Determination of key design parameters of a scooter-walker hybrid in scooter mode

by

Lucas Giesen

In partial fulfilment of the requirements for the degree of
Master of Science
at Delft University of Technology,
to be defended publicly on 10/07/2023

Faculty: 3mE
Department: Biomechanical Engineering
Programme: Biomechanical Design

Mentors / Supervisors: Heike Vallery
Robert Riener
Adrian Esser
Graduation committee: Heike Vallery
Gerard Verbiest
Adrian Esser

TU Delft

25-10-2017 KM

Keywords:
mobility
mobility scooter
walker
design parameters
optimization

Abstract
A new type of mobility device that functions as both a walker and a scooter is being developed to address some issues of current mobility solutions. Some key design parameters for the scooter mode of this scooter-walker hybrid must be optimized. Through literature, a number of simulated "tests", mostly about safety and manoeuvrability, are developed, and 2100 different possible configurations for the device are generated, with each configuration being rated on the performance for each test using the Analytic Hierarchy Process. Some design parameters tend to affect the test outcomes more, particularly the width of the front of the device and the length. In general, larger dimensions lead to a higher stability, but a lower manoeuvrability. A set of optimized key design parameters is identified, but the methodological framework developed is a useful tool by itself as well.

Determination of key design parameters of a scooter-walker hybrid in scooter mode

Lucas Giesen

***Abstract*—A new type of mobility device that functions as both a walker and a scooter is being developed to address some issues of current mobility solutions. Some key design parameters for the scooter mode of this scooter-walker hybrid must be optimized. Through literature, a number of simulated "tests", mostly about safety and manoeuvrability, are developed, and 2100 different possible configurations for the device are generated, with each configuration being rated on the performance for each test using the Analytic Hierarchy Process. Some design parameters tend to affect the test outcomes more, particularly the width of the front of the device and the length. In general, larger dimensions lead to a higher stability, but a lower manoeuvrability. A set of optimized key design parameters is identified, but the methodological framework developed is a useful tool by itself as well.**

I. Introduction

Mobility is a crucial aspect of human life. Being mobile grants people access to services and shops, but also gives them a feeling independence, control, and inclusion [1]. However, in the Netherlands, around 7.7% of the population suffers from one or more mobility impairments, and this figure increases to 18.3% for men and 36.5% for women aged 75+ [2]. Impaired mobility can result from various medical conditions, including but not limited to arthritis, multiple sclerosis, stroke, and Parkinson's disease [3]. For older adults, the onset of chronic conditions like arthritis and and lung problems are the most common reasons for a reduction in mobility [4].

Mobility aids such as walkers and wheelchairs have helped in improving the quality of life for many people with mobility impairments. However, traditional mobility aids have their limitations, particularly when it comes to users' energy levels. Powered mobility scooters, for example, allow their users to travel longer distances outside without needing to walk, increasing the quality of life and mental well-being of their users [5]. However, they also have their limitations, such as being difficult to manoeuvre indoors [6] [7] and possibly having an averse effect on physical health [5]. On the other hand, devices such as walkers offer a more compact footprint and more manoeuvrability, but only let the users go as far their energy levels allow them to walk. Users of mobility scooters have also been found to frequently use mobility devices like walkers inside [8].

To address some of these limitations, the author came up with the idea of a hybrid device that combines the features of a scooter and a walker. Currently, a team of students working at the Sensory Motor Systems lab at the ETH Zürich, including the author, is working on the concept. A prototype has already been constructed by student Olivier Nüssli (Figure 1). This scooter-walker hybrid allows users to switch between a walker mode and scooter mode. In walker mode, the device serves as a standard walker. However, when users wish to travel a longer distance, or when their energy levels demand it, they can switch to scooter mode, allowing them to rest and continue their journey. This hybrid device has the potential to improve the mobility and independence of people with mobility impairments significantly.

Lucas Giesen l.a.m.giesen@student.tudelft.nl

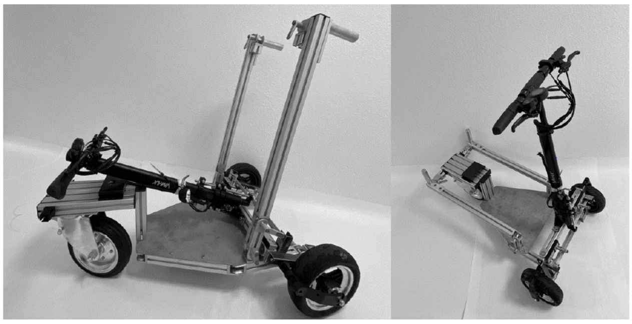


Fig. 1: Prototype of the scooter-walker hybrid built by ETH Zürich student Olivier Nüssli. The seat, which would be over the single rear wheel, is not included in the photo (image used with permission). [9]

Though similar hybrid walker devices already exist [10] [11], they differ in a key way, as they are electric wheelchair-walker hybrids, instead of mobility scooter-walker hybrids. Electric wheelchairs are made to be used at walking speed, since they have small caster wheels in the front that have more trouble handling bumps, have thinner wheels that are more prone to slipping, and lack suspension. Contrast this to mobility scooters, which have sturdy frames and larger, wider, suspended wheels to safely handle speeds higher than walking speed, especially on uneven road surfaces and small obstacles that can be encountered in urban environments. Thus our concept allows users to have a larger effective travel range compared to similar products currently offered.

However, the prototype in Figure 1 has some shortcomings that need to be addressed. First, switching between the modes requires users to get off the scooter and bend down to manipulate the handlebars, which would be difficult for many people in the target audience to achieve. Second, at 80 cm overall width, the scooter is quite large in walker mode, making it difficult to manoeuvre in tight spaces. The third main drawback is the fact that the device takes quite some effort to move in walker mode, due to its high inertia compared to a normal walker. Thus, it has been decided that the design will be reevaluated, posing an opportunity for more research and optimization to be done on it.

One important consideration is the number of wheels and their layout in scooter mode. Currently, mobility scooters come with three and four-wheeled configurations. The choice of the number of wheels depends on various factors, including stability, maneuverability, and user preference. Three-wheel configurations are typically more manoeuvrable, but four-wheel configurations are generally more stable.

For four-wheeled configurations, multiple options exist as well. The wheelbase could be rectangular, but an "in-between" solution with a trapezoid wheelbase, so with the front wheels being closer to each other than the rear wheels, is also possible.

Additionally, the hybrid nature of this concept yields additional challenges with regard to the number of wheels and their configuration, as the wheel layout may enable or constrain certain transformation mechanisms. Therefore it is important that the characteristics and consequences of the different types of wheelbases and the choice of other design parameters are understood well.

At the time of writing, one of the other students working on the project is developing the control system driving the wheels, while the other is revising the entire structure of the design. Therefore the author must be careful to make a contribution that is based on appropriate assumptions, while not constraining the design possibilities of the device as a whole too much.

An opportunity thus presents itself in optimizing several key design parameters. For scooter mode, it would have to be understood how these parameters interact with, for example, the stability of the design. In walker mode, the stability would not be an issue, as the control system that is in development ensures that speeds are kept low enough for the walker to operate safely. The main focus of this contribution will thus be on the scooter mode of the device.

Therefore, the research question of this thesis is "What are the optimal key design parameters for a scooter-walker hybrid in scooter mode?". To answer this question, literature will be explored and expanded upon to identify the what factors must be taken into account in answering this question. From that, a number of virtual "tests" will be derived, with each of them testing how well a possible configuration of the device would perform with regard to a requirement. These tests then define the scope of a general model with certain design parameters. Some of these design parameters will be optimized by generating a large number of possible configurations for the device, with each having a unique set of optimizable design parameters. Using a weighted scoring method, each configuration will be rated, from which a configuration with the most promising set of design parameters will emerge.

The results of these analyses can be used for further development of the scooter-walker hybrid. Moreover, the methodological framework as laid out in this research can be used as a tool in future work as well.

II. Methods

A. Legal background

It must be assured that the proposed design concept is actually allowed to be used in public, making it vital to know if the choice of some design parameters is affected by the law. Therefore, relevant legislature about mobility scooters has to be studied. Since this project is in collaboration between the TU Delft and the ETH Zurich, the legislature studied is limited to the Netherlands, Switzerland and Germany.

	Switzerland	Netherlands	Germany
Number of wheels	n.c.*	n.c.	n.c.
Max. speed (km/h)	30 [12]	45 [13]	15 [14]
Max. weight (kg)	n.c.	n.c.	300 [14]
Min. wheel diameter (m)	n.c.	n.c.	n.c.
Max. length (m)	n.c.	3.5 [15]	n.c.
Max. width (m)	1.0 [12]	1.1 [15]	1.1 [14]

TABLE I: Overview of relevant legal information in target countries.
*n.c. = not codified into law

An overview can be seen in Table I. When a law is not codified, it means that all relevant sources have been studied, but no information was found. A full overview of all legal information studied can be found in Appendix A-A.

B. Design scope

Since nothing about the inner workings or actual construction of the design is certain yet, that means that, for the purpose of this research, in scooter mode the scooter-walker hybrid can only be regarded as a "black box" with wheels and a place where someone sits down. Even the chair itself would be within this black box, as several seating options are still being considered. Assuming that the device becomes more compact in walker mode, this "black box" should shorten in its transformation to walker mode.

Everything that clearly exists outside of this model will be considered outside of the scope of this research. For example, if there is literature that examines the brake handle ergonomics of E-scooter, it would be deemed outside of the scope, as that is a detail existing within the "black box" that is not relevant for this research.

Furthermore, anything related to pre-existing design requirements will also be considered outside of the scope of this research, as those will not be changed anymore. The relevant design requirements are listed below.

R.1 The device must be able to function as both a walker and a scooter.
R.2 It can be assumed that the device will have a transformation between both modes that reduces the devices length in walker mode.
R.3 The device must be safe.
R.4 The device must weigh no more than 35 kg.

R.5 For the walker mode:
R.5.1 The user must be able to lean on the scooter and push it.
R.5.2 The front wheel(s) in walker mode will be one or two caster wheels. It is not clear whether this wheel/these wheels will be used in scooter mode as well.
R.5.3 The rear wheels will not be caster wheels and will be driven in walker mode, so that the user doesn't feel the full weight when pushing it.

R.6 For scooter mode:

R.6.1 The user must be able to sit on the scooter
R.6.2 The scooter must operate like a regular Motorized Mobility Scooter (MMS).
R.6.3 The front wheels will not be driven and will be steered.
R.6.4 The rear wheels will driven and will not be steered. These are the same wheels as the powered wheels in walker mode.
R.6.5 In case of two front wheels, the distance between these wheels will not be greater than between the rear wheels.
R.6.6 In case of two front wheels, Ackermann steering geometry will be applied.
R.6.7 The maximum speed is 15 km/h.

C. Mobility scooter literature

1) Safety: When determining key design parameters for the scooter-walker hybrid in scooter mode, it must be ensured that it can operate safely. Following requirement **R.6.2**, literature about normal mobility scooters can be consulted in order to get an overview of what must be taken into account to ensure a safe design. The SCOPUS search terms used can be viewed in Appendix A-B1.

Note that MMS accidents can be (partly) caused by multiple factors, such as inappropriately designed infrastructure [16], but making recommendations for infrastructure improvement falls outside of the scope of this research. However, it would be useful to see if these dangers can be negated with appropriate design choices.

There exists one study that analyzes mobility scooter accidents and relates them to design factors in a comprehensive manner. The study conducted by Davidse e.a. [16] analyzes MMS accidents in the Netherlands in detail, and categorizes them into four different types:

1) Rider squeezes gas handle to break and hits water.
2) MMS loses balance on contact with an obstacle or irregular surface, after which the rider falls.
3) Evasion manoeuvre prevents collision but causes rider to fall from scooter.
4) Crossing rider comes into collision with crossing motor traffic.

For accident type 4, the authors did not find any cause related to E-scooter design, but they did for accident types 1, 2 and 3. A possible cause for accident type 1 could be counter intuitive brake handle design [16], which falls outside of the scope for this research. For accident type 2 and 3, however, more relevant information is given. The authors attribute the accidents partly to insufficient stability [16]. They are particularly worried about three wheeled scooters, which in the majority of the cases are used by the riders because they got them through the Social Support Act (Wet Maatschappelijke Ondersteuning / Wmo), whereas most scooter owners who bought their vehicle themselves opt for four wheels [16].

The study also noted that, for accident type 3, the falls were caused by the rider making a turn at “a relatively high speed” of >10 km/h, causing the scooter to loose balance and tip over [16]. For accident type 2, the “turn’s radius is too small”, and that “bigger and suspended wheels” may increase the stability of the vehicle [16].

Another design factor that has been identified in literature that impacts scooter safety, is stability on slopes. According to Rentschler e.a., it is important to know when a scooter tips while braking when going down hill, and when it tips while standing still on downward, upward and sideways slopes [17].

2) Manoeuvrability: The user experience must also be considered when designing the scooter mode of the scooter-walker hybrid.

As was already mentioned in the introduction, mobility scooters’ users often suffer from a lack of manoeuvrability indoors. This can especially be an issue on public transport, where users need to navigate to an area where there is room for them. Entering and positioning correctly in this area was rated as the most difficult step for users of powered mobility scooters across all steps undertaken in a ride on public transport [7]. In the US, the clear floor area (760 mm x 1220 mm) is often not large enough for users of mobility scooters [18], which has to be kept in mind when designing a scooter [19].

Something else to take into account is how much space the device takes up when performing standard manoeuvres. Koontz e.a. examined this in a paper, using four different tests as performed by participants in various personal mobility devices (Appendix A-B2).

3) Vibrations: One factor that has been identified in literature to influence user comfort, is the vehicle’s interaction with the road surface. Rough road surface can lead to less comfort for the rider. The vibration response is determined by a number of factors, namely the scooter’s velocity, size and suspension type [20].

D. Expansion on literature

1) Scooter mode: Koontz e.a. mentioned that “wheelchairs equipped with seat functions such as tilt and recline effectively lengthen the wheelchair and can impair maneuverability in tight quarters” [21]. Though it can be assumed that the scooter-walker hybrid will not be used by people who need these large recline angles, it does highlight the issue of how posture affects the measurements of the device in scooter mode. Because no records could be identified on the appropriate sitting posture for mobility scooters, it can be assumed that a regular sitting posture can be used for mobility scooters. Then, body size and the angle of the lower legs affect the ideal length of the scooter.

2) Walker mode: Even though the goal of this research is to determine the values of key design parameters for the scooter mode, its possible ramifications for the walker mode cannot be neglected. Taking the scope of this research into account, with certain assumptions the manoeuvrability of the device in walker mode can still be evaluated, as will be explained in subsubsection II-E4.

E. Defining tests

The information in the previous section will be used to develop tests that give an overview of each configuration’s performance. Because of the large number of different configurations that will be rated, it is important that all models used for the test are very computationally efficient.

On top of that, it will be proven that simpler models are sufficient to compare different design configurations within the scope. A summary of all tests can be viewed in subsubsection II-E5.

1) Safety: As was made clear in subsubsection II-C1, sideways stability is an important factor in E-scooter safety, particularly for accident types 2 and 3 as described by Davidse e.a. [16]. To cover accident type 2, it can be analyzed how much energy is required to tip over the scooter as it is making a turn (**T.1**). Using an energy equation can give an insight into the scooter's stability during a turn, while foregoing the use of computationally intensive, suspension-dependent time variant simulations. Since Davidse e.a. also mentioned wheel size as something influencing the outcome in such a scenario [16], a basic test can be introduced to analyze the interaction between an obstacle on the road and the wheel. One can calculate the angle of the tangent on the wheel as it comes into contact with an obstacle (**T.2**). For smaller wheels, this angle will be larger, increasing the upward acceleration on the scooter when it comes into contact with the obstacle.

In the automotive industry, a benchmark test for evading obstacles is the "moose test" [22], where the driver must steer left at speed to avoid an obstacle, and then steer back afterward. Because the influence of suspension and tire properties are not taken account in the model used in this research, since a simplified model is sufficient for comparing different configurations, the moose test cannot be adapted to yield close to real life results. It will be assumed that to evade the obstacle, the rider will use the maximum steering angle. It can then be calculated what the threshold speed is at which the device will tip over (**T.3**). This will become test 3.

To cover performance on hills, it can be analyzed what the maximum acceleration of the scooter tips is while going uphill (**T.4**)), likewise for going downhill while braking (**T.5**). A slope of 15% will be used, as that gives a reasonable upper bound for streets that can be encountered in a hilly city like Zurich (Appendix A-C).

As the aforementioned tests already cover the lateral and longitudinal stability in ways that could actually matter in typical use cases, and because redundancy of test results must be avoided, the static tipping angles as described in Rentschler e.a. [17] will not be calculated.

2) Manoeuvrability: For the clear floor area, it can be analyzed how long it needs to be in order for the user to exit it while going forward with the maximum steering angle (**T.6**), without requiring the user to

The experimental set-ups as described by Koontz e.a. can also be used as inspirations for tests. In order to avoid redudancy, only test A and test D (see Appendix A-B2) will be used. For both tests, theoretical minima of the spaces (so with the scooters in theory grazing the walls) can be used, as adding spacing would not change configurations' performance with respect to each other. This results in one test that yields the theoretical minimum of the required hallway width for a 90° turn (**T.7**), while the other test yields the theoretical minimum of the required room width for a 360° turn (**T.8**).

3) Other scooter factors: Though tire and suspension properties play in important role in guaranteeing a comfortable ride and can impact safety as well, it can be assumed that, with the suspension analysis and manufacturing technologies of today, a satisfactory ride quality can be achieved for any configuration. More importantly perhaps, it is likely that all the moving parts required for the transformation mechanism make the frame compliant enough that getting realistic, useful results from a tire/suspension simulation simply is not possible in this stage of development. Therefore there will not be any tests or design parameters related to tire deformation or suspension.

Finally, to take into account the sitting posture, it will be calculated how much room is left in front of the feet, assuming an average rider sitting with their knees at 90° **T.9**.

4) Walker mode: If it is assumed that the device reduces in length with a certain factor in its transformation to walker mode, then a simple 2D model can be created to analyze the manoeuvrability in walker mode, just like for scooter mode.

Because of the reduced length and increased manoeuvrability of the device in walker mode, the length of the clear floor area in public transport should not be an issue. That leaves a test equivalent to test 7 or 8. It must be taken into account though, that because of the caster wheels used in walker mode, the walker can rotate around the middle of the rear axis. This manoeuvre may be difficult for users to perform however, because it requires them to walk sideways in a circular path while pulling with one arm and pushing with the other. Thus it can instead be analyzed how much room the walker needs when it spins about one of the rear wheels (**T.10**).

5) Overview: The tests, each one having one scalar quantity as a test result, are summarized below. Each test is accompanied by a requirement in the form of a minimum or maximum value (Appendix A-D).

- **T.1** Energy ($> 0\,\mathrm{J}$) required to tip the scooter while driving 5 km/h on a flat road with a 2 meter turning radius.
- **T.2** Angle (< 0.4472, fraction, unitless) of the tangent on the wheel in the contact point on a 20 mm tall obstacle.
- **T.3** Speed ($> 3\,\mathrm{km/h}$) at which the scooter will tip with the maximum steering angle.
- **T.4** Acceleration ($> 1\,\mathrm{m/s^2}$ at which the scooter starts tipping backward while driving straight up a 15° incline.
- **T.5** Deceleration ($> 4\,\mathrm{m/s^2}$ at which the scooter starts tipping forward while driving straight down a 15° incline.
- **T.6** Length ($< 2\,\mathrm{m}$) required for the clear floor area with the scooter driving forward with maximum steering angle, assuming a maximum steering angle.
- **T.7** Width ($< 1.5\,\mathrm{m}$) required for a hallway with a 90° bend in order for the device to drive through, assuming a maximum steering angle.
- **T.8** Width ($< 5\,\mathrm{m}$) required for a square room in order for the device to make a 360° turn in it, assuming a maximum steering angle.
- **T.9** Distance ($> 0\,\mathrm{m}$) between feet and front wheels.

T.10 Width ($< 0.9\,\mathrm{m}$) required for a hallway with a 90° bend in order for the device to go through in scooter mode, with the device turning around one rear wheel, assuming a certain shortening factor from scooter to walker mode.

F. Determining design parameters

1) General model assumptions: It must now be determined which design parameters will have assumed values, and what their values will be. First, general assumptions must be made about the models:

A.1 Suspension, tires and vehicle's body will all be assumed to be rigid.
A.2 Slipping does not occur.
A.3 A caster angle of zero is assumed, because its effect on handling are not relevant for the tests.
A.4 All wheels have the same diameter and thickness.
A.5 The wheelbase polygon's vertices define the track of the vehicle.
A.6 The wheelbase polygon's vertices are defined by the contact points in the middle of the tires when looking from above.
A.7 The vehicle's body does not stick out over the tires.
A.8 The scooter's CoM (Center of Mass) is only dependent on the vehicle's length.
A.9 The maximum steering angle of the front wheel(s) $\alpha_{\max}$ will be assumed to be 42° (close to observed values [23] [24]).
A.10 The rider will be modeled as a rigid mass fixed to the frame.
A.11 The rider's posture and anthropometry do not vary, with the rider sitting upright (knees and hips bent 90°) and having average body measurements and weight (See Appendix A-F2 and rider weight m_{p}).
A.12 In transition to walker mode, the body of the device will shorten such that the overall length is multiplied by a factor of 0.65 (Appendix A-E). The width in the front and in the back stay the same.
A.13 The front wheels in scooter mode will also function as caster wheels in walker mode.

2) Value assumption criteria: A design parameter's value will be determined through optimization unless one or more of the following criteria must apply:

C.1 Decreasing or increasing the value will only have a positive or negative effect on the test results.
C.2 Changing the value within its reasonable bounds only affects the test results marginally.
C.3 The value can be determined through the relationship with another design parameter.
C.4 The value can be determined straight from a design requirement or already made assumption.

3) Design parameter values: Below is a list of relevant design parameters and their respective assumptions, where all are about scooter mode unless stated otherwise. Bertocci e.a. (1997) [25] and Hitcock e.a. (2006) [18] were consulted to make sure that no relevant parameters are left out.

- Wheel diameter (d): The mean diameter for MMS wheels is 8.8 inches [26], but because bigger wheels are generally safer, wheel diameters of $8\,\mathrm{inch} \leq d \leq 12\,\mathrm{inch}$ will be analyzed for optimization.
- Wheel width (t): This can not vary much due to width required for traction, and does not have a significant influence on the safety manoeuvrability, thus criterion **C.2** applies. It will be assumed to be 80 mm [26]. See Appendix A-F1 for more detail.
- Scooter overall width at the rear (w_{s}): Will be optimized. Since all-round mobility scooters are around 65 cm [27], a range of $0.6\,\mathrm{m} \leq w_{\mathrm{s}} \leq 0.7\,\mathrm{m}$ will be investigated. If it were larger, it could not fit through some interior doors.
- Scooter overall width at the front (f_{s}): This design parameter is coupled to the scooter wheelbase width w_{sb}, and thus determines whether the wheelbase forms a triangle, trapezoid or a square. None of the criteria apply, f_{s} value will be optimized, with a range of $f_{\mathrm{s,min}} \leq f_{\mathrm{s}} \leq w_{\mathrm{s}}$, where

$$f_{\mathrm{s,min}} = d + 2q + u \tag{1}$$

Where $q = 0.05\,\mathrm{m}$ is the caster wheel offset in walker mode, and $u = 0.02\,\mathrm{m}$ is the minimum front wheel clearance in walker mode (see Figure 15). This definition of $f_{\mathrm{s,min}}$ is there to guarantee that, when the front wheels function as caster wheels in walker mode, they can point toward each other, allowing them to swivel freely for every configuration.
- Scooter overall length (l_{s}): The range for optimization will be $0.90\,\mathrm{m} \leq l_{\mathrm{s}} \leq 1.20\,\mathrm{m}$ (close to the mean ± SD [25]).
- Wheelbase width at the rear (w_{b}): With assumptions **A.6** and **A.7**, this parameter is coupled to w_{s} (criterion **C.3**), where

$$w_{\mathrm{b}} = w_{\mathrm{s}} - t \tag{2}$$

- Wheelbase width at the front (f_{b}): With assumptions **A.6** and **A.7**, this parameter is coupled to f_{s} (criterion **C.3**), where

$$f_{\mathrm{b}} = f_{\mathrm{s}} - t \tag{3}$$

- Wheelbase length (l_{b}): Using assumption **A.7**, the parameter is coupled to l_{s} (criterion **C.3**), where

$$l_{\mathrm{b}} = l_{\mathrm{s}} - d \tag{4}$$

- Number of wheels (N_{w}): Coupled to f_{b}, d and t (criterion **C.3**) as follows:

$$N_{\mathrm{w}} = \begin{cases} 3, & f_{\mathrm{b}} = 0 \\ \emptyset, & 0 < f_{\mathrm{b}} < f_{\mathrm{s,min}}(d) - t \\ 4, & f_{\mathrm{b}} \geq f_{\mathrm{s,min}}(d) - t \end{cases} \tag{5}$$

- Scooter CoM vertical position (h_{s}): Must be as low as possible for stability tests, as long as it is high enough to ride over obstacles (criterion **C.1**). It will be assumed to be mean value of h_{s} = 0.22 m (8.5 inches) with respect to the ground (see assumption **A.8**). [25].
- Scooter CoM horizontal position (p_{s}): The mean value here is 0.22 m (8.5 inches as well) forward from the rear axle [25]. Criterion **C.3** applies, as the CoM position changes when the length of the wheelbase changes (**A.8**). The mean wheelbase length is 80 cm

(31.6 inches) [25], thus the following estimation can be made to assume the value of this design parameter:

$$p_{\rm s} = \frac{3}{4} l_{\rm b} \tag{6}$$

- Scooter Mass m_s: See requirement R.4, will be maximum mass of 35 kg (criterion C.4) in order to not constrain the possible designs.
- Rider CoM vertical position ($h_{\rm p}$): Just like for the vehicle's vertical CoM position, the Rider's CoM must be as low as possible in order to ensure stability (criterion C.1). It is also constrained by the height of the seat. The height of the rider CoM will therefore be defined by adding the mean seat height $h_{\rm a}$, which is given by the vertical distance between the rear axle and the corner between the seat and the back support, and the person's CoM height with respect to that $h_{\rm b}$. Therefore, $h_{\rm p} = d/2 + h_{\rm a} + h_{\rm b}$, where $h_{\rm a} = 0.43\,{\rm m}$ [25]. Using D.A. Winter's Anthropometry model [28] and Munoz e.a.'s model for the CoM position [29] (which can be used because of assumption A.11), while assuming a mean German height of $1.69\,{\rm m}$ [30], it can be assumed that $h_{\rm b} = 0.19\,{\rm m}$ (Appendix A-F2). Accordingly,

$$h_{\rm p} = d/2 + 0.62\,{\rm m} \tag{7}$$

- Rider CoM horizontal position ($p_{\rm p}$): This design parameter's value will be optimized with a range of $0.050\,{\rm m} \leq p_p \leq 0.250\,{\rm m}$.
- Rider weight (m_p): Mean German weight of 73 kg is assumed [30] (assumption A.11, criterion C.4).

