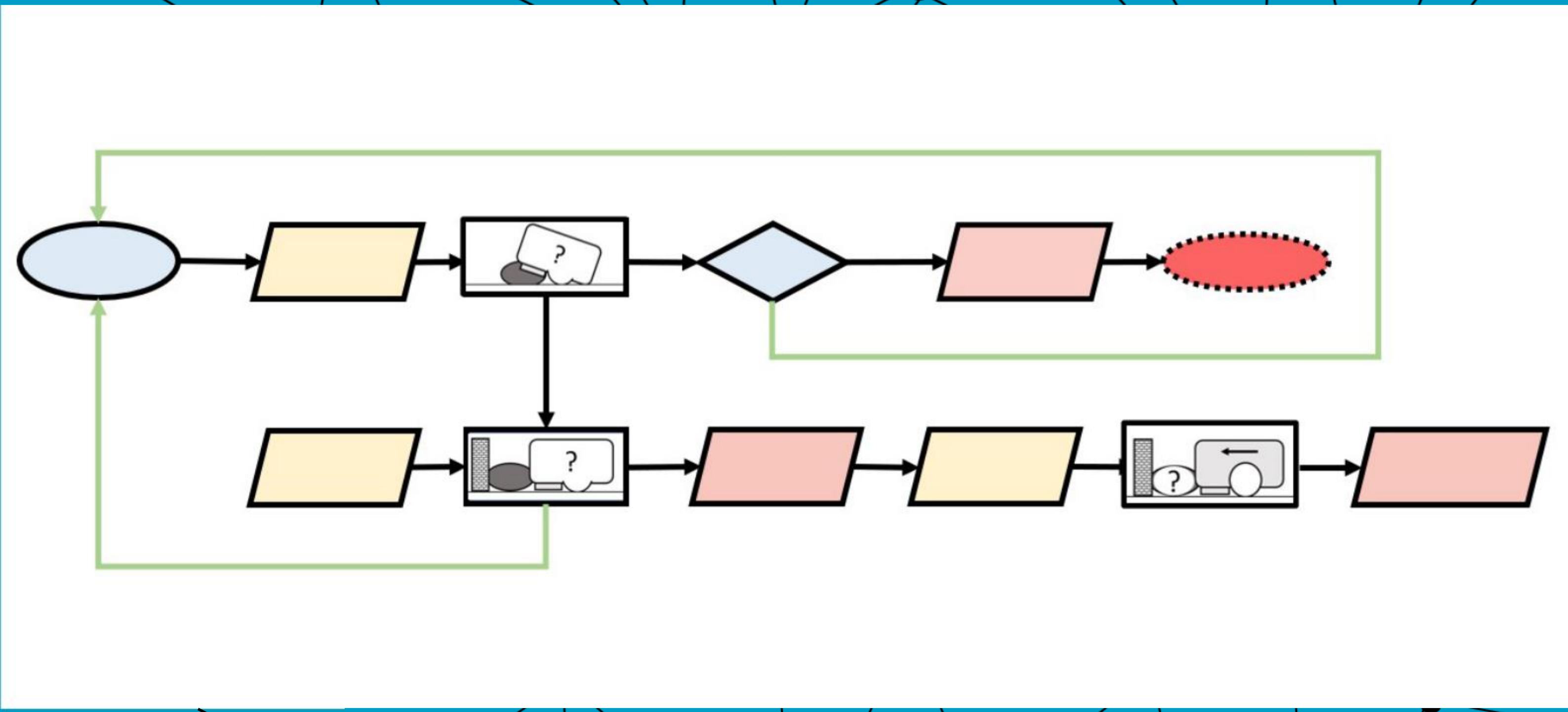
Blind collision detection and classification framework for a heavyweight, low-speed mobile robot colliding with soft and hard objects

E.C. Kooi



This page is left blank intentionally

Blind collision detection and classification framework for a heavyweight, low-speed mobile robot colliding with soft and hard objects

Master Thesis for the department of BioMechanical Engineering, Faculty of Mechanical, Maritime and Materials Engineering

To obtain the degree of Master of Science in Mechanical Engineering at Delft University of Technology
to be defended publicly on February 11, 2022.

by

Emma C. Kooi

Student number:	4301641	
Thesis committee:	Dr.ir. Y.B. Eisma	TU Delft, supervisor
	Dr.ir. J.C.F. de Winter	TU Delft
	Dr. D. Dodou	TU Delft
	Ir. L. Roelse	Lely, supervisor

This page is left blank intentionally

PREFASE

This research started with a specific problem; to recognize a particular type of collision with a heavyweight, low-speed mobile cleaning robot and no concept on which route to take to solve the problem. The most challenging aspect of this thesis was to define the method using preliminary experiments, with little literature available on this specific problem. To provide not only academic results for the research and experiments performed, but also generate result-oriented solutions that work in a soiled and dynamic environment was most unusual. At the same time, this was the most enjoyable part of the graduation project, as I had all the freedom to search for solutions with my feet in the dirt, and the results obtained could be implemented directly into the robot.

I want to thank my supervisors Lennart Roelse and Diederik van der Pant at Lely for their valuable insights, guidance and support during the whole time span of the project, and for giving me all the space to work with the robot and search for solutions. I am grateful for the opportunity to graduate at Lely and to get to know the company.

Furthermore, I would like to thank my daily supervisor from the TU Delft, Yke Bauke Eisma, for his enthusiasm and positivity in this project from day one, his academic point of view and the online and - if allowed by Covid-19 restrictions - personal meetings, which increased in the final writing phase of the thesis. I also thank Joost de Winter from the TU Delft for his accurate comments during the online meetings and thorough feedback on my draft report throughout the research.

I hope you will enjoy reading this thesis.

Delft, Februari 2022 E.C. Kooi

Table of contents

LIST OF TABLES

List of Abbreviations

Blind collision detection and classification framework for a heavyweight, low-speed mobile robot colliding with soft and hard objects

Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology

Kooi, E.C. (4301641)

DELFT, FEBRUARI 2022

Abstract— **Visual occlusion can cause object detection and classification systems to fail due to swift movement or a soiled, moist or dusty environment. Hence, this asks for a 'blind' collision detection and classification method. This paper presents a novel blind collision detection and classification scheme for a heavyweight, low-speed, wheeled mobile robot. By recognizing the target signal pattern in phase current data, the robot can detect whether the wheels are blocked caused by overload (G1). After this, using acceleration and phase current data, the robot will be able to differentiate between a collision against a wall or a collision with an object between the wall, or no collision of interest (G2). Additionally, an algorithm based on the pitch angle can detect if the robot slides on top of a soft object (G3). Three different algorithms are developed that address these goals. Finally, the developed algorithms are validated in a dynamic and soiled environment (G4). The first two goals are reached by training machine learning classifiers that identify the signal pattern of its target event and place uneventful data in a separate class (open-world classification). During training the classifiers Random Forest, Multi-layered Perceptron, Decision Tree and Logistic Regression are compared, and the two best perfoming classifiers are subsequently tested for a time-series open-world classification task. The third goal is reached with the development of a heuristic algorithm based on the running mean of the pitch angle.**

Experiments were performed in a controlled environment, creating collisions with varying floor conditions, robot weight and collision objects. The best performing algorithm for blind collision overload detection has achieved 98.6% detection accuracy. The object classification algorithm can differentiate between two types of soft objects, a wall or no_event with an accuracy of 93.3%. The algorithm based on the deviating pitch angle can detect events with 100% detection rate and 0% false positive rate. Detailed implementation schemes are provided for real-time implementation, illustrating a robust framework in soiled environments. The proposed solutions can be used to improve collision detection on blind mobile robots, as well as mapping the environment using the object classification model.

Index Terms— **Collision detection, object classification, mobile robot, blind sensing, visual occlusion**

I. INTRODUCTION

Automated guided vehicles have become more prominent in the last twenty years because of their relevant applications to the world today. A combination of a mobile robot, inexpensive sensors and advanced software technology can already ensure reliable navigation. However, detecting and interpreting collisions is an essential aspect of safe robot operations. The classification of different types of obstacles for mobile robots typically requires vision- or laser-based perception systems (Mahalingam & Subramoniam, 2017; Hernández, Gómez, Crespo, & Barber, 2016; Galvez, Vicerra, Bandala, Dadios, & Maningo, 2018). However, visual occlusion can cause object detection and classification systems to fail due to swift movement or a soiled, moist or dusty environment. Hence, this asks for 'blind' collision detection and classification system. Blind sensors in this research were defined as sensors that do not make use of light detection or radio waves, i.e. all sensors that are not affected in their performance when covered with a thick layer of dirt. This thesis will research collision detection and classification with soft objects for a heavyweight cleaning robot. The focus in practice will be to develop a feasible real-time object detection application in an uncontrolled, soiled environment with arbitrary robot movements. The method must be reproducible and applicable on all robots of the same type. The types of collisions that can occur when the robot is in operation are shown in Figure 1. The robot can push the object forward against a wall (type I), slide on top of the object (type II), or drive against it in free ride (type III).

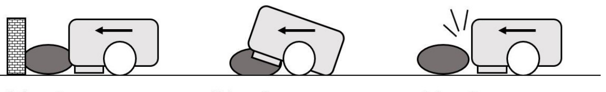


Fig. 1: Visualisation of different types of collisions from the robot and soft object: (a) push object against wall, (b) sliding on top of object and (c) collision impact. Collision type I and II will be detected when the research goals are achieved. Collision type III will transition to type I or II over time.

I-1 State of the art

A previously performed extensive literature review (summary in Appendix A) gives an overview of the existing blind collision detection algorithms and classification methods, where cameras, LIDAR or ultrasonic sensors were excluded. The results of the performed literature study show that collisions can be detected and distinguished based on blind sensing, using robotic arms or lightweight mobile robots. Thus far, the most significant part of the literature focuses on motor current and acceleration based detection methods. Geravand et al. used time-varying detection thresholds on the motor current based on the input trajectory of the robot, which resulted in a robust algorithm making the collision detection method not bound to a smooth trajectory (Geravand, Flacco, & De Luca, 2013). Additionally, low and high pass filters on this motor current could differentiate between hard and soft collisions. This concept is also shown in the research of Cho et al. (Cho, Kim, Kim, Song, & Kyung, 2012). (Park & Kim, 2021) showed that motor currents and machine learning methods based on Support Vector Machine(SVM) and Convolutional Neural Network(CNN) could accurately detect soft collisions (pushing, pulling, catching) and hard collisions (sharp impacts). The aforementioned studies describe a way of detecting collisions using the motor current and determining whether the collision object is hard or soft. These classification methods are limited to hard or soft, where no differentiation can be made between different soft or hard collisions.

When the load caused by a collision is too high for the motor, an event of overload will occur which will lock the rotor. A threshold on the motor current will not purely detect this event since high torques are sometimes desired in industrial robots. Mohar et al. (Mohar et al., 2021) detected overload in a three phase induction motor using a Neural Network with inputs the motor Current, Voltage, Torque and Speed. With only the phase current as input feature, Gao et al. showed that overload of the motor could be identified based on variations of the current signal in the frequency spectrum (Gao, Cecati, & Ding, 2015). Although this is used for fault detection in the motor, this can also be used for overload caused by a collision. A caveat to this method is that fluctuating load, speed and friction of the robot causes variations in the motor current, which can introduce false positives if the method is based on the frequency spectrum only.

Collisions in literature are also detected based on accelerometers. After pre-processing acceleration signals, a detection threshold was commonly used to identify collisions (He, Du, Sun, & Lin, 2007; Wisanuvej, Liu, Chen, & Yang, 2014; Moorits & Usk, 2012; Zug, Seidel, Beckhaus, & Winkelsträter, 2017; Speleers & Ebner, 2019). Except for Mericli et al., who detected collisions using the Fast Fourier Transform (FFT) of a sliding window to detect anomalies, and Becker et al., who trained a Logistic Regression (LR) classifier with normal data and two types of collision data to detect and classify collisions. (Meriçli & Levent Akın, 2008; Becker & Ebner, 2019). Acceleration signals lend themselves well to classify collisions after detection. Windau et al. could identify the hardness, elasticity, and stiffness using correlation techniques on known material properties (Windau & Shen, 2010). However, learning algorithms based on acceleration data performed best; Wisanuvej et al. could identify which material was tabbed against a robotic arm (Wisanuvej et al., 2014), and Vail et al. could identify the surface on which a legged robot was walking (Vail & Veloso, 2004). The study from (Adu & Bran-Melendez, 2018) optimized IMU based gesture classification. They compared the classification accuracy of Logistic regression, Support Vector Machines and Multi-Layered Perceptron (MLP) and concluded that MLP had the highest accuracy. The best performing classifiers with acceleration data in the discussed literature were DT, Neural Network and LR. However, signal patterns are different for each robot, and thus these results cannot directly be used on our mobile robot.

I-2 Knowledge gap

Nothing in literature could be found for blind collision detection and classification for a soft object with a heavyweight cleaning robot in a soiled environment. The robots from the literature study are lightweight with low driving friction, usually in a controlled environment. Papers found on mobile robots in a highly soiled environment focus on navigation instead of object detection and classification.
Additionally, a method based on the three-phase current for object detection and classification in a wheeled mobile robot could not be found. Techniques to detect overload using raw three-phase current signal patterns did not turn up. Lastly, acceleration signal patterns have shown to discriminate between the type of object, however, the patterns are different for each robot and thus a new algorithm needs to be trained for a heavyweight mobile robot.

I-3 Generic approach

The method used is derived from preliminary experiments and inspired by different methods presented in the literature, both from robotic arms and lightweight mobile robots. Currently, the data of the robot is logged with 10Hz. Based on the Nyquist-Shannon Sampling Theorem (Shannon, 1949), it can then only reconstruct signals with a frequency <5Hz. Therefore external sensors are placed on the robot to reach a higher sample rate of 1000 Hz, as the minimum sample rate used for object classification in literature is 125 Hz (Appendix A). Preliminary data acquisition and analysis showed that the collision type III (Fig. 1(c)) of a soft object (40 kg) could not be detected (see Appendix C for this analysis). Thresholds on acceleration and motor current data to determine the exact time of collision proved to be not applicable. Any signal changes due to a soft collision were visually undetectable during normal movement of the robot, despite different high- and low-pass filters applied. Furthermore, the excited frequencies of the acceleration in driving direction during movement with and without an object did not empirically change. The momentum caused by the robot's weight (being ten times heavier than the object) presumably overrules a noticeable impact from the soft object. In addition, the cleaning robot has high frictional forces with the ground due to rubber strips, which fluctuate enormously in time and varying ground conditions. Furthermore, the robot's speed is low, varying from 0.1 to

0.2 m/s; in combination with the softness of the object, this will significantly reduce the magnitude of the impact force. Therefore this thesis focuses on the detection of collisions type I and II (see Figure 1). Collision type III eventually transitions into type I or II, therefore this focus can be defended. For collision type I, a two-stage process is used to minimize false positives: 1) detection of overload and 2) perception with acceleration and three-phase current signal patterns. Acceleration signals in the impact direction are used for classification. To keep the classification task in the horizontal plane, this thesis will only focus on classification of type I collisions. For collision type II, the focus is placed on a heuristic, robust but straightforward algorithm based on the pitch angle, robust to dirt buildup in front of the robot, changes in the slope of the floor and drifting data and still identify collisions.

I-4 Aim

This research was aimed to close the knowledge gap by providing the following research goal:
Develop a blind collision detection and classification method for implementation in a heavyweight mobile cleaning robot in a soiled and uncontrolled environment.
The approach presented in this paper uses a data-driven strategy to address the main challenges associated with blind mobile robots in a soiled environment:

- detect when the wheels block by predicting motor overload based on the raw phase current signal pattern (G1),
- identify the type of object causing this overload by classifying the unique signal patterns (G2),
- determine if the robot slides on top of a soft object by deviating pitch angle(G3),
- validate the developed algorithms in a dynamic and soiled environment (G4).

This thesis aims to re-introduce previously unused sensors in the robot for a new purpose to improve the robot's object detection. To reach goals G1, G2 and G3, algorithms A1, A2 and A3 will be developed. With these algorithms, collision events in a predominantly uneventful environment will be detected, to be able to stop the movement of the robot. This measure can significantly decrease the risk of harmful injuries.

From the three-phase motor currents, information such as hardness can be obtained based on the rate of change (Cho et al., 2012). The wavelength of the three-phase current signal will change based on the specific object, depending on the hardness or elasticity of the object, determining how fast the wheels come to a stop. Acceleration signals give information about the hardness, elasticity, and stiffness. After impact, it is expected that each object has its unique signal pattern, like tapping sounds coming from different materials. With the combination of motor current and acceleration measurements, the hypothesis is that different soft objects can be distinguished from hard objects.

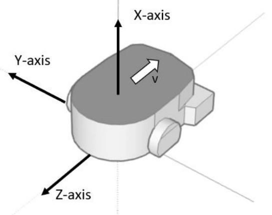
To test these expectations, three experiments were carried out in June and August 2021 to characterise the machine behaviour for different types of collisions and to develop and train the three algorithms in order to achieve the research goals. The method is described in Chapter II, including the hardware used and the experiments performed. Hereafter the results are presented in Chapter III, followed by a discussion of the results in chapter IV. Finally, a conclusion and future recommendations are presented in Chapter V.

II. METHOD

This chapter describes the methods used for detecting collision types I and II. Details on hardware are presented along with the performed experiments, the analysis method, and the algorithm development for detection and classification.

II-1 Materials

The used robot is a differential-drive mobile cleaning robot with rubber strips upfront. Its weight varies from 300 kg when empty and 700 kg when filled with water. Dimensions are approximately 1.0 x 1.2 x 0.5 meter, and the velocity range is 0.1 - 0.2 m/s. The robot logs internal data at 10 Hz, from which the pitch angle is extracted. To obtain data with a higher sampling frequency, an external IMU (ADIS 16505, 2000 Hz) is placed inside, and a 3-phase current logger (1000 Hz) is connected to the motors. The natural frequencies of the robot are primarily present within the bandwidth 300 - 450 Hz, determined using an impact test and FFT plot (Appendix B). A representation of the robot, its defined axes, and the specifications are given in Figure 2.



Specification	Value
Weight	300 - 700 kg
Velocity	0.1 - 0.2 m/s
Dimensions	1.0 x 1.2 x 0.5 m
IMU	2000 Hz
3-Phase current logger	1000 Hz

Fig. 2: Representation of the robot used and its defined axes (left). Specifications of the robot and the sensors used (right).

The three phase current signals were recorded in time-domain, acquired from both motors using an NI based data acquisition system and LabVIEW, and saved as CSV. The program iSensorFX3Eval was used for logging the IMU data from an USB-c cable and saved as CSV. The data was read and merged using Python, filling up any missing values with zeros after re-sampling the IMU data to 1000 Hz.

II-2 Machine Learning Classifiers

For this thesis, supervised learning was chosen as the preferred method to fulfill goals G1 and G2. Four different classifiers were compared for collision type I detection and classification of the object. This thesis could not compare the whole spectrum of machine learning algorithms due to the wide range of available algorithms and the continued development in this sector. When selecting the suitable classifiers, the following criteria needed to be met:

- Effective with little training data (<300 samples per class)
- Robust to noise in data

Based on the selection criteria and the literature discussed, this thesis compares the classifiers Random Forest (RF), MLP, LR and Decision Tree (DT) during training of the algorithms. After training, due to the significant computation time when implemented in time-series, only two classifiers with the best prediction accuracy are compared for open world classification in time-series.

With a part of the data obtained from the experiments, the classifiers are trained using the target signal patterns of collision detection and classification (training set). The remaining part is used for validation (test set). The train and test set contain respectively 80% and 20% of the whole dataset. The prediction accuracy on the test set will vary per run as a result of shuffling of test and train sequences. To systematically compare the methods, the average accuracy will be determined of 10 consecutive runs.
Table 1 presents an overview of the used classifiers, including the method, pros, cons and the specifications used for training in Python.

II-3 Experiments

Three experiments were performed to develop the methods and train the three algorithms in order to achieve the research goals. The experiments give information about the behaviour of the robot during collisions type I and II, and collects data for object characterisation. The experiments were performed in controlled lab conditions.

II-3.1 Experiment A: Driving Routes: The aim of the first experiment was to analyse the behaviour of the robot during normal operation and when colliding with a 40 kg soft object (leather bag filled with sand, see Appendix Fig. 44). This experiment determines the exact direction of the methods used to achieve the research goals. The three phase motor currents, acceleration in driving direction and pitch angle were collected.
Three types of driving routes were programmed which include the relevant actions of the robot during operation: (a) wallbump: driving against the wall; (b) wallfollow: scraping along the wall, with the wall to the right of the robot; (c) freeride: riding without touching the wall (Figure 3). Each route is driven ten times with and without a 40 kg soft object placed in the trajectory of the robot.

II-3.2 Experiment B: Stationary Robot Object Classification: The second experiment was performed with a stationary robot, to gather information from pure acceleration signals in impact direction. The goals of this experiment were to 1) determine the impact frequencies of the soft and hard object, 2) compare four different classification algorithms for these conditions and 3) compare the input features that

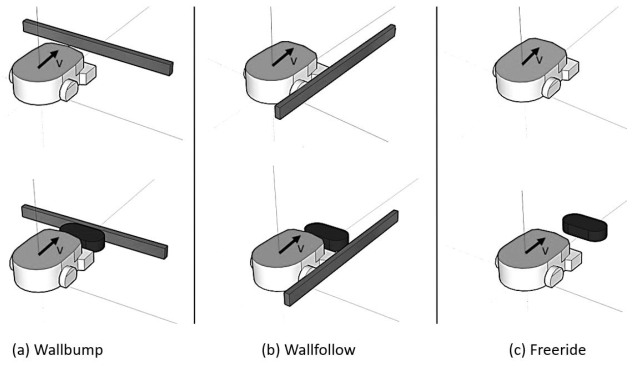


Fig. 3: Driving conditions for experiment A; Wallbump, Wallfollow, and Freeride. With and without soft object (bottom / top) of 40 kg placed in its trajectory. Used for analysis of the robots behaviour during type I and II collisions.

result in the most accurate classification. With this test, pure acceleration signals from impact with an object can be obtained without driving vibrations caused by movement of the robot (Fig. 4). For three representative object velocities, the acceleration of the robot in impact direction is obtained. The objects hang as a pendulum from the ceiling, free from the ground and are released against the stationary robot. The objects are a 40 kg wooden beam and a 40 kg bag filled with sand, and are released at horizontal speeds of 0.1, 0.2 and 0.24 m/s (1). The experiment is repeated 50 times for each object and its velocities, making 300 collisions.

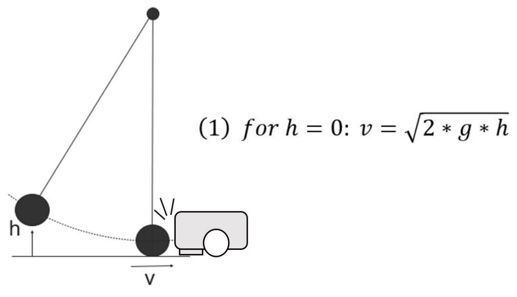


Fig. 4: Visualisation of Experiment B: pendulum test, and equation (1) for bottom velocity

II-3.3 Experiment C: Object Classification During Robot Movement: Experiment C collects the three phase currents and acceleration signals when the robot drives against two different types of soft objects between the robot and wall, and the wall without an object. The goals of this experiment were to train the object classification algorithm for goal G2, and reflect on the acceleration signals compared to the stationary robot data from Experiment B. The motor current that is needed for a given speed varies with the payload of the robot and the friction of the floor. Therefore it is important for future implementation to vary these properties. The robot's weight (300 kg, 450 kg, 700 kg) and the friction of the floor (with rubber strip, μ_s = 0.75 and 0.37)(Appendix 24) are independent variables. Dependent variables are the acceleration signal in impact direction and three phase motor currents at the time of collision with each object. The wall in

Classifiers	Model	Pro's	Con's	Specification settings Python
Decision Tree (DT)	Predicts the target variable class by learning simple rules of decision (<, =, >) on training data. The prediction arises from a series of splits based on input feature numbers, generating a tree-like graph (Alzubi, Nayyar, & Kumar, 2018)	• Effective with little training data. • Very simple, low in captivated memory space. • Less affected with irrelevant features.	•High accuracy requires heavy feature engineering.	default
Random Forest (RF)	Uses a set of 50 decision trees, internally trained with random features. The most votes among each outcome of the decision trees is the predicted class.	• Effective with little training data. • Very high accuracy and very robust, low variance. • Less affected with irrelevant features.	• Captivates most memory space. • Appears as a black box model structure.	n_estimators = 50

Multi-Layered Perceptron (MLP)	Feedforward neural network with backward propagation. It consists of input layers, hidden layers and an output layer. Learns by changing weights of the neurons that it assigns to itself (Gardner & Dorling, 1998)	• Effective to complex nonlinear problems.
		• Can reach very high accuracy.
		• Quick predictions after training.
		• Can reach high accuracy with lesser training data than other Neural Networks.
			• Requires more training data then DT, RF, LR for difficult classification tasks.
			• Performance variance is high for each training set.
			• Has a black box model structure.
				alpha = 1e-5,
				hidden_layer_sizes = (30,30,30),
				solver = 'adam',
				activation = 'relu',
				learning_rate_init = 0.01,
				learning_rate = 'adaptive'

Logistic Regression (LR)	Implements regularized logistic regression equation. For multiclass, a binary problem is fit for each label, the highest prediction score of all classes is chosen. (Alzubi et al., 2018)	• Effective with little training data.
		• Very simple, low in captivated memory space.
		• More accurate for simple training sets.
			• Less accurate with complex classification problems.
			• High accuracy requires heavy feature engineering.
			• Poor performance with irrelevant/correlated features.
				max_iter = 400,
				multi_class = 'ovr'

TABLE 1: Specifications of classifiers DT, RF, MLP and LR used to detect motor overload (G1) and predict type of object (G2).

the workshop is a rigidly fixed wooden beam to the floor. A Wallbump is defined as a collision type I without an object between robot and wall. The soft objects placed between robot and wall are representative for real objects in the field of operation of the robot and include 1) a 40 kg leather bag filled with sand and 2) a polyether foam block over the whole length of the wall. Each combination of object, weight and floor properties is executed ten times, collecting a total of 180 collisions; the test matrix is given in Table 2.

