters.

In [2], there is used a simple mean square error minimization technique to obtain an approximation of $\underline{m}$. In this way, the mass and inertia of the robot are obtained. These parameters are put back in the motions of equations of 29 and used as prediction model for the disturbance controller.

To conclude, the parameters of A_{ID}, m_{wh}, $J_{zz,wh}$ and J_rot, w can be learned offline, while m_b and Jzz, b have to be adapted during driving due to different loads on the robot. The simple mean square error minimization technique can be expanded with more advanced learning models and more test data.

D. DOB pseudo-code

```
# Set system parameters
M_dir = ...
B_T = ...
B_Fb = ...
# Set filter parameters
Q = ...
# Initialize system variables
q = 0
q_dot = 0
q_ddot = 0
tau = 0
Fb_hat = 0

# Implement filters and estimators
  using discretization method
# These formulas correspond to the
  discrete −time implementation of transfer functions
def discrete_filter ( last_value , new_value, dt):
    alpha = dt / (Q + dt)
    return alpha * new_value + (1 − alpha) * last_value

def update_system(q_meas, tau_input , dt):
    global q, q_dot, q_ddot, Fb_hat
    # Update system state using sensor measurements
```

```
q_ddot_new = (q_meas − 2*q + q_prev) / dt**2
q_dot_new = (q_meas − q_prev) / dt
q_prev = q
q = q_meas
# Inverse system model calculation
inverse_dynamics = (M_dir * q_ddot_new − B_T *
    tau_input) / B_Fb

# Disturbance estimation
Fb_hat = discrete_filter (Fb_hat, inverse_dynamics,
    dt)
# Use the estimated disturbance force to
 calculate the compensating torque
tau_compensated = tau_input + Fb_hat
# Return the compensated control input and
    disturbance force estimation
return tau_compensated, Fb_hat
```

E. BLDC motor characteristics

Unlike traditional DC motors, BLDC motors are electronically commutated, which means they use electronic switching to control the phase changes in the motor winding. The rotor of a BLDC motor is made up of permanent magnets and they interact with the stator's electromagnetic field to produce rotational motion. Figure 24 shows some of the main components of BLDC motor described above.

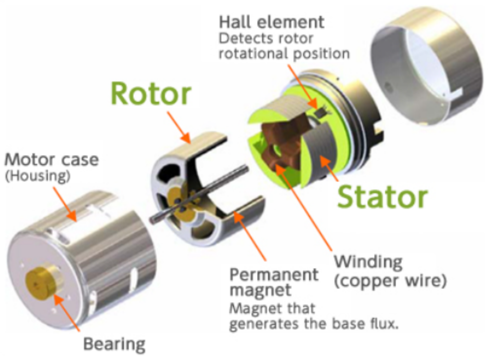


Fig. 24: BLDC motor and its parts

The rotational speed of BLDC motor can be calculated using the equation 33.

$$N = K_v \cdot V \tag{33}$$

Here N is the speed in RPM, K_v is motor's the KV rating (RPM per volt) which indicates how many RPM the motor will turn per volt applied and V is the supply voltage. The torque (T) generated by a BLDC motor can be calculated using the equation 34.

$$T = K_t \cdot I \tag{34}$$

Here, T is the torque in Newton-meters (Nm), $K_t is$ is the motor torque constant in Nm/A and I is the current in Ampere (A). The relation between speed and torque of a BLDC motor is given by equation 35.

$$T = \frac{P_{mech}}{\omega} \tag{35}$$

Here, P_{mech} is the mechanical power in watts (W), T is the torque in newton-meters (Nm) and ω is the angular velocity in radians per second (rad/s).

Equation 33 states that the speed of the motor is directly proportional to the Voltage. From equation 35, it can be seen that an increase in load (torque) decreases the speed of the motor. This in-turn forces the controller to compensate the decrease in speed by increasing the voltage (V) supplied to the motor. This increase is load also increases the stator current which can be inferred from equation 34.

It's important to note that the above formulae assume ideal conditions without accounting for losses and inefficiencies in the motor and speed controller, and just used to explain the overall concept.

F. Detailed description of simulation setup

The simulation setup consists of a BLDC based electrical drive and a load torque. The motor drive and its components were adopted from Simulink example, BLDC Motor Speed Control with Cascade PI Controllers where an ideal torque source acts as load to motor. This example uses the following Simulink add-ons listed below, which must be installed for running the simulation.

- Simscape Electrical
- Simulink Control Design
- Simulink
- Simscape

The details of the motor drive are as follows. The buck converter is modeled with MOSFETs and the inverter with IGBTs. Both the voltages of the DC-DC converter link and the inverter can be controlled by changing the semiconductor gate triggers, which control the speed of the motor. The Control subsystem uses a PI-based cascade control structure with an outer speed control loop and an inner dc-link voltage control loop. The dc-link voltage is adjusted through a DC-DC buck converter. The BLDC is fed by a controlled three-phase inverter. The gate signals for the inverter are obtained from hall sensors.

The above model is modified for this project, to measure current and voltage of the motor, by adding respective sensors. The ideal torque is replaced by the varying torque load (explained in section II-D3) in this project. Figure 25 shows the Simulink model used for case 1 (refer table I).