An overview of the design parameters can be seen in Figure 2.

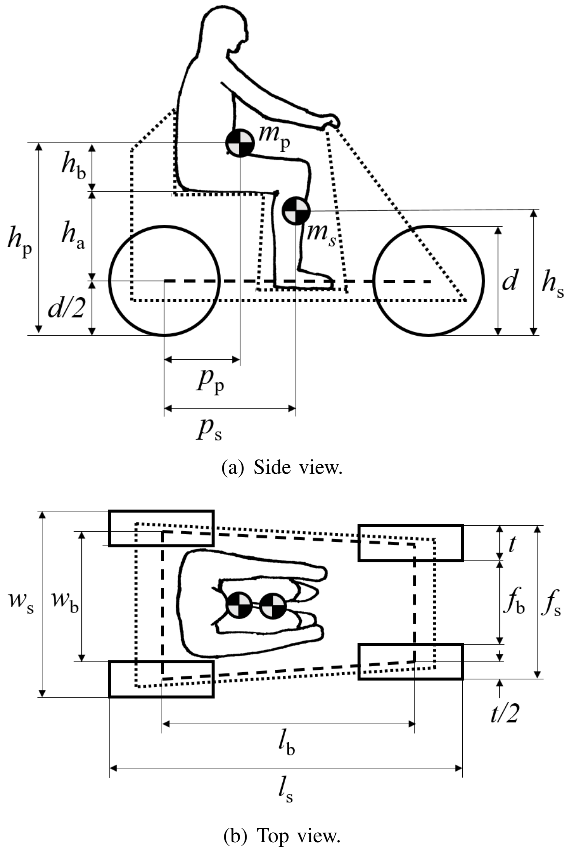


Fig. 2: Diagram of the scooter model with the wheels straight (steering angle $\alpha = 0$) including the design parameters. The configuration shown here is arbitrarily chosen.

G. Test details

A code framework was created in MATLAB® (See Appendix D for all code) that creates a certain number of different configurations, depending on ranges for the optimizable parameters. Each configuration has a unique combination of values for the optimizable design parameters. Using a 3D model featuring the wheels and overall center of gravity (Appendix A-G1), for each configuration a number of key geometrical values are calculated, which are then used as inputs for the tests. Many of the values are dependent on the steering angle as well. The outputs of the tests are saved, and used as inputs for the rating method, leading to a final grade for each configuration.

For verifications of the calculations for the tests, see Appendix B.

1) Tests 1-3: Both test 1 and test 3 have to do with lateral stability, so their models will be grouped together. Test 3 is the simpler case, since it is not dependent on the "tipping angle" of the vehicle.

While the model could be calculated using dynamical equations in three dimensions, that will not be necessary, since it is also possible using two. When making a left turn, the scooter will tip over line EF (Appendix A-G2). Suppose there is a body fixed frame $\mathcal{F}(\hat{\boldsymbol{u}}_{\rm x'}, \hat{\boldsymbol{u}}_{\rm y'}, \hat{\boldsymbol{u}}_{\rm z'})$, which is defined such that $\hat{\boldsymbol{u}}_{\rm x'} \perp EF$, then the acceleration along the x'-axis $a_{\rm C,x'}$ can be used. $a_{\rm C,x'}$ is the projection of $a_{\rm C}$ on the x'-axis, which is dependent on the path speed in C, $v_{\rm C}$. Note that, unless subscripts specify otherwise, v and r are the speed and turning radius respectively in the middle of the rear axis, which is to ease integration with the control system for further development. $v_{\rm C}$ can be calculated from v using the constant angular velocity, and geometry derived from the 3D model. A free body diagram drawn in the $\mathcal{F}$-frame can be seen in Figure 3. Here it is about to tip, meaning that it there is a moment balance, but no contact forces in the wheels on the inside of the turn (unstable equilibrium). Since the (angular) velocity is constant for tests 1 and 3, the following force-balance equations can be derived:

$$\Sigma F_{\rm x'} = -H_1 \qquad = m a_{\rm C,x'} \tag{8}$$
$$\Sigma F_{\rm z'} = -m_{\rm C} g + V \ = 0 \tag{9}$$
$$\Sigma M = kV - h_{\rm C} H_1 = 0 \tag{10}$$

Putting everything together, the following equation for the speed at which the scooter tips $v_{\rm tip}$ can be derived (Appendix A-H):

$$v_{\rm tip} = \sqrt{\frac{gks}{h_{\rm C} f^2 \cos\varphi}} \tag{11}$$

Where g is the gravitational acceleration, s is the turning radius in point C (distance from O to C), φ is the angle between $\hat{\boldsymbol{u}}_{\rm x'}$ and ${}^{\rm C}\boldsymbol{r}_{\rm O}$ (vector from O to C) and $f = \frac{s}{r}$

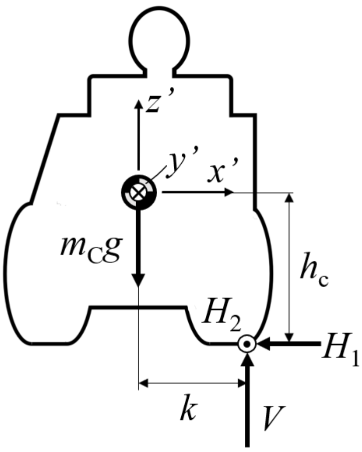


Fig. 3: Free body diagram of the scooter in drawn in the $x'z'$-plane (part of the $\mathcal{F}$-frame), in the situation where it is on the verge of tipping (test 3). V, H_1 and H_2 represent the sum of the forces on the tires on the outside of the turn, and grab on along line EF, where V is the component along the z'-axis, H_2 along the y'-axis (not relevant for calculations) and H_1 along the x'-axis. Note that the shape drawn for the scooter is arbitrary and not relevant.

is a scalar defined such that $v_C = vf$. If s, φ and f are calculated for the situation where the steering angle α_{max} is applied, v_{tip} is the result of test 3.

To get the results for test 1, the scooter must first be rotated along the tipping line EF before it reaches the point at which it is in an unstable equilibrium. This introduces a new angle γ with which the scooter rotates counter-clockwise around the line EF. Equation 11 can be used again, but because γ changes the position of the CoM, all values in the equation (except g) are affected, but these can be calculated in the 3D model. To find the angle γ at which the scooter is about to tip for the given turning radius $r = 2\,\mathrm{m}$ and speed $v = 5\,\mathrm{km/h}$, as must be done for test 1, an iterative method is used, where guesses for γ are made until $v_{\mathrm{tip}} = 5\,\mathrm{km/h}$. This angle is then used to calculate the difference in height of the CoM, which yields the energy required to tip the scooter for the given turning radius and speed.

Finally, for test 2, the angle with which the tangent on the wheel strikes the object is calculated with the following equation:

$$\frac{a}{l} = \frac{a}{\sqrt{ad - a^2}} \tag{12}$$

Where a/l is the angle, $a = 20\,\mathrm{mm}$ is the object height and d is the wheel diameter (see Appendix A-H1).

2) Tests 4-5: The free body diagram as shown in Figure 4 is used. Using this, the following force and moment balances can be derived:

$$\Sigma F_{\mathrm{Y}} = V - m_{\mathrm{C}} g \sin\kappa \ = m_{\mathrm{C}} a_{\mathrm{Y}} \tag{13}$$

$$\Sigma F_{\mathrm{Z}} = N - m_{\mathrm{C}} g \cos\kappa = 0 \tag{14}$$

$$\Sigma M = h_{\mathrm{C}} V - p_{\mathrm{C}} N \quad\ = 0 \tag{15}$$

Where $\kappa = 8.5308°$, which is the angle corresponding to a 15% gradient. The acceleration is given in Equation 17

$$a_{\mathrm{Y}} = g\left(\frac{p_{\mathrm{C}}}{h_{\mathrm{C}}}\cos\kappa - \sin\kappa\right) \tag{16}$$

Equation 17 can be adapted for test 5 as well if the direction of the vehicle is changed, leading to the following equation for the deceleration:

$$a_{\mathrm{Y}} = g\left(\frac{l_{\mathrm{b}} - p_{\mathrm{C}}}{h_{\mathrm{C}}}\cos\kappa - \sin\kappa\right) \tag{17}$$

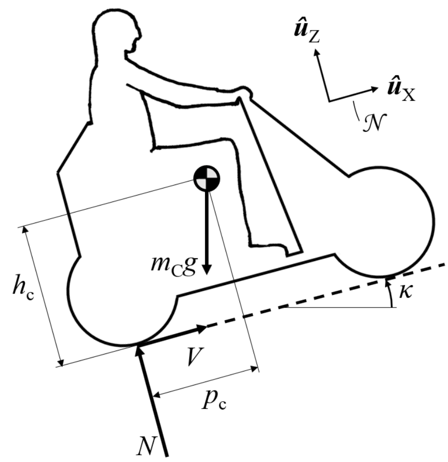


Fig. 4: Free body diagram for tests 4 and 5. N denotes the normal force in the rear tires, V the force perpendicular to that and κ the angle of the incline. An inertial frame of reference $\mathcal{N}(X, Y, Z)$ is used, though here its axis are aligned with the body-fixed frame $\mathcal{B}$.

3) Tests 6-8: These tests all have to do with the swept path of the scooter. For test 6 and 8, only the outer edge of the swept path must be known, but for test 7, the width of the path itself must be known. To calculate the radius of the outer edge of the swept path, one must consider the turning radii of all points on the scooter, and take the largest turning radius. To get the width of the swept path, the minimum turning radius of all points must be substracted from the maximum.

To calculate the result for 6, the intersection between the circle with the maximum turn radius and the line along the left side of the left rear wheel (assuming a left turn) can be calculated. The length of this line corresponds to the length of the clear floor area (Appendix A-H2).

For test 7, consider Appendix A-H3. The width of the hallway w can be calculated as follows:

$$w = r_{\mathrm{max}} - \frac{r_{\mathrm{min}}}{\sqrt{2}} \tag{18}$$

Where r_{max} is the maximum turning radius of all points on the scooter for when $\alpha = \alpha_{\mathrm{max}}$, and r_{min} is the minimum turning radius.

The width for test 8 is easily calculated by multiplying the maximum r_{max} by 2.

4) Test 9: The outcome of this test relies on both the physical dimensions of the rider, as well as the dimensions of the scooter. For the dimensions of the rider, anthropometric data by Winter (2009) [28] is used, with a mean German height of $h = 1.69\,\mathrm{m}$ [30]. Information on the rider CoM position while sitting down is taken from Munoz e.a. (2011) [29]. The author also made an estimation that the horizontal distance from the ankle to the tip of the toe is $\frac{4}{5}$ the foot length. See Appendix A-H4 for more details.

5) Test 10: To calculate how wide a hallway needs to be for a 90° turn, one must know $r_{\max}$ again, but for walker mode. It must also be noted that the turning radius is not bound by $\alpha_{\max}$ anymore, as the front wheels are free to rotate, and the center of the turn is now in the left rear wheel. The wheels rotate as caster wheels around a point forward from the front axis, for which the author will assume an offset of $q = 50\,\mathrm{mm}$. Equation 19 shows how the width of hallway can be calculated.

$$w = \frac{t}{2} + \max\left(r_{\mathrm{G}}, r_{\mathrm{H}}\right) \tag{19}$$

Where r_{G} and r_{H} are the radii of the swept paths as determined by the outer front and rear wheel respectively. See Appendix A-H5 for a diagram of this test.

H. Assessment method

To convert the performances on tests for each configuration into an overall score, each outcome of the test must first be normalized. This will be done using the upper and lower bounds mentioned in subsubsection II-E5. First, every concept that is not meet within required bounds for every tests will be discarded. Then, for every single test, the result that is furthest away from the required value will receive a score of 100, and all other results are scaled linearly.

To combine these test results into one overall score per configuration, the Analytic Hierarchy Process (AHP) is used [31], which uses pairwise comparisons to allow for a thorough and complete examination of the relative importance of all tests. An overview of these pairwise comparisons is given in the 10×10 matrix $\mathbf{A}$. Looking at the 5$^{\text{th}}$ column on the 1$^{\text{st}}$ row, for example, it can be seen that the outcome of test 1 was deemed 3 times as important as the outcome of test 5. All values have been heuristically assigned by the author.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 & 2 & 3 & 2 & 4 & 3 & 2 & 2 \\ \frac{1}{2} & 1 & 1 & 2 & 3 & 2 & 3 & 3 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 & 2 & 2 & 2 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 1 & 2 & 2 & 2 & 2 & 2 & \frac{2}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 2 & 1 & 2 & 2 & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{3} & \frac{1}{2} & 3 & \frac{1}{2} & 1 & 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 3 & \frac{1}{2} & 1 & 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 3 & 1 & 2 & 2 & 1 & 1 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{3}{2} & 3 & 2 & 3 & 3 & 1 & 1 \end{bmatrix}$$

Relative importance was given to the tests concerning sideways stability over the scooter manoeuvrability tests, as it can be assumed that scooter mode will mostly be used outdoors, making its performance in tight spaces less relevant compared to its safety. Accordingly, the manoeuvrability in walker mode was in general given more importance with respect to the other manoeuvrability tests.

Finally, to get the weights for each test, the principal eigenvector of $\mathbf{A}$ can be calculated [32]. This yields the vector $\mathbf{w}$ (see Appendix A-I or right column in Table IV). Multiplying the 1×10 vector containing test results for a configuration with $\mathbf{w}$ will yield its overall score. Sorting by final scores will then give a ranking of all concepts from overall best to worst.

III. Results

With 5 different wheel sizes, 3 different values for w_{s}, 4 different values for f_{s}, 7 different values for the overall length and 5 different values for p_{p}, $5 \cdot 3 \cdot 4 \cdot 7 \cdot 5 = 2100$ configurations have been generated. After removing the configurations that didn't meet the requirements, about half of the configurations (1007) are left.

A. Design parameters and test results

This subsection contains an overview of how the optimizable design parameters affect the test outcomes. For a full visual overview, refer to Appendix C-B.

1) Wheel diameter: The wheel diameter d mostly has an effect on the outcomes of tests 2, 4, 5 and 9 (Appendix C-A1). Especially for test 1, a clear monotonic correlation can be observed, where higher wheel diameters lead to lower angles. It can also be seen how larger wheels can lead to less space between the feet and the front wheels.

The deceleration values are almost $3\,\mathrm{m/s^2}$ removed from their minimum value, while the acceleration has a margin of almost $2\,\mathrm{m/s^2}$. Meanwhile, for test 10, there are many configurations that are close to the minimum required distance.

2) Overall width rear: For w_{s}, its influence on most test results is less pronounced relative to the other design parameters, with only tests 1 and 10 being clearly affected. Nonetheless, Appendix C-A2 shows how a wider rear correlates to a higher energy required to tip the vehicle during a turn, and how it correlates to needing more space to turn the device in walker mode.

3) Overall width front: As can be seen in Appendix C-A3, the width in the front has substantial influence over the majority of test results. There always is a gap in f_{s} because of the minimum required distance between the front wheels $f_{\mathrm{s,min}}$, as well as a gap in test performance. It can be seen how, for test 1 and test 3, a wider front correlates to a higher amount energy and speed, respectively, required to tip the scooter. On the other hand, the results for tests 6 through 8 demonstrate that a higher value for f_{s} can also correspond to more space required for manoeuvres. This can also be observed in the result for test 10.

It can also be noticed that for test 6 through 8, the results are well within the requirements, especially for the latter two.

Finally, in the results for test 1, even the best performing three wheeled configuration (in the lower band) has a lower energy than any four-wheeled configuration (upper band). This pattern can not be observed for the other lateral stability test, test 3, where there still is some overlap. The same goes for the other tests in the figure.

4) Overall length: Just like f_{s}, the overall scooter length l_{s} is relatively influential for the test outcomes

(Appendix C-A4). For tests 4 and 5, a bigger length correlates to a higher maximum acceleration and deceleration. On the other hand, a larger length can be linked to lower manoeuvrability, but also a higher amount of space between the feet and the front wheels.

5) Longitudinal rider CoM position: Lastly, p_{p} in Appendix C-A5 shows some correlation with the test results. First, there is a clear link where a more forward rider CoM seems to mostly result in a higher maximum acceleration, but a lower downhill deceleration. A higher value for p_{p} also correlates to a lower outcome value for test 10.

B. Top configurations

While subsection III-A showed how the the design parameters influenced the test results in general, this subsection will go into more detail about how design test scores relate to each other for single configurations. Tables containing results for the top 70 configurations can be found in Appendix C-D.

The main results, the best rated sets of optimizable design parameters, can be seen in Table II. Note that for all configurations, the maximum overall length and minimum rider CoM position have been chosen. None of the concepts have the minimum wheel size of $8\,\mathrm{inch}$ ($0.0203\,\mathrm{m}$). The top nine configurations have a trapezoidal wheel configuration, with the top three-wheeler coming in at place 9, while the first rectangular wheel configuration lands at place 34. Only four three-wheelers exist in the top 35.

Rank	1	9	34
Configuration #	1186	1151	1116
d (inch)	10	10	10
w_{s} (m)	0.700	0.700	0.650
f_{s} (m)	0.454	0.080	0.650
l_{s} (m)	1.200	1.200	1.200
p_{p} (m)	0.050	0.050	0.050

TABLE II: Overview of the optimizable parameters for the top configurations per wheelbase type. Configuration #1186 shows the most promising set of design parameters overall (trapezoidal wheelbase), #1151 for three wheels and #1116 for a rectangular wheelbase.

Table III and Table IV show the test results and scores for the top configurations per wheelbase type. Notice the differences between the three-wheeled and four-wheeled configurations (three-wheelers can be recognized by $f_{\mathrm{s}} = 0.080\,\mathrm{m}$). The three wheel concept has relatively poor performance in the sideways stability tests (tests 1 and 3), but performs well in the manoeuvrability tests (tests 6, 7, 8 and 10). Appendix C-C shows the best configuration in the MATLAB® 3D vizualizer. It can also be noticed how much higher the three wheeled concept scores on test 10 compared to the other concepts; this positive difference of more than 40 points is not seen for any other test.

Lastly, notice how all configurations have relatively high scores for test 9, with the lowest score being only 65.13, corresponding to a distance of $0.21\,\mathrm{m}$ between the feet and the front wheels.

Rank	1	9	34		
Config. #	1186	1151	1116	Unit	Req.*
Test 1	80.57	52.44	86.31	Energy (J)	>0
Test 2	0.29	0.29	0.29	Angle (-)	<0.4472
Test 3	8.32	6.63	8.99	Speed (km/h)	>3
Test 4	2.99	2.99	2.99	Accel. (m/s^2)	>1
Test 5	10.03	10.03	10.03	Decel. (m/s^2)	>4
Test 6	1.64	1.41	1.76	Length (m)	<2
Test 7	1.13	0.96	1.21	Width (m)	<1.5
Test 8	3.51	2.92	3.84	Width (m)	<5
Test 9	0.25	0.25	0.25	Distance (m)	>0
Test 10	0.85	0.74	0.90	Width (m)	<0.9

TABLE III: Overview of the test results for the top configurations per wheelbase type, together with the required minimum/maximum values.
*Req. = required minimum maximum values as per subsubsection II-E5

Rank	1	9	34	
Configuration #	1186	1151	1116	Weight
Test 1	78.48	51.08	84.07	0.1787
Test 2	84.99	84.99	84.99	0.1508
Test 3	86.78	59.18	97.68	0.1503
Test 4	42.40	42.40	42.40	0.0986
Test 5	85.54	85.54	85.54	0.0396
Test 6	39.01	64.37	26.02	0.0766
Test 7	49.79	71.58	39.24	0.0515
Test 8	52.71	73.60	40.97	0.0553
Test 9	76.76	76.76	76.76	0.0858
Test 10	17.48	53.72	0.60	0.1127
Overall	64.48	63.74	63.03	1.0000

TABLE IV: Overview of the test scores for the top configurations per wheelbase type. Test scores are given on a scale from 0 to 100, where 0 corresponds to the worst possible result (minimum/maximum required value).

IV. Discussion

A. Influence of parameters on test

1) Wheel diameter: The outcome of test 2 is solely dependent on the wheel diameter. However, the model used is a rather basic one for gaining insight in how a tire would interact with an object. This was done because, as was already explained, making accurate predictions would be outside the scope of this research. The current model for test 2 does not shed light on how exactly a collision affects the vehicle's dynamics, but it did demonstrate the very basic geometric relation that exists between a tire and a small obstacle, which has been used to quantify the positive influence of wheel size on safety. Thus, this positive effect could be included in the overall score. So while the test does give a satisfactory estimation in this design stage within the research scope, a new model could be developed, perhaps in a later design stage when estimations can be made about the suspension and frame compliance.

The reason d affects the outcomes of multiple tests can be explained by the fact that the l_{s} and d are coupled to each other through Equation 4. This explains why larger wheel diameters tend to lower the acceleration and deceleration, while increasing the amount of space required to manoeuvre the device.

2) Overall width rear: Since the maximum and minimum values of w_{s} are so close to each other, the influence of this design parameter on the test results seems the least strong compared to the other design parameters. The gap in Appendix C-A2 for test 1 can be explained by the gap in f_s values (because of the difference between 3 and 4-wheeled configurations), as it can be for all such "white bands" in the other figures. It also makes sense that higher values of f_s can be linked to a higher energy for test 1 and a greater width for test 10: a wider rear likely has a higher lateral stability, but also likely takes up more space, though exact test results still depend heavily on the other design parameters in a configuration.

3) Overall width front: The dichotomy between lateral stability and manoeuvrability can be observed for f_s most clearly. Its influence on the results is larger than for w_s, since the minimum and maximum values are further apart. It is also interesting to note the lack of overlap between three and four wheeled configurations in the results for test 1, while that overlap does exist for test 3. This means that tests 1 and 3 truly are distinct ways of defining lateral stability.

Apart from that, the higher overlap between three and four-wheeled configurations for the latter four test results suggests that choosing between three or four wheels is more likely to affect the lateral stability than the manoeuvrability.

4) Overall length: For l_s, a similar dichotomy exists as for the scooter width: logically, a greater length tends to result in a more longitudinally stable design, but also in a less manoeuvrable one. However, as for almost all relationships between single design parameters and test results, it is still dependent on the other design parameters.

5) Longitudinal rider CoM position: As can be expected, p_p mostly affects the maximum acceleration and deceleration for tests 4 and 5. For both tests, the results are well above the minimum required values, and are therefore not rated as important in the matrix $\mathbf{A}$. On the other hand, because the parameter affects the position of the feet, a higher value for p_p can correlate to the feet being closer to the wheels.

B. Top configurations

Given that most of the top configurations (30 out of the top 35) are trapezoidal wheel configurations, it is proven that they can offer a good compromise between lateral stability and manoeuvrability. Nonetheless, concepts with different wheelbases are still very close in terms of overall score. This is a direct result of the values in the relative importance matrix $\mathbf{A}$ and the required values for the test results, meaning that adjusting either slightly could already change the ranking of the configurations.

The effects of the weights on the overall scores in Table III are clear: because of the emphasis that was put on the outcomes of tests 1 through 4, the top concepts has the maximum values for l_s and w_s, meaning that overall, these concepts are more stable but less manoeuvrable. The three wheeled concept is the other way around, which can especially be seen in its result for test 10. Because of these differences, it still managed to land in the top 3 in terms of overall performance.