II-4 Blind Collision Overload Detection (G1)

There is a large uneventful interval around each collision, causing false positives to become more likely. Therefore a double verification process to detect a collision was used: first, detect overload by collision with a sliding window and an open-world classification algorithm identifying blocked wheels (goal G1), and afterwards predict the type of object OR if the overload is not caused by a collision (goal G2).

A plot of the three phase currents during a collision with the wall from Experiment A can be seen in Figure 5. The black line shows the point in time where the robot hits the wall (without object), and at the red line, the phase currents increase above the rated value (overload), causing the rotor to lock. Events will be detected using the characteristics of the waveforms during overload. Therefore the part after the red line will be used for the first goal G1: detection. The part between the black and red line gives information about the object between robot and wall, and will be used later in the report for the second goal: classification. Following this method, a robust detection system can be developed.

II-4.1 Development Algorithm - Training: The dataset for overload detection was made of 1000 "no_event" samples of normal phase current data and 360 samples of "overload"

Dependent variable	Weight robot [kg]	Floor	Collisions
Wallbump	300	High friction	10
		Low friction	10
	450	High friction	10
		Low friction	10
	700	High friction	10
		Low friction	10
Sandbag + wall	300	High friction	10
		Low friction	10
	450	High friction	10
		Low friction	10
	700	High friction	10
		Low friction	10
Foamblock + wall	300	High friction	10
		Low friction	10
	450	High friction	10
		Low friction	10
	700	High friction	10
		Low friction	10
Total			**180**

TABLE 2: Test matrix of Experiment C: used for training and validation of blind object classification algorithm. Variations of the robot's weight and floor friction are included for robustness of algorithm in new environments.

(Table 3). A signal of length 100 ms has been chosen for fast detection when applied in time-series data. Open world classification was used since real-time detection in an uneventful environment is wanted. No_event samples are harmonic phase currents in different amplitudes; Figure 6

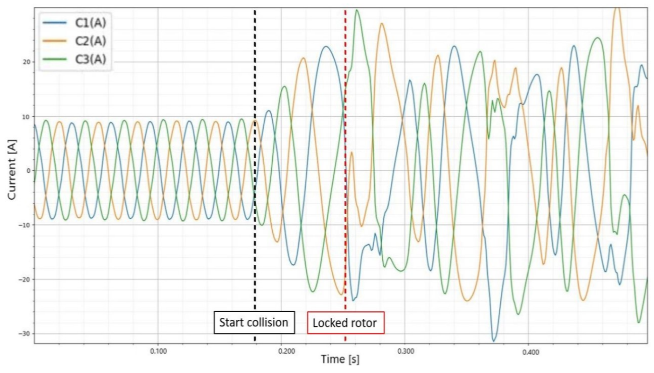


Fig. 5: Three phase currents of the right motor during start of collision type I without object (black dotted line) until locked rotor (red dotted line). Obtained in controlled lab conditions.

shows the phase current samples for a locked rotor and normal driving data. The test and training samples of bumps are made of 8 collisions, split in 15 samples per collision for each of the left motor three phase currents (C1, C2, C3), making a total of 360 sequences (Appendix Fig. 25). In the Appendix more phase current sequences of eventful and uneventful data can be seen (Figures 26, 27). The sequences are trained using Python, the script is presented in Appendix f.1 and f.2.

	Total sequences [#]	Length sequence [# datapoints]	Training sequences [#]	Test sequences [#]
No event	1000	100	797	203
Overload	360	100	291	69

TABLE 3: Characteristics of the dataset to train and test the classifiers for goal G1: detection of overload using the raw phase current.

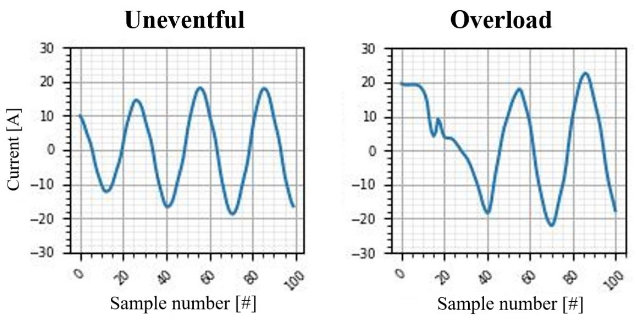


Fig. 6: Phase current signal patterns of an Uneventful and Overload sequence of 100 ms used to train the classifiers for goal G1: detection of overload using the raw phase current.

II-4.2 Real-Time Implementation: Due to the long run time across an entire time-series dataset, only the two best performing algorithms from the previous step are implemented for time-series detection. The classifiers will run through one hour of eventful and uneventful data using a sliding window. An improvement in the detection accuracy was to threshold the prediction probability of an overload detection. If the predicted sample of overload had 85 percent certainty, the detection was deemed successful; otherwise, the signal was ignored.

Only one out of three phases of the motor supply current will be monitored for real-time overload detection. Since the robot has two motors, the algorithm will run through both motor currents consecutive. When an overload is detected on the first motor (left wheel), the algorithm will directly check the second motor (right wheel). A collision event signal is triggered if both rotors are predicted as locked.

Sequences through the dataset were made using a sliding window as shown in Figure 7.

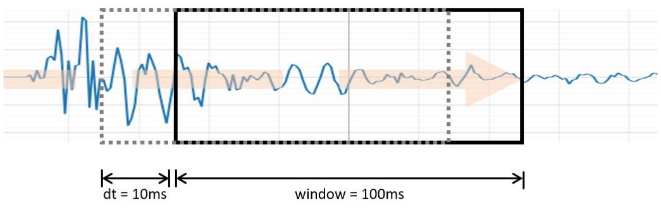


Fig. 7: Sliding window used for prediction collision type I in time-series data. The window contains 100 samples (100 ms) and moves with 10 samples (10 ms) each step

The window has the same length as the trained sequences, with a time step of 10 ms, the detection algorithms will run through time-series data. A smaller time step will slow down the algorithm, while a bigger time step can skip important data. The same collision can be detected multiple times due to the duration of the signal and the sliding window. Therefore, consecutive events with an interval shorter than 0.5 seconds are ignored. Furthermore a minimum of two consecutive bumps is required for successful event determination to reduce false positives. The detection algorithm (A1) provides the starting point for object classification, which allows for the collision signal to be segmented. In Appendix f.3 the Python script is presented.

II-5 Blind Object Classification (G2)

When an object hits different materials, varying patterns are produced in the data segments during impact. These signals can be processed using machine learning techniques for classification. To keep the classification task in the horizontal plane, this thesis will only focus on classification of type I collisions. Besides, it is assumed that collisions of type II will not happen with a hard object, and will push the object forward against a wall (transitioning into type I).

A plot of the three phase current of a collision with the wall from Experiment A was seen in Figure 5. The part between the black and red line gives information about the possible object between robot and wall. After detection of overload (red line), described in the previous section, the algorithm can go back a section in time to predict the type of object (goal G2).

Since the robot is in motion, the natural frequencies of the robot can be excited. To be able to filter out these frequencies, the specific impact frequency per object was determined from Experiment B: impact with a hard and soft object on a stationary robot. The impact frequencies per object were determined using an FFT plot. The prediction performance of classifiers RF, MLP DT and LR for classification of pure acceleration signals from a hard and soft object collision will be determined using Python. Next, these learning's will be used to develop the algorithm for collision classification during movement of the robot in Experiment C. Here the three phase motor currents are an important extra feature.

II-5.1 Development Algorithm - Training: The classifiers RF, MLP, DT and LR were trained and compared with data from Experiment B and subsequently trained and compared with data from Experiment C.

II-5.1.1 Experiment B: An FFT plot made from the collision sequences of the hard and soft objects in Experiment B showed the impact-specific frequencies of each object. Based on this data, the cut-off frequency of a Low-Pass (LP) filter to process the acceleration signal could be set at 100 Hz - to filter out the higher natural frequencies of the robot. The 300 collisions from Experiment B are labelled and shortened to sequences of 200ms. The length of the sequences was chosen based on the length of the signal. Data is normalized for the algorithm not to focus on magnitude but signal pattern ($Z_{acc}/max|Z_{acc}|$). The machine learning classifiers will be compared for different input features to determine the input that achieves the most accurate result for this type of object classification. The input features for acceleration are 1) raw data; 2) LP filtered (cut-off 100 Hz) and 3) the FFT of the sequence.

	Total sequences	Length sequence	Training sequences	Test sequences
	[#]	[# datapoints]	[#]	[#]
Soft object	150	200	121	29
Hard object	150	200	119	31

TABLE 4: Characteristics of the dataset to train and test the classifiers for Stationary Robot Object Classification

II-5.1.2 Experiment C: The same filters and acceleration signal processing features were used to train the algorithms for classification during robot movement, based on Table 2. The input features will be based on raw or processed phase current signals and acceleration data. Since the robot is differential wheeled, both left and right motor current needs to be implemented in the algorithm. To combine this into a single feature, the maximum of the three phase currents of each motor is determined and multiplied for the left and right motor (Maximum Power). This makes one signal dependent on both wheels, and could result in an increased performance by reducing the number of input features.
Segmented collision signals with a length of one second will be classified. Once again, the performance of the classifiers RF, MLP, DT and LR for different input features was compared. To apply the classifiers on multi-variate time-series data, e.q. acceleration and motor current, the sequences must be flattened into a vector. Each two dimensional sequence array is re-shaped from [l_samples, n_features] into [1,l_samples x n_features] (Figure 8), where $l\_samples$ is the length of the samples and $n\_features$ is the total number of features in a sample. Subsequently, the entire dataset will be of shape [n_samples, l_samples x n_features], where $n\_samples$ is the number of collision sequences. The Python script for training the classes is presented in Appendix f.5.

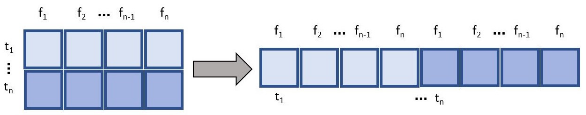


Fig. 8: Example of a time-series sequence flattened into a vector

II-5.2 Real-Time Implementation: During real-time implementation, the *detection* algorithm A1 can signal a false positive or event of no interest; this can happen when the robot scrapes along the wall during navigation, or due to a second push. Therefore the classifier should have an extra output of no_event, to place unseen data which can not be classified as $wallbump$ or $object$. The output of the final classification model will be a prediction in the classes [no_event, $wallbump$, $sandbag$, $foamblocks$].
From Experiment C, Table 2, the classifiers were trained with different types of floor frictions and payloads. This will improve the algorithms robustness when applied to different robots of the same type in a new environment. The Python script for predicting classes after detection (Algorithm A2) is presented in Appendix f.6.

II-6 Detection of Deviating Pitch Angle (G3)

The motor wont block if the robot slides on top of the soft object, ergo collision type II will not be detected using the same method as for collision type I.

II-6.1 Development Algorithm: A machine learning model is not necessary to reach the third goal, therefore a heuristic method was developed to detect sudden change in the pitch angle. However, data from Experiment A showed a drift in pitch angle, whereby a simple threshold would not be sufficient. Furthermore, it should be robust in an uncontrolled environment; slopes in the floor and dirt build-up in front of the robot. Filtering out the low frequencies to recover for drift in the sensor signals caused a loss of information in magnitude (Appendix Fig. 28). In Python, an algorithm is developed that can be run real-time to detect a sudden negative change in magnitude. A moving average of the pitch angle with a lagging window of 80 seconds is determined. Any deviation from this average that exceeds a threshold is considered an event. The moving threshold of five degrees deviation is determined based on two times the maximum outlier of uneventful data, determined using a boxplot (Appendix Fig. 29). This proved to be the optimum between eliminating false positives and detecting events. Eventful signals will not be included in the lagging window to calculate the threshold, this way the algorithm signals an event for the whole duration of the collision. The algorithm is presented in Python script in Appendix f.4.

II-7 Validation Algorithms in Dynamic Environment (G4)

The algorithms developed will be validated for real-time implementation in a different environment under soiled and dynamic conditions.

From goals G1 and G2 only the best performing classifier will be validated in an uncontrolled and dynamic environment due to the significant computation time when implemented in large time-series data-files. The hardware and sensors were set up on another robot of the same type operating in a soiled and dynamic environment with moving objects. The classifiers were tested for its true and false positives for each algorithm. Since more data was available at with a sample rate of 10 Hz, the algorithm for detection of deviating pitch angle could be validated on seven different robots, in a dynamic and soiled environment.

II-7.1 Implementation Pipeline: A pipeline for implementation of the algorithms was developed to use for real-time collision detection on the heavyweight mobile robot. This implementation scheme aims to reduce the number of false positives significantly by adding the expectation of hitting a wall, the pitch angle algorithm and true/false time duration statements. It can be used to robustly implement the developed methods in the field before developing future algorithm improvements.

III. Results

The results of the developed algorithms are represented in this chapter to reflect on the the research aim and goals. First, the blind collision detection outcomes for goal G1 will be evaluated, after which the blind collision classification outcomes for goal G2 will be examined. Thirdly the results of the detection of deviating pitch angle (G3) will be given followed by the validation of the algorithms in a dynamic and soiled environment.

III-1 Blind Collision Overload Detection (G1)

Events of collision type I are detected using the signal pattern characteristics of the phase current waveforms during overload. The results of training the algorithm are described in the next subsection followed by the results of implementation of the algorithm in a new data-set.

III-1.1 Development Algorithm - Training: Table 5 shows the performance of the compared algorithms on the test sequences. RF has the highest average detection accuracy of 98.6%, and a slightly less performance of MLP with an average accuracy of 97.9%, followed by DT (91.2%) and LR (82.4%). The confusion matrixes of the best performing algorithms can be found in Appendix Fig.30. The successive results in Table 5 concerns the best two performing classifiers for time-series implementation.

III-1.2 Real-Time Implementation: When run through one hour of time series data in lab conditions, both algorithms detected 100% of the 13 type I collisions (Appendix Figures 32 and 31). It can be seen that using the RF classifier, harmonic phase current signals but with a high amplitude are effectively ignored. MLP had a significantly larger amount of false positives than RF (59.0% versus 0%). When the robot hits the sidewall hard, the wheels can also block, however, this is defined as a Non_collision, not a false positive. These type of blocked wheels will be later on classified as a collision of no interest. The MLP based learning algorithm runs faster through 10 seconds of time series data than RF, with an 8.48 second duration versus 9.05 seconds. Both algorithms are faster-than-real-time. Table 5 shows the results for the detection of type I per learning algorithm.

Performance ***Detection***	RF	MLP	DT	LR
Accuracy test set	98.6%	97.9%	91.2%	82.4%
True Positive rate time-domain	100%	100%	-	-
False Positive rate time-domain	0%	59.0%	-	-
Overload detected of type Non_collision	6	6	-	-
Algorithm's speed over 10 s of data [s]	9.25	8.48	-	-

TABLE 5: Accuracy of classifiers compared for goal G1: detection of overload using the raw phase current. During training four algorithms were compared using the test set of sequences. Only the best performing two were compared for real-time implementation using a time-series data-set. A higher false positive rate means a higher number of false positives.

III-2 Blind Object Classification (G2)

This section examines the outcomes of the blind collision classification method by analysis of experiment B and C. First the results of training the algorithms on data from experiments B followed by C are described, whereafter the best performing algorithms are analysed for its accuracy in a time-series dataset.

III-2.1 Development Algorithm - Training:

III-2.1.1 Experiment B: The impact frequencies of each object are shown in Figure 9, together with the raw and filtered time-series acceleration signal. It can be observed that the impact frequencies are below 100 Hz, where the natural frequencies were most dominant above 300 Hz (Appendix B). Therefore, a low-pass filter with a cut-off frequency of 100 Hz is applied to the data. Figure 33 in the supplementary materials shows the importance of filtering these collision signals for classification; the raw collision signals with a soft object looks uncorrelated, where the filtered data shows significant similarities. Acceleration data and FFT plots from experiment B can be seen in the supplementary materials (Appendix Figures 34, 35, 36, 37, 38, 39)

The accuracy scores of the different algorithms with different input features is shown in Table 6. It can be seen that all algorithms perform good; RF has the highest score of 100% accuracy. LR and MLP follow up close (99.5% & 99.3%), and DT stays behind with 92.8%.

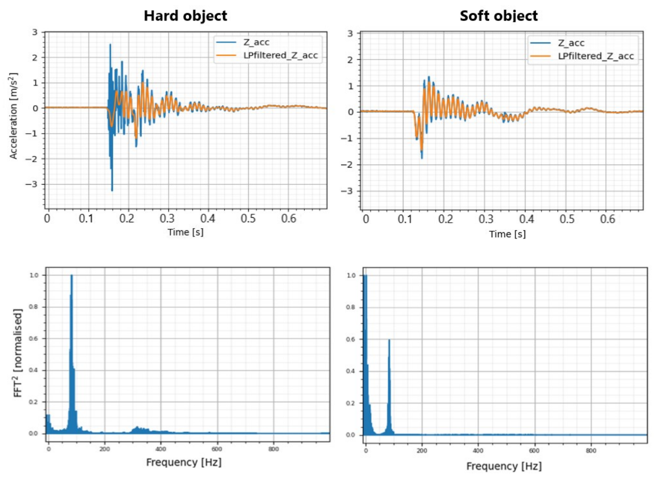


Fig. 9: Specific signal patterns of the pure collision signals (raw and filtered) for hard and soft objects (top), with their corresponding frequency spectra (bottom).

Performance	RF	MLP	DT	LR	Average
Raw acceleration data	100.0%	99.3%	92.8%	99.5%	**97.9%**
Low-pass filter (cut-off 100Hz)	99.4%	98.7%	96.4%	98.7%	**98.3%**
Fast Fourier Transform	99.4%	99.5%	98.1%	98.8%	**99.0%**
Average	**99.5%**	**99.3%**	**96.0%**	**98.8%**	

TABLE 6: Performance Stationary Robot Object Classification. Average accuracy of classification algorithms on the test set - compared for goal G2 using stationary robot collisions from Experiment B.

III-2.1.2 Experiment C: Figure 10 provides an illustration of the signal patterns for collision type I for each dependent variable. A clear difference can be seen in the slope of the phase current after a Wallbump, Sandbag or Foamblock collision. Furthermore, the deceleration of the robot during Wallbump is more significant in magnitude than the other objects. In the acceleration collision signal of the Sandbag the deceleration is more difficult to observe - though still present, but it has a drift upwards. This is due to the object causing the robot to tilt slightly, which skews the IMU and lets gravity influence the signal. Ten acceleration and phase current signals for each object can be found in Appendix Figure 43, 44 and 45.
Table 7 shows the performance of the algorithms for the different input features. It can be seen that Random Forest performs best, followed by DT, LR and MLP. However, in perspective with results from experiment B, the performance of the MLP based algorithm is falling short.

III-2.2 Real-Time Implementation: The detection algorithm of G1 defines the endpoint of the one-second segment for classification when applied to time-series data. For accurate implementation with unpredictable robot movements, it should be emphasized that both the (processed) motor current and acceleration signals must be used as input features. As an example, Figure 11 shows

Performance	RF	MLP	DT	LR	Average
Maximum power + LP filtered acc	93.3%	57.4%	86.2%	58.5%	**73.9%**
LP filtered acc	92.8%	64.0%	83.1%	60.3%	**75.1%**
C1 + C4 + raw acc	92.4%	65.2%	82.1%	55.1%	**73.7%**
Maximum power + raw acc	91.9%	59.2%	84.0%	62.2%	**74.3%**
Maximum power	91.8%	67.5%	85.1%	62.0%	**76.6%**
C1 + C4	89.4%	48.4%	83.5%	51.0%	**68.1%**
FFT^2 acc	59.5%	63.0%	48.3%	60.4%	**57.8%**
Average	**88.7%**	**60.0%**	**80.6%**	**60.3%**	

TABLE 7: Performance Object Classification During Robot Movement. Average prediction accuracy of classification algorithms on the test set - compared for goal G2 using moving robot collisions from Experiment C. Tested when trained with different input features in which the maximum power stands for: max(C1;C2;C3) * max(C4;C5;C6)

the phase motor current and filtered acceleration signal of a forward Wallbump collision versus a backward driving Wallbump collision. It can be observed that the motor currents have the same behaviour, but - as expected, the acceleration signal sign flips. The acceleration signals during collision with a hard and soft object on a stationary robot (Fig. 9) differ substantially from acceleration signals during collision with a hard and soft object in robot movement. This can be explained due to the low speed of the robot, making the acceleration signal caused by impact force to not rise above the driving vibrations. Therefore for time-series implementation the best performing algorithm trained with moving robot data from Experiment C will be used. Table 2 shows this to be the Random Forest classifier with input features Maximum Power and LP filtered acceleration.

Figure 12 shows the final overview of the classifier performance for goal G1: detection and goal G2: classification of type I collisions, and the false positives in test data of 1 hour of the overload detection algorithm.

III-3 Detection of deviating pitch angle (G3)

Events of collision type II are detected using a developed algorithm that identifies sudden changes in the pitch angle.
III-3.1 Development Algorithm: Figure 13 shows the pitch angle of the robot in controlled lab conditions. The smaller downward peaks are caused by the robot's acceleration from a standstill, causing it to tilt. The most prominent downward peaks are when the robot is on top of the object. The algorithm detects 10/10 type II collisions.

III-4 Validation Algorithms in Dynamic Environment (G4)

Since Random Forest outperformed all other classifiers for collision detection and classification, this was used for validation on a different robot of the same type. Table 8 shows the performance of the developed Random Forest detection and classification algorithms for collisions type I in one hour of data from a new environment.