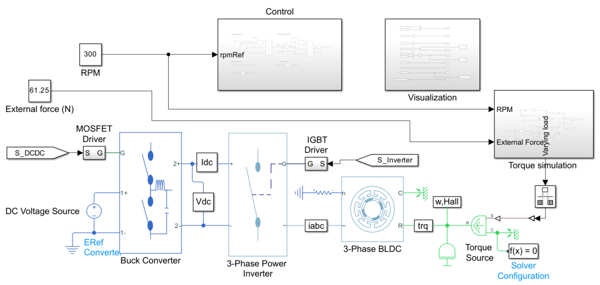


Fig. 25: Simulink model for case 1

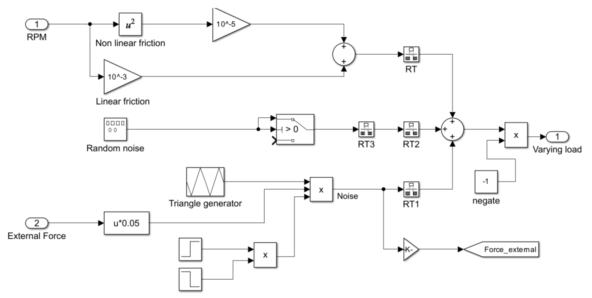
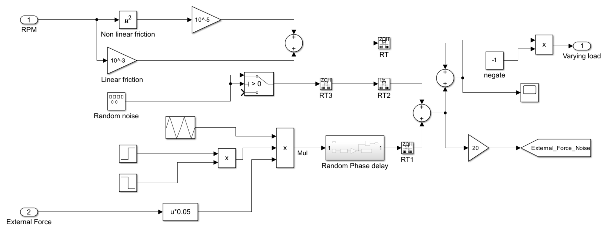


Fig. 26: Torque simulation block

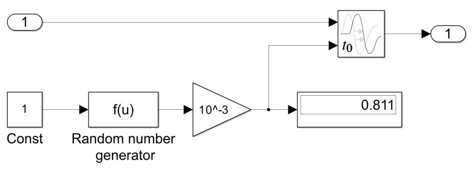
The time-varying torque is provided by the Torque Simulation block as seen in figure 25. A more detailed view of this block can be seen in figure 26. The noise in the torque is added using the Simulink Function Generator in random mode, which generates normally distributed random numbers. Non-linear friction is modelled as a square function, where the torque increases quadratically with respect to speed. For linear friction, a constant block with a value of 0.001 is used. Thus, the torque due to linear friction is obtained by scaling down the speed by a factor of 0.001.

The required data is logged and stored in desired location (within the local device) withing the Visualization block. This is done by using the 'To File' block, which incrementally write data into a variable and stores it in a specified path, as a MAT-file.

For case 5 and case 6, the external force is applied at an random time t, where t ∈ [0.125, 0.875]. For this, a Random Phase delay block is introduced within the torque simulation block, as seen in figure 27a. This phase delay block applies the force (torque) at a random time within the above time interval. A detailed view of torque simulation block can be seen in figure 27b.



(a) Torque simulation block with phase delay block



(b) Phase delay block

Fig. 27: Simulink model for Torque Simulation block with Random Phase delay block.

The simulink simulation (.slx) files and multiple simulation setup (.mldatx) files used for generating data for different cases is uploaded to github (refer section VI-A).

G. *Simulink - Multiple Simulations Panel*

The simulation setup described in section to VI-F is run multiple times to generate data, for all the six cases depicted in table I. The data is generated by varying the initial speed, final speed and external force for each simulation. Table III summarizes the number of simulation runs, and range of values of speed and external force, to generate data for different cases. The last column indicates whether both noise and actual force is used to train the ML model.

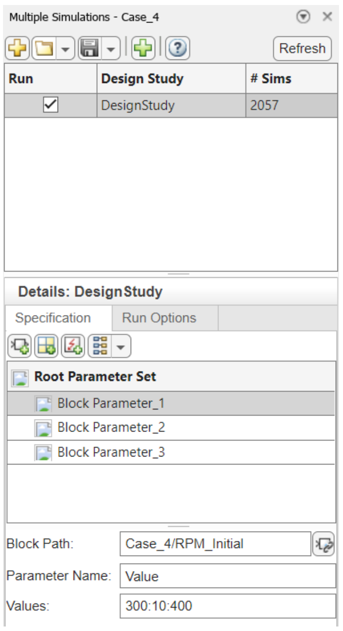
TABLE III: Number of simulations

Case	Number of simulations	Initial speed (in RPM)	Final speed (in RPM)	Training with noise
1	2074	300 - 400		No
2	2079	300 - 400		Yes
3	2541	300 – 400	500 - 600	No
4	1573	300 - 400	500 - 600	Yes
5	4961	300 – 400	500 - 600	No
6	4904	300 – 400	500 – 600	Yes

Figure 28 shows the image of Mutliple Simulations Panel of Simulink Editor. Using the panel, a design study can be created, where required block parameter(s) for which value is to be changes across simulations can be selected. By this way, the same simulation can be executed multiple times for different values of chosen block parameters.

The configuration (Design study) can be stored as multiple simulation (.mldatx) file. The multiple simulation files used for the various cases is uploaded in the repository (refer section VI-A).

The accuracy of ML model improves with an increase in training data. Ideally, a very high number of simulations is required, compared to the number of simulations depicted in the table. For example, running 2000 simulations takes about 12 to 14 hours in a 64-bit operating system, x64-based Intel(R) Core(TM) i5-8250U processor with 8GB RAM. Since running multiple simulations consumes a lot of time and requires high compute resources, the number of simulations was limited to the numbers as shown in the table. More focus was given to case 5 and case 6 and hence the higher number of simulations for those two cases.


Multiple Simulations - Case_4

Run	Design Study	# Sims
✓	DesignStudy	2057

Details: DesignStudy

Root Parameter Set
- Block Parameter_1
- Block Parameter_2
- Block Parameter_3

Block Path: Case_4/RPM_Initial

Parameter Name: Value

Values: 300:10:400

Fig. 28: Multiple Simulations Panel showing Design study (for case 4)