As for test 9, all models guarantee that the "mean rider" is able to sit with their knees bent at 90° or greater, and that even larger riders will have no issue with this, as the extra distance can still accommodate riders well above mean length.

By far the most deciding factor for the device's overall rating is the amount of wheels. Three wheeled devices clearly tend to be take up less space, especially in walker mode, while four wheeled devices are more stable. As was already mentioned in II-C1, most users who buy a mobility scooter themselves chose for four wheels [16], but there still clearly is a demand for both, and perhaps the extra manoeuvrability in walker mode would make three wheeled versions more attractive to the user. Based on the results, rectangular wheelbases do not seem worth it to develop, so it could be a possibility to first choose the top-rated configuration, and aim to bring a four wheeled version (like the highest scoring configuration #1186) with a trapezoidal wheelbase to the market. Later, a three wheeled version (like configuration #1571) could also be developed. This would allow individual users to pick a three wheeled version if they prefer to be manoeuvrable and are less concerned about safety, or the four wheeled version if their priorities are the other way around.

On the other hand, the increased mechanical complexity of a four wheeled version, could make it easier to develop a three-wheeled version first, but this depends on the inner workings of the concept that is being developed by the other student.

C. Recommendations for further research

Because this research has been done in an early stage of development, many assumptions and simplifications had to made in order to make a useful contribution to the project. However, the MATLAB® code framework that has been constructed is highly adaptable, and can be used for further research in this topic.

1) Suspension: As progress is being made in the design process of the scooter-walker hybrid, more details of the design will become clear. These details can then be used to optimize the design further, in a more detailed manner. For example, once a rough model for the transformation mechanism has been designed, its compliance can be analyzed, and used in a suspension model to simulate how the device will behave when it rides on bumpy roads or when it encounters an obstacle. This could also be integrated with a more accurate model for evaluating wheel size and other wheel properties, like grip on slippery roads. Perhaps certain already existing tools (like one by Garcia-Pozuelo e.a. [33]) can aid in predicting the interaction with bumps.

2) Slipping: It has also been assumed that no slip will happen, and that tests 1 and 3 happen at a constant (angular) velocity. It could be useful to challange these assumptions in further research, especially since slipping behavior may relate to p_p. In the moose test used in the automotive industry, it often happens that cars slip. This could be affected by the sudden input on the steering wheel as well as by the CoM position. Even though cars' CoMs are considerably lower relative to the wheelbase than mobility scooter CoMs, resulting in a higher likeliness to slip instead of slip, it could still be useful to analyze slipping behavior for the scooter. This could mean that certain configurations slip instead of tip when the rider tries to evade an accident. However, one must come up with a way to still have only one test score if a slipping model is incorporated into a test.

3) Maximum steering angle: Another assumption that could be challanged is that of the maximum steering angle. The author assumed that the angle is the same across all configurations, but this does not take into account that it is theoretically possible to rotate a single front wheel by $360°$. However, the steering angle is often still limited for three-wheeled vehicles in real life, as allowing the user to steer as far as they wish could lead to dangerous situations, but no literature could be found to support this. More research would be needed to determine a more accurate number for the maximum steering angle of three-wheeled vehicles, as this would affect the test outcomes.

4) Rider assumptions: Something else that could be looked into further, is the assumption that the rider is a fixed mass to the frame. Though that assumptions gave satisfactory results in this design stage, perhaps a more advanced model could be used in the future. This could be done, for example, using a lumped parameter model, like Asperti e.a. did for E-scooter modeling [34]. Alternatively, a model where the rider actively changes its posture could be used. A model like this could improve the scores of less stable configurations, as it could include the rider actively leaning into the turn, increasing the system's overall lateral stability. This model could be extended to be integrated with the wheel, suspension and slipping model to calculate a closer equivalent to the moose test. Alternatively, a physical test could be carried out, once a prototype is constructed.

What can also be improved upon during the design process, are the assumptions for the scooter's CoM position. For this research, the values extracted from literature give a satisfactory starting point, but as the model is being developed further, more accurate replacements for Equation 6 can be developed. The factor by which the device shortens in its transformation to walker mode could then also be re-evaluated. These could both be reassessed with the aid of a physical or digital model of the device, once this has been created.

It could also be analyzed how the test results are influenced by the assumptions that have been made about the rider itself. How would the results change for different rider body weights, sizes and proportions? As was already discussed, the results for test 9 indicate that tall riders will fit on the device as well, but a sensitivity analysis could be performed to see how exactly the other rider parameters influence the outcomes.

5) Assessment method: Finally, it could be valuable if the AHP matrix were to be reassessed using feedback from potential users. In this paper, the estimations were made according to the author's best knowledge based on the literature that has been reviewed, but perhaps users of the device would give different ratings, which could affect the final outcome. It would be particularly useful to focus on the trade-off between lateral stability and manoeuvrability, and to explore the difference between the two modes: when would users exactly use which mode, and how does that affect their preferences for the device? Perhaps this refinement of the weights could also create clarity on the issue of three versus four wheels.

However, it should be taken into account that, in the AHP process, the number of pairwise comparisons roughly scales quadratically with the number of tests. It would therefore be recommendable to put the focus on improving/changing the tests, and refining the values in the AHP matrix, rather than on adding more tests.

V. Conclusion

Through the use of literature about mobility scooters, and while maintaining a link to the walker mode of the hybrid device, a set of tests and key optimizable design parameters have been identified. Using the Analytic Hierarchy Process, this resulted in a set op optimal key design parameters for the scooter-walker hybrid, thereby answering the research question.

Apart from the set of design parameters that have been identified, perhaps the most important contribution of this research may be the methodological framework that has been developed in the process, and the code framework that forms a part of it. This serves as a highly adaptable tool that can be used by future future participants in the project, optimizing the design as it is being developed, and perhaps one day ultimately resulting in enhancing people's mobility and quality of life.

Acknowledgments

The author would like to thank Prof. Heike Vallery and Prof. Robert Riener for their supervision. Special thanks Adrian Esser for stepping in to take on the role of co-supervisor, and giving me his time for lots of support and advice. Thanks also to Olivier Nüssli for developing the first prototype, Karim Hassan for making the safety helmet (and thanks to both of them for the months worth of Swiss-German lessons), Ludovico Papavassiliou for making the control system, Amanda Felouzis for currently designing the second design iteration, and all of them for the support and fun along the way.

References

[1] C. Musselwhite and H. Haddad, "Mobility, accessibility and quality of later life," *Quality in Ageing and Older Adults*, vol. 11, no. 1, pp. 25–37, Jan 2010. [Online]. Available: https://doi.org/10.5042/qiaoa.2010.0153

[2] Dutch Ministry of Ministry of Health, Welfare, and Sport, "Functioneringsproblemen | leeftijd en geslacht," 2023. [Online]. Available: https://www.vzinfo.nl/functioneringsproblemen/leeftijd-en-geslacht#beperkingen-uitgesplitst

[3] University of Washington, "Mobility impairments." [Online]. Available: https://www.washington.edu/doit/mobility-impairments

[4] S. Musich, S. S. Wang, J. Ruiz, K. Hawkins, and E. Wicker, "The impact of mobility limitations on health outcomes among older adults," *Geriatric Nursing*, vol. 39, no. 2, pp. 162–169, Mar 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0197457217302057

[5] R. Thoreau, "The impact of mobility scooters on their users. does their usage help or hinder?: A state of the art review," *Journal of Transport & Health*, vol. 2, no. 2, pp. 269–275, Jun 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214140515000201

[6] S. Jang, W. Mortenson, L. Hurd, and R. Kirby, "Caught in-between: tensions experienced by community mobility scooter users," *Disability and Society*, vol. 35, no. 10, pp. 1577–1595, 2020.

[7] C. D'Souza, V. Paquet, J. Lenker, and E. Steinfeld, "Self-reported

difficulty and preferences of wheeled mobility device users for simulated low-floor bus boarding, interior circulation and disembarking," *Disability and Rehabilitation: Assistive Technology*, vol. 14, no. 2, pp. 109–121, 2019.

[8] M. P. LaPlante and H. S. Kaye, "Demographics and trends in wheeled mobility equipment use and accessibility in the community," *Assistive technology: the official journal of RESNA*, vol. 22, no. 1, pp. 3–17; quiz 19, 2010.

[9] O. Nüssli, "Design of a new e-scooter concept for people with mobility impairments," *ETH Zurich*, 2023.

[10] Rollz International, "Electric wheelchair for extra independence | rollz motion electric." [Online]. Available: https://www.rollz.com/rollator-walkers/electric-wheelchair/

[11] MovingStar, "Movingstar allinone - faltbarer elektro-rollator." [Online]. Available: https://www.moving-star.de/movingstar-allinone-faltbarer-elektrorollator

[12] Swiss Federal Council, "Sr 741.41 - verordnung vom 19. juni 1995 über die technischen anforderungen an strassenfahrzeuge (vts)," Jun 1997. [Online]. Available: https://www.fedlex.admin.ch/eli/cc/1995/4425_4425_4425/de

[13] M. v. B. Z. e. Koninkrijksrelaties, "Regeling voertuigen," last Modified: 2023-02-24. [Online]. Available: https://wetten.overheid.nl/BWBR0025798/2023-01-01

[14] Sweco, "Gehandicaptenvoertuigen: Landenscan regelgeving," Nov 2022. [Online]. Available: https://open.overheid.nl/documenten/ronl-c0389c5c3c23e00e294e6706a74da363c0480780/pdf

[15] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties, "Beleidsregel aanwijzing bijzondere bromfietsen," May 2019, last Modified: 2021-07-03. [Online]. Available: https://wetten.overheid.nl/BWBR0035848/2019-05-02

[16] R. Davidse, K. Duijvenvoorde, I. W. Louwerse, M. Boele-Vos, A. Stelling-Kończak, and A. Algera, "Ongevallen voorkomen letsel beperken levens redden," *Stichting Wetenschappelijk Onderzoek Verkeersveiligheid*, 2018.

[17] A. J. Rentschler and R. A. Cooper, "Comparison of the dynamic and static stability of power wheelchairs versus scooters," vol. 1, 1999, p. 613.

[18] D. Hitcock, M. Hussey, S. Burchill, and M. Galley, "A survey of occupied wheelchairs and scooters," *Mobility and Inclusion Unit of the Department for Transpor*, 2006. [Online]. Available: https://www.livewellyork.co.uk/Resources/myLife/library/StuffNotStaff-Equipment/A%20survey%20of%20occupied%20wheelchairs%20and%20scooters.pdf

[19] A. Bharathy and C. D'Souza, "Revisiting clear floor area requirements for wheeled mobility device users in public transportation," *Transportation Research Record*, vol. 2672, no. 8, pp. 675–685, 2018.

[20] K. Tomiyama and K. Moriishi, "Pavement surface evaluation interacting vibration characteristics of an electric mobility scooter," *Lecture Notes in Civil Engineering*, vol. 76, pp. 893–900, 2020.

[21] A. Koontz, E. Brindle, P. Kankipati, D. Feathers, and R. Cooper, "Design features that affect the maneuverability of wheelchairs and scooters," *Archives of Physical Medicine and Rehabilitation*, vol. 91, no. 5, pp. 759–764, 2010.

[22] International Organization for Standardization, "Iso 3888-2:2011." [Online]. Available: https://www.iso.org/standard/57253.html

[23] D. Eck, T. Heim, R. Hess, and K. Schilling, "A narrow passage assistance functions on a mobility scooter for elderly people," in *ROBOTIK 2012; 7th German Conference on Robotics*, May 2012, pp. 1–6.

[24] J. Suzurikawa, S. Kurokawa, H. Sugiyama, and K. Hase, "Estimation of steering and throttle angles of a motorized mobility scooter with inertial measurement units for continuous quantification of driving operation," *Sensors*, vol. 22, no. 9, 2022.

[25] G. Bertocci, P. Karg, and D. Hobson, "Wheeled mobility device database for transportation safety research and standards," *Assistive Technology*, vol. 9, no. 2, pp. 102–115, 1997.

[26] R. Montgomery, Y. Li, T. Dutta, P. Holliday, and G. Fernie, "Quantifying mobility scooter performance in winter environments," *Archives of Physical Medicine and Rehabilitation*, vol. 102, no. 10, pp. 1902–1909, 2021.

[27] R. Heeasakkers, "Afmetingen scootmobiel: meten is weten!" *Scootmobielberging*, 2020. [Online]. Available: https://www.scootmobielberging.nl/blog/afmetingen-scootmobiel-meten-weten

[28] D. A. Winter, *Biomechanics and motor control of human movement*, 4th ed. Hoboken, N.J: Wiley, 2009.

[29] F. Munoz and P. R. Rougier, "Estimation of centre of gravity movements in sitting posture: application to trunk backward tilt," *Journal of Biomechanics*, vol. 44, no. 9, pp. 1771–1775, Jun 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021929011003137

[30] Normenausschuss Ergonomie (NAErg) im DIN, "Din spec 91279 - wesentliche masse des menschlichen körpers für die technische gestaltung," 2011.

[31] J. A. Jagoda, S. J. Schuldt, and A. J. Hoisington, "What to do? let's think it through! using the analytic hierarchy process to make decisions," 2020. [Online]. Available: https://kids.frontiersin.org/articles/10.3389/frym.2020.00078

[32] R. Ramanathan, *Multicriteria Analysis of Energy*. New York: Elsevier, 2004, pp. 77–88. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B012176480X002400

[33] D. Garcia-Pozuelo, A. Gauchia, E. Olmeda, and V. Diaz, "Bump modeling and vehicle vertical dynamics prediction," *Advances in Mechanical Engineering*, vol. 6, p. 736576, Jan 2014. [Online]. Available: https://doi.org/10.1155/2014/736576

[34] M. Asperti, M. Vignati, and F. Braghin, "Modelling of the vertical dynamics of an electric kick scooter," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9266–9274, 2022. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85112660454&doi=10.1109%2fTITS.2021.3098438&partnerID=40&md5=b0d5bc2238eea7470b93ad0a8b3fdcbc

[35] Stadt Zürich Tiefbauamt, "Leitfaden "standards fussverkehr"," 2020. [Online]. Available: https://www.stadt-zuerich.ch/content/dam/stzh/ted/Deutsch/taz/Verkehr/Publikationen_und_Broschueren/Leitfaden_Standards_Fussverkehr_Trottoirbreiten.pdf

[36] F. Niesar, K. Treichler, and P. Regli, "BegegnungsfÂ€lle und fahrbahnbreiten," 2017.

[37] Barrierefrei Leben, "Ratgeber "tipps- und lösungsbeispiele: Flur und garderobe anpassen - das ausgehen erleichtern" - online-wohn-beratung.de." [Online]. Available: https://www.online-wohn-beratung.de/wohnungsanpassung-barrierefrei-(um-)-bauen/ratgeber-wohnungsanpassung-barrierefrei-(um-)-bauen-tipps-loesungsbeispiele/flur-und-garderobe/

[38] J. C. Huston, B. J. Graves, and D. B. Johnson, "Three wheeled vehicle dynamics," *SAE Transactions*, vol. 91, pp. 591–604, 1982. [Online]. Available: https://www.jstor.org/stable/44631967

[39] Jos, May 2023. [Online]. Available: https://nl.mathworks.com/matlabcentral/fileexchange/10064-allcomb-varargin

[40] M. Bartholdt. [Online]. Available: https://nl.mathworks.com/matlabcentral/answers/183311-setting-default-interpreter-to-latex#answer_803656

[41] [Online]. Available: https://nl.mathworks.com/matlabcentral/answers/271016-how-to-get-the-vector-from-a-point-orthogonal-to-a-vector#answer_211994

[42] W. Rose. [Online]. Available: https://nl.mathworks.com/matlabcentral/answers/1869077-how-to-plot-a-cylinder-from-a-specified-axis#answer_1118047

[43] A. Danz. [Online]. Available: https://nl.mathworks.com/matlabcentral/answers/1594904-how-do-i-rotate-the-view-of-a-cylinder-created-using-surf-plot#answer_839699

[44] June 2023. [Online]. Available: https://nl.mathworks.com/matlabcentral/fileexchange/7470-plot-2d-3d-vector-with-arrow

APPENDIX A
METHODS

A. Full legal overview table

	Switzerland E-scooter	Switzerland Mobility scooter	Netherlands E-scooter	Netherlands Mobility scooter	Germany E-scooter	Germany Mobility scooter
amount of wheels	not codified	not codified	not codified	not codified	not codified	not codified
maximum velocity (km/h)	20 (±10%)	30 (±10%)	25	6 (sidewalk), 30 (cycling lanes in built-up areas), 40 (cycling lanes outside built-up areas), 45 +5 km/h(on roads)	20	6 (sidewalk), 15?
maximum power (W)	500	1000	4000	not codified	500	not codified
maximum weight (kg)	200	not codified	125	not codified	55	300
wheel diameter (m)		not codified		not codified		not codified
maximum length (m)		n.c.		3.5		not codified
maximum width (m)		1		1.1		1.1
type approval required (Typengenehmigung / typegoedkeuring)	no	>10 km/h (EU type appproval)	yes (Dutch type approval)	?	yes (German type approval)	?
driver's license required	14-16 y/o: cat.M	14-16 y/o or >20 km/h: cat.M	no	no	no	> 14 y/o
license plate required	no	>10 km/h	no	no	yes	no
vehicle admission test	no	>10 km/h	no	no	no	no
helmet required	no	no	no	no	no	no
use of cycling path allowed	yes	yes	yes	yes	yes	no
liability insurance required	no	> 10 km/h	yes	yes	yes	> 6 km/h
notes		no definition found	may not be disability vehicle! no front wheel drive			only for physically impaired

https://www.astra.admin.ch/astra/de/home/themen/verkehrsregeln/vorschriften-trendfahrzeuge.html
https://www.fedlex.admin.ch/eli/cc/1995/4425_4425_4425/de
https://www.fedlex.admin.ch/eli/cc/1995/3997_3997_3997/de
https://www.rijksoverheid.nl/onderwerpen/voertuigen-op-de-weg/bijzondere-bromfiets/elektrische-step
https://www.rijksoverheid.nl/onderwerpen/voertuigen-op-de-weg/gehandicaptenvoertuig
https://wetten.overheid.nl/BWBR0035848/2019-05-02
https://wetten.overheid.nl/BWBR0025798/2023-01-01#Hoofdstuk1_Afdeling1_Artikel1.1
https://open.overheid.nl/documenten/ronl-c0389c5c3c23e00e294e6706a74da363c0480780/pdf
https://wissenswertes.ergoflix.de/elektrischer-rollstuhl-strassenverkehr/
https://www.gesetze-im-internet.de/ekfv/BJNR075610019.html
https://www.tcs.ch/de/camping-reisen/reiseinformationen/wissenswertes/fahrzeugvorschriften/elektro-scooter-europa-regeln.php#anchor_europa

Fig. 5: Full overview of legal information.

B. Literature

1) SCOPUS search terms: The following search terms have been used in SCOPUS to gather records. Relevant records were screened for related sources as well.

- `TITLE-ABS-KEY ( mobility AND scooter AND safety AND NOT ( "e-scooter" OR "electric scooter" OR "e scooter" OR "electric-scooter" OR "kick scooter" ) )`
- `TITLE-ABS-KEY ( mobility AND scooter AND accessibility AND NOT ( "ELECTRIC SCOOTER" OR "E-SCOOTER" OR "ELECTRIC-SCOOTER" OR "E SCOOTER" ) )`
- `TITLE-ABS-KEY ( mobility AND scooter AND manoeuvrability AND NOT ( "ELECTRIC SCOOTER" OR "E-SCOOTER" OR "ELECTRIC-SCOOTER" OR "E SCOOTER" ) )`

2) Koontz e.a. tests: See Figure 6.

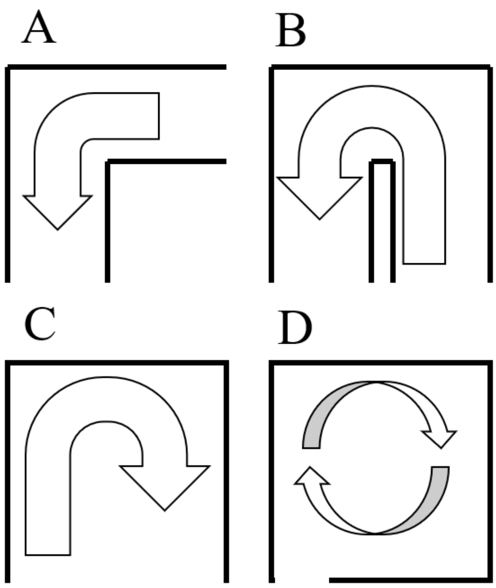


Fig. 6: Tests performed in Koontz e.a. [21]: 90° turn in hallway (A), 180° turn in hallway (B), U-turn (C) and 360° turn (D).

C. Steep street in Zürich

See Figure 7.

D. Test requirement values

Below are some brief motivations for the values that have been picked as required values for each test:

Test	Unit	Required	Motivation
1	Energy (J)	> 0	Guarantees that turn can be made.
2	Angle (-)	< 0.45	Impact angle for an object that's 1/6 the diameter of the wheel.
3	Speed (km/h)	> 3	Slow walking pace, should thus be the minimum.
4	Accel. (m/s	> 1	To get to slow walking pace within seconds. Can always be limited with control.
5	Decel.	> 4	Roughly 0.4 g, to match te braking performance of a car.
6	Length (m)	< 2	Approximate length of the clear floor area as measured by the author on a Bombardier Flexity tram in Zürich.
7	Width (m)	< 1.5	Based on minimum sidewalk width in Zürich, including potential obstacles [35].
8	Width (m)	< 5	Wide enough to turn around in on a normal street [36] or in a large interior space.
9	Distance (m)	> 0	To guarantee that the average person can sit with proper posture.
10	Width (m)	< 0.9	Wide enough to navigate interior hallways according to German standards [37].

TABLE V: Brief motivations for test bounds.

E. Walker shortening

The shortening factor is based on 2 LEGO® prototypes that have been constructed. As can be seen by the values on the ruler in Figure 8, the shortening factor can be estimated as $(8/12 + 7.5/11.5)/2 = 0.65$.

F. Design parameters

1) Wheel width: The wheel width had been changed to $t = 0.040\,\mathrm{m}$ and $t = 0.120\,\mathrm{m}$ to verify if criterion **C.2** did indeed apply, and the change did not affect the final rankings. Thus, it can be said that changing the criterion within its reasonable bounds, as both values are reasonable wheel widths, only affected test results marginally (if at all), making criterion **C.2** valid.