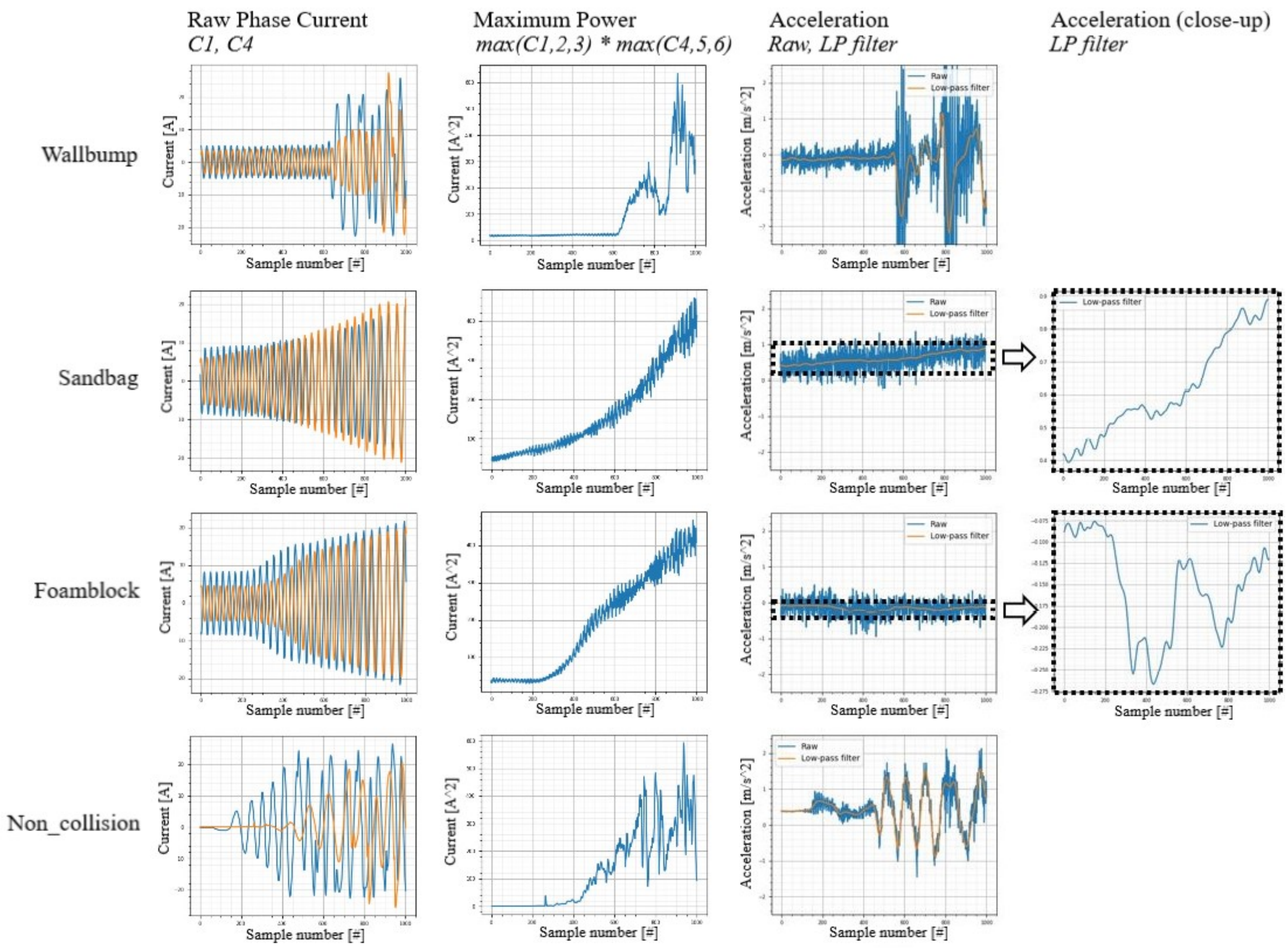


Fig. 10: One second signal patterns of Phase Motor Current, the calculated Maximum Power and Acceleration data during collision with each dependent variable, trained in the algorithm for blind object classification (G2) (shown splitted in full size in Appendix Figures 42 and 41)

Validation Dynamic Environment	Detection	Wallbump	Sandbag	Foamblock
True Positive rate time-domain	100%	100%	100%	-
False Positive rate time-domain	2.67%	8.18%	0%	2.72%

TABLE 8: Performance of the developed best algorithms for goals G1: detection and G2: classification of collisions type I in a soiled and uncontrolled environment. Since no foamblocks could be placed in this environment, no true positives are available. A higher false positive rate means a higher number of false positives.

In the data from experiment A the pitch deviation from collision is dominantly visible; however, this is more difficult in the uncontrolled field, and the probability of false positives is high. Therefore the algorithm was tested on 24 hours of uneventful data from a similar robot in dynamic and soiled conditions, where the algorithm gives no false positives (Appendix Fig.46). The algorithm is additionally validated on seven different robots in different soiled environments, where no false positives are given as well (Appendix Fig.47). Lastly, in a log of eventful data in a dynamic and soiled environment the collision was detected (Appendix Fig.48).

III-4.1 Implementation Pipeline: Figure 14 shows the suggested pipeline for implementation of the developed algorithms to use for real-time collision detection and classification. For each red circle, action is desired from the robot operator. The wall expectation statement reduces the amount of non-interesting overload detection events that need to be classified. When a wall is not expected, the overload duration needs to be at least five seconds to signal and classify the event. For collision type II, a duration penalty of five seconds is suggested to ensure it is not a short tilt reaction but still minimizes harm.

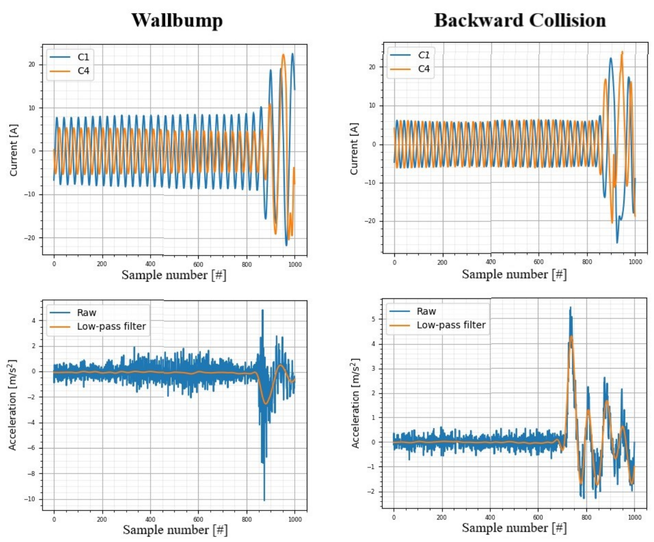
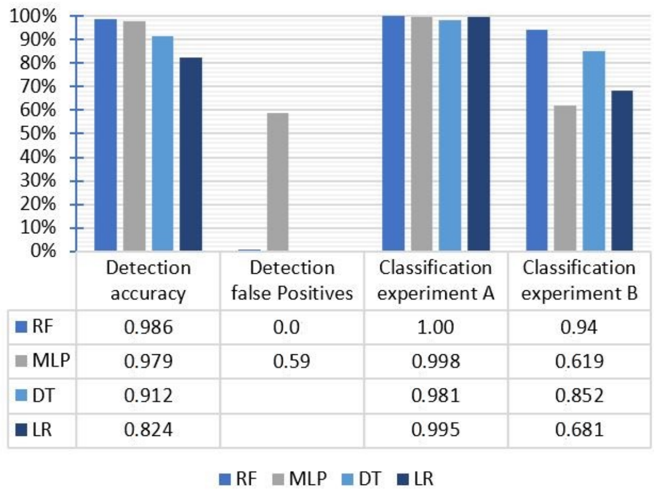


Fig. 11: Top: Phase Motor Current C1 (left motor) (C1) and Phase Motor Current C4 (right motor) (C4) and bottom: raw and filtered acceleration signal. Shows similarities in forward Wallbump collision and a backwards driving Wallbump in the phase current signal, however a clear difference in acceleration signal can be seen



	Detection accuracy	Detection false Positives	Classification experiment A	Classification experiment B
RF	0.986	0.0	1.00	0.94
MLP	0.979	0.59	0.998	0.619
DT	0.912		0.981	0.852
LR	0.824		0.995	0.681

Fig. 12: Overview of classifier performance for goal G1: detection and goal G2: classification of type I collisions, and the false positives for test data of 1 hour of the overload detection algorithm. A higher false positive rate means a higher number of false positives.

IV. Discussion

The primary aim of this thesis was to *develop a blind collision detection and classification method for implementation in a heavyweight, low-speed mobile cleaning robot in a soiled and uncontrolled environment.*

With a data-driven approach and by re-introducing previously unused acceleration and three-phase motor current signals in the robot, four machine learning algorithms were trained and compared for detection and classification. With the pitch angle, a heuristic algorithm was developed to detect sliding on top of a soft object. The final algorithms were validated in a dynamic and soiled environment. This collision detection and classification framework was specially developed for a type of robot whose size and dynamics had no agreement with the robots seen in the literature concerning collision detection and classifications.

To summarise the main findings, the detection method has proven to support the first goal G1; overload of the motor can be detected based on the specific signal pattern: interruption of the harmonic phase current. Additionally, the second goal G2 was achieved; obstacles that caused this overload can be identified by classifying the unique signal pattern, provided these also appeared in training data. Thirdly, goal G3 is fulfilled; determine if the robot slides on top of a soft object based on the pitch angle. Finally, the last goal G4 is reached; the methods work in a dynamic and soiled environment.

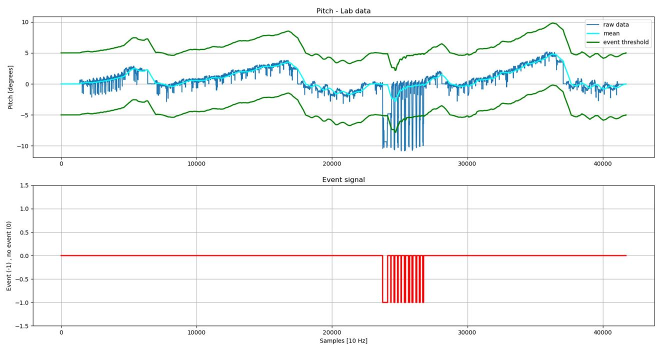


Fig. 13: Tilt detection algorithm on lab environment data, detecting all type II collisions marked with a corresponding event signal (bottom)

IV-1 *Blind Collision Overload Detection (G1)*

The results of the experiments show that sequences of overload versus normal harmonic phase current can be correctly classified during training with 98.6% accuracy using an RF classifier. For one hour of an unseen time-series dataset, overload due to a collision was detected 100% using open world classification. This is a different approach than motor current-based detection methods used in literature, which rely on thresholds and a rapid transition of the current value (Geravand et al., 2013; Cho et al., 2012) or variations in the frequency spectrum (Gao et al., 2015). The method from Geravand et al. (2013) and Cho et al. (2012) was an accurate method to detect hard and soft collisions in their research, however, the motor currents in Figure 31 indicates a threshold would deploy to many false positives in this data-set. Irregularities in the floor or acceleration of the robot can cause a rapid transition of the measured current as well, generating a number of false positives. Although the literature showed that optimal performance can be reached when using a time-varying detection threshold on the motor current based on the input trajectory model of the robot, this thesis demonstrated a robust method based on recognising the overload signal pattern. This is especially useful in cases where high phase currents are often desired but at the same time blocked wheels need to be detected. In general, algorithms based on machine learning have a

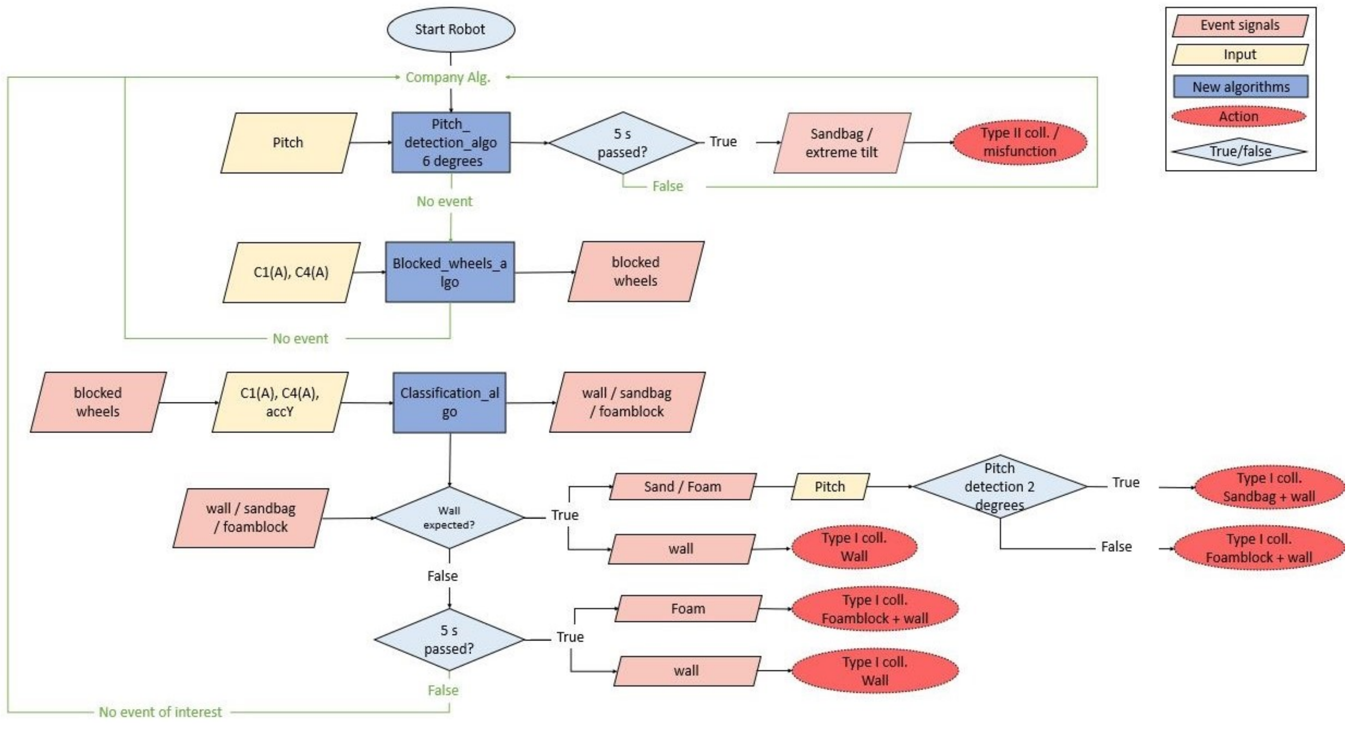


Fig. 14: Pipeline for implementation of the algorithms A1: Blocked_wheels_algo; A2: Classification_algo and A3: Pitch_detection_algo for robust implementation of real-time collision detection

longer response time then a simple algorithm. However, the algorithms developed in this research have shown to be 'faster' then real-time, which is satisfactory.

IV-2 Blind Object Classification (G2)

From pure, normalized acceleration signals of a hard and soft object hitting a stationary robot, RF could classify hard from soft with an accuracy of 100%. Logistic regression and MLP performed second best with a maximum accuracy of 99.5% for both algorithms. DT performed least with an accuracy of 98.1% - however, still satisfactory. These results are in line with the statement from Wisanuvej et al. (2014) , "The accuracy (...) in terms of object identification proves that accelerometer has a very high potential to capture the difference in vibration patterns from different materials (p.2254)." However, the acceleration patterns from stationary robot collisions (Experiment B) were not similar to the acceleration patterns during robot movement (Experiment C). The classifiers perform different on training data from both experiments.

The impact signals from soft object collisions with the robot do not rise above movement noise/friction in acceleration data, also seen in the supplementary material (Appendix C). Therefore, although all classifiers performed well for classifying a pure hard / soft / no_collision impact signal on a stationary robot, the trained classifiers from Experiment B could not directly be applied to acceleration data during robot movement. Experiment B did show that each object's impact frequency bandwidth was very consistent and could therefore be used to determine the frequency bandwidth that should be preserved after filtering the data.

The RF algorithm could classify the type of impact from movement data with an accuracy of 93.3% on the test samples, where DT, LR and MLP reached a maximum accuracy of respectively 86.2%, 67.5%, 62.2%. The least performing algorithms might need more collisions for training for this more difficult classification task. There is a clear difference in phase current and acceleration signal patterns from collisions against a wall with and without a soft object in between. In line with Geravand et al. (2013) and Cho et al. (2012), information such as hardness can be obtained based on the rate of change of the motor current. A rapid transition of the current signal indicates a hard collision, and a slow rising slope indicates a soft collision, which can be traced back to the type of object. Geravand et al. (2013) and Cho et al. (2012) did not show that differentiation between different soft objects using motor current was possible, where the result from this thesis can differentiate between a foam block and a leather bag filled with sand.

IV-3 Detection of deviating pitch angle (G3)

A heuristic algorithm based on the change in pitch angle was developed to detect if the robot slides on top of a soft object. Assuming the robot would not manage to slide on top of a hard object.

A simple threshold would suffice if no drift in the pitch angle data was present. Furthermore, drift in integrated senor data is a well-known concept, demonstrating the importance of an algorithm that handles this well.

A running mean with a threshold on the deviation from this mean allowed pitch signals caused by collision type II to be accurately detected. A similar concept using a running mean on acceleration signals detected robot collisions in the aforementioned literature (He et al., 2007). The developed algorithm in this research shows that this type of method can also be applied to sensor data experiencing drift. It identified all events in lab conditions and showed robust to false positives in data from a robot in a different, uncontrolled environment. Therefore, existing slopes in the floor and dirt build-up in front of the robot did not reduce the algorithm's accuracy.

As the classification task was limited to the horizontal plane, the proposed method does not provide an opportunity to classify the type of object for this specific collision (type II).

IV-4 Validation Algorithms in Dynamic Environment (G4)

The performance of the algorithms developed for goals G1 and G2 was slightly less in a dynamic environment than achieved in training data, with 8.2% false positives on Wallbump detection and 2.7% false positives on Foamblock detection. However, the algorithm still performed satisfactorily in this dynamic and soiled environment.

The scientific articles in the blind collision detection and classification branch keep the environment and the robot's payload constant. Furthermore, no article was found that applies its blind collision detection and classification method in a soiled environment. When implementing the algorithms according to the developed pipeline, this method is innovative and robust in dynamic environments.

V. Conclusion and Recommendations

To summarise the main findings, 100% of overload events with the developed data-driven method can be detected, and the type of impact with an accuracy of 93.3% using Random Forest. Furthermore, all events of the robot sliding on top of a soft object were detected. With the developed algorithms and the implementation scheme, we will be able to detect collision events in a predominantly uneventful environment to stop the robot's movement. This measure can significantly decrease the risk of harmful injuries.

V-1 Recommendations

The Random Forest classifier has shown to outperform all other machine learning based algorithms in this thesis. However, MLP was computationally faster, and RF uses a bigger amount of memory space on the processor then other algorithms. If this is not desired, another type of machine learning algorithm can be used and optimized by adding more samples or an extensive feature engineering method.

The present work opens up opportunities for environment mapping as well as detecting a variety of types of collisions with a clear collision signature when the algorithm is trained to do so. The classification method can be trained for any type of robot when data from different collision classes are present. If the robot shows uniformity with other robots, the algorithm can be duplicated to these kinds of robots, as proven by validation of the method with another robot of the same type (Table 8).

Due to the weight and speed of the robot used, classical detection methods with a threshold could not be used. The developed method can be used where collision detection is not possible employing a threshold. The patterns of motor current overload in other induction motors will correspond. In the used differential wheeled robot, both wheels had to block simultaneously for collision detection of type I. This can be easily adapted when a single induction motor drives both wheels.

In short, the developed object detection and classification methods have much potential for improving the behavior of different types of blind (mobile) robots in different environments (i.e. dynamic, soiled, dusty or foggy). The developed method especially adds value to collision detection and classification for heavyweight and low-speed mobile robots, where impact signals from soft object collisions during robot movement do not rise above movement noise in acceleration data.

References

Adu, P., & Bran-Melendez, S. (2018). *Optimizing imu based gesture classification.* Retrieved from `http://stanford.edu/class/ee267/Spring2018/report_adu_bran-melendez.pdf`

Alzubi, J., Nayyar, A., & Kumar, A. (2018). *Journal of Physics: Conference Series.* doi: 10.1088/1742-6596/1142/1/012012

Becker, F., & Ebner, M. (2019). Collision detection for a mobile robot using logistic regression. *ICINCO*(2), 167–173. doi: 10.5220/0007768601670173

Cho, C. N., Kim, J. H., Kim, Y. L., Song, J. B., & Kyung, J. H. (2012). Collision detection algorithm to distinguish between intended contact and unexpected collision. *Advanced Robotics*(16), 1825–1840. doi: 10.1080/01691864.2012.685259

Galvez, R. L., Vicerra, R. R. P., Bandala, A. A., Dadios, E. P., & Maningo, J. M. Z. (2018). Object detection using convolutional neural networks. *Proceedings of TENCON 2018 - 2018 IEEE Region 10 Conference.*

Gao, Z., Cecati, C., & Ding, S. X. (2015, 6). A survey of fault diagnosis and fault-tolerant techniques-part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, *62*, 3757-3767. doi: 10.1109/TIE.2015.2417501

Gardner, M., & Dorling, S. (1998). Artificial neural networks (the multilayer perceptron) — a review of applications in the atmospheric sciences. *Atmospheric Environment*, *32*(14), 2627-2636. doi: https://doi.org/10.1016/S1352-2310(97)00447-0

Geravand, M., Flacco, F., & De Luca, A. (2013). Human-robot physical interaction and collaboration using an industrial robot with a closed control architecture. *IEEE International Conference on Robotics* *and Automation*, 4000–4007. doi: 10.1109/ICRA.2013.6631141

He, F., Du, Z., Sun, L., & Lin, R. (2007). Collision signal processing and response in an autonomous mobile robot. *Neural, Oarallel Scientific Computations*, *15*, 319-334.

Hernández, A. C., Gómez, C., Crespo, J., & Barber, R. (2016, 8). Object detection applied to indoor environments for mobile robot navigation. *Sensors (Switzerland)*, *16*. doi: 10.3390/s16081180

Mahalingam, T., & Subramoniam, M. (2017). A robust single and multiple moving object detection, tracking and classification. *Applied Computing and Informatics*, *17*, 2-18. doi: 10.1016/j.aci.2018.01.001

Meriçli, e., Tekin, & Levent Akın, H. (2008). A Robust Statistical Collision Detection Framework for Quadruped Robots. *Lecture Notes in Computer Science*, *5399*, 145–156. doi: 10.1007/978-3-642-02921-9_13

Mohar, N. A., Mid, E. C., Suboh, S. M., Baharudin, N. H., Ahamad, N. B., Rahman, N. A., … Hadi, D. A. (2021, 6). Fault detection analysis for three phase induction motor drive system using neural network. *Journal of Physics: Conference Series*, *1878*. doi: 10.1088/1742-6596/1878/1/012039

Moorits, E., & Usk, A. (2012). Buoy collision detection. *54th IEEE International Symposium ELMAR*, 109–112.

Park, K. M., & Kim, J. F. C., Jihwan. (2021). Learning-Based Real-Time Detection of Robot Collisions without Joint Torque Sensors. *IEEE Robotics and Automation Letters*, *6*(1), 103–110. doi: 10.1109/LRA.2020.3033269

Shannon, C. (1949, jan). Communication in the presence of noise. *Proceedings of the IRE*, *37*(1), 10–21. doi: 10.1109/jrproc.1949.232969

Speleers, P., & Ebner, M. (2019). Acceleration Based Collision Detection with a Mobile Robot. *3rd IEEE International Conference on Robotic Computing*, 437–438. doi: 10.1109/IRC.2019.00088

Vail, D., & Veloso, M. (2004). Learning from accelerometer data on a legged robot. *IFAC Proceedings Volumes*, *37*, 822-827. doi: 10.1016/S1474-6670(17)32082-7

Windau, J., & Shen, W. M. (2010). An inertia-based surface identification system. *Proceedings - IEEE International Conference on Robotics and Automation*, 2330-2335. doi: 10.1109/ROBOT.2010.5509310

Wisanuvej, P., Liu, J., Chen, C. M., & Yang, G. Z. (2014). Blind collision detection and obstacle characterisation using a compliant robotic arm. *IEEE International Conference on Robotics and Automation*, 2249–2254. doi: 10.1109/ICRA.2014.6907170

Zug, S., Seidel, M., Beckhaus, J., & Winkelsträter, N. (2017). Collision detection for RoboCup@Work Competitions. *Springer Nature Switzerland*(33). doi: 10.1007/978-3-030-00308-1_31

A. Appendix - Literature Review Abstract And Results

A-1 *Abstract*

Detecting and interpreting collisions is an important aspect of safe robot operations. When the environment does not allow for visual based detection technologies, ‘blind’ collision detection systems could be used. This literature study will focus on different methods suitable for blind mobile robots to detect collisions after impact and distinguish different collision objects. This can be put into a singular research question: ”Which collision detection algorithm is suitable for a blind mobile robot to distinguish intended contact from unexpected collisions?”. A systematic search was performed to find related articles to answer the research question. The results can be classified based on the method and sensors used for the collision detection system. Methods found in the literature are divided into model-based and non-model-based methods. Model-based methods rely on internal control: current sampling, torque sensors, the velocity deviation or position deviation. Non-model-based methods rely on an accelerometer, microphone, tactile surface sensors or current sampling. The results confirm that collisions can be detected and distinguished based on blind sensing. Depending on the desired type of classification task, various methods are suitable to distinguish intended contact from unexpected collisions. Hard collisions can be accurately distinguished from soft collisions using multiple thresholds and parallel use of high- and low- pass frequency filters on acceleration or motor current data. Additionally, processing sound data performed good in discriminating hard from soft collisions. To classify collisions based on the specific material of the object, learning algorithms based on acceleration data performed best. Though, for a mobile robot driving on an uneven floor, it can be difficult to obtain a clear collision signal in order to classify the collision object material.

A-2 *Result Tables*

A number of 24 useful articles were found, of which 10 are model-based, and 14 non-model-based. Two Tables with all the results from literature are developed, where each article is classified and the goal, method, type of robot, sample rate and results of the CD-method are described (Table 9, 10).

TABLE 9: Literature overview of model-based collision detection methods

Year	Author	Times cited	Category	Title	Goal	Collision detection method	Sample rate	Robot used	Test & Environment	Results
2008	**Haddadin S. De Luca**	247	**Joint torque sensor**	**Collision Detection and Reaction: A Contribution to Safe Physical Human-Robot Interaction**	Show how reactive control strategies, to detect and distinguish unexpected collisions from intended cooperation, can make a significant contribution to safe physical human robot interaction	Two collision detection methods are described. The first method observes the disturbance of the angular momentum based on the angular velocity and determines an estimated version of the external torque. In the second method the external torque is determined using the dynamic model and the measured joint torque. The collision detection thresholds are 10% of the nominal torque.	1000 Hz	Robotic manipulator arm	The robotic arm collides with a human standing in the path of the robot.	All collisions could be detected with both methods. The robot is able to detect and distinguish unexpected collisions from intended interaction. The robustness of the second method is higher since only the joint torque sensor noise is present, which is considerably lower than the velocity sensor noise in the first method.
2012	**Cho C et al**	40	**Joint torque sensor**	**Collision Detection Algorithm to Distinguish Between Intended Contact and Unexpected Collision**	Propose a novel collision detection algorithm which can distinguish intended contacts and unexpected collisions.	The residual external torque is estimated based on the generalized momentum. When this exceeds a predefined threshold, a collision will be detected. Unexpected collisions show a faster rate of change of the external torque than intended contacts. Thus, a high-pass filter is added to the collision detection algorithm to extract the high-frequency components and detect unwanted collisions.	1000 Hz	Robotic manipulator arm	The experiments were done while the robot in 'teach and play-back' mode, were intended contacts occur frequently. An unintended collision is simulated by colliding with a human standing in the path of the robot.	The robot can distinguish intended contact from unexpected collisions. It only reacted to collisions, and not to input motion commands from the human. The results show no accumulation of errors and detects a collision within 2 ms.
2013	**Geravand et al**	126	**Current**	**Human-Robot Physical Interaction and Collaboration using an Industrial Robot with a Closed Control Architecture**	To present and evaluate an approach to collision handling for industrial robots with a closed control architecture, without prior knowledge from the robots dynamic model.	The system does not need to estimate joint velocities, but processes motor current measurements during robot motion and compares them with time-varying thresholds that depend on the input trajectory. By parallel adding a high- and low-pass filter on the motor currents, it is possible to distinguish accidental collisions from intentional soft contacts.	83.33 Hz	Robitic arm	The robot collides with a human while executing motion and should stop moving. Secondly, a soft intended contact is applied to the robot which should then switch to collaboration mode.	It is estimated that collisions producing more than 0.2 Ampere in any of the motor currents will be detected, however, an exact number is not known. All intentional and unintentional collisions could be distinguished.
2013	**Ho C, Song J**	17	**Joint torque sensor**	**Collision Detection Algorithm Robust to Model Uncertainty**	Build a collision detection algorithm which does not require the use of acceleration data to be robust to model uncertainties.	The external torque from the measured joint torques is extracted without estimating acceleration: a butterworth 2nd-order Band Pass filter removes torque due to motion, after which a high pass filter on the filtered torque signal is used to detect the hard collisions.	1000 Hz	Robotic manipulator arm	A model was build to test the proposed collision detection algorithm, where the actuators and the harmonic drives are intentionally not modelled. A collision detection experiment was performed 4 times with varying payload. The robotic arm collides with a load cell attached on a base.	The experimental results show that the proposed algorithm can detect the collisions even with the applied, severe, model error. The proposed algorithm detected the collisions within 6 ms
2013	**Kim K et al**	7	**Position**	**Contact sensing and mobility in rough and cluttered environments**	Develop a method to precisely track a planned path on an inclined surface and detect and respond to unknown collisions	Making use of absolute position/orientation sensing and current control, based on the wheel odometry the actual position can be compared with the desired trajectory.	2000 Hz	Omnidirectional mobile robot	The robot collides with an unknown wall while trying to follow his desired circular trajectory. The orientation of the wall is changed to various angles for different collisions.	The wall is for all experiments correctly detected. The omnidirectional robot is able to quickly respond to contacts through the use of absolute position sensing and current control.
2015	**Golz et al**	42	**Joint torque sensor**	**Using Tactile Sensation for Learning Contact Knowledge: Discriminate Collision from Physical Interaction**	To develop a classification system by combining collision monitoring and contact state estimation from proprioceptive sensation.	The external torque is estimated according to the method of Haddadin et al. (2008). Based on a simple physical model, different contact profiles between human and robot are identified by their linear and non-linear features. Only joint torque sensors and labeled training data per contact class are used and processed in a SVM-based algorithm.	1000 Hz	Robotic manipulator arm	65 training samples are labeled and implemented; in the classes head, chest and interaction.	The method resulted in accurate classification results for both simulated and real experiments. The correct classification rate is 90% for head impacts, 100% for chest impacts and 90% for interaction.

(continued) - Overview of model-based collision detection methods

Year	Author	Times cited	Category	Title	Goal	Collision detection method	Sample rate	Robot used	Test & Environment	Results
2016	**Kim K et al**	14	**Torque sensors**	**Full-body collision detection and reaction with omnidirectional mobile platforms: a step towards safe human–robot interaction**	Develop general sensing and control methods for quickly reacting to collisions in statically stable mobile bases.	The methods rely on wheel drive torque sensing instead of IMU to determine the direction and magnitude of the collision forces. A model for the base of the omnidirectional mobile robot is build with estimated roller friction and dynamic effects. Three rotary torque sensors were incorporated in the drivetrain. The external force measurement is based on a moving average filter of the torque sensor signals. When the external forces exceed the threshold, the robot detects the collisions.	2000 Hz	Omni-directional mobile robot	Drives in an office environment where the robot collides with objects and humans while moving	All collisions were detected. Overall a maximum error of 45% was measured for the magnitude, 3.3% for the direction and 18 % for the location of the external forces. The low accuracy of the magnitude can be due to mechanical limitations of the robot and the connections of the wheels to the torque sensors.
2017	**Narukawa et al**	8	**Force and Torque**	**Real-time collision detection based on one class SVM for safe movement of humanoid robot**	To develop a new real-time collision detection method for the safe movement of humanoid robots, by only providing the normal movement data.	The method is based on a one class Support Vector Machine . Only normal motion data is provided, since collision data is difficult to obtain. Real time collision data that does not appear in the training data can be detected as an outlier. The difference between the expected and measured values is used, using force and toque sensors.	1000 Hz	Humanoid robot	Training data contains 14 types of motion data. These were first tested without collision to detect false positives. Next, 12 experiments with collisions were performed, by pushing a metal bar against part of the robot's body while moving.	All collisions were detected within 0.01s after the robot was pushed, without any false positives. The method is practical for real-time collision detection since the detection delay is short enough.
2017	**Wenzhong X et al**	2	**Velocity deviation**	**Sensorless Robot Collision Detection Based on Optimized Velocity Deviation**	Develop a new method to detect robot collision by using the ideal velocity and the actual velocity deviation.	Due to the delay of the robots servo system, the actual velocity lags behind the calculated velocity. A digital low-pass filter is used to delay the calculated velocity to run equal with the actual velocity. A collision detection threshold is set on the obtained velocity deviation for the algorithm to detect a collision.	1000 Hz	Robotic manipulator arm	The robotic arm follows a smooth path and a soft object causes a collision with the robot.	The optimized calculated velocity corresponds very well with the actual velocity. The detection algorithm can detect the collisions accurately. The proposed scheme is simple and does not affect the normal motion control of the robot with good real-time and high precision.
2021	**Park K**	0	**Current based**	**Learning-Based Real-Time Detection of Robot Collisions Without Joint Torque Sensors**	The development of learning-based real-time collision detection methods that compensate for unmodeled friction and uncertain dynamic effects (e.g., parameter uncertainties, measurement noise, backlash, and deformations).	The learning-based detection algorithms are based on a support vector machine (SVM) and a convolutional neural network (CNN). A friction model or manual tuning of collision detection thresholds is not required	1000 Hz	Robotic manipulator arm	Collisions performed by applying hard and soft collisions on the links of the robot	The learning-based detection methods can accurately detect hard and soft collisions in real-time while compensating for uncertainties in the model. The CNN based method shows better performance if more training data is available, whereas the SVM based method performs better with less data. Detection failures: CNN: 0/578 hard & 2/209 soft and for the CVMR 1/578 hard and 0/209 soft.

TABLE 10: Literature overview of non-model-based collision detection methods

Year	Author	Times cited	Category	Title	Goal	Collision detection method	Sample rate	Robot used	Test & Environment	Results
2001	**Femmam S et al**	23	**Microphone**	**Perception and Characterization of Materials Using signal processing Techniques.**	Identify sensor-derived measures related to the properties of a material, which can be used to classify objects into their various material classes.	Sound is digitized and the discrete signal is analyzed in four steps: 1) Compute the spectogram using the time-frequency representation based on the Fourier transform. 2) Determine where the transient contact ends and the signal begins. 3) Determine the bands of concentrated signal energy. 4) Calculate the angle of internal friction for each band.	22050 Hz	Legged robot	Thin sample rods of duralumin, wood, plexiglass, soil, sand, carpet and plastic are produced. Each sample rod is struck with the leg of the robot. Signals are acquired using the microphone sensor attached to the leg of the robot.	Based on band frequency and amplitude energy various materials could be identified. However, a higher sampling rate is needed to acquire reliable measurements from non-metals.
2004	**Vail D, Veloso M**	65	**Acceleration**	**Learning from accelerometer data on a legged robot**	Show that a legged robot can identify the type of surface that it is walking on by using accelerometer data.	Method based on supervised learning. Uses a one second sliding window to determine the variance in x, y, and z accelerations as well as the (x, y), (x, z), and (y, z) correlation coefficients over each window. Labeled training data was passed to the C4.5 decision tree learning program to generate the tree and performed an online calculation.	125 Hz	Legged robot	Concrete, soft carpet and grass ground evaluated, where the robot marched in place on each surface.	Overall performance: 84.9 % correctly catergorized, 15.1% incorrect. The classifier has the most trouble disambiguating between the soft carpet and grass, as they are very similar and produce similar accelerometer data.
2007	**He et al**	3	**Acceleration**	**Collision signal processing and response in an autonomous mobile robot**	Extract pure collision information trough a detecting window based on a median filter.	The standard deviation of the acceleration signal is computed using median filtered acceleration data obtained from a running window of 7 consecutive values. This standard deviation of the collision acceleration is compared against a threshold to detect collisions.	200 Hz	Differential drive mobile robot.	Experiment is carried out in a room, the robot is running and a soft volleyball is rolled quickly towards the robot in the direction of approximately 0 , 45, 90 , 135 and 180 degrees.	The detecting rates for the bumping direction angles of 0, 45, 90A, 90B, 180 are respectively 100% 60% 53% 50% 80%.
2008	**Mericli et al**	6	**Acceleration**	**A Robust Statistical Collision Detection Framework for Quadruped Robots**	To discriminate collisions from normal walking data based on accelerometer signal patterns.	A smoother is applied to the acceleration signal. This is transformed to the frequency domain using the Fast Fourier Transform. Based on the Sum of Squared Differences (SSD), the change in frequency/power distribution could be determined to detect anomalies. A T-test is performed on the SSD data, where the mean of a sliding window is compared to the learned mean of the normal walking data.	125 Hz	Quadruped Robot	The performance of two robots during a robot soccer game on a flat ground was evaluated. One with the collision detection system and one without.	The collision detection framework proved to detect all collisions and reducing the response time remarkably. It is computationally cheap and fast, and independent to walk speed and surface, as long as the surface is flat.
2009	**Je H et al**	16	**Current based**	**A Study of the Collision Detection of Robot Manipulator without Torque sensor**	Develop a collision detection method for the robot manipulator without torque sensors in an unknown environment, only based on current value and commands of the controller.	The collision observer signal is based on the commander's input and the variation of the current value of the servo system. A fast transition of the current value is determined as a collision, assuming that the motion of the robot arm follows a well-made smooth trajectory behavior.	1000 Hz	Robotic manipulator arm	The value of the motor current is measured using the servo driver monitor. An external force will hit the robot arm while performing a moving task.	All collisions were detected. The proposed collision detection method excluded complex calculations and formulas.
2012	**Guillaume Doisy**	13	**Current based**	**Sensorless Collision Detection and Control by Physical Interaction for Wheeled Mobile Robots**	Apply a sensorless collision detection method developed for robotic arms (De Luca et al, 2005) to a wheeled mobile robot.	The torque output is monitored from the motors. Detection threshold is 80% of the total torque value averaged over the last second.	20 Hz	Differential drive mobile robot.	The robot is traveling at a constant speed on a flat ground, its inertia is limited and it can stop immediately. Heavy to light objects are placed in front of the robot.	Experiments show that collisions can be reliably detected even with light objects. The sensitivity of this method allowed a human operator to stop the robot by lightly pushing the top of the robot.

(continued) - Overview of non-model-based collision detection methods

Year	Author	Times cited	Category	Title	Goal	Collision detection method	Sample rate	Robot used	Test & Environment	Results
2012	**Moorits E, Usk A**	3	**Acceleration**	**Buoy Collision detection**	Develop a low resource demanding collision detection method that is feasible for application on navigational buoys.	Because of the limited amount of computational capability and memory, the (x,y,z) acceleration signals are summed instead of calculating the vector length. This results in a much higher acceleration then in the case of using vectors. The output is filtered with an IRR filter and squared to eliminate negative outputs. The modification of the output is beneficial for the sensitivity of the collision detection threshold.	50 Hz	Buoy	Simulations and tests were performed to verify the method. An artificial collision was created by mounting the sensor on a heavy object and this object is collided with another object. No real collisions with vessels could be recorded.	The method is capable of detecting collisions in real time, even at low acceleration signals. The method was successful: all simulation results were in line with expectations.
2014	**Wisanuvej P et al**	6	**Acceleration**	**Blind collision detection and obstacle characterisation using a compliant robotic arm**	Develop a method that: 1) can detect collisions and the location, direction and magnitude of the collision; 2) identify material properties and; 3) explore and map an unknown environment.	Two thresholds are set to detect the collision: a magnitude threshold and a time threshold. The time window accepts only short peaks, since the vibration of the collision is usually very short. The baseline is determined by calculating the median of the acceleration signal. The vibration signal was extracted with sound processing software YAAFE and processed with various machine learning techniques.	3200 Hz	Compliant robotic arm	A solenoid is used as a tool to hit the robot arm in different locations. The tip of the solenoid is interchanged with four different materials corresponding to: a rubber band, an eraser, a tire and a plastic helmet. For each material, 100 collisions are performed.	The method has achieved 98% collision detection sensitivity. Furthermore, by using sound feature extraction and machine learning techniques, the classifier produces an accuracy of 98% for classifying four different impact materials. Neural network, linear regression and decision trees are the most accurate classification methods.
2016	**Fritzsche et al**	17	**Tactile sensor**	**A Large Scale Tactile Sensor for Safe Mobile Robot Manipulation**	Outline the application of their developed tactile sensor for large sized mobile robots	The tactile sensor is build from piezoelectric material sandwiched between two electrodes and formed in an array. If the sensor signals exceed a given threshold, the motion of the robot can be stopped.	500 Hz	Mobile robot	Extensive collision experiments were performed, using a test setup simulating the physical properties of a human leg.	The signal generation of the tactile sensor is very fast, however the delay of the robot to perform action is considerably bigger. This delay is mainly caused by mechanical inertia, data processing and system reaction time. A safe collision detection for the mobile robot used could only be guaranteed at very low velocities (¡ 0.1 m/s).
2017	**Zug S et al**	0	**Acceleration**	**Collision-Detection for RoboCup@Work-Competitions**	Determine the optimal collision detection method of three different concepts for mounting the acceleration sensors: robot centric, mounted on the environment and both.	Three different detection algorithms are implemented and evaluated: 1) an acceleration gradient based approach and a threshold; 2) a statistical evaluation of the quantiles of the last 25 samples related to the actual measurement; 3) a T-test for the means of historic (n= 100) and a current sample set (n=10).	200 Hz	Omni-directional mobile robot	Experiments are conducted in a room where the characteristics of the floor are unpredictable. The robot collides with a wall at different speeds. A series of bumpers in front of the robot generate the ground truth signal.	For the first algorithm, the number of false-positives increases significantly at higher speeds. The second algorithm provides a constant number of false detections for all velocities. The T-test generates a reliable but delayed result for lower speeds, but is not accurate for higher speeds. Therefore, correct identification of collisions requires a combination of methods. These algorithms by themselves cannot generate reliable output for different speeds.
2019	**Becker F, Ebner M**	0	**Acceleration**	**Collision Detection for a Mobile Robot using Logistic Regression**	Logistic regression to detect and classify collisions from data measured by an accelerometer on a mobile robot.	After recording training data, pattern recognition with logistic regression was applied based on 19 data points. The method is based on motor currents and acceleration data. Ground truth data of impacts is obtained by pressing a button when a collision is detected.	125 Hz	Differential drive mobile robot	The mobile robot drives in a cluttered office environment. A button is manually pushed when a collision is observed.	13 out of 14 collisions were detected. Classifications of the collisions were possible, however it did not classify collisions of type A. These were probably not well represented by the available training data.
2019	**Speleers P, Ebner M**	1	**Acceleration**	**Acceleration Based Collision Detection with a Mobile Robot**	Evaluate and compare tree methods for acceleration based collision detection.	Three different methods for CD are evaluated: 1) a threshold on the acceleration signal along the driving direction without filter; 2) A running median filter; 3) a frequency based averaging filter. Method (2) is developed by He et al. and (3) by Moorits and Usk. Thresholds are determined after normal driving behaviour is measured	67 Hz	Differential drive mobile robot	Data is collected during a) normal driving behaviour, b) decelerating to a complete stop and c) by placing an obstacle in front of the robot. Test data is obtained for collisions with 3 different velocities.	The method of Moorits and Usk is the only method that is able to accurately detect all collisions (8/8). The method of He et al detected 7 out of 8 collisions, and one false positive.

B. Appendix - Natural frequency determination

The natural frequencies of the robot are determined with a hammer and the IMU sensor. The robot was hit 50 times with a rubber hammer from the front and side in horizontal direction, while measuring the acceleration in the direction of impact. From the FFT plot 16 it can be deduced that the excited frequencies are mainly above 300 Hz, with the major peak at 370 Hz.

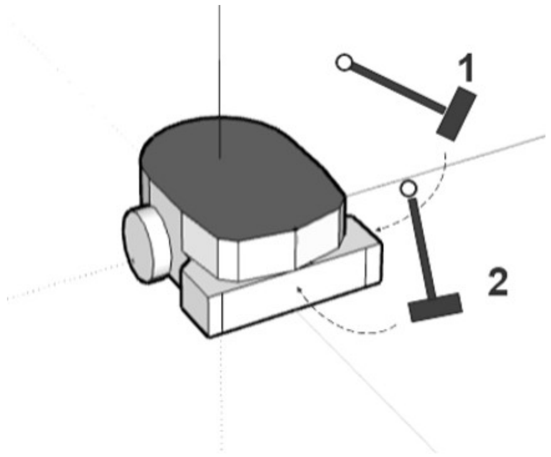


Fig. 15: Location of impact from rubber hammer concerning natural frequency determination: front and side   of the robot in horizontal direction

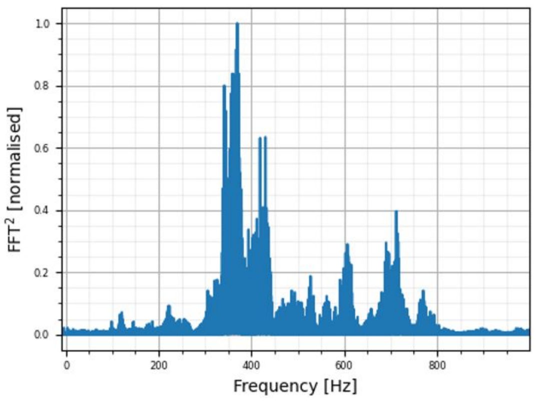


Fig. 16: Power spectral density of natural frequencies of the robot excited with a rubber hammer 50x

C. Appendix - Type III detection

C-1 *Visual inspection of raw data for anomalies*

Figure 17 shows the wheelspeed, calculated force and motor current during collision type III (impact in free ride of robot). An increase in energy of 11.7% was expected due to collision with a 40 kg soft object. However, no causal changes due to the object can be seen. Furthermore, in the acceleration signal in impact direction there was no visual change in frequency or magnitude.

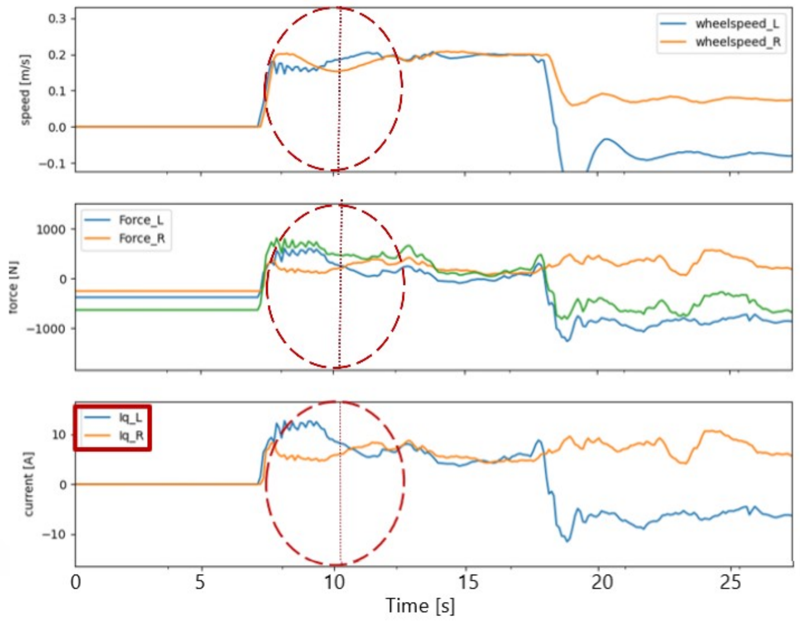


Fig. 17: Wheelspeed, calculated force and motor current during collision type III (red cirlce)

C-2 *Data filtering*

The acceleration collision data is filtered with a low pass filter of 100 Hz, to filter out the natural frequencies. However, the moment of collision could still not be visually determined. Secondly, the frequency spectrum of collision type III data is determined with an FFT plot and compared with no_collision data. The hypothesis was that impact with an object would change the frequency bandwith. However, no recurring difference in the frequency spectrum could be seen before and after collision with the soft object.

C-3 *Machine learning collision detection on acceleration data*

From experiment B, pure collision signals of acceleration data in impact direction was gathered. The suggested method to detect collisions was to classify the data with open-world classification into 'hard_object', 'soft_object' and 'no_collision' category. 10,000 sequences of 'no collision' are randomly generated from experiment B, together with 300 labelled hard and soft collisions. Classifiers MLP and Random Forest are trained with 80% training data. The confusion matrix of Random Forest and MLP on 20% test data is shown below. Random Forest false predicts three more labels then the MLP based algorithm.

Using a sliding window of 100ms, acceleration data gathered during robot movement is passed trough thealgorithms.
In Figure 20 below, 10 collisions of the robot with the hard wall can be seen. It shows the predicted detected hard objects with MLP detection classifier (in green) and the true collisions(in red). It can be seen that 9/10 hard collisions were detected, and 2 false positives are generated.
However, detection accuracy of collision with a soft object is poor(Fig. 21. The dotted black line shows the predicted softbumps, while the red line indicates true softbumps. Many false positives and false negatives were predicted.

C-4 *Conclusion*

The impact signal from collisions with a *soft* object of 1/10th the weight of the robot could not be detected during free movement of the robot. Thresholds on acceleration and motor current data proved to be not applicable. The acceleration and power change at the time of collision was also visually undetectable, despite different high- and low-pass filters applied. The frequency spectrum of movement with and without object did not empirically change.