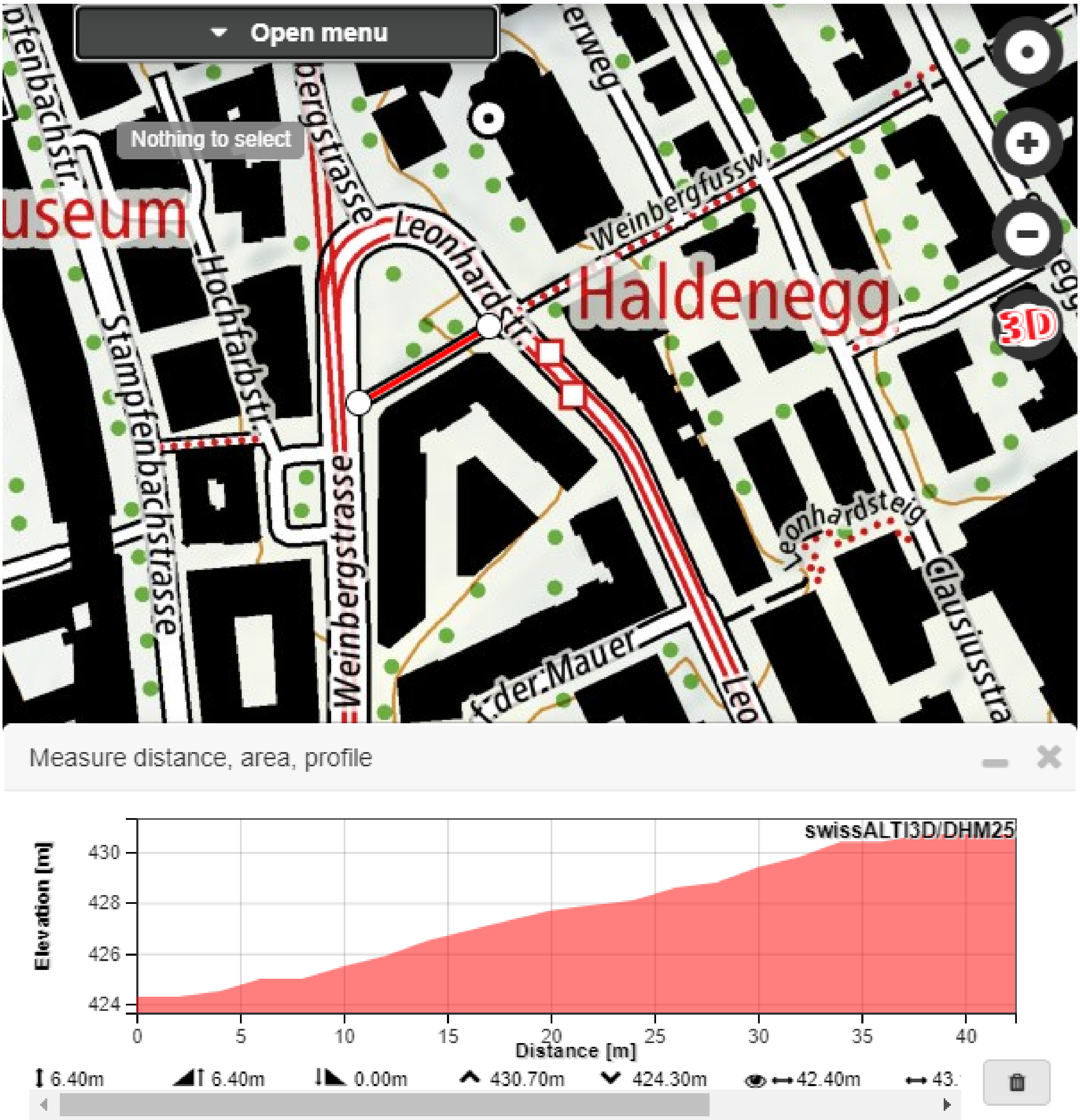


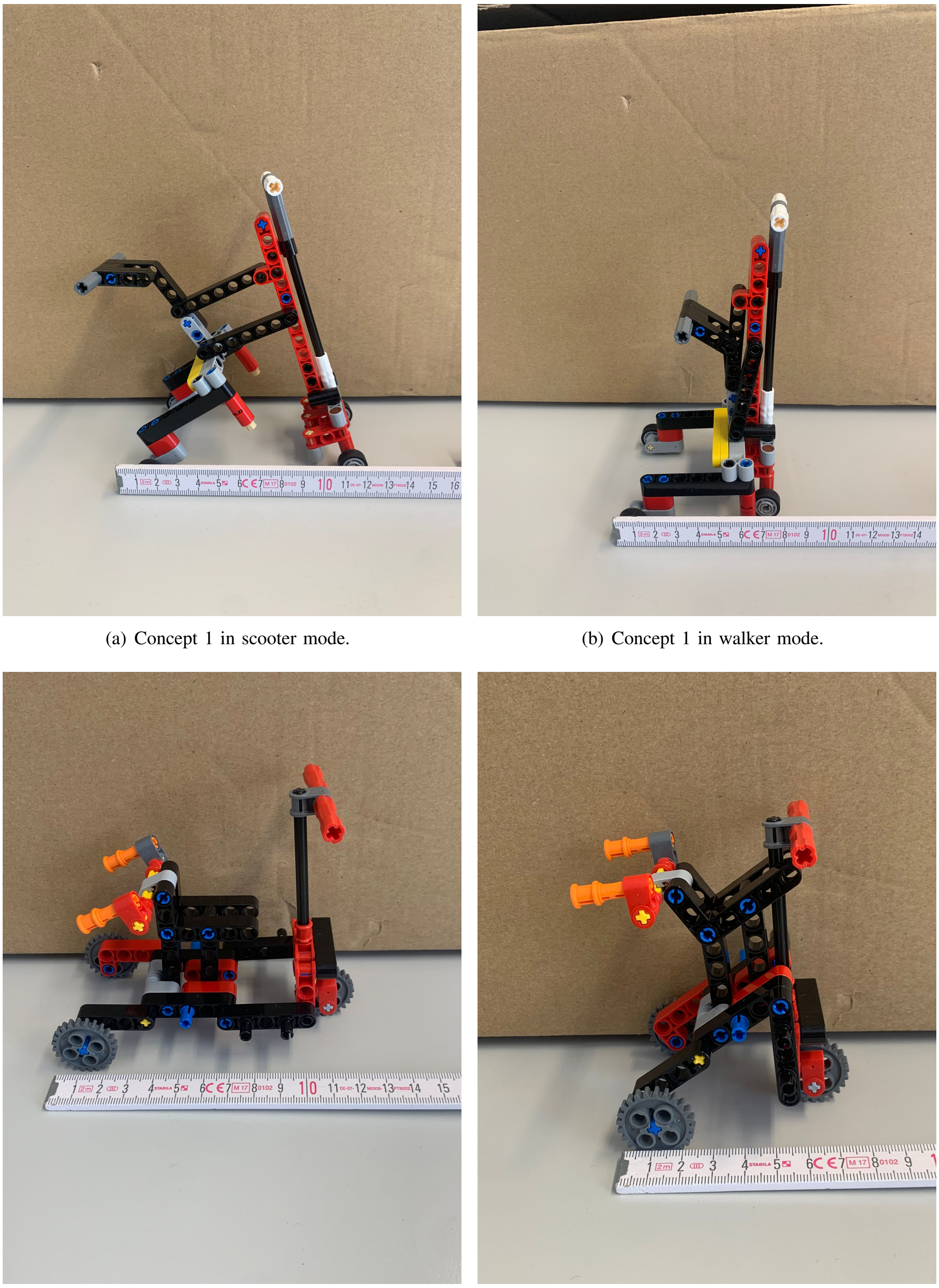
Fig. 7: A steep street in Zürich: the Weinbergfussweg at Haldenegg. Plugging in the numbers yields a gradient of $100\% \cdot (6.40/42.40) = 15.09\%$. Image made in Swisstopo online map viewer.

2) Rider CoM vertical position: The average height in Germany, with data for men and women respectively, is $(1750\,\mathrm{mm} + 1625\,\mathrm{mm})/2 = 1687.5\,\mathrm{mm} \approx 1.69\,\mathrm{m}$ [30]. According to Winter (2009) [28], the distance between the hip joint and buttocks (seat) thus is $(0.530 - 0.520) \cdot 1.69\,\mathrm{m} = 0.0169\,\mathrm{m}$. The height from the hip joint to the CoM then is $0.1 \cdot 1.69\,\mathrm{m} = 0.169\,\mathrm{m}$ [29]. Thus the height from the buttocks (seat) to CoM is $h_\mathrm{b} = 1.69\,\mathrm{m} + 0.169\,\mathrm{m} = 0.1859\,\mathrm{m} \approx 0.19\,\mathrm{m}$.

G. Test detail figures

1) 3D model figure: See Figure 9.

2) Test 3 diagram: See Figure 10.



(a) Concept 1 in scooter mode.

(b) Concept 1 in walker mode.

(c) Concept 2 in scooter mode.

(d) Concept 2 in walker mode.

Fig. 8: LEGO®prototypes used for estimating the shortening factor to walker mode. Top prototype built by the author, bottom one by student Amanda Felouzis. Photos used with permission from Amanda Felouzis.

H. Test 3 calculations

Let us start with defining the speed and acceleration at the CoM:

$$\frac{v_{\mathrm{C}}}{s} = \frac{v}{r}$$
$$\rightarrow v_{\mathrm{C}} = \frac{s}{r}v = fv$$
$$\rightarrow a_C = \frac{v_{\mathrm{C}}^2}{s} = \frac{f^2v^2}{s}$$
$$\rightarrow a_{\mathrm{C,x'}} = a_{\mathrm{C}}cos(\varphi) = \frac{f^2v^2}{s}\cos(\varphi)$$

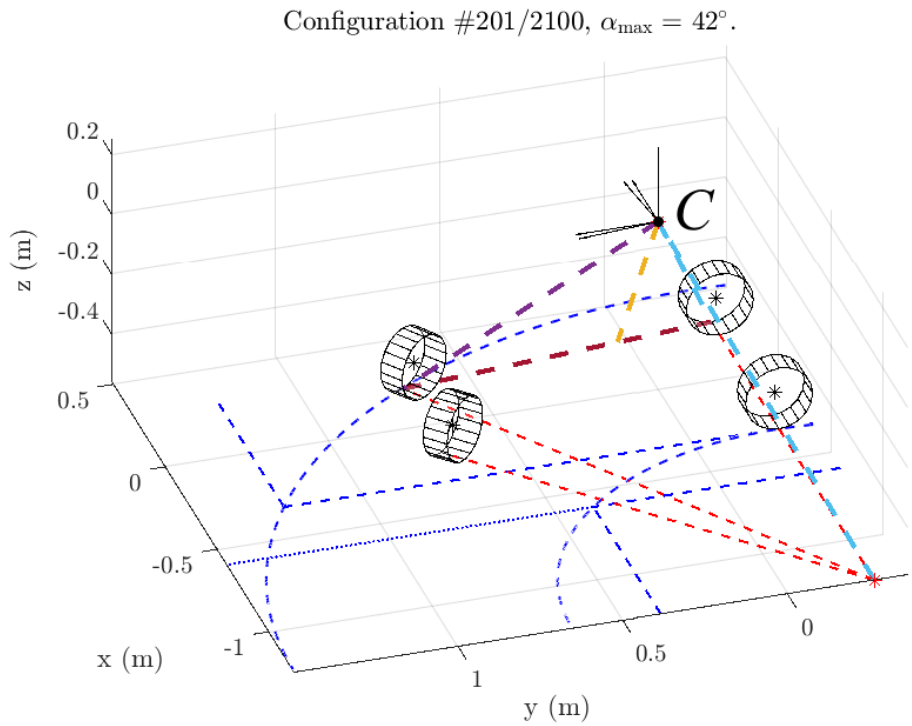


Fig. 9: 3D view of the scooter model, for an arbitrary configuration. Note that the scooter body is not visually represented, only the wheels and the overall CoM with mass m_C in point C.

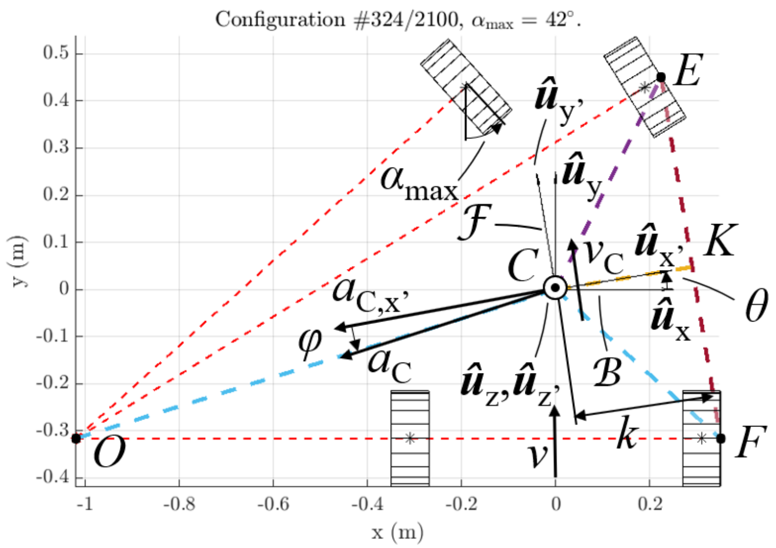


Fig. 10: Top view of the 3D scooter model, with annotations of relevant speeds, accelerations, angles and coordinate systems.

Combining that with Equation 8 through Equation 10, it follows that:

$$(\text{Eq. } 8) \rightarrow H_1 = -m a_{C,x'}$$
$$(\text{Eq. } 9) \rightarrow mg = V$$
$$(\text{Eq. } 10) \rightarrow kV = h_C H_1$$
$$\rightarrow gk = h_C a_{C,x'} = h_C \cos(\varphi) \frac{f^2 v^2}{s}$$
$$\rightarrow v_{tip} = v = \sqrt{\frac{gks}{h_C f^2 \cos(\varphi)}} \quad (\text{Eq. } 11)$$

1) Test 2 diagram: See Figure 11.

2) Test 6 diagram: For the calculation of the clear floor area for test 6, consider Figure 12. The clear floor area is drawn in the rectangle around the scooter, with the length along the y-axis defining the length of the clear floor area.

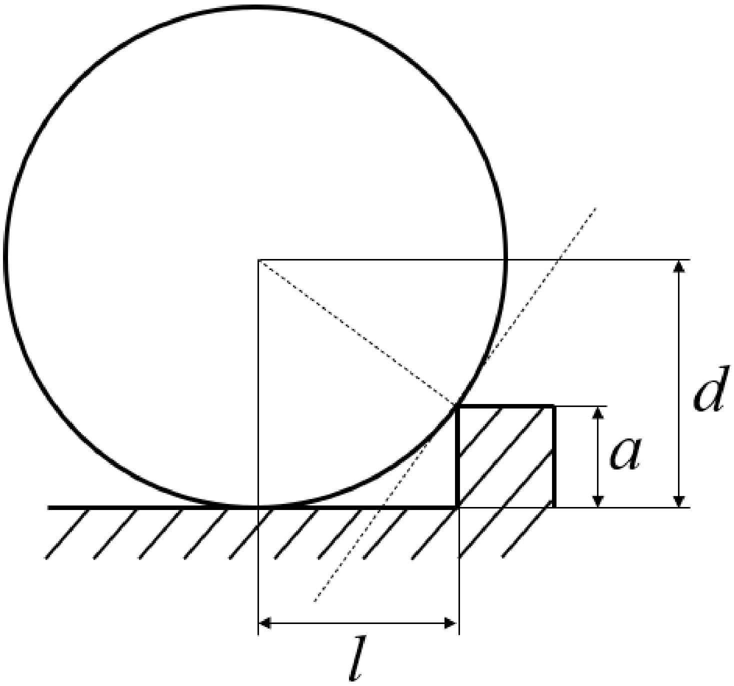


Fig. 11: Diagram for calculating the angle at which the tangent on the wheel strikes an object as part of test 2. d is the wheel diameter, a the obstacle height and l the distance between the contact point of the wheel and the base of the obstacle.

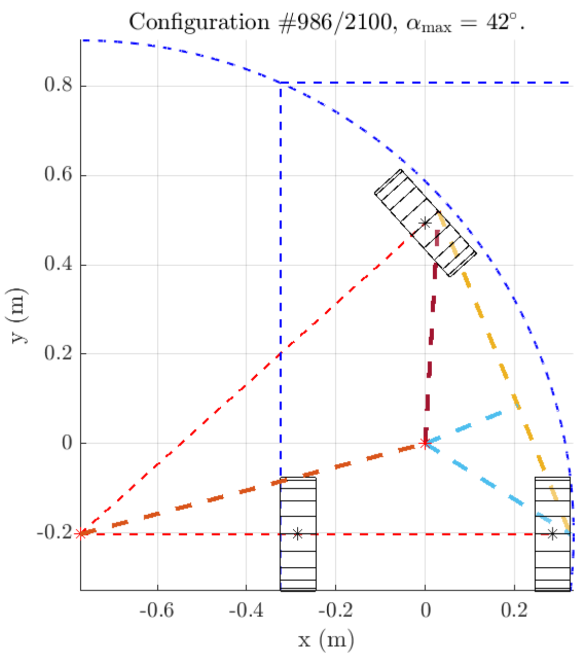


Fig. 12: Top view of a certain configuration, with relevant lines for the clear floor area (test 6). Note how the outer edge of the swept path is defined by the rear wheel, and how the edge in turn defines the length of the clear floor area.

3) Test 7 diagram: See Figure 13.

4) Test 9 calculations and figure: See Figure 14 for an overview of all relevant parameters. Here, $p_b = 0.060h$ [29], $l_{\text{leg}} = (0.530 - 0.285)h$ [28], $l_{\text{a2t}} = \frac{4}{5} l_{\text{foot}}$ (author's estimation), $l_{\text{foot}} = 0.152h$ [28] and $h = 1.69\,\text{m}$ (mean height in Germany [30]). Adding and substracting everything according to Figure 14 yields l_{t2w}.

5) Test 10 diagram: See Figure 15.

I. Weights vector

See below.

$$\boldsymbol{w} = \begin{bmatrix} 0.1787 & 0.1508 & 0.1503 & 0.0986 & 0.0396 & 0.0766 & 0.0515 & 0.0553 & 0.0858 & 0.1127 \end{bmatrix}^{\mathrm{T}}$$

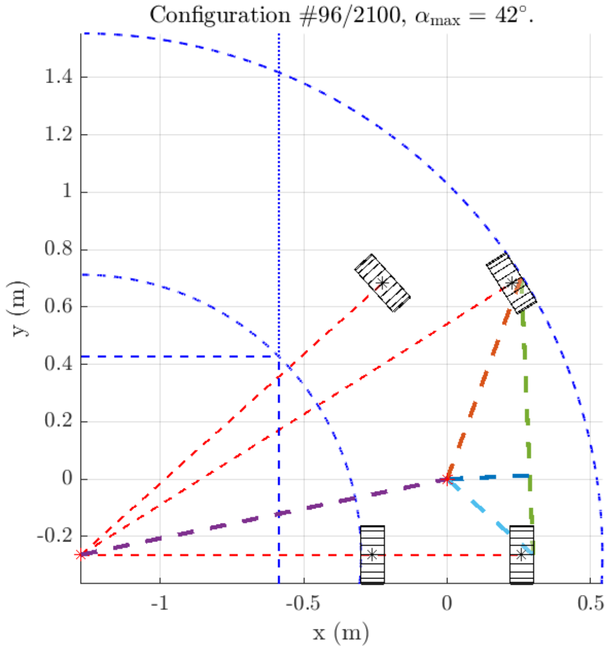


Fig. 13: Top view of a certain configuration, with relevant lines for the hallway (test 7). Note how in this case, it is the front wheel that defines the outer edge of the swept path.

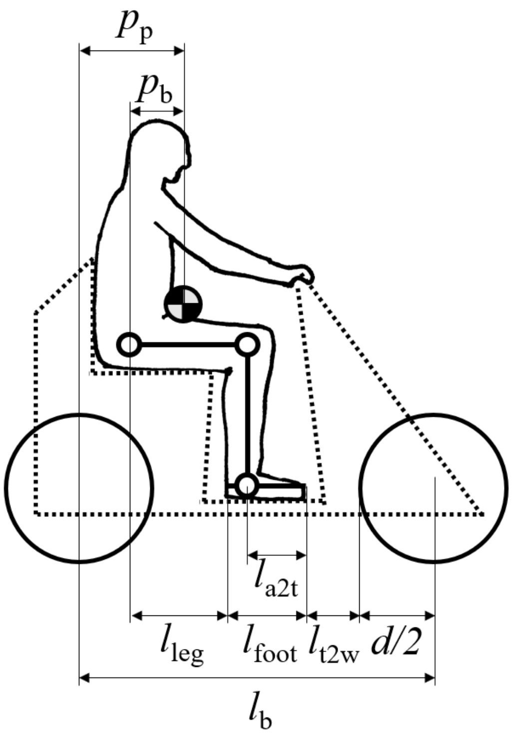


Fig. 14: Diagram for all relevant parameters for test 9. The target value is l_{t2w}.

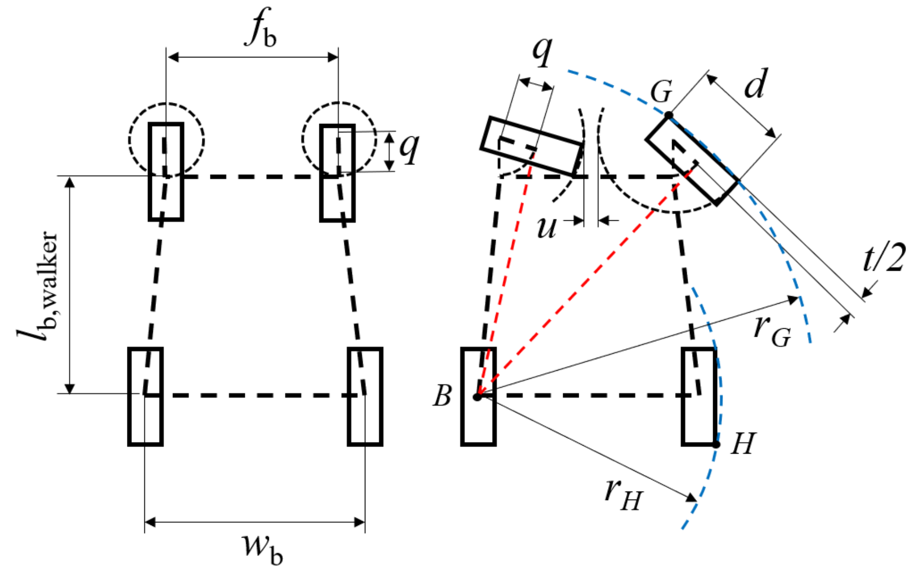


Fig. 15: Diagram for test 10: top view of the device in walker mode. On the left, the device can be seen with the wheels straight, and on the right while making a turn around the middle of the left rear wheel. Note how, unlike in scooter mode, the front wheels do not rotate about their centers, but about a point with an offset q, because they are functioning as caster wheels.

Appendix B
Verification

A. Tests 1-3

The results of test 1 have been verified using Huston e.a. (1982) [38], by applying the following values from their paper.

- $l_{\mathrm{b}} = 2\,\mathrm{m}$
- $f_{\mathrm{b}} = 1.25\,\mathrm{m}$
- $h_{\mathrm{C}} = 0.6\,\mathrm{m}$
- $r = 100\,\mathrm{m}$

The results are as seen in Table VI, where it is clear that the calculations were correct. Slight changes in the decimal values in for four wheels can be due to the different definitions of "speed", since the author defines it at the middle of the rear axis, whereas Huston e.a. define it at the CoM. Since test 1, because of the iterative method, uses the same Equation 11, but for different values regarding the geometry, the author knew that the results for test 1 must also be correct if the different geometry values are carefully checked. Therefore, several 3D plots were made to test if the calculated values for s, h_{C} and φ were correct. Once that was verified, the results for test 1 were thus verified.

$p_{\mathrm{C}} = l_{\mathrm{b}}/4$	Four wheels (Huston e.a.)	Three wheels (Huston e.a.)	Four wheels (author's results)	Three wheels (author's results)
$p_{\mathrm{C}} = l_{\mathrm{b}}/4$	32.0	27.7	31.9682	27.6636
$p_{\mathrm{C}} = l_{\mathrm{b}}/3$	32.0	26.1	31.9688	26.0752
$p_{\mathrm{C}} = l_{\mathrm{b}}/2$	32.0	22.6	31.9706	22.5714
$p_{\mathrm{C}} = 2l_{\mathrm{b}}/3$	32.0	18.5	31.9730	18.4213
$p_{\mathrm{C}} = 3l_{\mathrm{b}}/4$	32.0	16.0	31.9745	15.9499

TABLE VI: Comparison of calculated values for test 1 compared to Huston e.a. [38]

Test 2 was a very simply equation that was easily verified by plugging in different values.

B. Tests 4-5

Since tests 4 and 5 rely on what is essentially the same equation, only test 4 was verified. This was done by first setting κ to zero. In that case, $\frac{h_{\mathrm{C}}}{p_{\mathrm{C}}} = \frac{g}{a_{\mathrm{Y}}}$ should be true. This was verified for several configurations. Additionally, there is a special case where $\kappa = \mathrm{atan}\left(\frac{p_{\mathrm{C}}}{h_{\mathrm{C}}}\right)$, then $a_{\mathrm{Y}} = 0$. What happens here, is that κ has an angle such that the CoM is right above the contact point of the wheel, meaning that the acceleration must be zero. This condition was also verified for several configurations.

C. Tests 6-10

Since these tests are all only dependent on geometry, each test was verified by carefully cross referencing the output values with (3D) plots in MATLAB®.

Appendix C
Results

A. *Test results per optimizable design parameter*

1) Wheel diameter: See Figure 16.

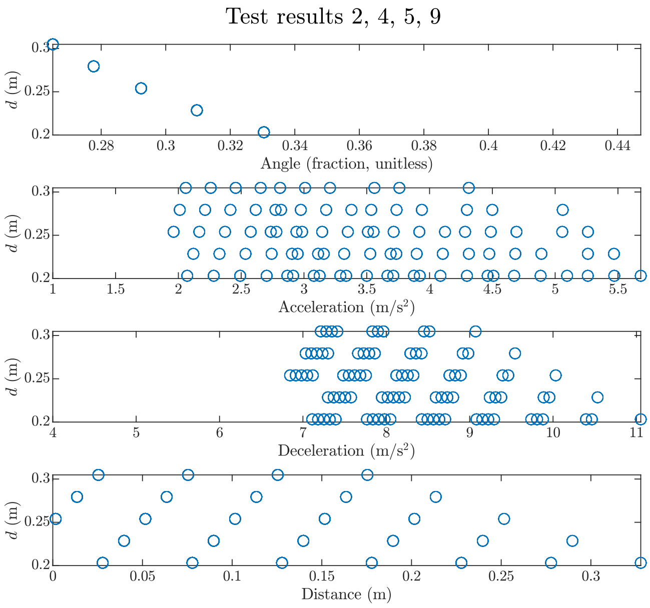


Fig. 16: Test results 2, 4, 5 and 9 for wheel diameter d.

2) Overall width rear: See Figure 17.

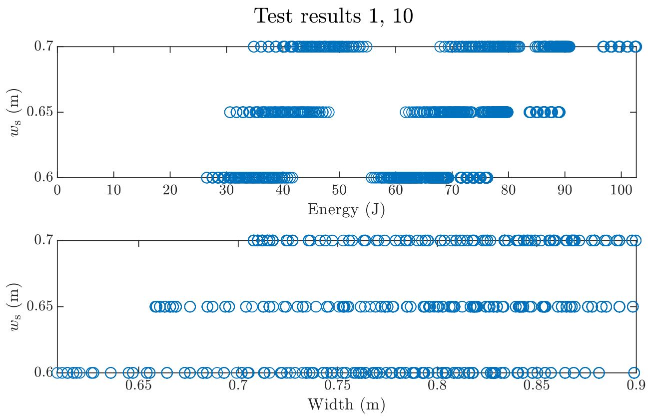


Fig. 17: Test results 1 and 10 for overall width at the rear w_s.