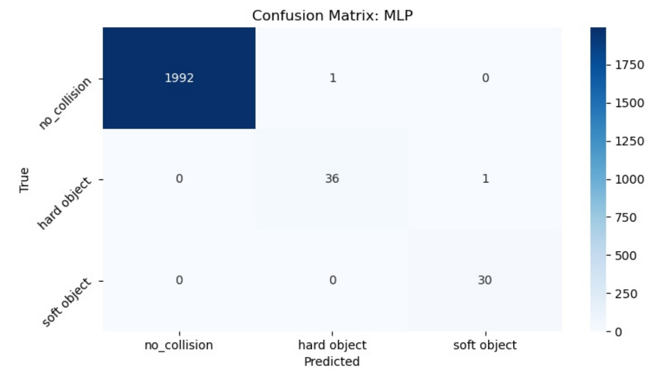


Fig. 18: Confusion matrix of MLP classifier

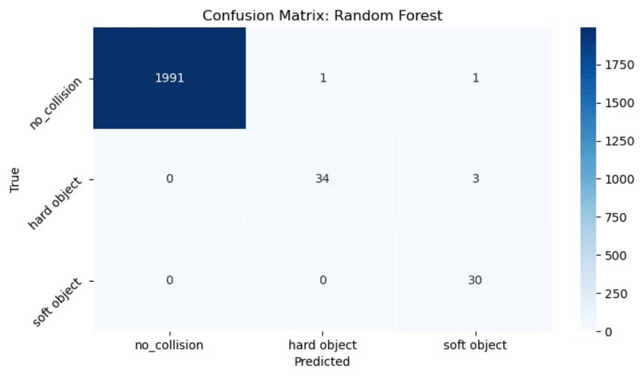


Fig. 19: Confusion matrix of Random Forest classifier

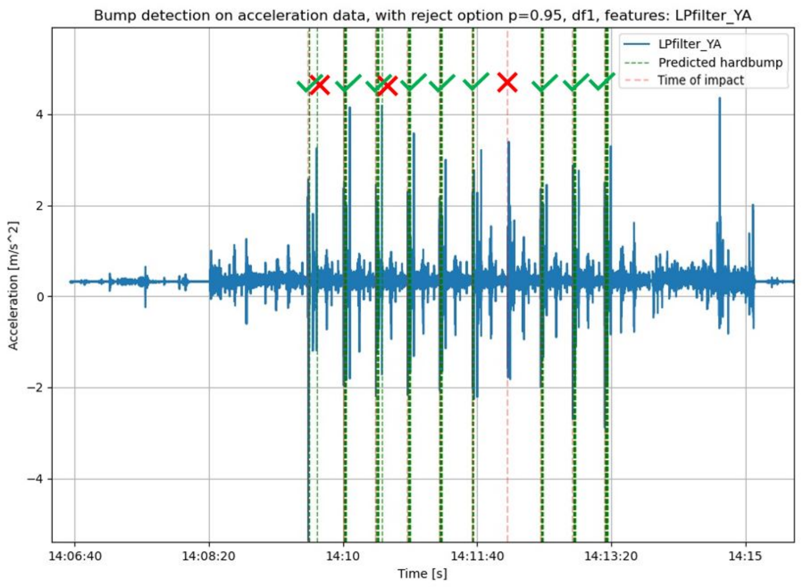


Fig. 20: Ten wallbumps from Test 2 with frictionless skits, predicted with MLP algorithm with low-pass filter Y-acceleration data as input feature

This can be explained by the weight of the robot being ten times heavier than the object; the momentum presumably overruling a noticeable impact from the soft object. In addition, the cleaning robot has high frictional forces with the ground due to rubber strips, which fluctuate enormously in time and varying ground conditions. Furthermore, the robot's speed is low, varying from 0.1 - 0.2 m/s; in combination with the softness of the object, this will significantly reduce the magnitude of the impact force.

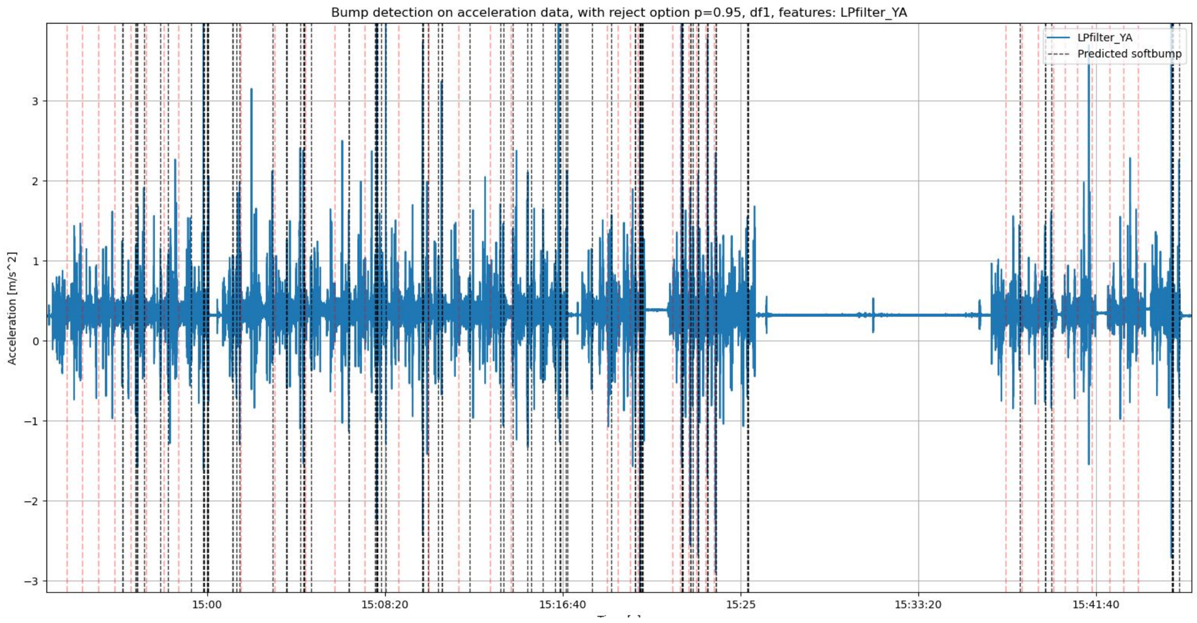


Fig. 21: Big section of the acceleration dataset during robot movement. Shows the poor accuracy of predicted soft collisions (black) compared to true collisions (red) with the MLP algorithm

D. Appendix - Materials

D-1 Sandbag

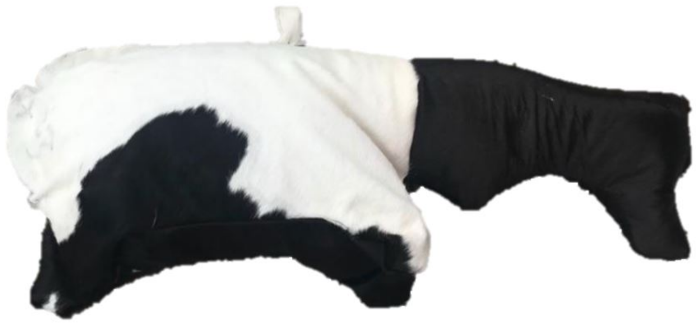
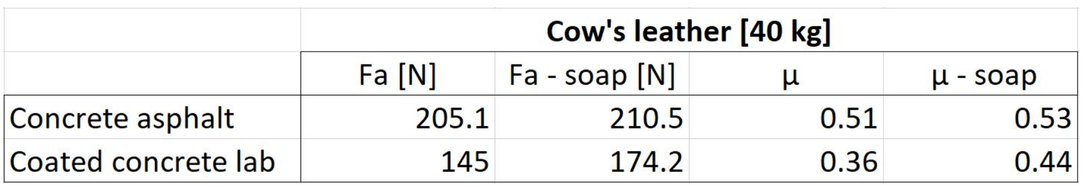


Fig. 22: Dummy leather bag filled with sand, representing 'sandbag'

D-2 Friction properties of floor in lab environment with different materials



	Cow's leather [40 kg]			
	Fa [N]	Fa - soap [N]	µ	µ - soap
Concrete asphalt	205.1	210.5	0.51	0.53
Coated concrete lab	145	174.2	0.36	0.44

Fig. 23: Friction coefficients of 40 kg cow's leather bag filled with sand versus different floor types in lab environment. Dry and soapy floor, object used in all experiments as 'sandbag'.

	Rubber strip [7 kg]			
	Fa [N]	Fa - soap [N]	µ	µ - soap
Coated concrete lab	5.27	2.58	0.75	0.36

Fig. 24: Friction coefficients of rubber strip with 7 kg weight on top versus coated concrete in lab environment. Dry and soapy floor

E. Appendix - Figures

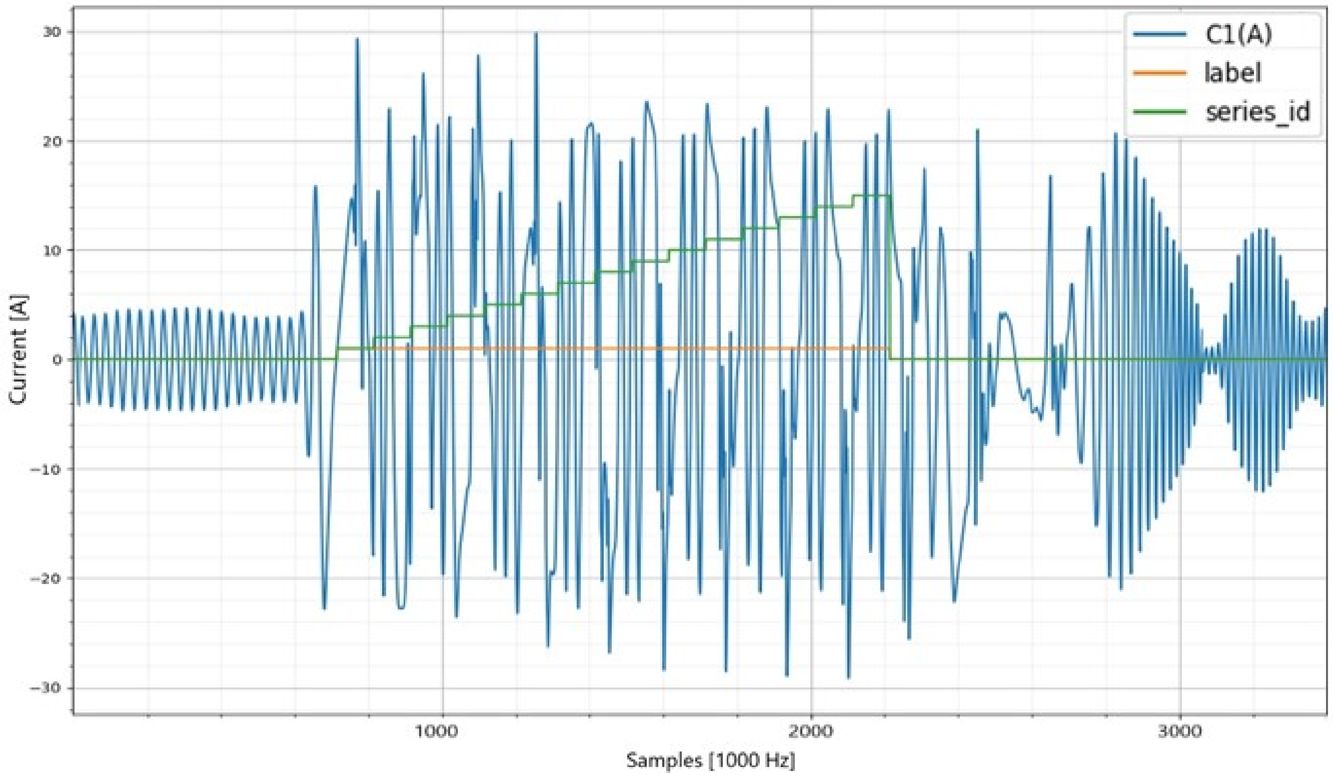


Fig. 25: One out of tree phase's stator current (C1), labbelled as 0 (uneventful) or 1 (collision). Eventful data of one collision is splitted in 15 sequences of 100ms for training

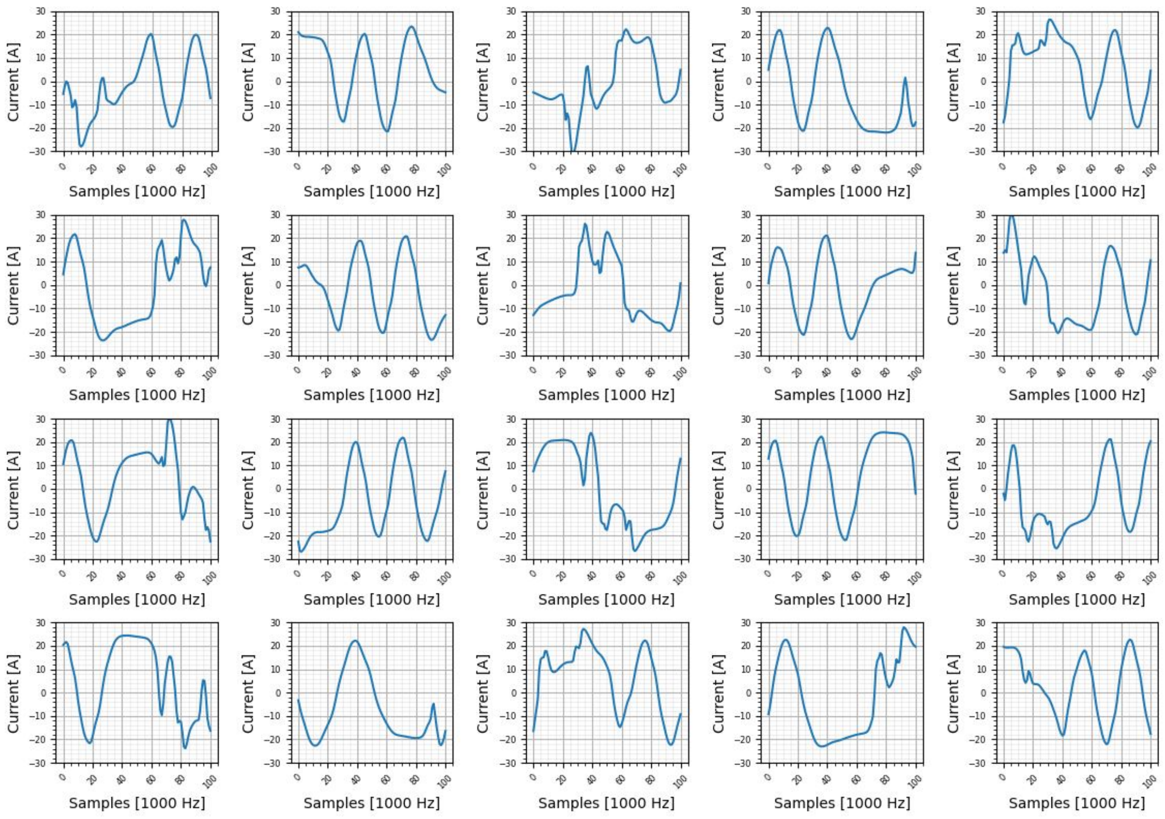


Fig. 26: (20 out of 380) Phase current sequences for learning detection algorithm EVENTFUL data during locked rotor

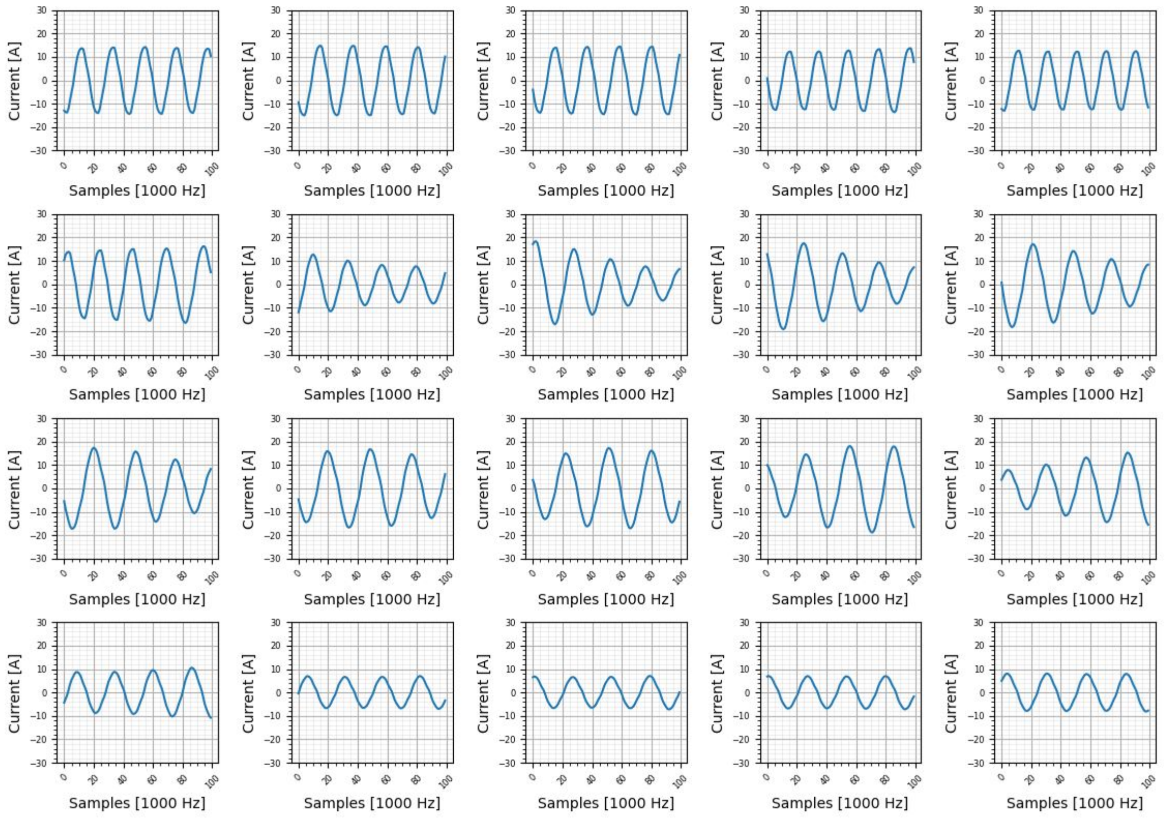


Fig. 27: (20 out of 1000) Phase current sequences for learning detection algorithm UNEVENTFUL data during normal movement

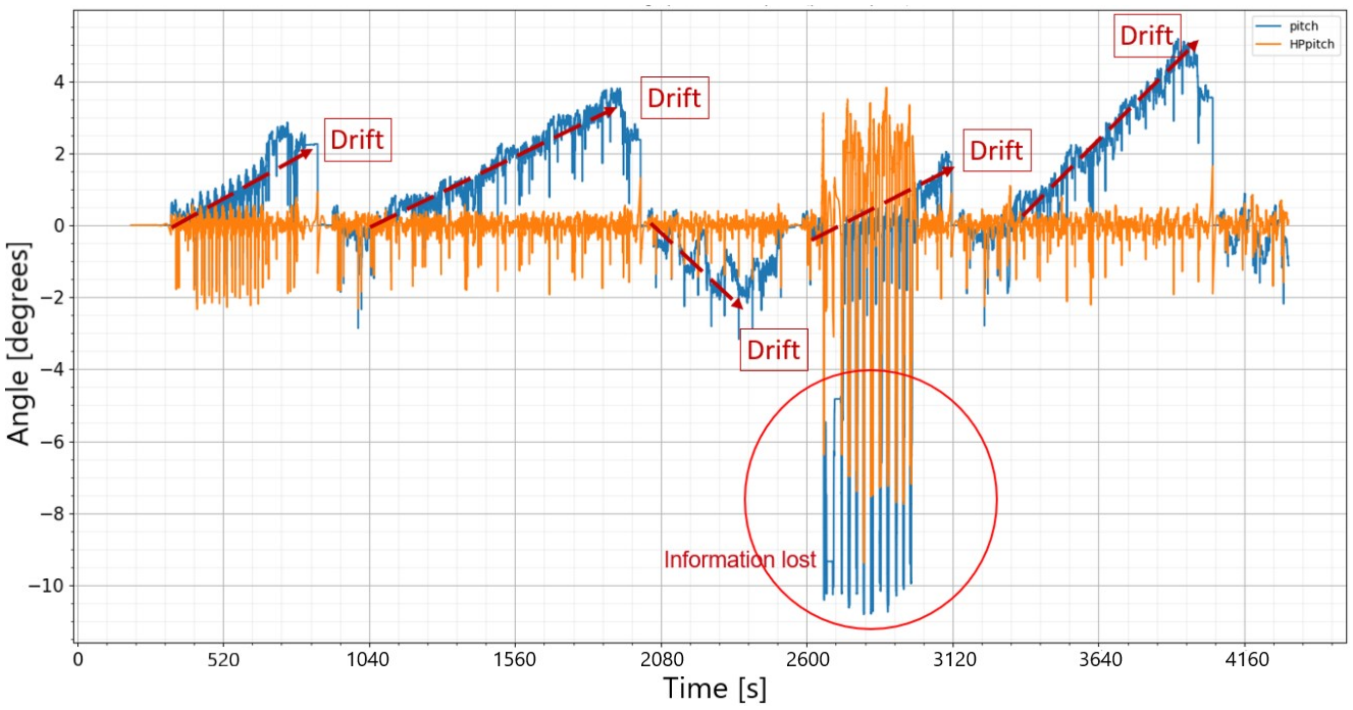


Fig. 28: High-pass (0.01Hz) filter on pitch angle data including drift, sampled at 10 Hz. Shows loss of information in magnitude with respect to raw data

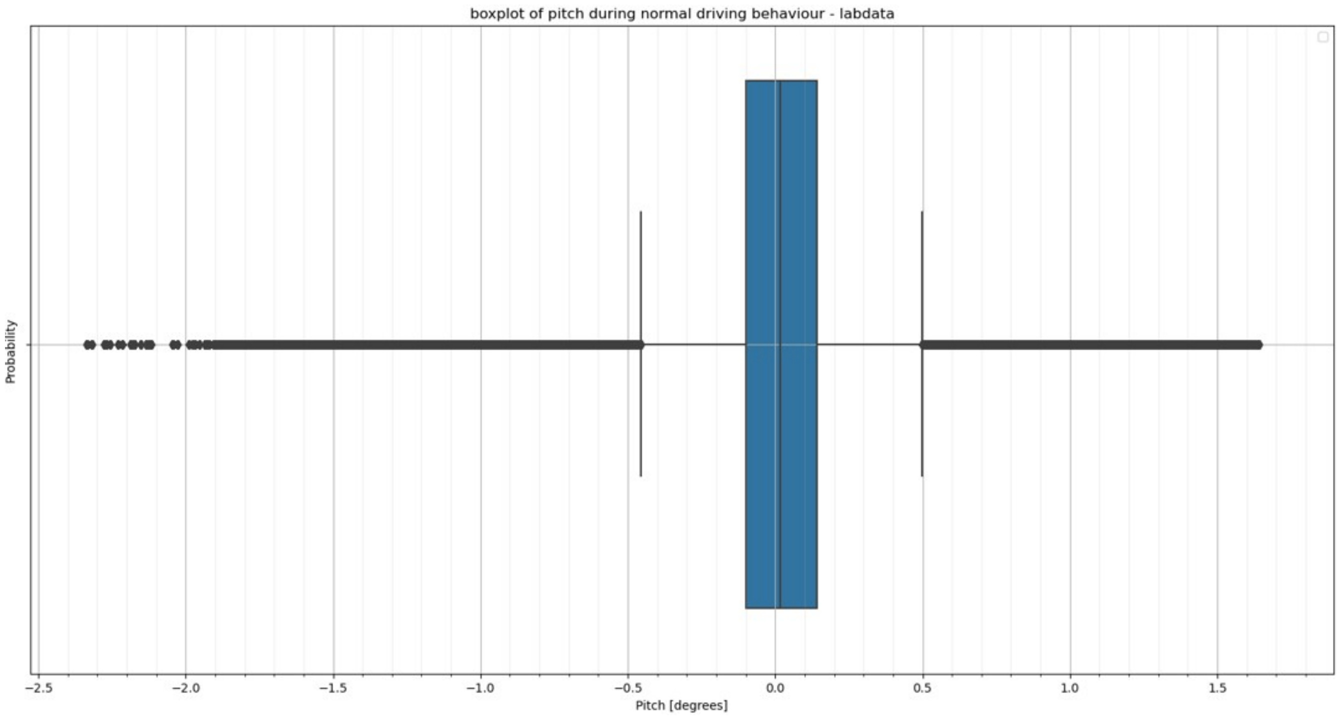


Fig. 29: Boxplot of pitch angle data during uneventful driving conditions. Threshold for eventful data is set on -5.0 degrees, to eliminate false positives in future unseen data

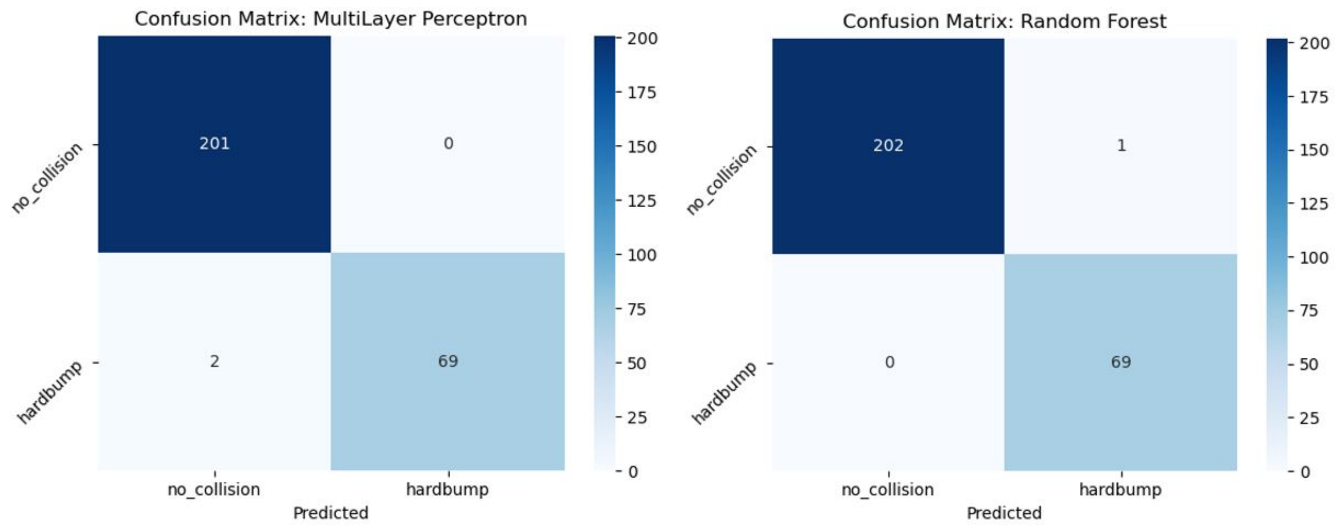


Fig. 30: Best performing confusion matrix for blocked wheels of MultiLayer Perceptron resp. Random Forest after 10 iterations

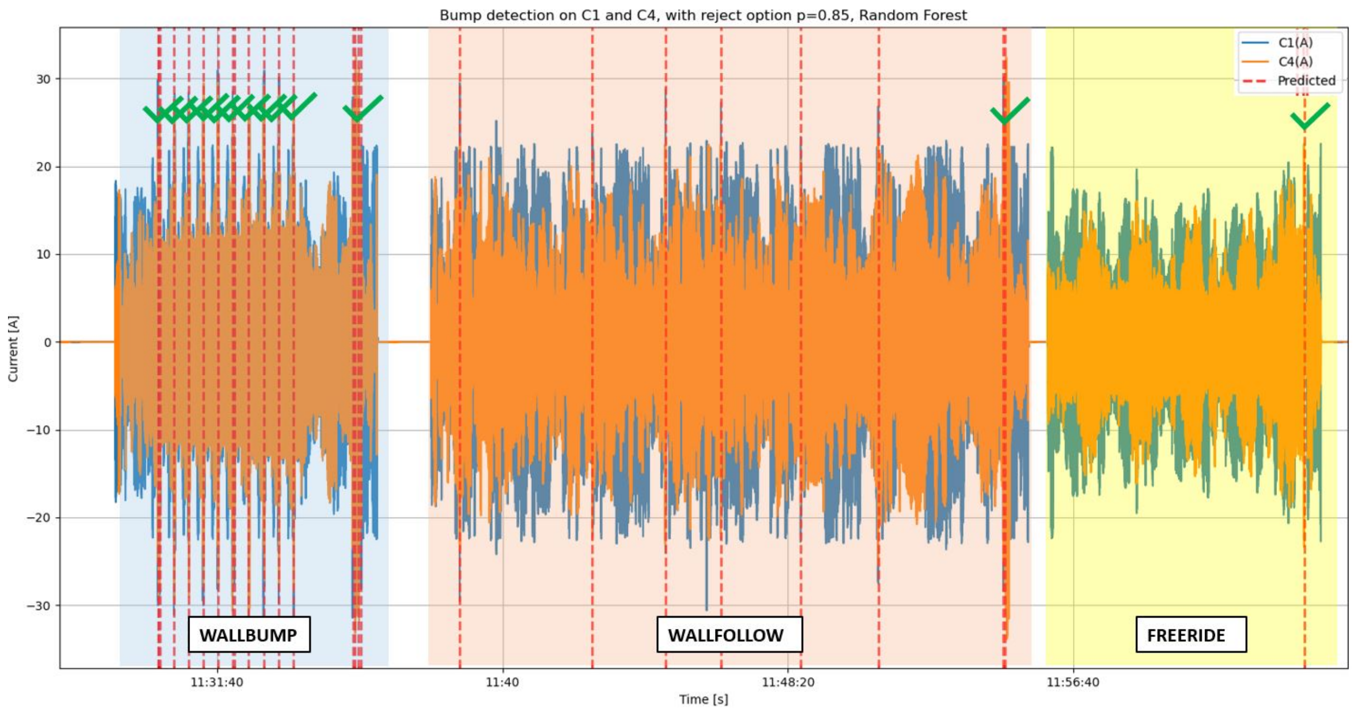


Fig. 31: Test data experiment A with 13 collisions (green V). Phase currents from left (C1) and right (C2) motors. Predicted collisions by RF based algorithm have been marked with a red dotted line.

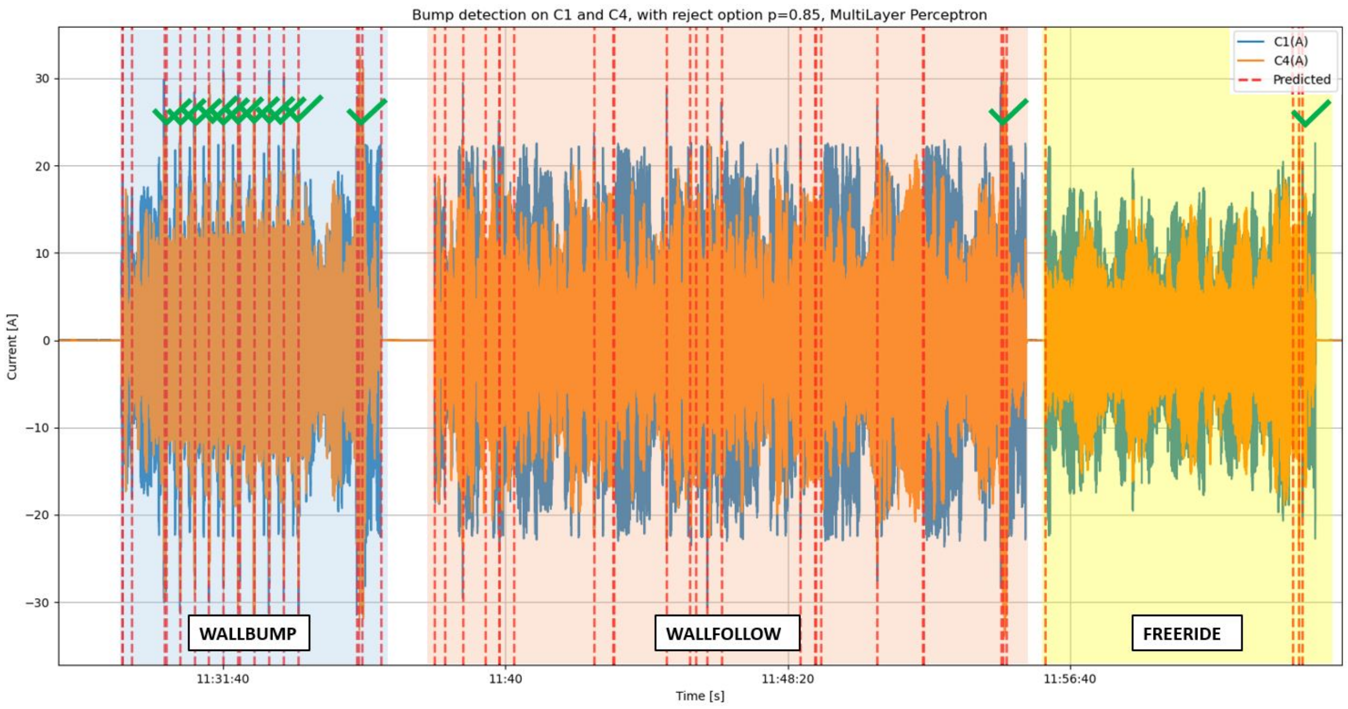


Fig. 32: Test data set with 13 collisions (green V). Phase currents from left (C1) and right (C2) motors. Predicted collisions by MLP based algorithm have been marked with a red dotted line.

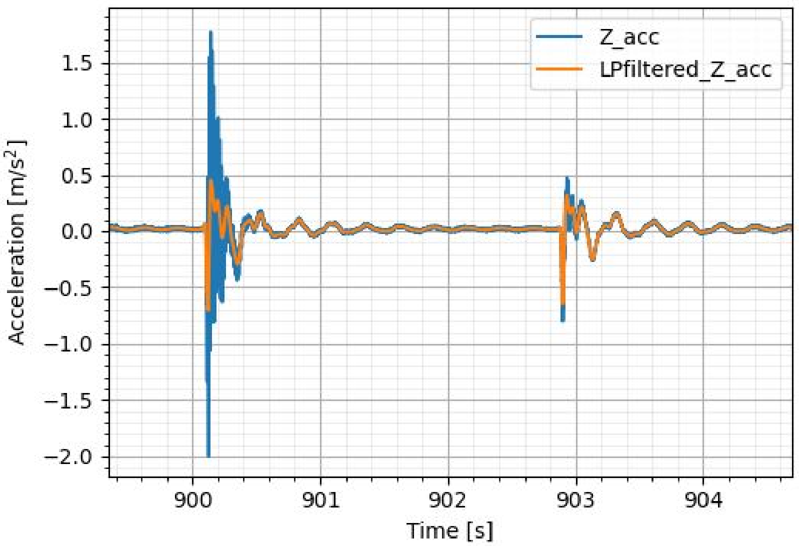


Fig. 33: Shows importance of filtering acceleration signals for classification. Acceleration signal from two collisions in impact direction with a soft object in experiment A