3) Overall width front: See Figure 18.
4) Overall length: See Figure 19.
5) Longitudinal rider CoM position: See Figure 20.

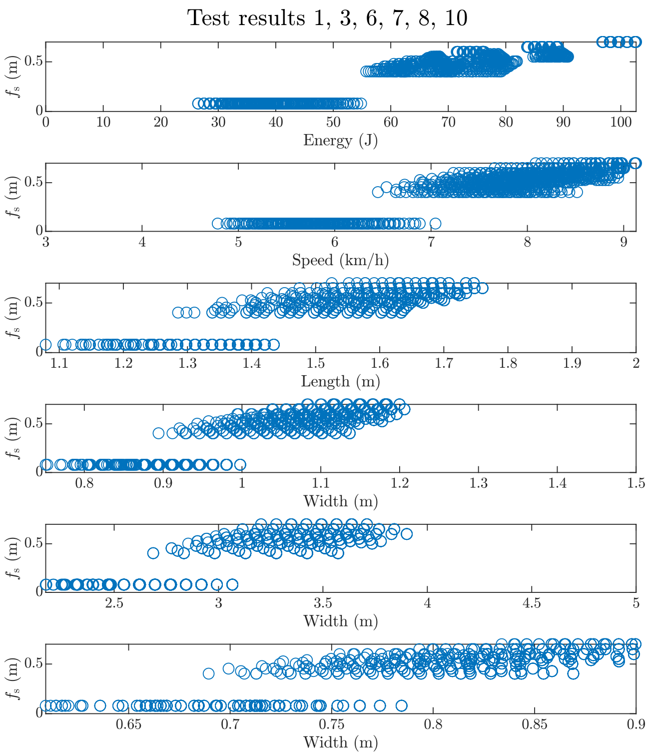


Fig. 18: Test results 1, 3, 6, 7, 8 and 10 for the overall width in the front f_s.

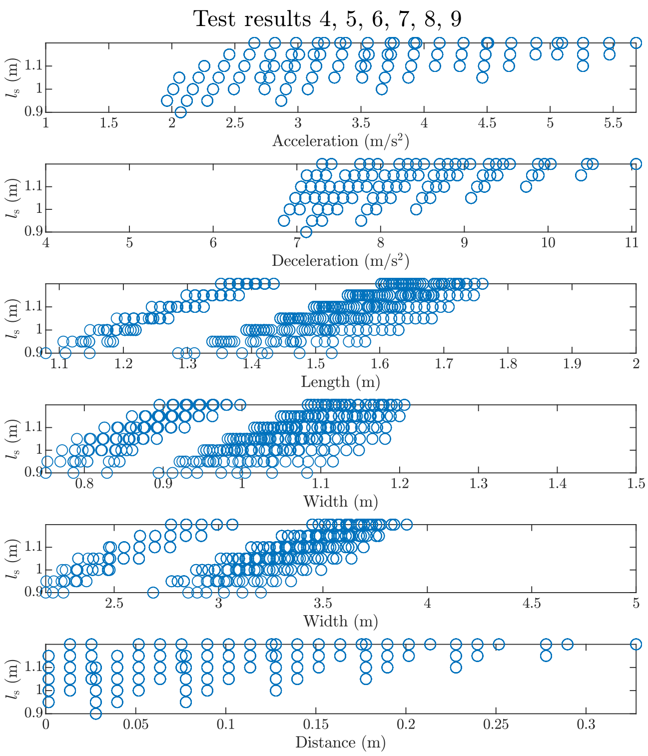


Fig. 19: Test results 4, 5, 6, 7, 8 and 9 for the overall length l_s.

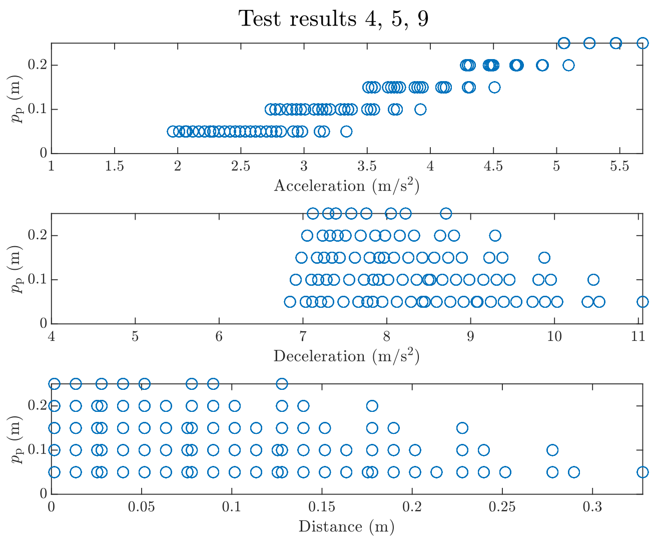


Fig. 20: Test results 4, 5 and 9 for the longitudinal rider CoM position p_{p}.

B. *Full result plots*

See Figures 21 through 23.

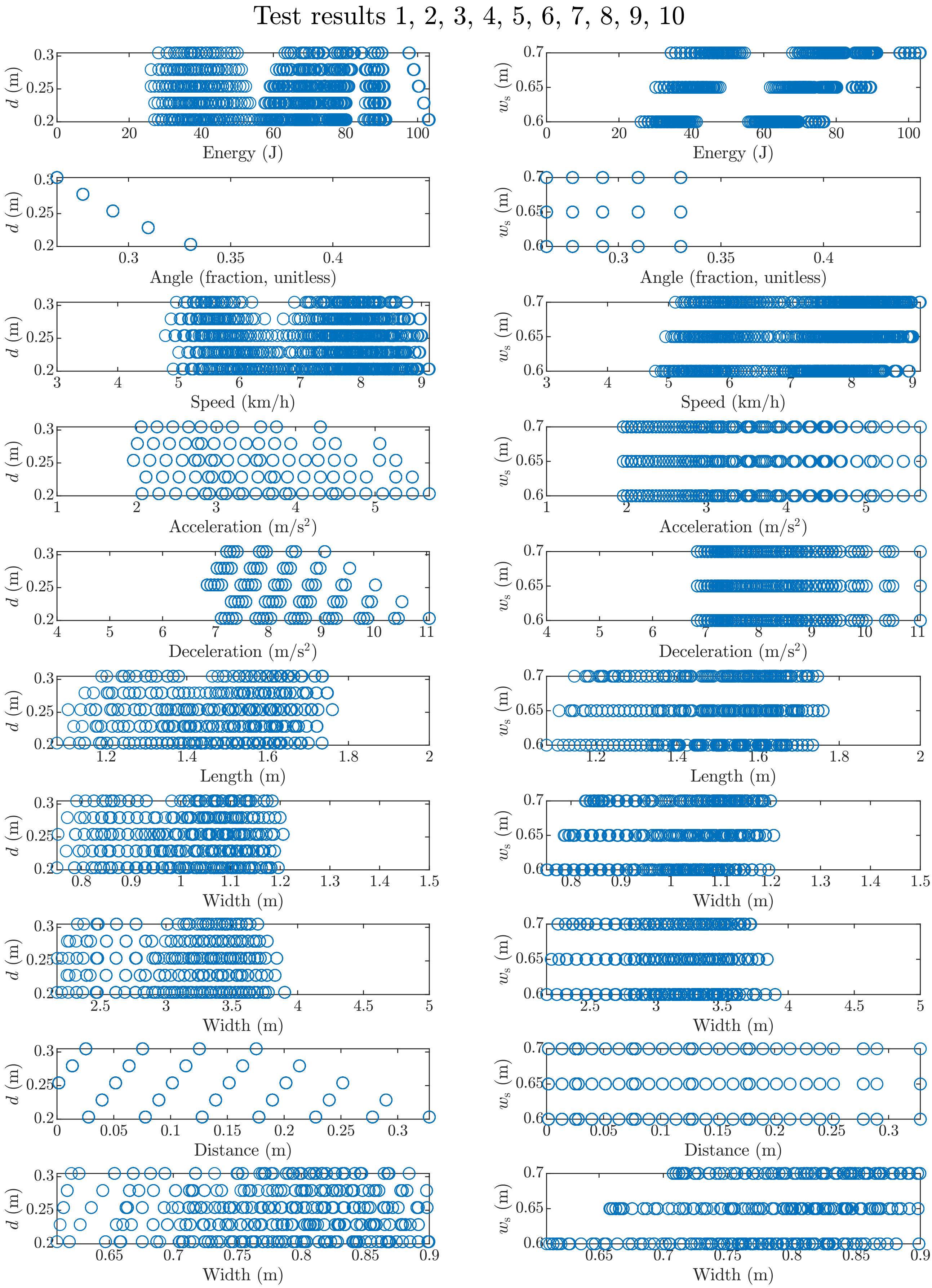


Fig. 21: All test results for d and w_s.

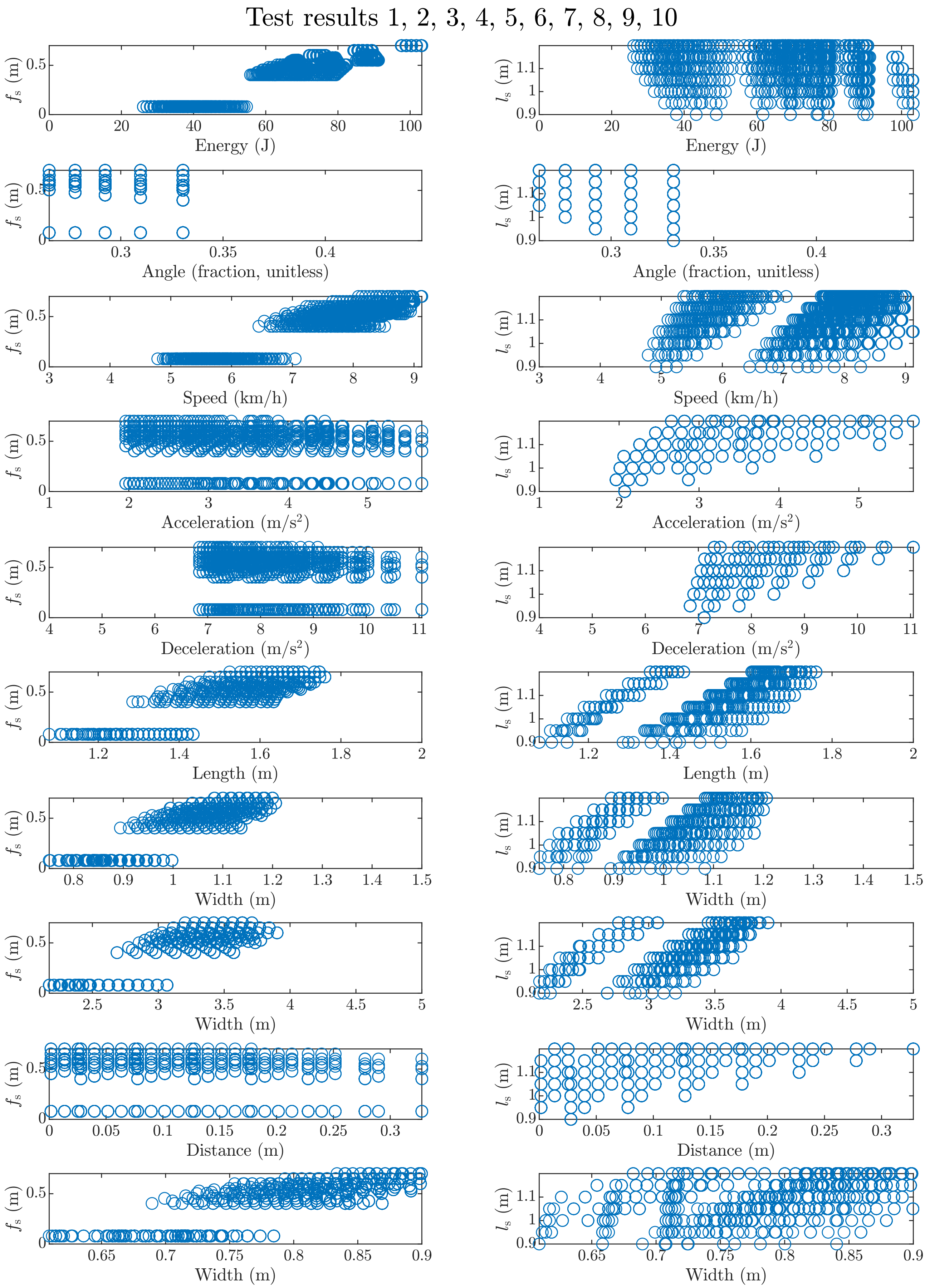


Fig. 22: All test results for f_s and l_s.

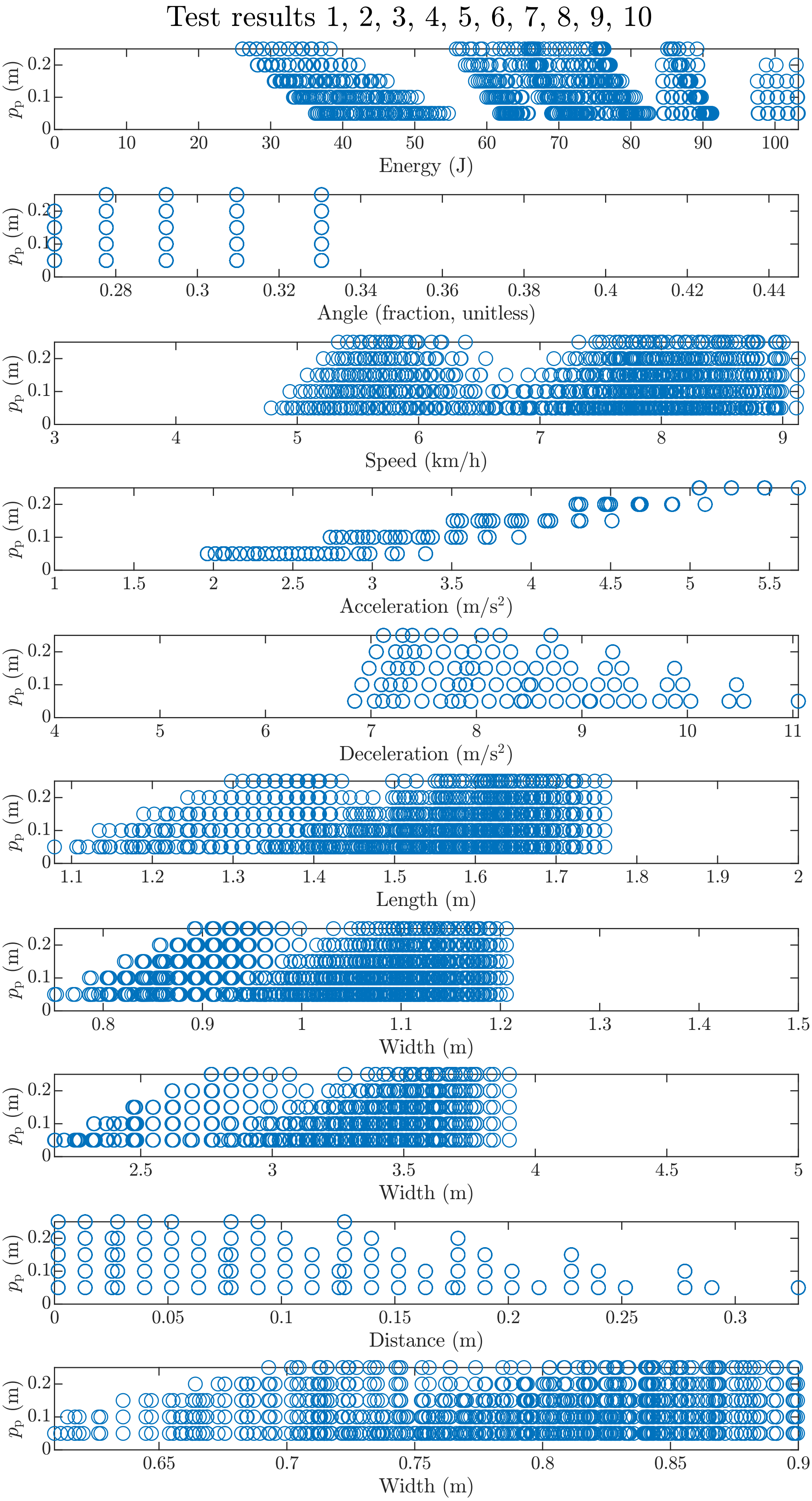
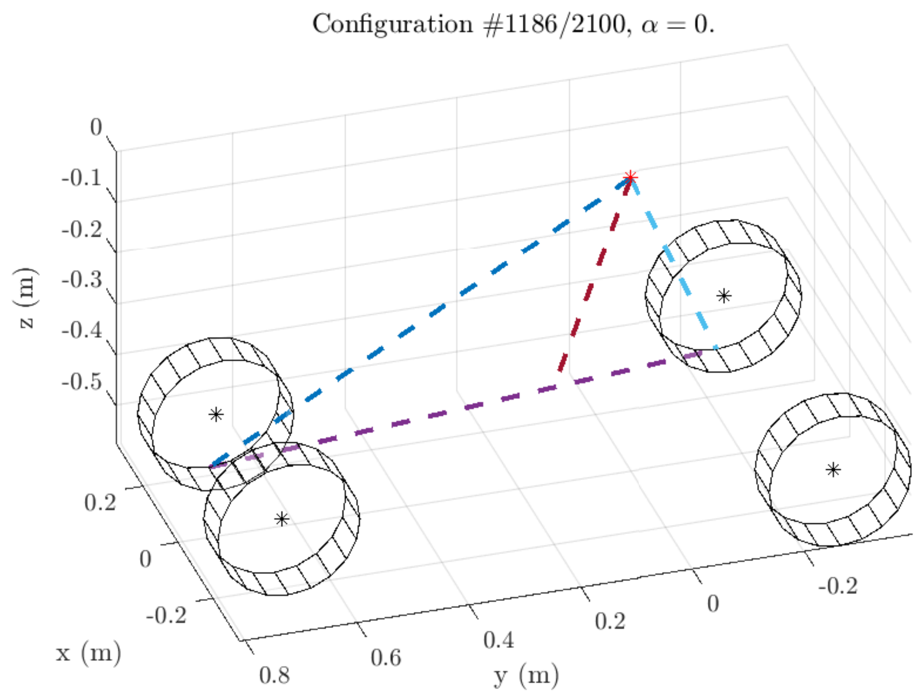


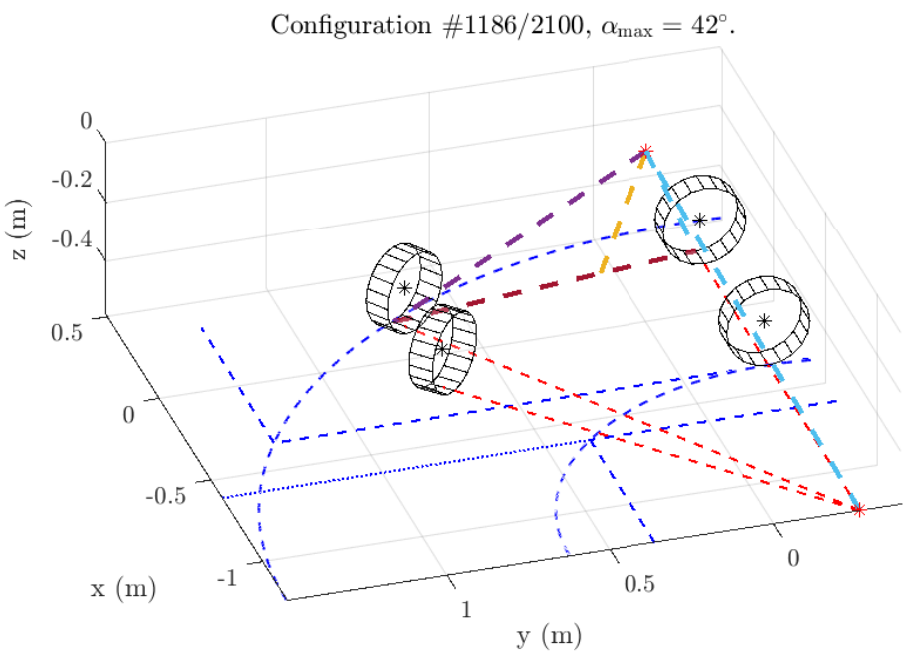
Fig. 23: All test results for and p_{p}.

C. Best concept in visualization

See Figure 24



(a) With wheels straight.



(b) With wheels at maximum angle.

Fig. 24: 3D representation of the concept with the optimal set of design parameters.

D. *Top 50 configurations tables*

See Table VII, Table VIII and Table IX.

Rank	Configuration #	d (m)	w_s (m)	f_s (m)	l_s (m)	pp (m)
1	1186	10	0.700	0.454	1.200	0.050
2	1221	10	0.700	0.577	1.200	0.050
3	766	9	0.700	0.429	1.200	0.050
4	1606	11	0.700	0.479	1.200	0.050
5	1641	11	0.700	0.590	1.200	0.050
6	346	8	0.700	0.403	1.200	0.050
7	2026	12	0.700	0.505	1.200	0.050
8	2061	12	0.700	0.602	1.200	0.050
9	1151	10	0.700	0.080	1.200	0.050
10	1571	11	0.700	0.080	1.200	0.050
11	1222	10	0.700	0.577	1.200	0.100
12	1181	10	0.700	0.454	1.150	0.050
13	761	9	0.700	0.429	1.150	0.050
14	731	9	0.700	0.080	1.200	0.050
15	796	9	0.700	0.564	1.150	0.050
16	1216	10	0.700	0.577	1.150	0.050
17	1187	10	0.700	0.454	1.200	0.100
18	1642	11	0.700	0.590	1.200	0.100
19	1601	11	0.700	0.479	1.150	0.050
20	767	9	0.700	0.429	1.200	0.100
21	1607	11	0.700	0.479	1.200	0.100
22	1046	10	0.650	0.454	1.200	0.050
23	1636	11	0.700	0.590	1.150	0.050
24	1081	10	0.650	0.552	1.200	0.050
25	376	8	0.700	0.552	1.150	0.050
26	1466	11	0.650	0.479	1.200	0.050
27	626	9	0.650	0.429	1.200	0.050
28	661	9	0.650	0.539	1.200	0.050
29	341	8	0.700	0.403	1.150	0.050
30	2062	12	0.700	0.602	1.200	0.100
31	1991	12	0.700	0.080	1.200	0.050
32	1501	11	0.650	0.565	1.200	0.050
33	2027	12	0.700	0.505	1.200	0.100
34	1116	10	0.650	0.650	1.200	0.050
35	2021	12	0.700	0.505	1.150	0.050
36	311	8	0.700	0.080	1.200	0.050
37	1223	10	0.700	0.577	1.200	0.150
38	1671	11	0.700	0.700	1.150	0.050
39	347	8	0.700	0.403	1.200	0.100
40	1146	10	0.700	0.080	1.150	0.050
41	1886	12	0.650	0.505	1.200	0.050
42	1536	11	0.650	0.650	1.200	0.050
43	2056	12	0.700	0.602	1.150	0.050
44	1431	11	0.650	0.080	1.200	0.050
45	1643	11	0.700	0.590	1.200	0.150
46	241	8	0.650	0.527	1.200	0.050
47	1217	10	0.700	0.577	1.150	0.100
48	797	9	0.700	0.564	1.150	0.100
49	1011	10	0.650	0.080	1.200	0.050
50	206	8	0.650	0.403	1.200	0.050
51	1921	12	0.650	0.577	1.200	0.050
52	726	9	0.700	0.080	1.150	0.050
53	1851	12	0.650	0.080	1.200	0.050
54	1176	10	0.700	0.454	1.100	0.050
55	756	9	0.700	0.429	1.100	0.050
56	1182	10	0.700	0.454	1.150	0.100
57	1117	10	0.650	0.650	1.200	0.100
58	791	9	0.700	0.564	1.100	0.050
59	1082	10	0.650	0.552	1.200	0.100
60	1637	11	0.700	0.590	1.150	0.100
61	1211	10	0.700	0.577	1.100	0.050
62	1672	11	0.700	0.700	1.150	0.100
63	2091	12	0.700	0.700	1.150	0.050
64	1188	10	0.700	0.454	1.200	0.150
65	762	9	0.700	0.429	1.150	0.100
66	591	9	0.650	0.080	1.200	0.050
67	1041	10	0.650	0.454	1.150	0.050
68	662	9	0.650	0.539	1.200	0.100
69	1956	12	0.650	0.650	1.200	0.050
70	1047	10	0.650	0.454	1.200	0.100

TABLE VII: Overview of the optimizable parameters for the top 70 configurations.