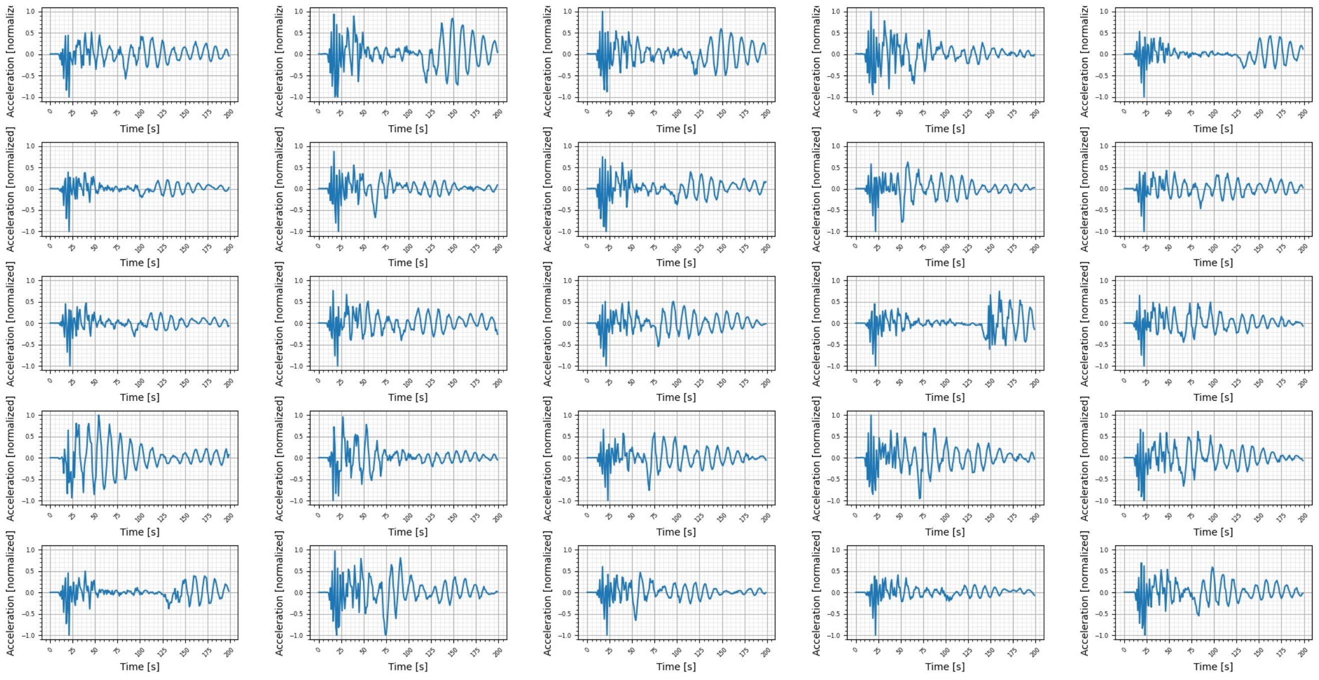


Fig. 34: Hard object pendulum test 25 bumps raw data

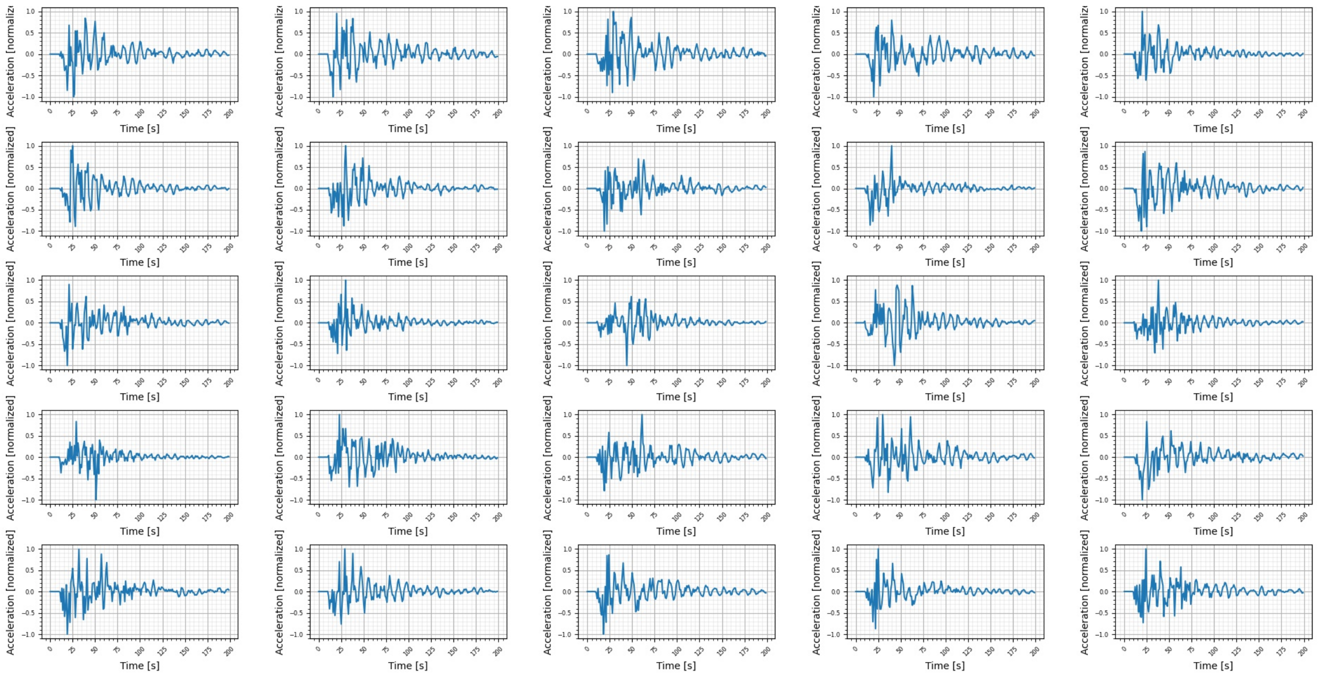


Fig. 35: Soft object pendulum test 25 bumps raw data

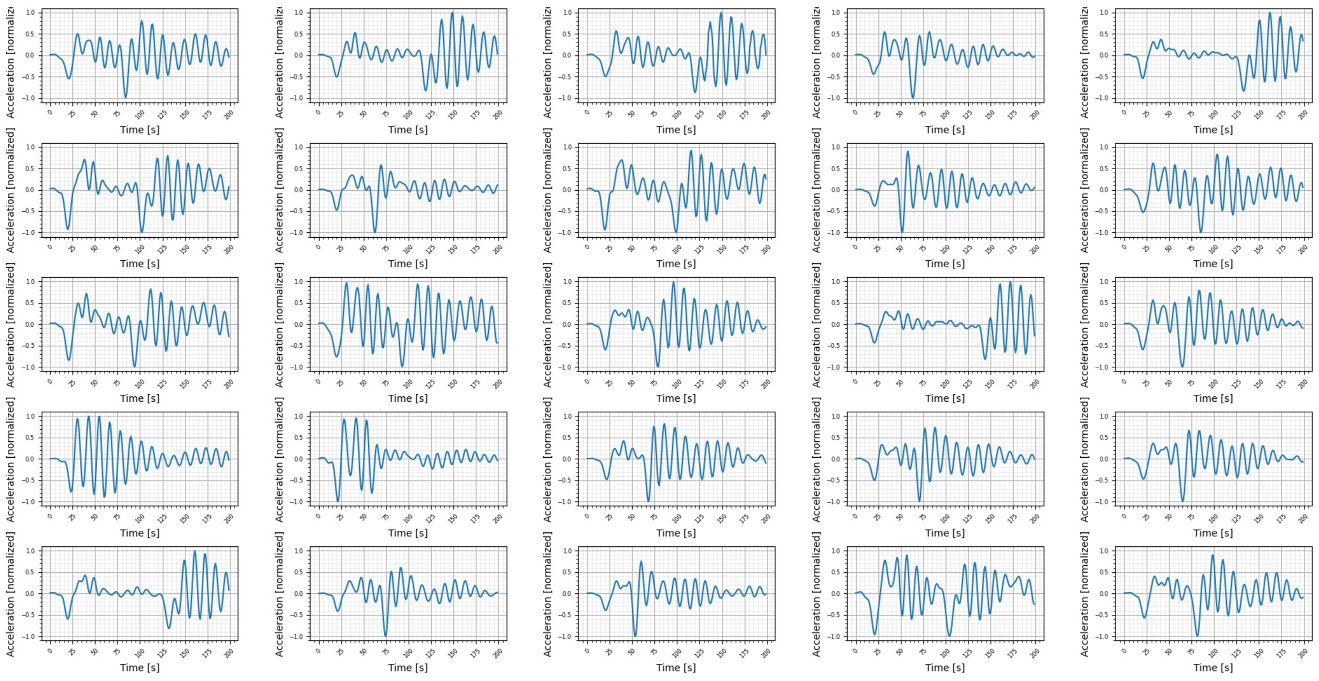


Fig. 36: Hard object pendulum test 25 bumps LP filtered 100 Hz cutoff

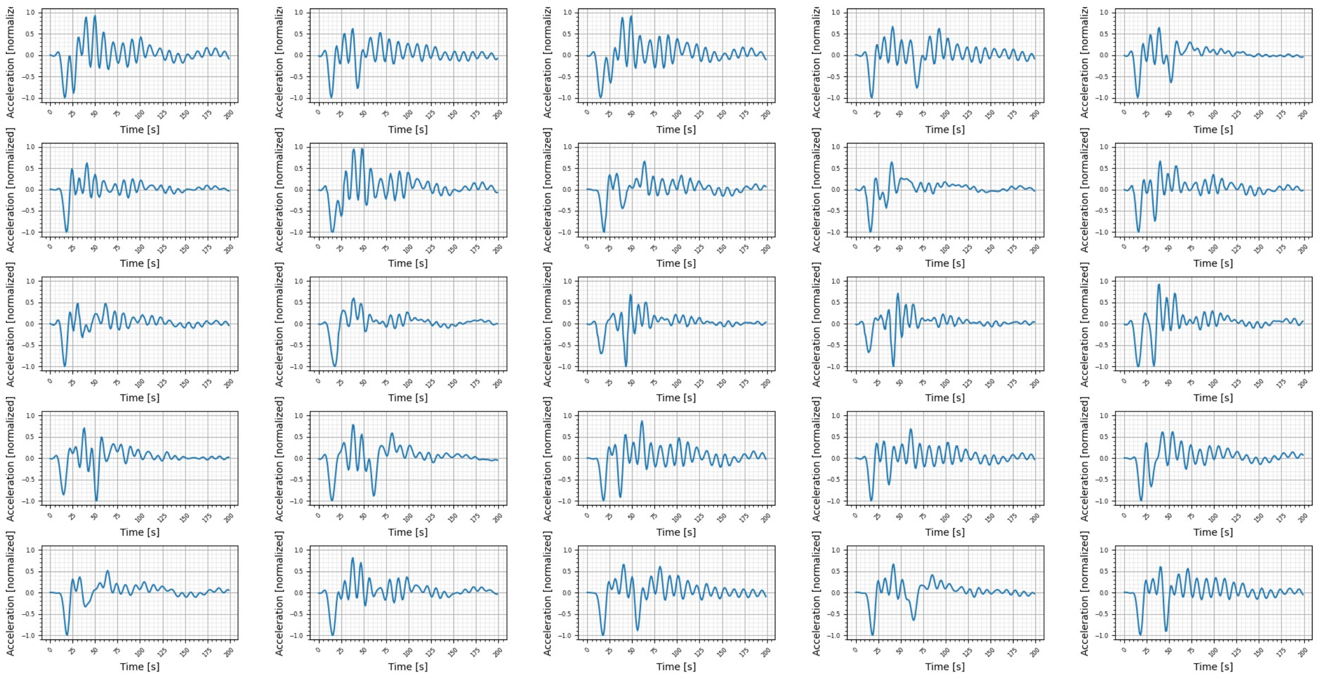


Fig. 37: Soft object pendulum test 25 bumps LP filtered 100 Hz cutoff

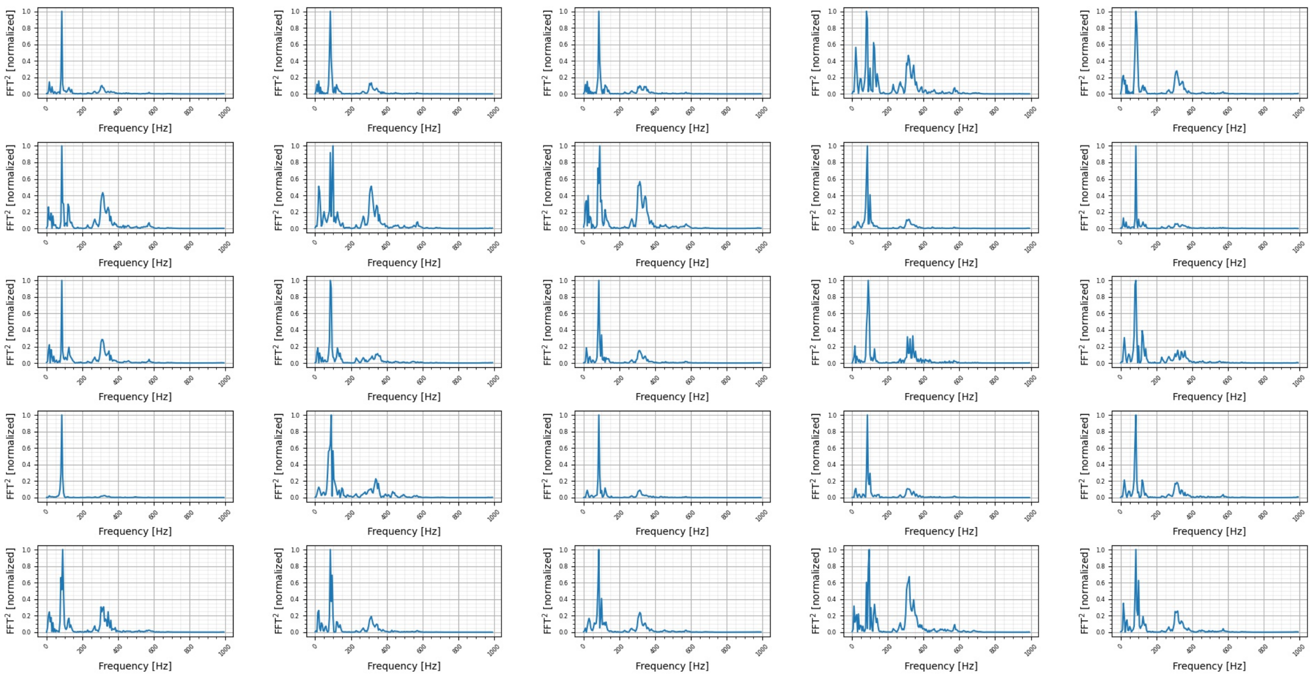


Fig. 38: Hard object pendulum test 25 bumps FFT plot

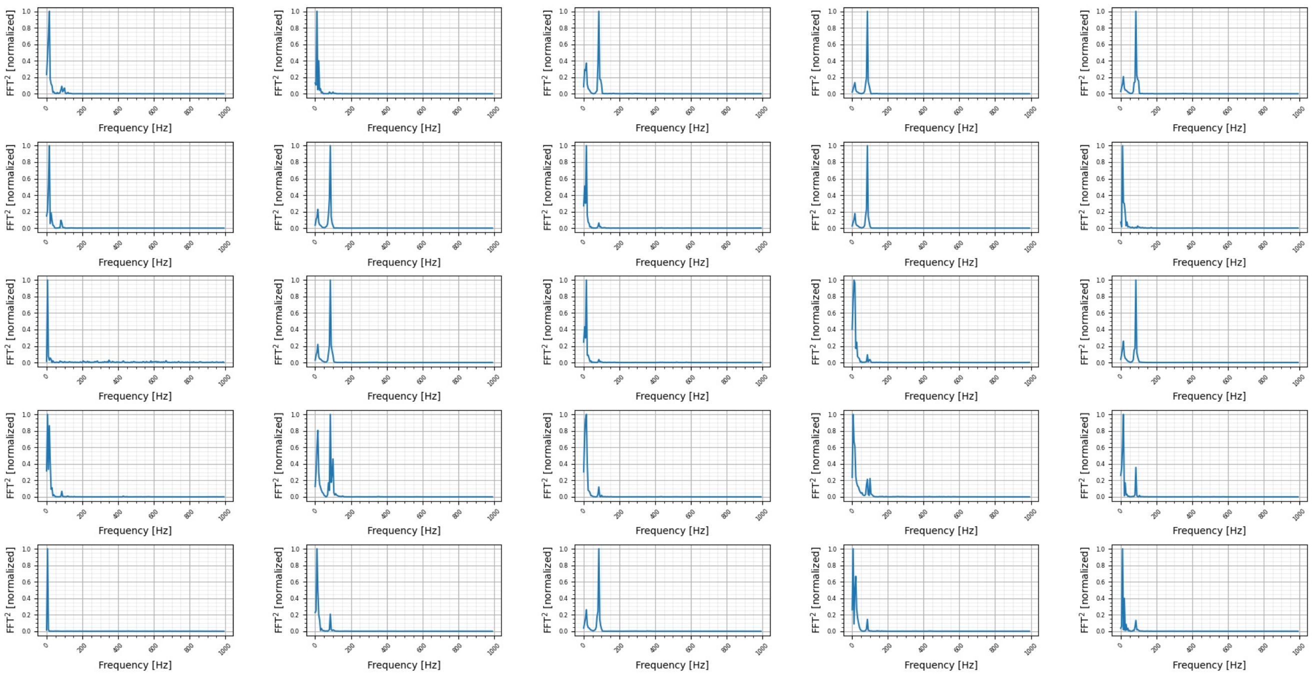


Fig. 39: Soft object pendulum test 25 bumps FFT plot

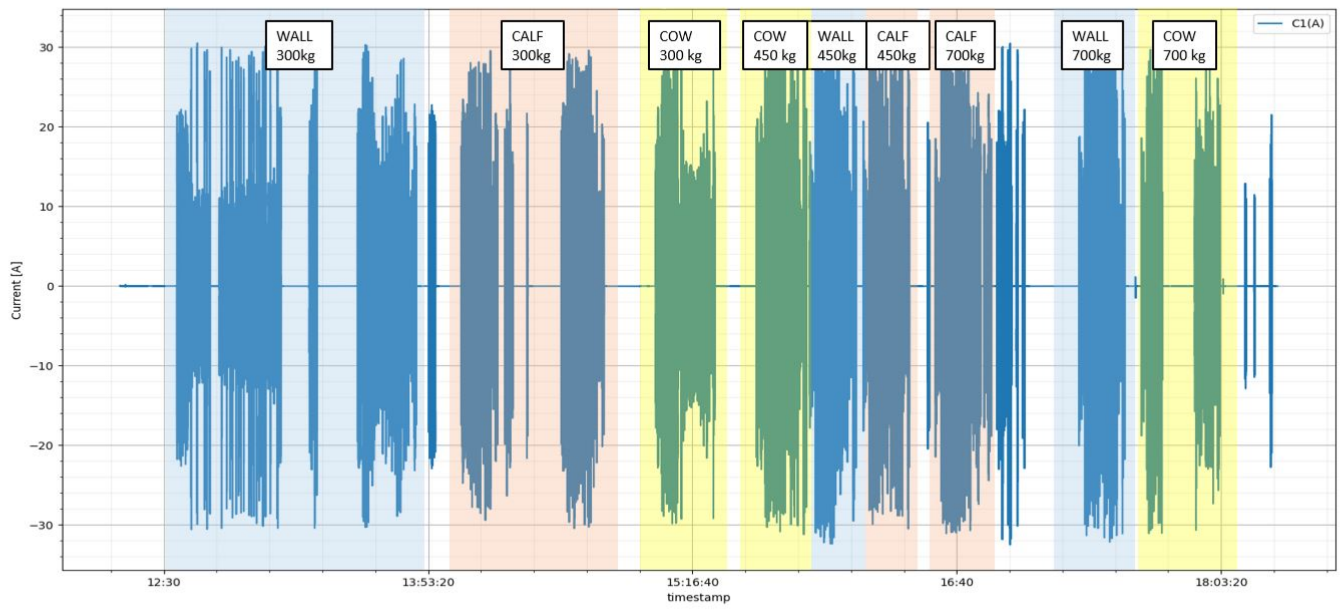


Fig. 40: Overview of experiment B time series data, phase current C1, robot weight and type of collision

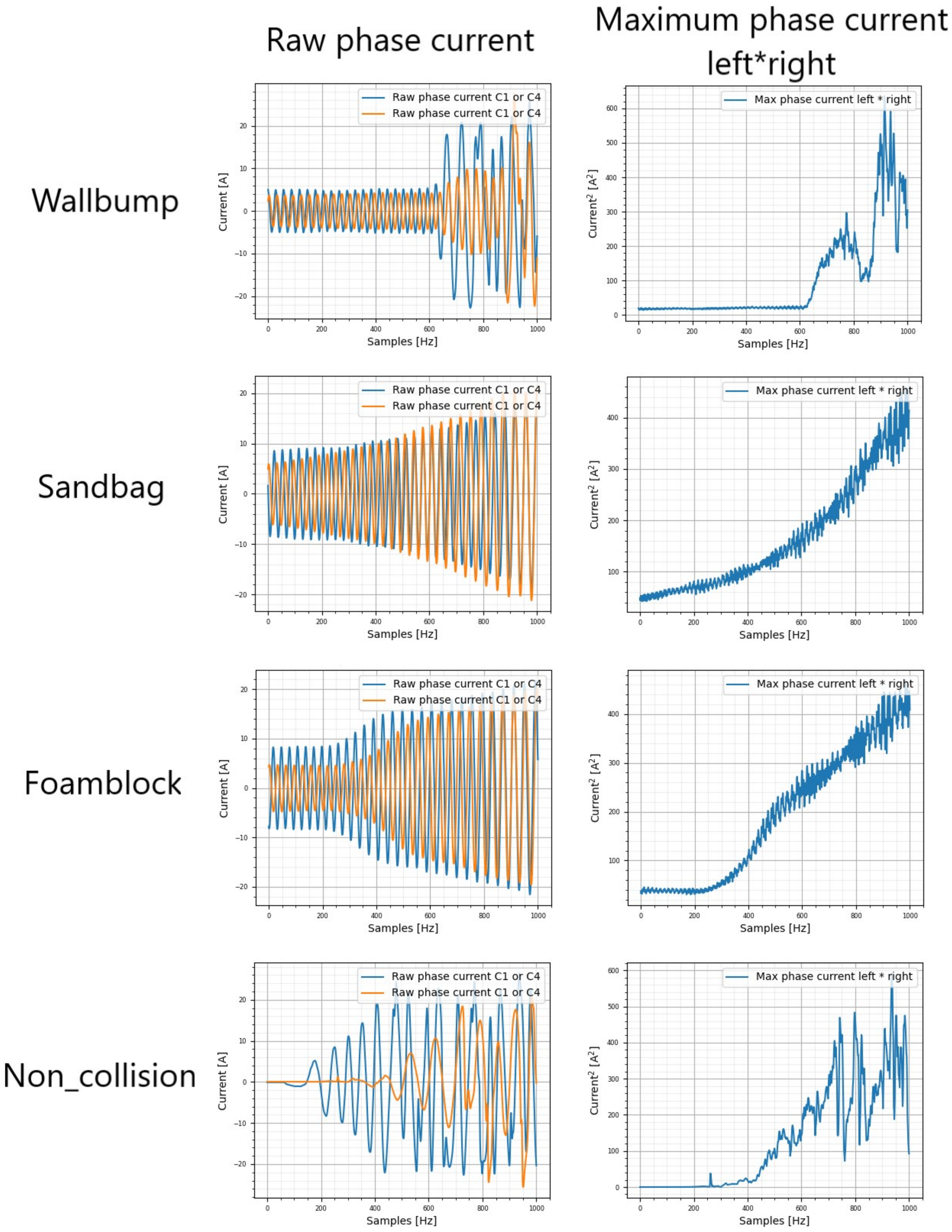


Fig. 41: Phase motor current signals from experiment B per dependent varable

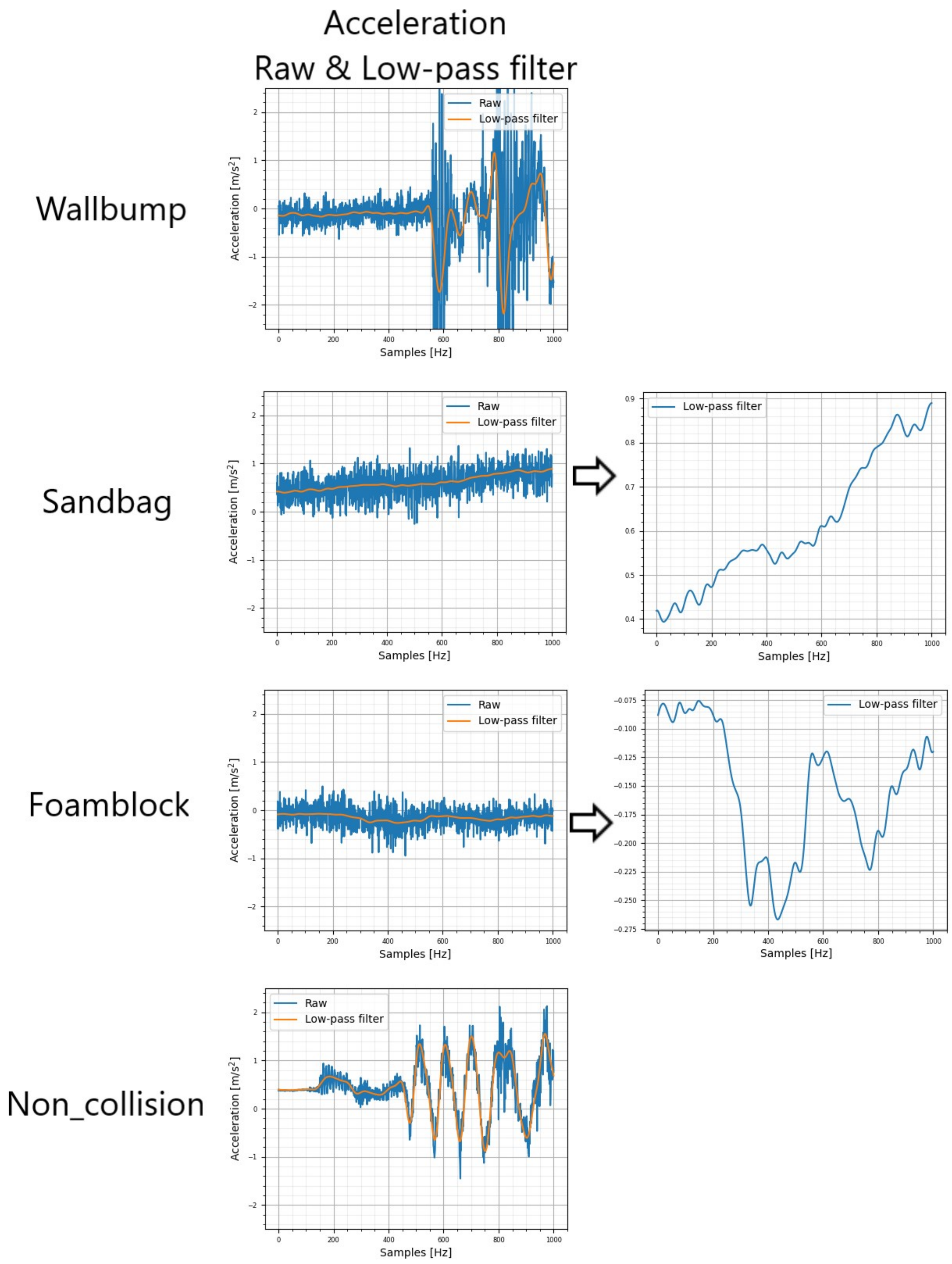


Fig. 42: Acceleration signals from experiment B per dependent varable

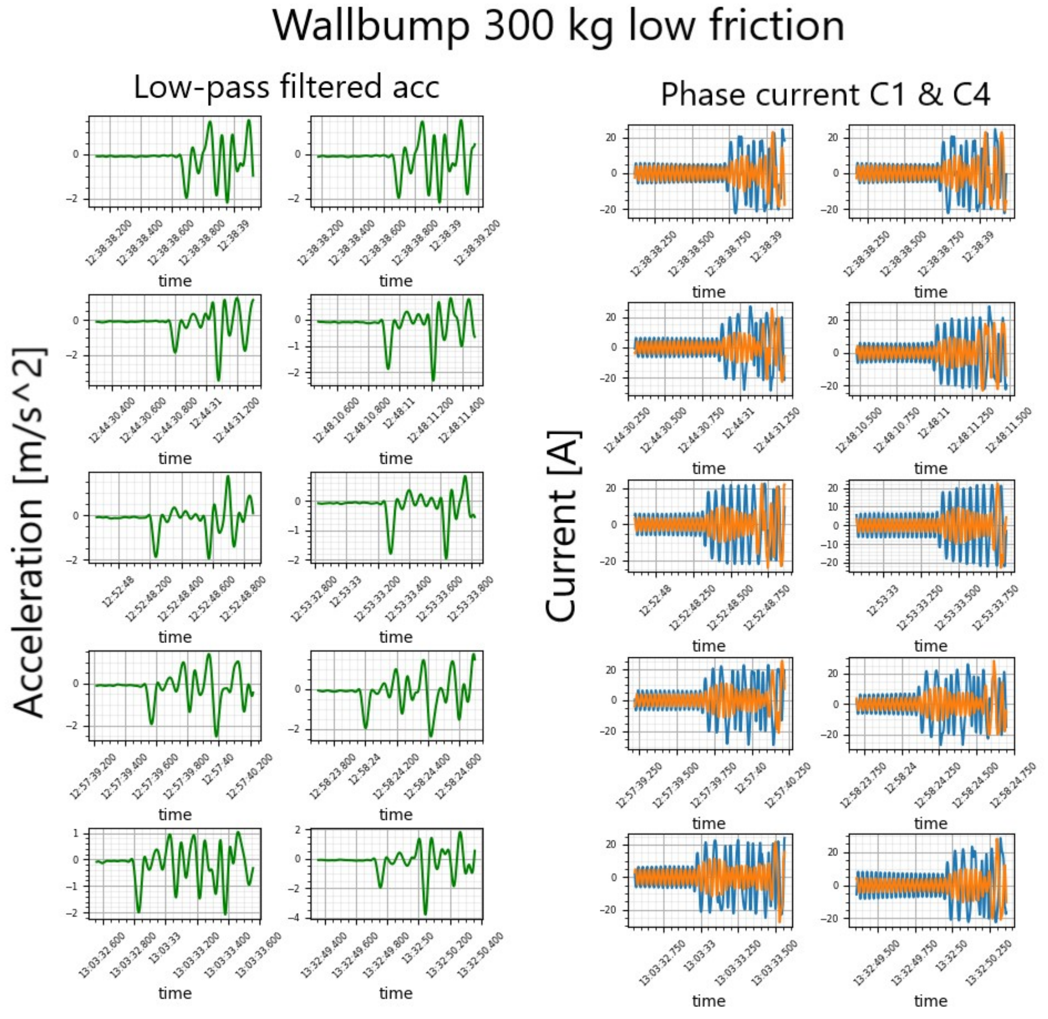


Fig. 43: Acceleration and phase current signals for experiment B: wallbump with a robot weight of 300 kg and low floor friction

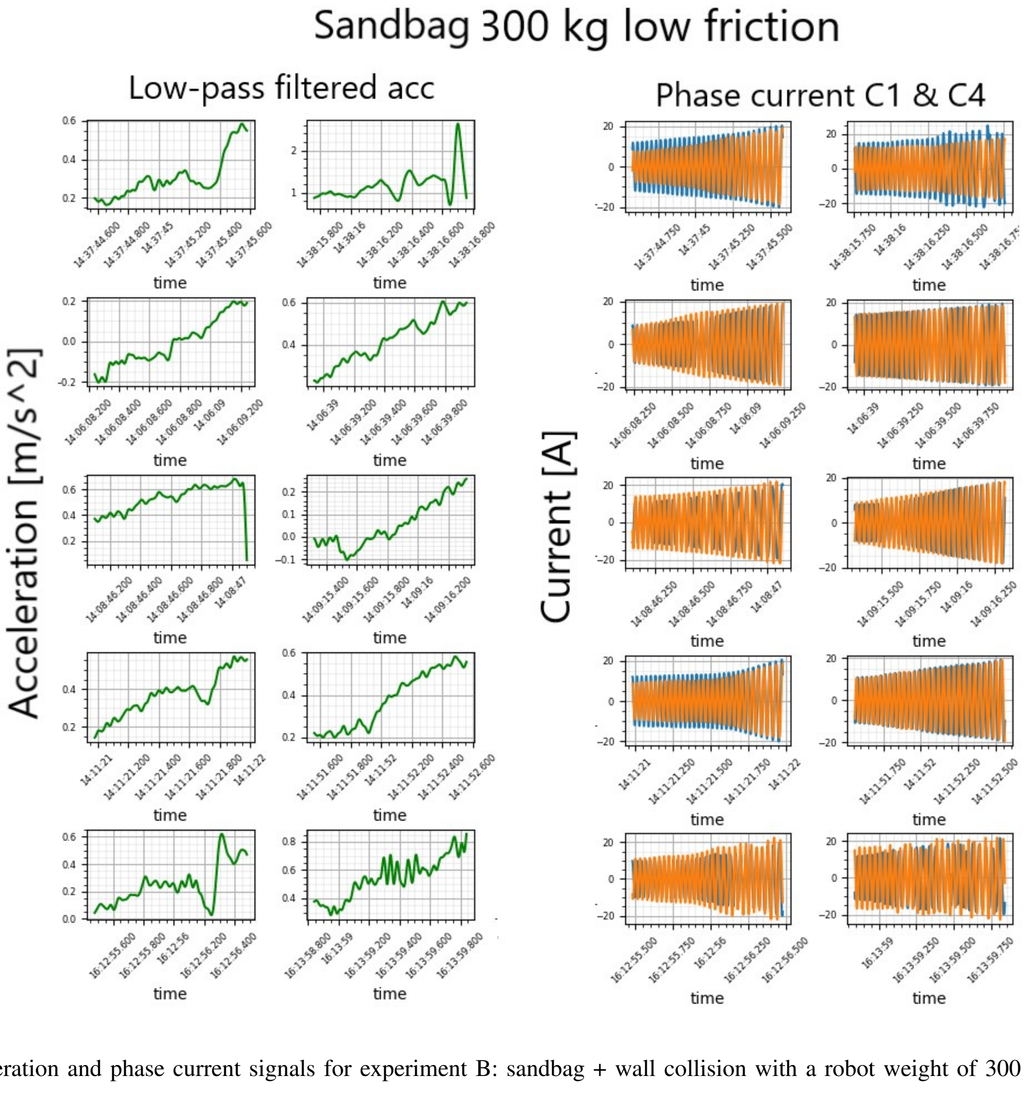


Fig. 44: Acceleration and phase current signals for experiment B: sandbag + wall collision with a robot weight of 300 kg and low floor friction

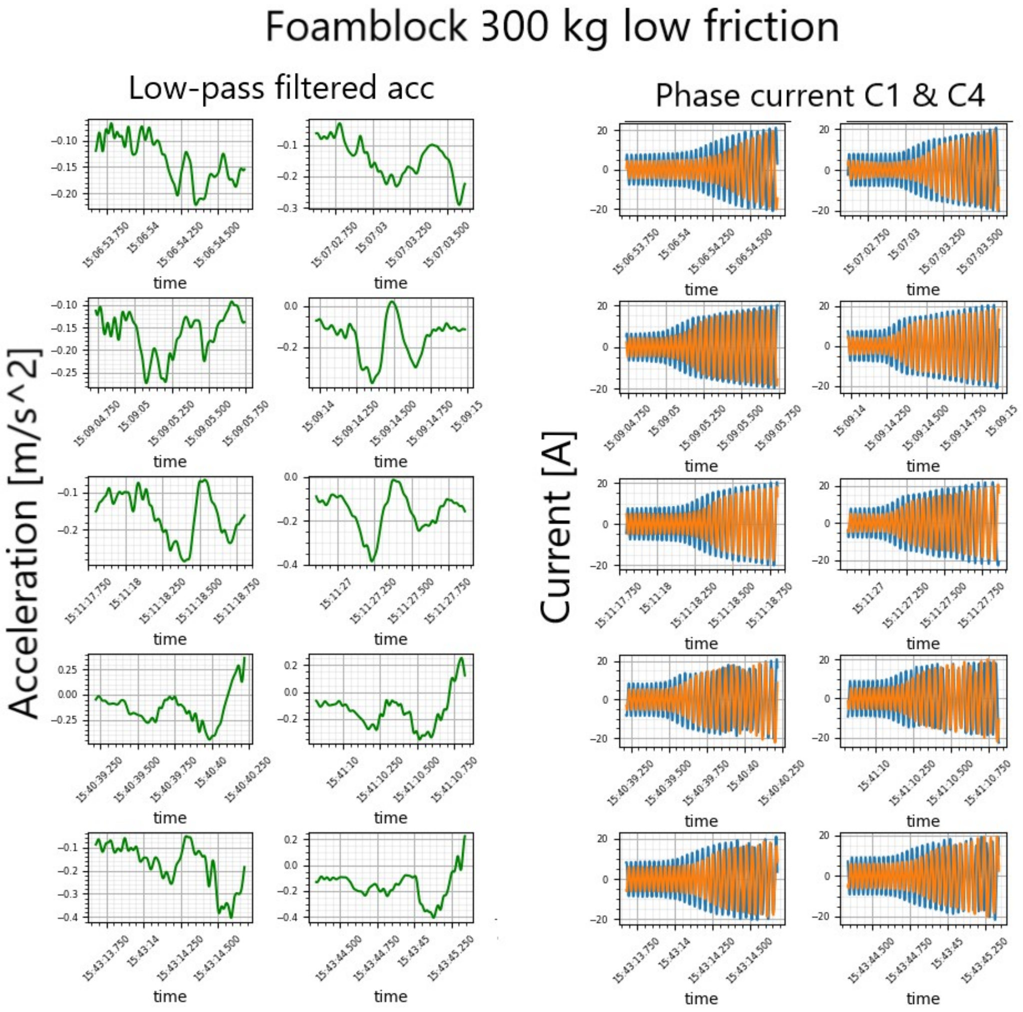


Fig. 45: Acceleration and phase current signals for experiment B: foamblock + wall collision with a robot weight of 300 kg and low floor friction