Rank	Configuration #	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10
1	1186	80.57	0.29	8.32	2.99	10.03	1.64	1.13	3.51	0.25	0.85
2	1221	90.17	0.29	8.88	2.99	10.03	1.72	1.19	3.72	0.25	0.89
3	766	79.89	0.31	8.42	3.16	10.53	1.64	1.13	3.54	0.29	0.86
4	1606	81.26	0.28	8.22	2.82	9.54	1.64	1.12	3.48	0.21	0.84
5	1641	89.82	0.28	8.72	2.82	9.54	1.72	1.18	3.67	0.21	0.88
6	346	79.22	0.33	8.52	3.34	11.05	1.63	1.14	3.57	0.33	0.87
7	2026	81.94	0.26	8.12	2.65	9.07	1.65	1.12	3.45	0.18	0.83
8	2061	89.48	0.26	8.56	2.65	9.07	1.72	1.17	3.62	0.18	0.87
9	1151	52.44	0.29	6.63	2.99	10.03	1.41	0.96	2.92	0.25	0.74
10	1571	51.25	0.28	6.42	2.82	9.54	1.39	0.95	2.85	0.21	0.72
11	1222	88.99	0.29	8.85	3.55	9.46	1.72	1.19	3.72	0.20	0.89
12	1181	80.35	0.29	8.06	2.78	9.39	1.59	1.09	3.36	0.20	0.83
13	761	79.65	0.31	8.16	2.95	9.89	1.58	1.10	3.39	0.24	0.83
14	731	53.64	0.31	6.84	3.16	10.53	1.42	0.98	2.99	0.29	0.76
15	796	90.47	0.31	8.78	2.95	9.89	1.68	1.16	3.62	0.24	0.88
16	1216	90.11	0.29	8.62	2.78	9.39	1.67	1.15	3.57	0.20	0.87
17	1187	78.38	0.29	8.25	3.55	9.46	1.64	1.13	3.51	0.20	0.85
18	1642	88.73	0.28	8.69	3.38	8.98	1.72	1.18	3.67	0.16	0.88
19	1601	81.06	0.28	7.95	2.62	8.92	1.59	1.09	3.33	0.16	0.82
20	767	77.54	0.31	8.34	3.74	9.96	1.64	1.13	3.54	0.24	0.86
21	1607	79.24	0.28	8.16	3.38	8.98	1.64	1.12	3.48	0.16	0.84
22	1046	72.15	0.29	8.10	2.99	10.03	1.63	1.11	3.51	0.25	0.83
23	1636	89.76	0.28	8.46	2.62	8.92	1.67	1.14	3.52	0.16	0.86
24	1081	79.27	0.29	8.54	2.99	10.03	1.69	1.16	3.67	0.25	0.86
25	376	90.84	0.33	8.94	3.12	10.40	1.68	1.17	3.67	0.28	0.89
26	1466	72.83	0.28	8.00	2.82	9.54	1.63	1.11	3.48	0.21	0.82
27	626	71.47	0.31	8.19	3.16	10.53	1.62	1.11	3.54	0.29	0.84
28	661	79.56	0.31	8.69	3.16	10.53	1.70	1.17	3.73	0.29	0.88
29	341	78.96	0.33	8.26	3.12	10.40	1.58	1.10	3.42	0.28	0.84
30	2062	88.49	0.26	8.54	3.21	8.52	1.72	1.17	3.62	0.13	0.87
31	1991	50.06	0.26	6.21	2.65	9.07	1.38	0.93	2.77	0.18	0.72
32	1501	78.99	0.28	8.39	2.82	9.54	1.69	1.15	3.62	0.21	0.85
33	2027	80.10	0.26	8.07	3.21	8.52	1.65	1.12	3.45	0.13	0.83
34	1116	86.31	0.29	8.99	2.99	10.03	1.76	1.21	3.84	0.25	0.90
35	2021	81.77	0.26	7.85	2.46	8.45	1.60	1.08	3.30	0.13	0.81
36	311	54.84	0.33	7.05	3.34	11.05	1.43	1.00	3.07	0.33	0.78
37	1223	87.81	0.29	8.82	4.12	8.89	1.72	1.19	3.72	0.15	0.89
38	1671	98.23	0.28	8.97	2.62	8.92	1.75	1.20	3.71	0.16	0.90
39	347	76.71	0.33	8.44	3.92	10.47	1.63	1.14	3.57	0.28	0.87
40	1146	51.71	0.29	6.35	2.78	9.39	1.35	0.93	2.77	0.20	0.72
41	1886	73.51	0.26	7.91	2.65	9.07	1.63	1.10	3.45	0.18	0.82
42	1536	85.07	0.28	8.78	2.82	9.54	1.75	1.19	3.77	0.21	0.88
43	2056	89.43	0.26	8.30	2.46	8.45	1.66	1.13	3.47	0.13	0.85
44	1431	44.99	0.28	6.21	2.82	9.54	1.38	0.93	2.85	0.21	0.71
45	1643	87.65	0.28	8.67	3.94	8.42	1.72	1.18	3.67	0.11	0.88
46	241	79.86	0.33	8.85	3.34	11.05	1.70	1.18	3.78	0.33	0.89
47	1217	88.87	0.29	8.59	3.35	8.83	1.67	1.15	3.57	0.15	0.87
48	797	89.13	0.31	8.75	3.53	9.31	1.68	1.16	3.62	0.19	0.88
49	1011	46.03	0.29	6.41	2.99	10.03	1.39	0.95	2.92	0.25	0.73
50	206	70.79	0.33	8.28	3.34	11.05	1.62	1.12	3.57	0.33	0.85
51	1921	78.72	0.26	8.24	2.65	9.07	1.68	1.14	3.57	0.18	0.84
52	726	52.95	0.31	6.56	2.95	9.89	1.37	0.95	2.84	0.24	0.74
53	1851	43.95	0.26	6.01	2.65	9.07	1.37	0.91	2.77	0.18	0.69
54	1176	80.10	0.29	7.78	2.58	8.76	1.53	1.06	3.21	0.15	0.80
55	756	79.38	0.31	7.89	2.74	9.24	1.53	1.06	3.25	0.19	0.81
56	1182	78.05	0.29	7.99	3.35	8.83	1.59	1.09	3.36	0.15	0.83
57	1117	86.26	0.29	8.99	3.55	9.46	1.76	1.21	3.84	0.20	0.90
58	791	90.38	0.31	8.52	2.74	9.24	1.62	1.13	3.47	0.19	0.86
59	1082	78.36	0.29	8.52	3.55	9.46	1.69	1.16	3.67	0.20	0.86
60	1637	88.62	0.28	8.43	3.18	8.36	1.67	1.14	3.52	0.11	0.86
61	1211	90.03	0.29	8.35	2.58	8.76	1.62	1.12	3.42	0.15	0.85
62	1672	98.18	0.28	8.97	3.18	8.36	1.75	1.20	3.71	0.11	0.90
63	2091	96.87	0.26	8.75	2.46	8.45	1.73	1.18	3.64	0.13	0.88
64	1188	76.21	0.29	8.19	4.12	8.89	1.64	1.13	3.51	0.15	0.85
65	762	77.19	0.31	8.08	3.53	9.31	1.58	1.10	3.39	0.19	0.83
66	591	47.08	0.31	6.62	3.16	10.53	1.41	0.96	2.99	0.29	0.75
67	1041	72.02	0.29	7.84	2.78	9.39	1.57	1.07	3.36	0.20	0.81
68	662	78.56	0.31	8.66	3.74	9.96	1.70	1.17	3.73	0.24	0.88
69	1956	83.85	0.26	8.57	2.65	9.07	1.74	1.17	3.70	0.18	0.87
70	1047	70.46	0.29	8.04	3.55	9.46	1.63	1.11	3.51	0.20	0.83

TABLE VIII: Overview of the test results for the top 70 configurations.

Rank	Configuration #	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Overall
1	1186	78.48	84.99	86.78	42.40	85.54	39.01	49.79	52.71	76.76	17.48	64.48
2	1221	87.83	84.99	95.92	42.40	85.54	29.91	41.78	45.39	76.76	3.34	64.42
3	766	77.82	75.50	88.39	46.09	92.66	39.42	49.18	51.61	88.38	14.25	64.39
4	1606	79.15	93.05	85.16	38.82	78.62	38.57	50.37	53.79	65.13	20.31	64.32
5	1641	87.49	93.05	93.34	38.82	78.62	30.33	43.12	47.19	65.13	7.24	64.20
6	346	77.16	64.10	90.00	49.89	100.00	39.79	48.51	50.48	100.00	10.64	63.98
7	2026	79.82	100.00	83.54	35.34	71.91	38.09	50.89	54.85	53.51	22.75	63.96
8	2061	87.15	100.00	90.76	35.34	71.91	30.74	44.42	48.97	53.51	10.84	63.80
9	1151	51.08	84.99	59.18	42.40	85.54	64.37	71.58	73.60	76.76	53.72	63.74
10	1571	49.92	93.05	55.76	38.82	78.62	65.85	73.81	76.19	65.13	60.41	63.74
11	1222	86.68	84.99	95.40	54.55	77.47	29.91	41.78	45.39	61.50	3.34	63.70
12	1181	78.27	84.99	82.48	38.03	76.50	44.77	54.39	57.92	61.50	25.74	63.60
13	761	77.59	75.50	84.14	41.65	83.49	45.18	53.79	56.83	73.12	22.79	63.53
14	731	52.25	75.50	62.60	46.09	92.66	62.87	69.33	71.00	88.38	46.82	63.52
15	796	88.12	75.50	94.34	41.65	83.49	35.14	44.97	48.77	73.12	7.21	63.52
16	1216	87.77	84.99	91.71	38.03	76.50	35.59	46.32	50.57	61.50	11.17	63.51
17	1187	76.35	84.99	85.69	54.55	77.47	39.01	49.79	52.71	61.50	17.48	63.51
18	1642	86.43	93.05	92.89	50.80	70.67	30.33	43.12	47.19	49.88	7.24	63.50
19	1601	78.96	93.05	80.82	34.51	69.72	44.32	54.94	58.99	49.88	28.29	63.41
20	767	75.53	75.50	87.19	58.43	84.47	39.42	49.18	51.61	73.12	14.25	63.38
21	1607	77.18	93.05	84.19	50.80	70.67	38.57	50.37	53.79	49.88	20.31	63.38
22	1046	70.28	84.99	83.15	42.40	85.54	40.63	52.15	52.71	76.76	22.95	63.33
23	1636	87.43	93.05	89.08	34.51	69.72	36.01	47.63	52.36	49.88	14.83	63.27
24	1081	77.22	84.99	90.41	42.40	85.54	33.42	45.80	46.89	76.76	12.07	63.23
25	376	88.48	64.10	96.97	45.39	90.69	34.68	43.60	46.95	84.75	2.94	63.22
26	1466	70.94	93.05	81.63	38.82	78.62	40.17	52.73	53.79	65.13	26.00	63.21
27	626	69.62	75.50	84.67	46.09	92.66	41.06	51.54	51.61	88.38	19.51	63.20
28	661	77.50	75.50	92.89	46.09	92.66	32.98	44.42	45.07	88.38	7.63	63.16
29	341	76.91	64.10	85.79	45.39	90.69	45.56	53.15	55.72	84.75	19.43	63.14
30	2062	86.19	100.00	90.37	47.15	64.07	30.74	44.42	48.97	38.26	10.84	63.11
31	1991	48.76	100.00	52.34	35.34	71.91	67.32	76.01	78.77	53.51	62.79	63.10
32	1501	76.94	93.05	87.93	38.82	78.62	33.84	47.14	48.70	65.13	16.21	63.07
33	2027	78.03	100.00	82.69	47.15	64.07	38.09	50.89	54.85	38.26	22.75	63.06
34	1116	84.07	84.99	97.68	42.40	85.54	26.02	39.24	40.97	76.76	0.60	63.03
35	2021	79.65	100.00	79.15	31.10	63.13	43.83	55.44	60.03	38.26	30.41	63.02
36	311	53.42	64.10	66.02	49.89	100.00	61.37	67.06	68.39	100.00	39.73	63.01
37	1223	85.53	84.99	94.89	66.71	69.40	29.91	41.78	45.39	46.25	3.34	62.99
38	1671	95.68	93.05	97.37	34.51	69.72	27.48	40.07	45.60	49.88	0.73	62.98
39	347	74.72	64.10	88.69	62.42	91.68	39.79	48.51	50.48	84.75	10.64	62.95
40	1146	50.37	84.99	54.65	38.03	76.50	70.30	76.28	78.87	61.50	63.35	62.91
41	1886	71.60	100.00	80.10	35.34	71.91	39.67	53.25	54.85	53.51	28.63	62.89
42	1536	82.86	93.05	94.25	38.82	78.62	27.36	41.41	43.53	65.13	5.99	62.85
43	2056	87.11	100.00	86.46	31.10	63.13	36.41	48.91	54.13	38.26	18.16	62.84
44	1431	43.82	93.05	52.39	38.82	78.62	67.34	76.17	76.19	65.13	64.42	62.83
45	1643	85.38	93.05	92.44	62.77	62.72	30.33	43.12	47.19	34.63	7.24	62.80
46	241	77.79	64.10	95.37	49.89	100.00	32.52	43.02	43.24	100.00	2.91	62.79
47	1217	86.56	84.99	91.18	50.18	68.43	35.59	46.32	50.57	46.25	11.17	62.79
48	797	86.81	75.50	93.74	53.99	75.29	35.14	44.97	48.77	57.87	7.21	62.78
49	1011	44.84	84.99	55.69	42.40	85.54	65.89	73.94	73.60	76.76	57.61	62.77
50	206	68.96	64.10	86.18	49.89	100.00	41.45	50.87	50.48	100.00	15.69	62.76
51	1921	76.67	100.00	85.45	35.34	71.91	34.24	48.46	50.49	53.51	20.04	62.72
52	726	51.57	75.50	58.14	41.65	83.49	68.79	74.03	76.27	73.12	56.59	62.71
53	1851	42.81	100.00	49.08	35.34	71.91	68.78	78.37	78.77	53.51	71.01	62.70
54	1176	78.02	84.99	78.03	33.66	67.46	50.54	58.97	63.13	46.25	33.78	62.66
55	756	77.32	75.50	79.73	37.22	74.31	50.97	58.40	62.05	57.87	31.12	62.62
56	1182	76.03	84.99	81.38	50.18	68.43	44.77	54.39	57.92	46.25	25.74	62.60
57	1117	84.02	84.99	97.74	54.55	77.47	26.02	39.24	40.97	61.50	0.60	62.60
58	791	88.03	75.50	90.03	37.22	74.31	40.84	49.51	53.95	57.87	15.05	62.59
59	1082	76.33	84.99	89.99	54.55	77.47	33.42	45.80	46.89	61.50	12.07	62.58
60	1637	86.32	93.05	88.62	46.49	61.77	36.01	47.63	52.36	34.63	14.83	62.56
61	1211	87.69	84.99	87.35	33.66	67.46	41.29	50.83	55.74	46.25	18.77	62.56
62	1672	95.63	93.05	97.44	46.49	61.77	27.48	40.07	45.60	34.63	0.73	62.55
63	2091	94.35	100.00	93.77	31.10	63.13	28.83	42.19	48.13	38.26	5.43	62.54
64	1188	74.24	84.99	84.62	66.71	69.40	39.01	49.79	52.71	46.25	17.48	62.54
65	762	75.18	75.50	82.92	53.99	75.29	45.18	53.79	56.83	57.87	22.79	62.50
66	591	45.86	75.50	58.98	46.09	92.66	64.42	71.69	71.00	88.38	50.59	62.50
67	1041	70.15	84.99	79.00	38.03	76.50	46.34	56.75	57.92	61.50	31.40	62.50
68	662	76.52	75.50	92.41	58.43	84.47	32.98	44.42	45.07	73.12	7.63	62.50
69	1956	81.67	100.00	90.82	35.34	71.91	28.70	43.55	46.07	53.51	11.14	62.50
70	1047	68.63	84.99	82.28	54.55	77.47	40.63	52.15	52.71	61.50	22.95	62.48

TABLE IX: Overview of the test scores for the top 70 configurations.

Appendix D
MATLAB®Code

main.m

```matlab
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Main file to run. Run latex_interpreter.m once before first run.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

clc; clearvars; close all
%% ratumed valaue input

tic
% par is a struct containing all design parameters
par.g = 9.81;       % m/s^2, gravitational acceleration (not design pam.)
par.t = 0.080;      % m, wheel width
par.v_max = 15/3.6; % m/s, maximum scooter speed
par.h_s = 0.22;     % m, scooter CoM vertical position, from ground
par.m_s = 35;                   % kg, scooter mrat
par.m_p = 73;                   % kg, rider mrat
par.alpha_max = deg2rad(42);    % rad, maximum steering angle
par.wal.short = 0.65;
    % factor by which l_s is multiplied in transition to walker mode
par.wal.q = 0.05;       % m, caster wheel offset in walker mode
par.wal.u = 0.02;       % m, min. clearance between caster wheels

%% Optimizable value input

optVal_amt = 5;     % amount of optimizable values

% wheel diameter
itv.d_min = 8;          % m, min. value
itv.d_max = 12;         % m, max. value
itv.d_amt = 5;          % amount of values
itv.d = linspace(itv.d_min, itv.d_max, itv.d_amt);
itv.d = itv.d*0.0254;       % conversion from inch to m
    % itv is a struct containing all values relevant for the optimizable
    % parameter intervals and setting up the configurations

% scooter overall width at the rear
itv.w_s_min = 0.6;      % 0.6 m, min. value
itv.w_s_max = 0.7;      % 0.7 m, max. value
```

```
itv.w_s_amt = 3;          % amount of values
itv.w_s = linspace(itv.w_s_min, itv.w_s_max, itv.w_s_amt);

% scooter overall width at the front (must be calculated)
itv.f_s_amt = 4;              % amount of values

% scooter overall length
itv.l_s_min = 0.90;       % m, min. value
itv.l_s_max = 1.20;       % m, max. value
itv.l_s_amt = 7;          % amount of values
itv.l_s = linspace(itv.l_s_min, itv.l_s_max, itv.l_s_amt);

% rider CoM horizontal position, from rear axis (must be calculated still)
itv.p_p_min = 0.05;       % m, min. value
itv.p_p_max = 0.25;       % m, max value
itv.p_p_amt = 5;          % amount of values
itv.p_p = linspace(itv.p_p_min, itv.p_p_max, itv.p_p_amt);

itv.optVal = {itv.d itv.w_s '' itv.l_s itv.p_p};
    % values for all optimizable parameters in one cell aray. the value for
    % f_s is dependent on the other optimizable design parameters and is
    % yet to be calculated
par.cfg.optVal_label = ["$d$ (m)", "$w_\mathrm{s}$ (m)",...
    "$f_\mathrm{s}$ (m)","$l_\mathrm{s}$ (m)", "$p_\mathrm{p}$ (m)"];
    % quantities and unit of optimizable variables in LaTeX format

%% Optimizable design parameter calculations

itv.config_nums = allcomb(1:itv.d_amt, 1:itv.w_s_amt,...
    1:itv.f_s_amt,1:itv.l_s_amt,1:itv.p_p_amt);
    % matrix containing numbers that determine which value must be picked
    % for the optimizable design parameters. Each row represents one
    % configuration
par.config_amt = height(itv.config_nums); % amount of configurations
par.cfg.optVal = zeros(par.config_amt,optVal_amt);
    % cfg is a struct within par containing all configuration dependent
    % design parameters
    % optVal is a matrix containing values of all optizimizable design
    % parameters. Each row represents one configuration.
for i = 1:par.config_amt
    for j = [1 2 4 5]
        optVal_vec = itv.optVal{j};
        optVal_vec_idx = itv.config_nums(i,j);
        par.cfg.optVal(i,j) = optVal_vec(optVal_vec_idx);
    end
    [par.cfg.optVal(i,3), ~] =...
        overall_width_front(par, itv.config_nums, itv.f_s_amt, i);
end
par.cfg.d    = par.cfg.optVal(:,1);
par.cfg.w_s  = par.cfg.optVal(:,2);
par.cfg.f_s  = par.cfg.optVal(:,3);
par.cfg.l_s  = par.cfg.optVal(:,4);
par.cfg.p_p  = par.cfg.optVal(:,5);

%% Calculating other configuration-dependent design parameters

par.cfg.w_b = zeros(par.config_amt,1);
par.cfg.f_b = zeros(par.config_amt,1);
par.cfg.l_b = zeros(par.config_amt,1);
par.cfg.N_w = zeros(par.config_amt,1);
```

```
par.cfg.p_s = zeros(par.config_amt,1);
par.cfg.h_p = zeros(par.config_amt,1);
for i = 1:par.config_amt
    par.cfg.w_b(i) = wheelbase_width_rear(par,i);
    par.cfg.f_b(i) = wheelbase_width_front(par,i);
    par.cfg.l_b(i) = wheelbase_length(par,i);
    [~, par.cfg.N_w(i)] =...
        overall_width_front(par, itv.config_nums, itv.f_s_amt, i);
    par.cfg.p_s(i) = scooter_CoM_horz(par,i);
    par.cfg.h_p(i) = rider_CoM_vert(par,i);
end
par.cfg.config_num = (1:par.config_amt)';

save('design_parameters.mat','par')

%% Calculating important geometric properties for tests

tes.rSpec = 2;                    % m, specified turn radius
    % tes is a struct containing values relevant for the tests
tes.aSpec = deg2rad(30);          % rad, specified inner wheel steering angle
tes.vSpec = 5/3.6;                % m/s, specified speed
tes.obstacle_height = 0.020;      % m, obstacle height
tes.slope = 15;                   % %, slope
geo = geometric_values(par,tes);
    % geo is a struct containing geometric values that have been derived
    % from the design parameters and are useful in multiple tests
disp([num2str(par.config_amt) ' configurations generated!'])

%% Performing tests

tes.test_amt = 10;            % amount of tests
tes.res.results = zeros(par.config_amt, tes.test_amt);
    % array with test results, one column per test
    % res is a struct contain the test results and anything related to it
tes.res.quantities = strings(1,tes.test_amt);
    % array with the respective quantities
tes.res.units = strings(1,tes.test_amt);
    % array with the respective units
tes.res.misc = cell(1,tes.test_amt);
    % cell with miscual information
for i = 1:tes.test_amt      %1:tes.test_amt
    fun=str2func(strcat('test',num2str(i)));    % create function for test
    [tes.res.results(:,i), tes.res.units(i), tes.res.quantities(i),...
        tes.res.full_quantities(i), tes.res.misc{i}]...
        = fun(par,geo,tes);
    disp(strcat("Test ",num2str(i)," performed."))
end

%% Filtering and ratings

% Filtering test results
tes.minOrMax = [0 1 0 0 0 1 1 1 0 1];
    % when 0 larger outcome is better, when 1 vice-verca, so for 0 a lower
    % bound will be applied, while for 1 a higher bound will be applied
tes.bound = [0 0.4472 3 1 4 2 1.5 5 0 0.9];
    % upper or lower bound for each test outcome (requirements)
fil = results_filter(par,tes);  % filter according to bounds
    % fil is a struct containing anything related to filtering data
    % according to test outcomes

```

```
% Assessment
rat.Pmax = 100;      % maximum possible amount of points for assessment
rat.dispLtx = 1;     % If 1, weighting matrix and vector will be displayed
rat.dispTop = 1;     % if 1, display top 3 worst and best concepts
rat = assessment(tes,fil,rat);
rank_final = rat.ran.config_num;     % save final ranking
config_num_best = rank_final(1);     % save number of best configuration
optVal_ranked = rat.ran.optVal;      % optimizable design parameters, ranked
toc

%% Plot

pltSet.fileFrmt = 'epsc';    % file format for plots to save

% Creating 3D image
pln = config_num_best;       % configuration number to plot
structName = 'aMax';
    % must be either 'aMax', 'aSpec','aZero' or 'rSpec'
pltSet.viewType = 'fancy';
    % view type: either 'default', 'fancy', 'Fframe' or 'top'
pltSet.drawTriad = 0;        % if 1 draw triads for body-fixed rames B and F
pltSet.savePlot = 0;         % if 1 image is saved
pltSet.filePath = ['C:\Users\lucas\OneDrive\Documenten\Vakken\Master' ...
    ' Biomechanical Design\Afstuderen' ...
    ' ETH\Verslag\LaTeX\figures'];  % file path for saving figure
scooter_3D(par,geo,tes,structName,pln,pltSet)

% Plotting test results
pltSet.plotAll = 0;              % if 1, all results plotted, ranges ignored
pltSet.PxT = 0;
    % if 1, optimizable design parameters are plotted on subplot columns,
    % tests on rows. Otherwise vice-verca.
pltSet.optVal_range = 5;         % (vector) optimizable parameter(s) to plot
pltSet.test_range = [4 5 9];     % (vector) test(s) to plot
pltSet.savePlotRes = 1;          % if 1, image is saved
results_plotter(optVal_amt,par,tes,fil,pltSet)

% save data
save('test_parameters.mat','tes')
save('geometric_values.mat','geo')
save('test_parameters.mat','tes')
save('filter_parameters.mat','fil')
save('ratesment_outcomes.mat','rat')
```

`allcomb.m` Unmodified function from MATLAB® file exchange [39]. Not included.

`assessment.m`

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Rates all configuration and creates a ranked list.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
```

```
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function rat = assessment(tes,fil,rat)
%% Set-up

% Import variables
A_raw = readmatrix('AHP_matrix.xlsx');
optVal = fil.optVal;
N = fil.config_amt;
results = fil.results;
config_num = fil.config_num;
test_amt = tes.test_amt;
bound = tes.bound;
Pmax = rat.Pmax;
dispLtx = rat.dispLtx;
dispTop = rat.dispTop;

%% Calculations and saving variables

% AHP weighting matrix and vector
A_raw(isnan(A_raw)) = 0;     % replace NaN (empty Excel cells) with zeros
A_rawCompl = 1./A_raw';      % transpose, invert cell-wise
A_rawCompl(isinf(A_rawCompl)|isnan(A_rawCompl)) = 0;     % remove Inf
A = A_raw + A_rawCompl - eye(length(A_raw));             % preference matrix
[V, gamma] = eig(A);
[~, idx_princ] = max(diag(gamma));  % find principle eigenvalue
w = V(:,idx_princ);                 % find principle eigenvector
w = abs(w./sum(w));      % normalize principle eigenvector to get weights

% Test ratings
scores_tests = zeros(N,test_amt);
    % array containing scores for each single test per configuration
for i = 1:test_amt
    res_delta = abs(results(:,i) - bound(i));
        % difference between bound and result
    res_deltaMax = max(res_delta);
        % maximum difference (best test result)
    scores_tests(:,i) = (res_delta./res_deltaMax)*Pmax;
end
scores_final = scores_tests*w;
ranking = flipud(sortrows([scores_final config_num scores_tests optVal...
    results]));
scores_final_ranked = ranking(:,1);
config_num_ranked = ranking(:,2);
scores_tests_ranked = ranking(:,2+(1:test_amt));
optVal_ranked = ranking(:,2+test_amt+(1:width(optVal)));
results_ranked = ranking(:,2+test_amt+width(optVal)+(1:width(results)));

%% Saving variables

rat.A = A;
rat.w = w;
```

```
rat.score_tests = scores_tests;
rat.scores_final = scores_final;
rat.ran.scores_final = scores_final_ranked;
    % ran is a struct containing data by rank (sorted according to overall
    % test score, best overall score on top)
rat.ran.config_num = config_num_ranked;
rat.ran.scores_tests = scores_tests_ranked;
rat.ran.optVal = optVal_ranked;
rat.ran.results = results_ranked;
if dispLtx == 1
    dispLatex(A,'frac')
    dispLatex(w,'float')
end
if dispTop == 1
    disp('Top 3 best (best to worst configuration numbers)')
    disp(config_num_ranked(1:3))
    disp('Top 3 worst (worst to bit less worse configuration numbers)')
    disp(config_num_ranked(flip(end-2:end)))
end
end
```

dispLatex.m

```
% Function created by Lucas Giesen for Multibody Dynamics B HW11
% Modified for master thesis (ME51035, 4596102)
% Used to display stuff in LaTeX formatting.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function dispLatex(value,type)
% struct containting strings to be replaced
LTXold = {'\\','\left(\begin{array}','\end{array}\right)',...
    '{cccccccccc}','{c}','A','w'};
    % raw LaTeX expressions to be replaced
LTXnew = {'\\[1ex]','\begin{bmatrix}','\end{bmatrix}',...
    '','','\mathbf{A}','\mathbf{w}'};
    % custom LaTeX expressions to be replaced by

% check if value is numeric OR if sym does not contain variables to create
% decimal output
value = sym(value);                         % convert numeric value to sym
if strcmp(type,'float')
    sympref('FloatingPointOutput',true);    % set to float output
```

```
elseif ~strcmp(type,'frac')
    error('False type input, must be either ''float'' or ''frac''.')
end
out = replace(latex(value),LTXold,LTXnew);   % create output
disp(strcat(inputname(1), " = ", out))       % display variable name
sympref('default');                          % set sympref to default
format default
```

geometric_values.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Function creates important geometric values.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function geo = geometric_values(par,tes)
%% Unpacking variables

d = par.cfg.d;
t = par.t;
% w_s = par.cfg.w_s;
% f_s = par.cfg.f_s;
% l_s = par.cfg.l_s;
w_b = par.cfg.w_b;
f_b = par.cfg.f_b;
l_b = par.cfg.l_b;
h_s = par.h_s;
p_s = par.cfg.p_s;
m_s = par.m_s;
h_p = par.cfg.h_p;
p_p = par.cfg.p_p;
m_p = par.m_p;
N = par.config_amt;

%% Calculations for universal geometric values

m_c = m_p + m_s;  % total mass
p_c = (p_p*m_p + p_s*m_s)./m_c;
    % horizontal position of overall CoM with respect to rear axis
h_c = (h_p*m_p + h_s*m_s)./m_c;
    % vertical position of overall CoM with respect to ground
B_r_AwC = [w_b/2, -p_c, -h_c];
    % position of point A (right rear wheel middle of contact patch) with
```

```matlab
    % respect to point C (overal center of mass) in body-fixed frame B
    % (alligned with vehicle)
B_r_FwC = B_r_AwC + [t/2 0 0];
    % position of point F (outer right tipping point of right rear wheel)
    % with respect to point C
B_r_HwC = B_r_FwC + [zeros(N,1) -d/2 zeros(N,1)];
    % position of point H (rear right corner point of right rear wheel)
    % with respect to point C
B_r_DwC = [f_b/2, l_b-p_c, -h_c];
    % position of point D (right front wheel middle of contact patch) with
    % respect to point C;
B_centers = {[w_b/2, -p_c, -h_c+d/2] [-w_b/2, -p_c, -h_c+d/2]...
[-f_b/2, l_b-p_c, -h_c+d/2] [f_b/2, l_b-p_c, -h_c+d/2]};
    % coordinates of the center of each wheel

%% Saving variables

geo.m_c = m_c;
geo.p_c = p_c;
geo.h_c = h_c;
geo.B_r_AwC = B_r_AwC;
geo.B_r_FwC = B_r_FwC;
geo.B_r_HwC = B_r_HwC;
geo.B_r_DwC = B_r_DwC;
geo.B_centers = B_centers;

%% Calculations for steering angle dependent geometric values

geo.aMax = geometric_values_steering(par,geo,tes,'aMax');
    % aMax is a struct containing geometric values for the case when
    % maximum steering angle is applied
geo.aSpec = geometric_values_steering(par,geo,tes,'aSpec');
    % aSpec is a struct containing geometric values for when the inner
    % wheel's steering angle is specified
geo.aZero = geometric_values_steering(par,geo,tes,'aZero');
    % aZero is a struct containing geometric values for when the wheels are
    % pointed straight ahead (alpha = 0)
geo.rSpec = geometric_values_steering(par,geo,tes,'rSpec');
    % rSpec is a struct containing geometric values for when the turning
    % radius is specified

end
```

geometric_values_steering.m

```matlab
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Function creates important geometric values that are dependent on
% steering angle.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
```

```
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function out = geometric_values_steering(par,geo,tes,stn)
% stn is the struct name that the variables will be saved under inside the
% struct geo

%% Unpacking variables

N = par.config_amt;
d = par.cfg.d;
t = par.t;
w_s = par.cfg.w_s;
% f_s = par.cfg.f_s;
% l_s = par.cfg.l_s;
% w_b = par.w_b;
f_b = par.cfg.f_b;
l_b = par.cfg.l_b;
% h_s = par.h_s;
% p_s = par.cfg.p_s;
% m_s = par.m_s;
% h_p = par.cfg.h_p;
% p_p = par.cfg.p_p;
% m_p = par.m_p;
alpha_max = par.alpha_max;
r_turn = tes.rSpec;
alpha_spec = tes.aSpec;
% B_r_AwC = geo.B_r_AwC;
B_r_FwC = geo.B_r_FwC;
B_r_HwC = geo.B_r_HwC;
B_r_DwC = geo.B_r_DwC;

%% Calculations

alpha = zeros(N,1);
    % angle of the inner wheel
beta = zeros(N,1);
    % angle of the outer wheel
r = zeros(N,1);
    % turn radius: distance from turning center O to middle of rear axis
B_r_EwD = zeros(N,3);
    % position of point E (outer right tipping point of right front wheel)
    % with respect to point D
B_r_GwE = zeros(N,3);
    % position of point G (front right corner of right front wheel)
    % with respect to point E

if alpha_spec > alpha_max
    warning('Specified steering angle larger than maximally allowed!')
end

e2 = 1; % warning counter
for i=1:N
    % calculations for given inner wheel steering angle
    if strcmp(stn,'aMax') || strcmp(stn,'aSpec') || strcmp(stn, 'aZero')
```

```
        if strcmp(stn,'aMax')
            alpha(i) = alpha_max;
        elseif strcmp(stn,'aSpec')
            alpha(i) = alpha_spec;
        elseif strcmp(stn,'aZero')
            alpha(i) = 0;
        end
        if alpha(i) ~= 0   % to prevent invalid turn radius
            r(i) = f_b(i)/2 + l_b(i)/tan(alpha(i));
                % calculate turning radius for given angle of inner wheel
            beta(i) = atan(l_b(i)/(r(i)+f_b(i)/2));
                % calculate angle of outer wheel
        else
            r(i) = Inf;
            beta(i) = 0;
        end
    % calculations for given turning radius
    elseif strcmp(stn,'rSpec')
        r(i) = r_turn;
        alpha(i) = atan(l_b(i)/(r(i)-f_b(i)/2));
            % calculate angle of inner wheel for given turning radius
        beta(i) = atan(l_b(i)/(r(i)+f_b(i)/2));
            % calculate angle of outer wheel
        if max(alpha(i)) > alpha_max && e2 == 1
            warning(['Specified turn radius rSpec too small for' ...
            ' given maximum steering angle alpha_max!'])
            e2 = e2+1;
        end
    else
        warning(['False structName input. Must be either' ...
            '''aMax'', ''aSpec'', ''aZero'' or ''rSpec''.'])
    end
    B_r_EwD(i,:) = [cos(beta(i))*t/2, sin(beta(i))*t/2, 0];
    B_r_GwE(i,:) = [-sin(beta(i))*d(i)/2, cos(beta(i))*d(i)/2, 0];
end
B_r_EwC = B_r_EwD + B_r_DwC;
B_r_GwC = B_r_EwC + B_r_GwE;
B_r_KwC = perpendicular_vector(B_r_FwC, B_r_EwC,[0 0 0 ]);
B_r_OwC = [-r(:,1) B_r_FwC(:,2) B_r_FwC(:,3)];  % vector from turn center O to
    CoM C
k = sqrt(B_r_KwC(:,1).^2 + B_r_KwC(:,2).^2);

% F-frame: aligned along "tipping line" EF
theta = zeros(N,1); % transformation angle from B to F frame
phi = zeros(N,1);   % angle between B_r_KwC and B_r_OwC in x-y plane
F_R_B = cell(N,1);  % rotation matrices from B to F frame in x-y plane
for i = 1:N
    vec_a = [1 0];                % x unit vector in B frame in x-y plane
    vec_b = B_r_KwC(i,1:2);       % B_r_KwC projection on x-y plane
    theta(i) = acos(dot(vec_a,vec_b)/(vecnorm(vec_a)*vecnorm(vec_b)));
        % z-angle between x-axis and x'-axis
    F_R_B{i} = rotmat(theta(i),'z').';
        % rotation matrix to transform vectors from B to F frame
    vec_c = -B_r_OwC(i,1:2);      % B_r_CwO projection on x-y plane
    phi(i) = acos(dot(vec_b,vec_c)/(vecnorm(vec_b)*vecnorm(vec_c)));
        % z-angle between x'-axis and B_r_CwO (vector from O to C)
end
factor_v_CoM = vecnorm(B_r_OwC,2,2)./r;
    % multiplication factor of the path velocity at the CoM compared to
    % rear axle
```

```
% points for smallest and largest turning radius
r_min = r - w_s./2;              % min. turning radius (at inner rear wheel)
B_r_GwO = B_r_GwC - B_r_OwC;     % vector from O to G
B_r_HwO = B_r_HwC - B_r_OwC;     % vector from O to H
r_G = sqrt(B_r_GwO(:,1).^2 + B_r_GwO(:,2).^2);  % turning radius of point G
r_H = sqrt(B_r_HwO(:,1).^2 + B_r_HwO(:,2).^2);  % turning radius of point H
r_max = zeros(N,1);
r_max_point = strings(N,1);
for i =1:N
    if r_G(i) > r_H(i)
        r_max(i) = r_G(i);
        r_max_point(i) = 'G (front wheel)';
    else
        r_max(i) = r_H(i);
        r_max_point(i) = 'H (rear wheel)';
    end
end

%% Saving variables

out.alpha = alpha;
out.beta = beta;
out.r = r;
out.B_r_EwD = B_r_EwD;
out.B_r_EwC = B_r_EwC;
out.B_r_GwE = B_r_GwE;
out.B_r_GwC = B_r_GwC;
out.B_r_KwC = B_r_KwC;
out.B_r_OwC = B_r_OwC;
out.B_r_GwO = B_r_GwO;
out.B_r_HwO = B_r_HwO;
out.r_min = r_min;
out.r_G = r_G;
out.r_H = r_H;
out.r_max = r_max;
out.r_max_point = r_max_point;
out.k = k;
out.theta = theta;
out.phi = phi;
out.F_R_B = F_R_B;
out.factor_v_CoM = factor_v_CoM;
end
```

`latex_interpreter.m` Function from MATLAB® file exchange [40], modified by author.

```
% https://nl.mathworks.com/matlabcentral/answers/183311-setting-default
% -interpreter-to-latex#answer_803656
clear all
% This script changes all interpreters from tex to latex.
list_factory = fieldnames(get(groot,'factory'));
index_interpreter = find(contains(list_factory,'Interpreter'));
for i = 1:length(index_interpreter)
    default_name = strrep(list_factory{index_interpreter(i)},'factory','default'
        );
    % set(groot, default_name,'default');
    set(groot, default_name,'latex');
end
```

`overall_CoM.m`

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
```

```

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [p, h] = overall_CoM(par)
    p = (par.cfg.p_p*par.m_p + par.cfg.p_s*par.m_s)./(par.m_)
end
```

`perpendicular_vector.m` Includes code snippet from MATLAB® user forms [41].

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function x = perpendicular_vector(a,b,c)
    N = height(a);
    x = zeros(N,3);
    for i = 1:N
        A = a(i,:);
        B = b(i,:);
        C = c;
        x(i,:) = (A-C)-dot(A-C,A-B)/dot(A-B,A-B)*(A-B);
        % https://nl.mathworks.com/matlabcentral/answers/271016-how-to-get
        % -the-vector-from-a-point-orthogonal-to-a-vector#answer_211994
    end
```

```
    % vector x is from point C perpendicular to vector between A and B
end
```

results_filter.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Filters results based on certain conditions.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function fil = results_filter(par,tes)
%% Set-up

% Unpacking variables
config_num = par.cfg.config_num;
optVal = par.cfg.optVal;
test_amt = tes.test_amt;
results = tes.res.results;
minOrMax = tes.minOrMax;
bound = tes.bound;

%% Calculations

config_num_fil  = config_num;   % start with unfiltered list of configs
for i = 1:test_amt
    if minOrMax(i) == 0      % when a lower bound is given
        idx = find(results(:,i) >= bound(i));
            % configuration numbers where test result is higher than lower
            % bound (requirement met)
        config_num_fil = intersect(config_num_fil,idx);
            % filter list of configurations so that configurations that did
            % not pass requirement are dropped
    elseif minOrMax(i) == 1 % when an upper bound is given
        idx = find(results(:,i) <= bound(i));
            % configuration numbers where test result is lower than higher
            % bound (requirement met)
        config_num_fil = intersect(config_num_fil,idx);
            % filter list of configurations so that configurations that did
            % not pass requirement are dropped
    else
        error('minOrMax can only take 0 or 1 !')
    end
end
```

```

%% Saving variables

fil.config_num = config_num_fil;
fil.results = results(config_num_fil,:);
fil.optVal = optVal(config_num_fil,:);
fil.config_amt = height(config_num_fil);
end
```

results_plotter.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Plots all results.
function results_plotter(optVal_amt,par,tes,fil,plotSettings)
%% Set-up

% Unpacking variables
optVal_label = par.cfg.optVal_label;
test_amt = tes.test_amt;
units = tes.res.units;
quantities = tes.res.quantities;
minOrMax = tes.minOrMax;
bound = tes.bound;
results = fil.results;
optVal = fil.optVal;
fileFrmt = plotSettings.fileFrmt;
filePath = plotSettings.filePath;
savePlot = plotSettings.savePlotRes;
plotAll = plotSettings.plotAll;
PxT = plotSettings.PxT;
optVal_range = plotSettings.optVal_range;
test_range = plotSettings.test_range;

%% Dealing with plotting ranges

if plotAll == 1
    optVal_range = 1:optVal_amt;     % plot for all optimizable parameters
    test_range = 1:test_amt;         % plot for all tests
    plotTitle = 'Overview of all test results.';
    plotName = 'test_results_all';
else
    optVal_amt = length(optVal_range);
    test_amt = length(test_range);
    plotTitle = strcat("Test results ",...
        regexprep(num2str(test_range), '  ', ', '));
    plotName = strcat('test_results',...
        regexprep(num2str(test_range), '  ', '_'),...
        "optVal",regexprep(num2str(optVal_range), '  ', '_'));
end

%% Calculating x-axis ranges

xrange = zeros(test_amt,2);

for i = test_range
    if minOrMax(i) == 0      % when a lower bound is given
        xrange(i,:) = [bound(i) max(results(:,i))];
    elseif minOrMax(i) == 1 % when an upper bound is given
        xrange(i,:) = [min(results(:,i)) bound(i)];
    else
        error('minOrMax can only take 0 or 1 !')
    end
```

```
end

%% Plot

if plotAll == 1
    figure('Name',plotName,'WindowState','fullscreen')
else
    figure('Name',plotName)
end
% figure('Name',plotName,'Renderer', 'painters', 'Position', [10 10 1000 900])
l = 1;      % counter to get correct position in subplot
for i = optVal_range
    k = 1;  % second counter to get correct position in subplot
    for j = test_range
        if PxT == 1
            subplot(optVal_amt,test_amt,(l-1)*test_amt+k)
        else
            subplot(test_amt,optVal_amt,(k-1)*optVal_amt+l)
        end
        plot(results(:,j),optVal(:,i),'o')
        xlabel(strcat(quantities(j)," (",units(j),")"))
        xlim(xrange(j,:))
        ylabel(optVal_label(i))
        k = k+1;
    end
    l = l+1;
end

if plotAll ~= 1
    pos = get(gcf, 'Position');                     % for changing plot size
    set(gcf, 'Position',pos+[0 -200 0 200])         % for 6 plots
    % set(gcf, 'Position',pos+[0 -67 0 67])           % for 4 plots
    % set(gcf, 'Position',pos+[0 100 0 -100])         % for 2 plots
    % set(gcf, 'Position',pos+[0 -600 200 600])         % for 10 plots
end

%% Plot settings

sgtitle(plotTitle)
filename = fullfile(filePath, plotName);
if savePlot == 1
    saveas(gca,filename,fileFrmt);       % save plot
end
end
```

rider_CoM_vert.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
```

```
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function h_p = rider_CoM_vert(par, i)
[h_b, ~] = rider_dimensions;
h_a = 0.43;              % mean seat height from rear axle
h_p = par.cfg.d(i)/2 + h_a + h_b;
    % rider CoM height w/ respect to rear axle
end
```

rider_dimensions.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)

% This function calculates key body measurements as described in
% Biomechanics and motor control of human movement (Winter, 2009), Munoz
% e.a. (2011) and DIN Spec 91279.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [h_b, l_CoM2toe] = rider_dimensions
height_male = 1.750;      % m, average German male height
height_female = 1.625;    % m, average German female height
height = mean([height_male, height_female]);     % average German height

height_buttocks2hipJoint = (0.530-0.520)*height;
height_hipJoint2buttocks = 0.1*height;
h_b = height_buttocks2hipJoint + height_hipJoint2buttocks;
h_b = round(h_b,2);

l_leg = (0.530-0.285)*height;
l_ankle2toe = 0.152*(4/5)*height;    % foot length from ankle joint
l_hipJoint2toe = l_leg + l_ankle2toe;
l_hipJoint2CoM = 0.060*height;
l_CoM2toe = l_hipJoint2toe - l_hipJoint2CoM;
end
```

rotmat.m

```matlab
% Function created by Lucas Giesen for Multibody Dynamics B HW11
% Modified for master thesis (ME51035, 4596102)

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function R = rotmat(angle,axis)
% Creates rotation matrix for given axis and angle, so if frame B rotates
% about frame N, function Returns N_R_B
if lower(axis) == 'x'
    R = [1 0 0;0 cos(angle) -sin(angle);0 sin(angle) cos(angle)];
elseif lower(axis) == 'y'
    R = [cos(angle) 0 sin(angle);0 1 0;-sin(angle) 0 cos(angle)];
elseif lower(axis) == 'z'
    R = [cos(angle) -sin(angle) 0;sin(angle) cos(angle) 0;0 0 1];
else
    error('''axis'' should be ''x'', ''y'' or ''z''')
end
```