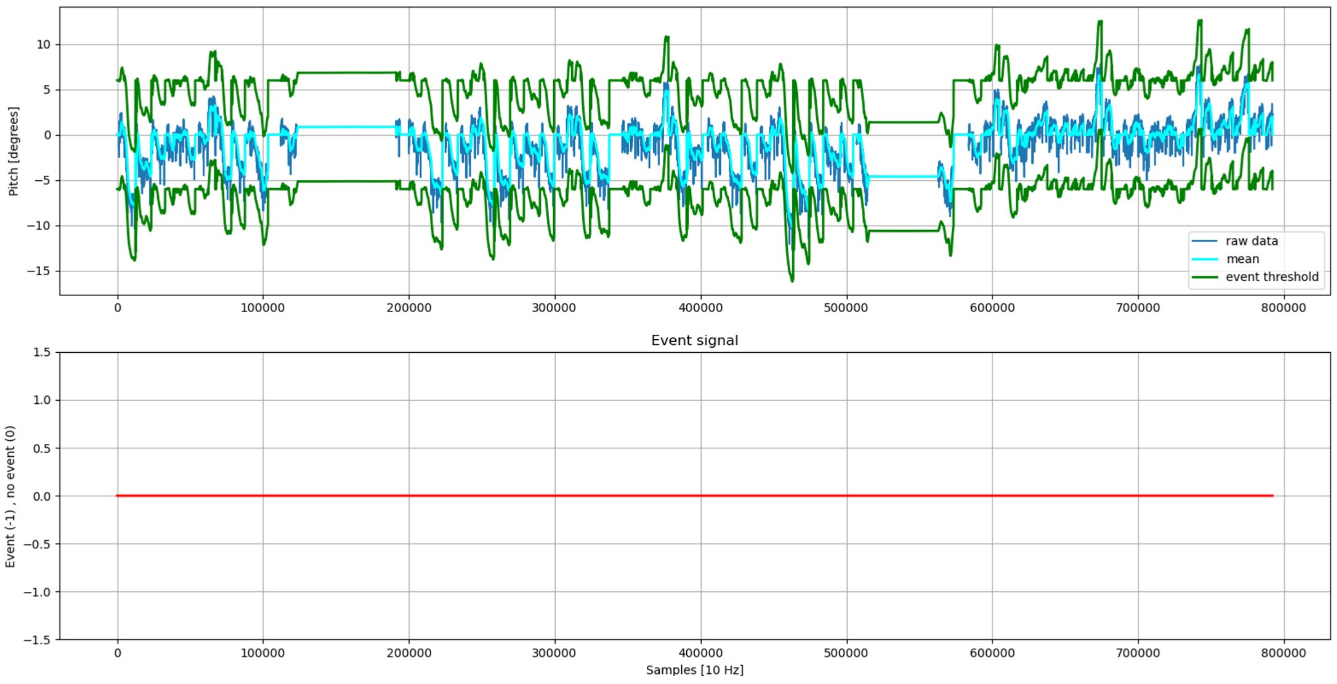


Fig. 46: Tilt detection algorithm on 24h of uneventful data in an uncontrolled environment (top) and corresponding event signal (bottom)

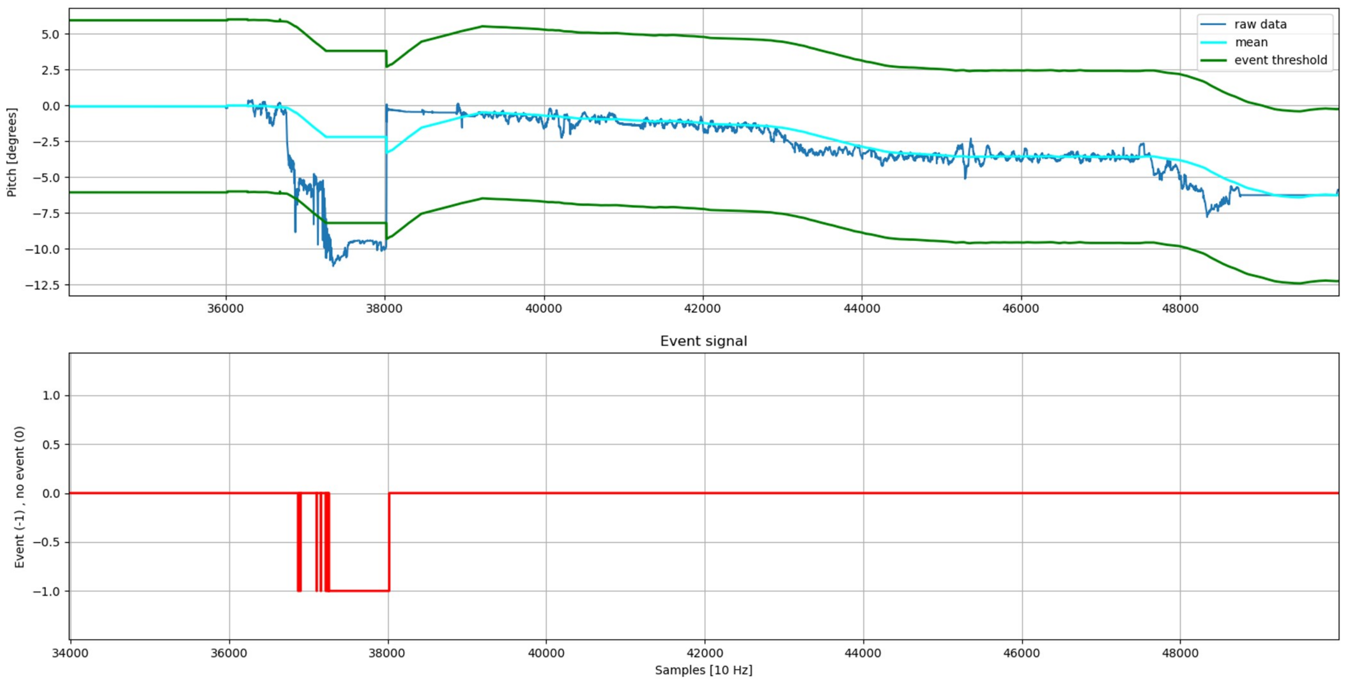


Fig. 47: Tilt detection algorithm on eventful data in an uncontrolled environment (top) and corresponding event signal (bottom)

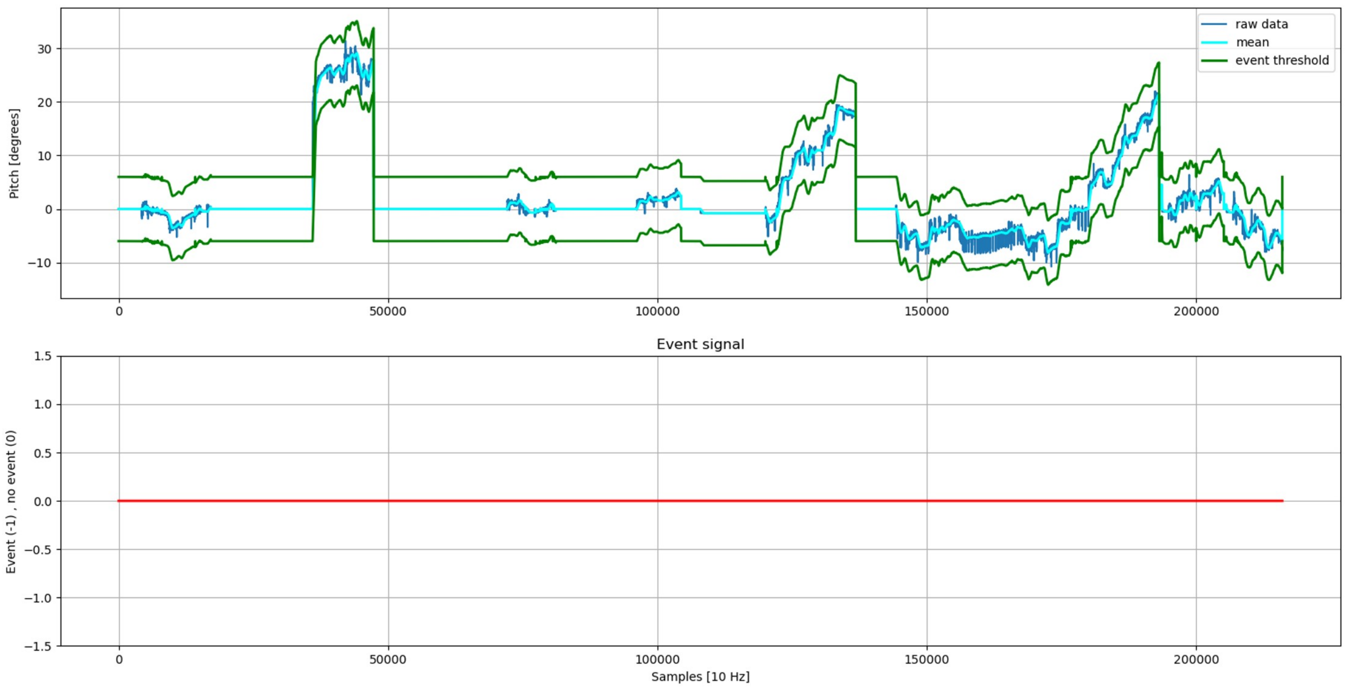


Fig. 48: Tilt detection algorithm on uneventful data of 7 different robots in an uncontrolled environment (top) and corresponding event signal (bottom)

F. Appendix - Python files

Python files have been shortened by removing less relevant and comparable codes for the readability.

F-1 Type I collision detection - sequences, label for training

```python
# -*- coding: utf-8 -*-
"""
@author: Emma Kooi
"""
text = ['non_of_the_above','wheelblock']
nr = [0,1]


# Label and series_id for events.
series_id = np.zeros(len(datafile))
series_id = series_id.astype(int)
datafile['series_id'] = series_id # add column of zeros to datafile
datafile['label'] = series_id # add column of zeros to datafile
series_id = datafile['series_id']
label = datafile['label']

window = 0.1
for i in starth:
    for m in range(15):
        h = j + datetime.timedelta(0,window)
        k = h.time()
        l = j.time()
        starth2.append(l)
        endh.append(k)
        j = h

# Label the bumps
for i in range(len(starth2)):
    series_id.loc[starth2[i]:endh[i]] = i+1
    label.loc[starth2[i]:endh[i]] = 1

# Drop all zeros  for making sequences of collisions
newdata = df1[~df1['label'].isin([0])]

loc01 = pd.to_datetime('12:16:00')
loc02 = pd.to_datetime('12:19:30')
newdata2 = df.loc[loc01.time():loc02.time()]
jkl = newdata2[FEATURE_COLUMNS]
jkl = np.array_split(jkl, 1000)  # for non_collision sequences

# Make sequences of datalist
sequences = []

for series_id, group in newdata.groupby("series_id"):
    sequence_features = group[['C2(A)']]
    sequences.append(sequence_features)

for series_id, group in newdata.groupby("series_id"):
    sequence_features2 = group[['C2(A)']]
    sequences.append(sequence_features2)

for series_id, group in newdata.groupby("series_id"):
    sequence_features3 = group[['C3(A)']]
    sequences.append(sequence_features3)

window = 0.1
# make all samples equal length
for i in range(len(sequences)):
    if len(sequences[i]) > int(window * 1000) :
        sequences[i].drop(sequences[i].tail(1).index,inplace=True)

for i in range(len(jkl)):
    if len(jkl[i]) > int(window * 1000):
        jkl[i].drop(jkl[i].head(len(jkl[i])- int(window*1000)).index,inplace=True)

# Make one list of labels
ll = np.ones((3*len(starth2)))
sequences0 = list(zip(jkl, np.zeros(len(jkl)).astype((int))))
sequences = list(zip(sequences, ll))
sequences = sequences + sequences0
```

F-2 *Split and vectorize sequences for training*

```
# split into training and test sequences
shuffle(sequences)

train_sequences, test_sequences = train_test_split(sequences, test_size=0.2)

X_train, X_test = [row[0] for row in train_sequences] , [row[0] for row in test_sequences]
y_train, y_test = [row[1] for row in train_sequences] , [row[1] for row in test_sequences]
xx_train = np.array(X_train)
xx_test = np.array(X_test)

# classififers require shape to be a vector
nsamples = len(train_sequences)
nx = int(window*1000) # length sequences
ny = len(features) # lengt features
train_dataset = xx_train
nsamples, nx, ny = train_dataset.shape
xx_train = train_dataset.reshape((nsamples,nx*ny))

nsamples = len(test_sequences)
nx = int(window*1000) # length sequences
ny = len(features) # lengt features
test_dataset = xx_test
nsamples, nx, ny = test_dataset.shape
xx_test = test_dataset.reshape((nsamples,nx*ny))
```

F-3 Type I collision detection - motor current

```
# -*- coding: utf-8 -*-
"""
@author: emmak
"""

F = 1000 # Hz
timestep = 1/F
features = ['C1(A)']


def rollingclf(starttime, endtime, classifier, dataset):
    start1 = pd.to_datetime(starttime)
    end1 = pd.to_datetime(endtime)
    window = 0.1 # 100 milliseconds
    timestepT = 0.02
    nx = int(window*1000) # length sequences
    ny = len(features) # length features
    currenttime = time.time()
    len_df = len(dataset.loc[start1.time():end1.time()])
    len_loop = int(len_df/timestepT/1000)
    print("length of dataframe (seconds) = ", len_df/1000)
    print("# of windows = ", len_loop)
    print('... Waiting for loop finish...')
    oldtime = start1

    for i in range(len_loop - 100 ): # whole dataset: range(len_df/window)
        Rolling = oldtime + datetime.timedelta(0,window)
        Rolling2 = Rolling + datetime.timedelta(0,2) # when there are missing values, window extra big
        newtime = oldtime # Update newtime for rolling window
        dfrol = dataset[features].loc[oldtime.time(): Rolling2.time() ]
        if len(dfrol) > int(window*1000): # adjust length to window
            dfrol.drop(dfrol.tail(len(dfrol)- int(window*1000)).index,inplace=True)
        X  = np.array(dfrol)
        nx, ny = X.shape
        X = X.reshape((1,nx*ny))

        """machine learning tool"""
        Ypred = classifier.predict(X)
        proba = classifier.predict_proba(X)

        """ set cutoff prediction probability:"""
        cutoffpred = 0.85
        if Ypred ==1 and proba[0][1] > cutoffpred:

            """ Start second wheel """
            dfrol2 = dataset['C4(A)'].loc[oldtime.time(): Rolling2.time() ]
            if len(dfrol2) > int(window*1000): # adjust length to window
                dfrol2.drop(dfrol2.tail(len(dfrol2)- int(window*1000)).index,inplace=True)
            X2  = np.array(dfrol2)
            X2 = X2.reshape((1,nx*ny))
            Ypred2 = classifier.predict(X2)
            proba2 = classifier.predict_proba(X2)
            if Ypred2 ==1 and proba2[0][1] > cutoffpred:
                bumplist.append(Rolling.time())
                print (Rolling.time(), "double bump", proba[0][1])

                """Implement classification algorithm here for real time application"""

        oldtime = oldtime + datetime.timedelta(0,timestepT) # 10 ms timesteps
        if i\%5000 == 0: # shows time passed for every 1.5 minute in data
            print ((time.time()- currenttime)/60, "minutes passed, currently at", Rolling.time() )

        """ Set break time (minutes)"""
        breaktime = 150# minutes
        if (time.time()- currenttime) > (60*breaktime): # 60 * n-max minutes loop
            print ("MAX SET TIME PASSED, stopped at", Rolling.time() )
            break
    print ("Loop = done")
    elapsedtime = (time.time()- currenttime)/60

    print( "Elapsed time (minutes): ", elapsedtime)
    return bumplist
```

```
""" Set wallbump function parameters """
bumplist = []

# Open Random Forest Algorithm
import pickle
with open("...",
          "rb") as fp:
    b = pickle.load(fp)

starttime = '12:15:00'
endtime = '12:15:10'
start1 = pd.to_datetime(starttime)
end1 = pd.to_datetime(endtime)
dfcut = df.loc[start1.time():end1.time()]
dataset = dfcut
classifier = b
bumplist = rollingclf(starttime, endtime, classifier, dataset)

def finalbumplist(whichbumplist):
    from datetime import datetime, date
    count = 0
    bumplistfinal = []
    bumplistfinal.append(whichbumplist[0])
    for i in range(1,len(whichbumplist)-1): # 1: i  for real time implementation i-1
        dt = datetime.combine(datetime(1,1,1,0,0,0), whichbumplist[i]) - datetime.combine(datetime(1,1,1,
                                                                        0,0,0), whichbumplist[i-1])
        dt = dt.total_seconds()
        if abs(dt) > 0.5: # 0.5 seconds in between
            count = 0
        elif abs(dt) < 0.20:
            count = count + 1
            if count ==1: # A minimum of two consequtive bumps required
                bumplistfinal.append(whichbumplist[i])
                print(whichbumplist[i])
            if count > 500: # 500 * (bumplist timestep = 0.02) = 10 seconds
                print("WARNING: stuck for at least 10 seconds = foam / obstruction")
    return bumplistfinal

bumplistX = finalbumplist(bumplist)
```

F-4 Type II collision detection - pitch

```
# -*- coding: utf-8 -*-
"""
@author: Emma Kooi
"""

def thresholding(y, lag, threshold):
    ignorePeak = 1
    calcY = np.array(y)
    avgY = [0]*len(y)
    avgY[lag - 1] = np.mean(y[0:lag]) # initialize variables
    for i in range(lag, len(y) - 1):
        """first if statement eliminates errors due to calibration at charger. Remove this when used real
                                                                        -time. Change next elif to if."""
        if abs(y[i]) < 0.05 and abs(abs(y[i-1]) - abs(y[i])) < 0.005:
            calcY[i] = 0
            avgY[i] = 0
        elif abs(y[i] - avgY[i-1]) > threshold: # Update signals = 0 here for real time application
            calcY[i] = y[i]
            avgY[i] = np.mean(calcY[(i-lag-ignorePeak):i-ignorePeak])
            calcY[i] = avgY[i]
            ignorePeak = ignorePeak + 1
        else: # Update signals = -1 here for real time application
            calcY[i] = y[i]
            avgY[i] = np.mean(calcY[(i-lag):i])
            ignorePeak = 1
    return avgY, calcY


def updateSignals(y, avgY):
    signals = np.zeros(len(y))
    for i in range(lag, len(y) - 1):
        if abs(y[i]) < 0.05:
            signals[i] = 0
        elif abs(y[i] - avgY[i-1]) > threshold:
            if y[i] > avgY[i-1]:
                signals[i] = 0
            else:
                signals[i] = -1
        else:
            signals[i] = 0
    return signals

dfHz = 1000
lag = 60 * dfHz # 60 seconds
threshold = 6 #degrees

# Run algo with settings from above
avgY, calcY = thresholding(y, lag=lag, threshold=threshold)
signals = updateSignals(y, avgY)

avgY = np.asarray(avgY)
signals  = np.asarray(signals)
```

F-5 *Blind object classification - sequences, label for training*

```
# -*- coding: utf-8 -*-
"""
@author: Emma Kooi
"""

theta = 0.136
df['Z_correct'] =  df.Z_ACCL_OUT * np.cos(theta) - df.X_ACCL_OUT * np.sin(theta)
df['X_correct'] =  df.Z_ACCL_OUT * np.sin(theta) + df.X_ACCL_OUT * np.cos(theta)

dffoam['Z_correct'] =  dffoam.Z_ACCL_OUT * np.cos(theta) - dffoam.X_ACCL_OUT * np.sin(theta)
dffoam['X_correct'] =  dffoam.Z_ACCL_OUT * np.sin(theta) + dffoam.X_ACCL_OUT * np.cos(theta)

# make sequences
def fingerprint(namebumplist):
    from datetime import datetime, date
    start = []
    window = pd.to_datetime('00:00:01,00').time()
    for i in namebumplist:
        j =  datetime.combine(datetime(1,1,1,0,0,0), i) - datetime.combine(datetime(1,1,1,0,0,0), window)
        l = pd.to_datetime(str(j)).time()
        start.append(l)
    return start

startwall = fingerprint(wallbumplist)
startsand = fingerprint(sandbumplist)
startfoam = fingerprint(foambumplist)
startnon_collisions = fingerprint(non_collisions)

# Feature selection motor current
datafile['C1max'] = abs(datafile[['C1(A)','C2(A)', 'C3(A)']]).values.max(axis=1)
datafile['C4max'] = abs(datafile[['C4(A)','C5(A)', 'C6(A)']]).values.max(axis=1)
datafile['C1C4max'] = datafile['C1max'].values * datafile['C4max'].values

# low or high pass filter on acceleration
fhp,flp     = 1 ,                 50               # Cut-off frequencies resp. high pass & low pass
                                                    filter
whp, wlp    = fhp / (fs / 2),     flp / (fs / 2)   # Normalize the frequencies
b, a        = signal.butter(4, wlp, btype='low', analog = False) # btype = high/low/band band: [whp,wlp]

featuresA = ['Z_ACCL_OUT']
newfeaturesA = ['LPfilter_ZA']
for i in range(len(featuresA)):
    outputA = signal.filtfilt(b, a, dataset[featuresA[i]])
    dataset[newfeaturesA[i]] = outputA

# make and label FFT sequences

#define timestep
F = 1000 # Hz
timestep = 1/F #* 1000
```

```
l1 = np.ones(len(wallbumplist))
l2 = np.ones(len(sandbumplist))*2
l5 = np.ones(len(foambumplist))*3
l3 = np.ones(len(non_collisions))*4
l4 = np.ones(len(foambumplist2))* 3

l1FFT = np.concatenate((l1,l2,l5,l4,l3,l6)).astype(int)

import datetime

sequencesFFT = []

for i in range(len(wallbumplist)):
    dfFFT = df['HPZ_correct'].loc[startwall[i]:wallbumplist[i]]
    n1        = len(dfFFT)
    n12       = int(n1 / 2 )
    ps1       = np.abs(np.fft.fft(dfFFT.values))**2
    ps1 =       ps1/ps1.max()
    sequencesFFT.append((ps1))

for i in range(len(sequencesFFT)):
    if len(sequencesFFT[i]) > int(window * 1000) :
        sequencesFFT[i] = sequencesFFT[i][:-1]

freqs1    = np.fft.fftfreq(n1,timestep)
sequencesFFT = list(zip(sequencesFFT, l1FFT))


# LABELLING
text = ['wallbump','sand', 'foam', 'none']
nr = [1,2,3,4]

# Label and series_id for events.
series_id = np.zeros(len(datafile))
series_id = series_id.astype(int)
datafile['series_id'] = series_id # add column of zeros to datafile
datafile['label'] = series_id # add column of zeros to datafile
series_id = datafile['series_id']
label = datafile['label']

# Label the wallbumps
for i in range(len(wallbumplist)):
    series_id.loc[startwall[i]:wallbumplist[i]] = i+1
    label.loc[startwall[i]:wallbumplist[i]] = 1

# Label the sandbumps
for i in range(len(sandbumplist)):
    series_id.loc[startsand[i]:sandbumplist[i]] = len(wallbumplist) + i+1
    label.loc[startsand[i]:sandbumplist[i]] = 2

# Label the foambumps
for i in range(len(foambumplist)):
    series_id.loc[startfoam[i]:foambumplist[i]] = len(wallbumplist) + len(sandbumplist) + i+1
    label.loc[startfoam[i]:foambumplist[i]] = 3

# Label the non-collisions
for i in range(len(non_collisions)):
    series_id.loc[startnon_collisions[i]:non_collisions[i]] = len(wallbumplist) + len(sandbumplist) + len
                                                         (foambumplist) + i+1
    label.loc[startnon_collisions[i]:non_collisions[i]] = 4

datafile['series_id'] = series_id
datafile['label'] = label

FEATURE_COLUMNS = ['C1C4max', 'LPfilter_ZA']
features = FEATURE_COLUMNS

datafile = df
newdata1 = datafile[~datafile['label'].isin([0])]

s1 = []
label1 = []

for series_id, group in newdata1.groupby("series_id"):
    sequence_features = group[features]
    s1.append(sequence_features)
    label1.append(group['label'][1])

window = 1 #[seconds]
remove_index = []
setS = [s1]#,s4]
for j in setS:
    for i in range(len(j)):
        if len(j[i]) > int(window * 1000) :
            j[i].drop(j[i].tail(len(j[i])- int(window*1000)).index,inplace=True)
        if len(j[i]) < int(window * 1000):
            # print(setS[i])
            print('index number', i)
            remove_index.append(i)

seq = [s1,label1]
for j in seq:
    j.pop(remove_index[-1])

seq = [s1]
s = []
for i in seq:
    sequences1 = list(zip(i, label1))
    s = s + sequences1
sequences = s

# Check for shape
for i in range(len(sequences)):
    print(sequences[i][0].shape, i)
```

F-6 Blind collision classification - Predict class

```
# -*- coding: utf-8 -*-
"""
@author: Emma Kooi
"""

F = 1000 # Hz
timestep = 1/F
features = ['C1C4max','LPfilter_ZA','signals']

with open("...",
          "rb") as fp:
    classifier = pickle.load(fp)

def classification_algo(classifier,dataset,startbumplist,bumplist):
    window = 1 # seconds
    nsamples = 1 #real time
    nx = int(window*1000) # length sequences
    ny = len(features) # length features
    currenttime = time.time()

    for i in range(len(bumplist)): # whole dataset: range(len_df/window)
        dfrol = dataset[features].loc[startbumplist[i]: bumplist[i]]
        if len(dfrol) > int(window*1000): # adjust length to window
            dfrol.drop(dfrol.tail(len(dfrol)- int(window*1000)).index,inplace=True)
        if len(dfrol) < int(window * 1000):
            print('too few datapoints')
            continue
        X  = np.array(dfrol)
        nx, ny = X.shape
        X = X.reshape((1,nx*ny))

        """machine learning tool"""
        Ypred = classifier.predict(X)
        proba = classifier.predict_proba(X)
        print(bumplist[i], 'Ypredicted', Ypred, proba)

    print ("Loop = done")
    elapsedtime = (time.time()- currenttime)/60
    print( "Elapsed time (minutes): ", elapsedtime)


classification_algo(classifier,dataset,startWP,bumplistWP)
```