`scooter_3D.m` Includes code snippets from MATLAB® user forms [42] [43].

```matlab
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Function plots scooter for given steering position.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
```

```
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function scooter_3D(par,geo,tes,stn,pln,plotSettings)

% This function makes a 3D drawing of the scooter for a specified
% configuration number, turning state and viewing angle.
%% Turning scenario check

if ~any(strcmp({'aMax','aSpec','aZero','rSpec'},stn))
    error(['False structName input. Must be either ''aMax'',' ...
        ' ''aSpec'', ''aZero'' or ''rSpec''.'])
end

%% Unpacking variables

t = par.t;
d = par.cfg.d(pln);
w_s = par.cfg.w_s(pln);
h_c = geo.h_c(pln);
B_centers = geo.B_centers;
B_r_AwC = geo.B_r_AwC(pln,:);
B_r_DwC = geo.B_r_DwC(pln,:);
B_r_FwC = geo.B_r_FwC(pln,:);
B_r_HwC = geo.B_r_HwC(pln,:);
B_r_EwC = geo.(stn).B_r_EwC(pln,:);
% B_r_EwD = geo.(stn).B_r_EwD(pln,:);
B_r_HwO = geo.(stn).B_r_HwO(pln,:);
% B_r_GwC = geo.(stn).B_r_GwC(pln,:);
% B_r_GwE = geo.(stn).B_r_GwE(pln,:);
% B_r_GwO = geo.(stn).B_r_GwO(pln,:);
B_r_OwC = geo.(stn).B_r_OwC(pln,:);
beta = geo.(stn).beta(pln);
r_min = geo.(stn).r_min(pln);
r_max = geo.(stn).r_max(pln);
r_H = geo.(stn).r_H(pln);
test6res = tes.res.results(pln,6);
test7res = tes.res.results(pln,7);
r = geo.(stn).r(pln);
Alpha = geo.(stn).alpha(pln);
    % Alpha with capital letter because of function aplha for transparency
    % used later on for the wheels
theta = geo.(stn).theta(pln);
viewType = plotSettings.viewType;
drawTriads = plotSettings.drawTriad;
savePlot = plotSettings.savePlot;
fileFormat = plotSettings.fileFrmt;
filePath = plotSettings.filePath;
N = par.config_amt;

%% Plot set-up

if strcmp(stn,'aMax')
    pltTitle = strcat(" $\alpha_\mathrm{max}=$ ",...
        num2str(round(rad2deg(Alpha),2)),"$^\circ$.");
    pltName = "aMax";
elseif strcmp(stn,'aZero')
    pltTitle = strcat(" $\alpha=0$.");
    pltName = "aZero";
elseif strcmp(stn,'aSpec')
    pltTitle = strcat(" $\alpha=$ ",...
```

```
        num2str(round(rad2deg(Alpha),2)),"$^\circ$.");
    pltName = "aSpec";
elseif strcmp(stn,'rSpec')
    pltTitle = strcat(" $r=$ ",num2str(r)," m.");
    pltName = "rSpec";
end
pltName = strcat("N",num2str(N),"conf",num2str(pln),"_",...
    viewType,"_",pltName);
figure('Name',pltName)

%% Plot wheels

plot3(0,0,0,'r*')        % Plot CoM
hold on
for i = 1:4
    p0=B_centers{i};     % wheel center location in B frame
    p0=p0(pln,:);
        % https://nl.mathworks.com/matlabcentral/answers/1869077-how-to-
        % plot-a-cylinder-from-a-specified-axis#answer_1118047
    plot3(p0(1),p0(2),p0(3),'k*')    % plot wheel centers
    [X,Y,Z] = cylinder(d/2);         % create cylinder
        % https://nl.mathworks.com/matlabcentral/answers/1594904-how-do-i-
        % rotate-the-view-of-a-cylinder-created-using-surf-
        % plot#answer_839699
    Z = Z*t-t/2;             % scale cylinder and move center
    if i==3&&Alpha>0 || i==4&&Alpha<0
        M1=makehgtform('translate',p0(1),p0(2),p0(3),...
        'xrotate',0,'yrotate',pi/2,'zrotate',0);    % rotate and translate
        M2=makehgtform('translate',0,0,0,...
            'xrotate',-Alpha,'yrotate',0,'zrotate',0);
            % apply rotation for steering for inner wheel with Alpha
        M=M1*M2;
    elseif i==4&&Alpha>0 || i==3&&Alpha<0
        M1=makehgtform('translate',p0(1),p0(2),p0(3),...
        'xrotate',0,'yrotate',pi/2,'zrotate',0);    % rotate and translate
        M2 = makehgtform('translate',0,0,0,...
            'xrotate',-beta,'yrotate',0,'zrotate',0);
            % apply rotation for steering for outer wheel with beta
        M=M1*M2;
    else
        M=makehgtform('translate',p0(1),p0(2),p0(3),...
            'xrotate',0,'yrotate',pi/2,'zrotate',0);
            % translate rear wheels
    end
    hold on
    % s = surf(X,Y,Z,'Parent',hgtransform('Matrix',M),...
    %     'FaceColor','#80B3FF', 'EdgeColor','none');
    % alpha(s,.2) % transparent blue wheels
    s = surf(X,Y,Z,'Parent',hgtransform('Matrix',M),...
        'FaceColor','#FFFFFF', 'EdgeColor','#000000');
    alpha(s,.2) % transparent wireframe wheels
end

%% Plot lines

hold on
% plot3([0 B_r_AwC(1)],[0 B_r_AwC(2)],[0 B_r_AwC(3)],...
%     '--','LineWidth',2);                % plot B_r_AwC
plot3([0 B_r_FwC(1)],[0 B_r_FwC(2)],[0 B_r_FwC(3)],...
    '--','LineWidth',2)                 % plot B_r_FwC
```

```
% plot3([0 B_r_DwC(1)],[0 B_r_DwC(2)],[0 B_r_DwC(3)],...
%     '--','LineWidth',2)                    % plot B_r_DwC
% plot3([0 B_r_HwC(1)],[0 B_r_HwC(2)],[0 B_r_HwC(3)],...
%     'b--','LineWidth',1)                   % plot B_r_HwC
% plot3([0 B_r_GwC(1)],[0 B_r_GwC(2)],[0 B_r_GwC(3)],...
%     'b--','LineWidth',1)                   % plot B_r_GwC

% withing the loop plot only things when a turn is made
if Alpha ~= 0
    plot3([-r -B_r_DwC(1)],[B_r_AwC(2) B_r_DwC(2)],[0 0]-h_c,...
        'r--','LineWidth',1)
    % plot line from turning center to inner front wheel
    plot3([-r B_r_DwC(1)],[B_r_AwC(2) B_r_DwC(2)],[0 0]-h_c,...
        'r--','LineWidth',1)
    % plot line from turning center to outer front wheel
    plot3([-r B_r_AwC(1)],[B_r_AwC(2) B_r_AwC(2)],[0 0]-h_c,...
        'r--','LineWidth',1)
    % plot line from turning center to rear axis
    % plot3([0 B_r_GwO(1)]+B_r_OwC(1),...
    %     [0 B_r_GwO(2)]+B_r_OwC(2),...
    %     [0 B_r_GwO(3)]+B_r_OwC(3),...
    %     'b--','LineWidth',1);                      % plot r_GwO
    % plot3([0 B_r_HwO(1)]+B_r_OwC(1),...
    %     [0 B_r_HwO(2)]+B_r_OwC(2),...
    %     [0 B_r_HwO(3)]+B_r_OwC(3),...
    %     'b--','LineWidth',1);                      % plot r_HwO

    % minimum and maximum turning radius circle plots
    sigma_rmin_min = -asin(0.5*d/r_min);
    sigma_rmin_max = 0.5*pi;
    % sigma_rmax_min = acos(B_r_HwO(1)/r_max);
    %     % plot circle up to outer side
    sigma_rmax_min = asin(B_r_HwO(2)/r_max);
        % plot circle all the way down
    sigma_rmax_max = 0.5*pi;%asin(B_r_GwO(2)/% r_max);
    sigma_rmin = sigma_rmin_min:0.01:sigma_rmin_max;
    sigma_rmax = sigma_rmax_min:0.01:sigma_rmax_max;

    circle_rmin_x = r_min*cos(sigma_rmin) + B_r_OwC(1);
    circle_rmin_y = r_min*sin(sigma_rmin) + B_r_OwC(2);
    circle_rmax_x = r_max*cos(sigma_rmax) + B_r_OwC(1);
    circle_rmax_y = r_max*sin(sigma_rmax) + B_r_OwC(2);
    plot3(circle_rmin_x,circle_rmin_y,-h_c...
        *ones(1,numel(sigma_rmin)),'b--','LineWidth',1)
    plot3(circle_rmax_x,circle_rmax_y,-h_c...
        *ones(1,numel(sigma_rmax)),'b--','LineWidth',1)

    if strcmp(stn,'aMax')
        % plotting clear floor area
        plot3(w_s/2*[-1 -1],[0 test6res]+B_r_HwC(2),-h_c*[1 1],...
            'b--','LineWidth',1)
        plot3(w_s/2*[-1 0]+(r_H-r)*[0 1],(test6res+B_r_HwC(2))*[1 1],...
            -h_c*[1 1],'b--','LineWidth',1)

        % plotting hallway inner corner
        plot3((r_min/sqrt(2)+B_r_OwC(1))*[1 1],...
            [B_r_HwC(2) r_min/sqrt(2)+B_r_OwC(2)],...
            -h_c*[1 1],'b--','LineWidth',1)
        plot3(B_r_OwC(1) + [0 r_min/sqrt(2)],...
            (r_min/sqrt(2)+B_r_OwC(2))*[1 1],...
```

```
            -h_c*[1 1],'b--','LineWidth',1)
        plot3((r_min/sqrt(2)+B_r_OwC(1))*[1 1],...
            r_min/sqrt(2)+B_r_OwC(2) + [0 test7res],...
            -h_c*[1 1],'b:','LineWidth',1)
    end
end

% plot3([0 B_r_EwD(1)]+B_r_DwC(1),[0 B_r_EwD(2)]+B_r_DwC(2),...
%     [0 B_r_EwD(3)]+B_r_DwC(3),'--','LineWidth',2);      % plot r_EwD
% plot3([0 B_r_GwE(1)]+B_r_EwC(1),[0 B_r_GwE(2)]+B_r_EwC(2),...
%     [0 B_r_GwE(3)]+B_r_EwC(3),'b--','LineWidth',1);     % plot r_GwE
plot3([0 geo.(stn).B_r_KwC(pln,1)],[0 geo.(stn).B_r_KwC(pln,2)],...
    [0 geo.(stn).B_r_KwC(pln,3)],'--','LineWidth',2)    % plot r_KwC
plot3([0 B_r_EwC(1)],[0 B_r_EwC(2)],...
    [0 B_r_EwC(3)],'--','LineWidth',2)          % plot r_EwC
plot3(B_r_OwC(1),B_r_OwC(2), B_r_OwC(3),'r*')   % plot turn center
plot3([B_r_OwC(1), 0],[B_r_OwC(2) 0],...
    [B_r_OwC(3), 0],'--','LineWidth',2)         % plot r_OwC
B_r_FwE = [B_r_EwC; B_r_FwC];                   % plot r_FwE
plot3(B_r_FwE(:,1), B_r_FwE(:,2), B_r_FwE(:,3),'--','LineWidth',2);

%% Plot triads B and F

triad_length = 0.25;   % length to plot unit vectors with
F_uxpr = triad_length*[0 0 0;1 0 0];
F_uypr = triad_length*[0 0 0;0 1 0];
F_uzpr = triad_length*[0 0 0;0 0 1];
B_R_F = geo.(stn).F_R_B{pln}.';
    % rotation matrix to transform vectors from F to B frame
B_uxpr = (B_R_F*F_uxpr.').';
    % double transpose is necessary because rotation matrices are made for
    % column vectors
B_uypr = (B_R_F*F_uypr.').';
B_uzpr = (B_R_F*F_uzpr.').';
B_ux = triad_length*[0 0 0;1 0 0];
B_uy = triad_length*[0 0 0;0 1 0];
if drawTriads == 1
    vectarrow(B_uxpr(1,:),B_uxpr(2,:),'k'); hold on
    vectarrow(B_uypr(1,:),B_uypr(2,:),'k'); hold on
    vectarrow(B_uzpr(1,:),B_uzpr(2,:),'k'); hold on
    vectarrow(B_ux(1,:),B_ux(2,:),'k'); hold on
    vectarrow(B_uy(1,:),B_uy(2,:),'k'); hold on
end

%% Plot settings

hold off

axis equal
grid on
xlabel('x (m)')
ylabel('y (m)')
zlabel('z (m)')

% select view
if strcmp(viewType,'default')
    view(3)
elseif strcmp(viewType,'fancy')
    view(-37.5-70,30)
elseif strcmp(viewType,'Fframe')
```

```
    view(rad2deg(theta),0)
elseif strcmp(viewType,'top')
    view(0,90)
else
    error('False view type input!')
end

title(strcat("Configuration \#",num2str(pln),"/",num2str(N),",",...
    pltTitle))
filename = fullfile(filePath, pltName);
if savePlot == 1
    saveas(gca,filename,fileFormat);    % save plot
end
```

scooter_CoM_horz.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function p_s = scooter_CoM_horz(par, i)
    p_s = (3/4)*par.cfg.l_b(i);
end
```

test1.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Test 1: "Disturbance test": energy required to make scooter tip at a
% speed of vSpec at a sSpec turn radius.

% The energy will be calculated using an iterative method. For a given
% tipping angle gamma, the speed at which scooter will tip over will be
% calculated. Then, using an iterative method, the angle is adjusted until
% the difference between the calculated tipping speed and the target speed
% vSpec is negligable. The function also has the energy as an output.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
```

```
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [tipping_energy, unit, quantity, quantity_full, misc]...
    = test1(par,geo,tes)
%% Set-up

% defining output unit and quantity
unit = 'J';
quantity = 'Energy';
quantity_full = strcat('Energy required to make the scooter tip with',...
    " v = ", num2str(tes.vSpec*3.6),...
    " km/h and r = ", num2str(tes.rSpec),'.');
misc = [];  % no miscual information to save

% unpacking variables
N = par.config_amt;
v_target = tes.vSpec;

%% Calculations

gamma = zeros(N,1);      % array containing tipping angles gamma
E = zeros(N,1);          % array containing energy
v_tolerance = 0.00001;  % tolerance for the final guess of the speed
guess_amt_max = 50;      % maximum amount of guesses per configuration
v = zeros(N,1);

for i = 1:N
    gamma_min = deg2rad(0);       % rad, minimum gamma guess
    gamma_max = deg2rad(60);      % rad, maximum gamma guess
    for j = 1:guess_amt_max
        % v_target = tes.results(i,3);disp(v_target) verification: E is
        % zero for tipping speeds where it should tip with gamma = 0
        gamma_est = (gamma_max+gamma_min)/2;
        [v_guess, E_guess] = test1_estimator(par,geo,gamma_est,i);
        if isreal(v_guess)
            if v_guess > v_target   % speed too high, higher tipping angle
                gamma_min = gamma_est;
                gamma(i) = gamma_est;
                if abs(v_guess - v_target) < v_tolerance
                    E(i) = E_guess;
                    v(i) = v_guess;
                    break
                end
            else
                gamma_max = gamma_est;  % speed to low, lower tipping angle
                gamma(i) = gamma_est;
                if abs(v_guess - v_target) < v_tolerance
                    E(i) = E_guess;
```

```
                    v(i) = v_guess;
                    break
                end
            end
            E(i) = E_guess;
            v(i) = v_guess;
        else
            gamma_max = gamma_est;
                % if v_est is imaginary, gamma is already so high that the
                % vehicle is past its tipping point. Thus the next guess
                % must be lower.
        end
    end
end

%% Saving variables

tipping_energy = E;
end
```

test1_estimator.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
function [v_tip, E] = test1_estimator(par,geo,gamma,i)
% gamma is the estimated tipping angle, i is the configuration number

% This file calculates the position the speed at which the scooter will tip
% for a given tipping angle gamma, which is depend on T, which is the CoM C
% in its tipped state (rotated around point K). Thus F_r_KwC will be
% transformed to the F frame, so that it can be rotated around the y'-axis
% by gamma. After transforming back to the B-frame, B_r_TwO can be
% calculated to calculate the turning radius at T. After calculating the
% angle psi between r_TwO and r_KwC, it is known by which factor cos(psi)
% the the centripetal acceleration in T must be multiplied in order to get
% the centripetal acceleration in the tipping plane (x'z'-plane).

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

%% Set-up

stn = 'rSpec';

% Unpacking variables
```

```
B_r_OwC = geo.(stn).B_r_OwC(i,:);
B_r_KwC = geo.(stn).B_r_KwC(i,:);
F_R_B = geo.(stn).F_R_B{i};
g = par.g;
h_c = geo.h_c(i);
m = geo.m_c;
r = geo.(stn).r(i);
w_s = par.cfg.w_s(i);
d = par.cfg.d(i);

%% Calculations

B_r_CwK = -B_r_KwC;
R_tip = rotmat(gamma,'y');

F_r_CwK = (F_R_B*B_r_CwK.').';
    % transformation of r_CwK to F frame
    % double transpose is necessary because rotation matrices are made for
    % column vectors
F_r_TwK = (R_tip*F_r_CwK.').';
    % rotation r_CwK around y' anti-clockwise with gamma to simulate
    % tipping, yields r_TwK with T being CoM in tipped state
B_R_F = F_R_B.';
B_r_TwK = (B_R_F*F_r_TwK.').';

B_r_TwC = B_r_KwC + B_r_TwK;
B_r_CwO = -B_r_OwC;
B_r_TwO = B_r_TwC + B_r_CwO;

vec_a = B_r_KwC(1:2);     % B_r_KwC projection on x-y plane
vec_b = B_r_TwO(1:2);     % B_r_CwO projection on x-y plane
psi = acos(dot(vec_a,vec_b)/(vecnorm(vec_a)*vecnorm(vec_b)));
    % z-angle between B_r_KwC and B_r_CwO, so like phi but for the
    % tipped state

h_c_tip = F_r_TwK(3);
k_tip = -F_r_TwK(1);
s_tip = sqrt(B_r_TwO(1).^2 + B_r_TwO(2).^2);     % turning radius at T
r_tip = r +  w_s/2 - sin(atan(d/w_s)+gamma)*sqrt(0.25*(d^2+w_s^2));
    % radius of the middle of the rear axis when tipping
factor_v_CoM = s_tip/r_tip;
    % factor by which to multiply v when tipping
v_tip = sqrt((g*k_tip.*s_tip)./(h_c_tip.*cos(psi).*factor_v_CoM.^2));
    % tipping speed for the estimated value of gamma

E = (h_c_tip - h_c)*m*g;
end
```

test2.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Test 2: Angle of impact on the wheel.

% This function calculates the angle of impact of an obstacle in the road
% hitting the wheel.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
```

```
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [impact_angle, unit, quantity, quantity_full, misc]...
    = test2(par,~,tes)
%% Set-up

% defining output unit and quantity
unit = 'fraction, unitless';
quantity = 'Angle';
quantity_full = strcat("Impact angle of a ",...
    num2str(tes.obstacle_height*1000)," mm tall obstacle on the wheel.");
misc = [];  % save miscual information here

% unpacking variables
d = par.cfg.d;
a = tes.obstacle_height;

%% Calculations

l = sqrt(a.*d-a.^2);

%% Saving variables

impact_angle = a./l;
end
```

test3.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Test 3: "Moose test": speed at which scooter tips voor maximum steering
% angle

% This function calculates the speed at which the scooter tips for the
% maximum steering angle alpha_max.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
```

```
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [tipping_speed, unit, quantity, quantity_full, misc]...
    = test3(par,geo,~)
%% Set-up

% defining output unit and quantity
unit = 'km/h';
quantity = 'Speed';
quantity_full = 'Speed at which scooter tips for maximum steering angle';
misc = [];  % no miscual information to save

% Unpacking variables
B_r_OwC = geo.aMax.B_r_OwC;
k = geo.aMax.k;
phi = geo.aMax.phi;
h_c = geo.h_c;
g = par.g;
factor_v_CoM = geo.aMax.factor_v_CoM;

%% Calculations

% v_C = factor_v_CoM*v_tip_sym;    % trajectory speed at C
% a_C = v_C.^2./r;                 % centripetal acceleration at C
% a_Cxpr = a_C.*cos(phi);          % centripetal accel. along x' direction

s = sqrt(B_r_OwC(:,1).^2 + B_r_OwC(:,2).^2);     % turning radius at C
v_tip = sqrt((g*k.*s)./(h_c.*cos(phi).*factor_v_CoM.^2));
v_tip_kmh = v_tip*3.6;                     % conversion to km/h

%% Saving variables

tipping_speed = v_tip_kmh;
end
```

test4.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Test 4: Acceleration while going up a slope.

% This function calculates how fast the scooter can accelerate uphill for a
% certain slope angle.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
```

```
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [a_max, unit, quantity, quantity_full, misc] = test4(par,geo,tes)
%% Set-up

% defining output unit and quantity
unit = 'm/s$^2$';
quantity = 'Acceleration';
quantity_full = strcat("Maximum acceleration on a ",...
    num2str(rad2deg(tes.slope))," deg slope.");
misc = [];  % save miscual information here

% unpacking variables
p_c = geo.p_c;
h_c = geo.h_c;
g = par.g;
slope = tes.slope;

%% Calculations

kappa = atan(slope/100);    % calculating angle from slope

%% Saving variables

a_max = g.*((p_c./h_c).*cos(kappa) - sin(kappa));
end
```

test5.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Test 5: Braking while going down a slope.

% This function calculates how fast the scooter can brake downhill for a
% certain slope angle.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [a_max, unit, quantity, quantity_full, misc] = test5(par,geo,tes)
```

```
%% Set-up

% defining output unit and quantity
unit = 'm/s$^2$';
quantity = 'Deceleration';
quantity_full = strcat("Maximum acceleration on a ",...
    num2str(rad2deg(tes.slope))," deg slope.");
misc = [];  % save miscual information here

% unpacking variables
p_c = geo.p_c;
l_b = par.cfg.l_b;
h_c = geo.h_c;
g = par.g;
slope = tes.slope;

%% Calculations

kappa = atan(slope/100);    % calculating angle from slope

%% Saving variables

a_max = g.*(((l_b-p_c)./h_c).*cos(kappa) - sin(kappa));
end
```

test6.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Test 6: Minimum length for a clear floor area, assuming maximum steering
% angle.

% Calculates above.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [l_f, unit, quantity, quantity_full, misc] = test6(par,geo,~)
%% Set-up

% defining output unit and quantity
unit = 'm';
quantity = 'Length';
quantity_full = 'Minimum length for a clear floor area.';
misc = [];  % save miscual information here
```

```

% unpacking variables
r_max = geo.aMax.r_max;
r = geo.aMax.r;
d = par.cfg.d;
w_s = par.cfg.w_s;

%% %% Calculations - Saving variables

l_f = d/2 + sqrt(r_max.^2 - (r - w_s./2).^2);
end
```

test7.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Test 7: Width required for a hallway with a 90 degree bend in order for
% the device to drive thruogh, assuming maximum steering angle.

% Calculates above.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [width, unit, quantity, quantity_full, misc] = test7(~,geo,~)
%% Set-up

% defining output unit and quantity
unit = 'm';
quantity = 'Width';
quantity_full = "Minimum required width for a hallway with a 90 " +...
    "degree bend for the scooter to ride through.";
misc = [];  % save miscual information here

% unpacking variables
r_max = geo.aMax.r_max;
r_min = geo.aMax.r_min;

%% Calculations - Saving variables

width = r_max - r_min/sqrt(2);
end
```

test8.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Test 7: Width required for a square room in order for the device to make
% a 360 degree turn in it, assuming maximum steering angle.

% Calculates above.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [width, unit, quantity, quantity_full, misc] = test8(~,geo,~)
%% Set-up

% defining output unit and quantity
unit = 'm';
quantity = 'Width';
quantity_full = 'Minimum required with for a room for the scooter to'+...
    " make a 360 degree turn in.";
misc = [];  % save miscual information here

% unpacking variable
r_max = geo.aMax.r_max;

%% Calculations - Saving variables

width = 2*r_max;
end
```

test9.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Test 9: Distance between feet and front wheels.

% This function assumes body measurements as described in Biomechanics and
% motor control of human movement (Winter, 2009).

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
```

```
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [distance, unit, quantity, quantity_full, misc] = test9(par,~,~)
%% Set-up

% defining output unit and quantity
unit = 'm';
quantity = 'Distance';
quantity_full = 'Distance between the feet and front wheels.';
misc = [];  % save miscual information here

% unpacking variables
d = par.cfg.d;
l_b = par.cfg.l_b;
p_p = par.cfg.p_p;

%% Calculations

[~, l_CoM2toe] = rider_dimensions;
l_CoM2wheel = l_b - p_p - d/2;

%% Saving variables

distance = l_CoM2wheel - l_CoM2toe;
end
```

test10.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Test 10: Width required for a hallway with a 90 degree bend in walker
% mode.

% This function first calculates several geometric values for the device in
% scooter mode, after which the turning radii for several points are
% calculated. The largest one then dictates the test output.

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
```

```matlab
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function [width, unit, quantity, quantity_full, misc] = test10(par,~,~)
%% Set-up

% defining output unit and quantity
unit = 'm';
quantity = 'Width';
quantity_full = "Minimum required width for a hallway with a 90 " +...
    "degree bend for the walker to ride through.";

% unpacking variables
d = par.cfg.d;
t = par.t;
l_s_scooter = par.cfg.l_s;
f_b = par.cfg.f_b;
w_b = par.cfg.w_b;
f = par.wal.short;
    % factor by which the length shortens in transition to walker mode
q = par.wal.q;                  % m, caster wheel distance in walker mode
N = par.config_amt;

%% Calculations

l_s = l_s_scooter*f;     % overall length of the device in walker mode
l_b = l_s-d+q;           % length of the wheelbase in walker mode
MB = sqrt(l_b.^2 + (w_b./2 + f_b./2).^2);
    % distance from point B (inner rear wheel, center of turn) to M, the
    % point where the front wheel swivels around as a caster wheel.
BD = sqrt(MB.^2 - q^2);
r_G = sqrt((BD+t/2).^2 + (d./2).^2);
r_H = sqrt((w_b + t/2).^2 + (d./2).^2);

width = zeros(N,1);
r_max_point = strings(N,1); % shows which point forms the largest radius
for i = 1:N
    if r_G(i) > r_H(i)
        width(i) = r_G(i) + t/2;
        r_max_point(i) = 'G (front wheel)';
    else
        width(i) = r_H(i) + t/2;
        r_max_point(i) = 'H (rear wheel)';
    end
end
%% Saving variables

misc = r_max_point;
end
```

testx_template.m

```matlab
% File created for Lucas Giesen's master thesis (ME51035, 4596102)
% Test x: test output

% if necessary, short explanation of how function works
function [output, unit, quantity, quantity_full, misc] = testx(par,geo,tes)

% Copyright 2023 Lucas A.M. Giesen
%
```

```
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

%% Set-up

% defining output unit and quantity
unit = 'unit';
quantity = 'quantity';
quantity_full = 'quantity with conditions';
misc = [];  % save miscual information here

% unpacking variables

%% Calculations

[];

%% Saving variables

output = [];
end
```

`vectarrow.m` Function from MATLAB® file exchange [44], modified by author.

```
function vectarrow(p0,p1,color)
%Arrowline 3-D vector plot.
%   vectarrow(p0,p1) plots a line vector with arrow pointing from point p0
%   to point p1. The function can plot both 2D and 3D vector with arrow
%   depending on the dimension of the input
%
%   Example:
%       3D vector
%       p0 = [1 2 3];   % Coordinate of the first point p0
%       p1 = [4 5 6];   % Coordinate of the second point p1
%       vectarrow(p0,p1)
%
%       2D vector
%       p0 = [1 2];     % Coordinate of the first point p0
%       p1 = [4 5];     % Coordinate of the second point p1
%       vectarrow(p0,p1)
%
%   See also Vectline

%   Rentian Xiong 4-18-05
%   $Revision: 1.0
%   Modified by Lucas Giesen to include color option
```

```
if max(size(p0))==3
    if max(size(p1))==3
        x0 = p0(1);
        y0 = p0(2);
        z0 = p0(3);
        x1 = p1(1);
        y1 = p1(2);
        z1 = p1(3);
        plot3([x0;x1],[y0;y1],[z0;z1],color);    % Draw a line between p0 and
            p1

        p = p1-p0;
        alpha = 0.1;  % Size of arrow head relative to the length of the
            vector
        beta = 0.1;  % Width of the base of the arrow head relative to the
            length

        hu = [x1-alpha*(p(1)+beta*(p(2)+eps)); x1; x1-alpha*(p(1)-beta*(p(2)+
            eps))];
        hv = [y1-alpha*(p(2)-beta*(p(1)+eps)); y1; y1-alpha*(p(2)+beta*(p(1)+
            eps))];
        hw = [z1-alpha*p(3);z1;z1-alpha*p(3)];

        hold on
        plot3(hu(:),hv(:),hw(:),color)  % Plot arrow head
        grid on
        xlabel('x')
        ylabel('y')
        zlabel('z')
        hold off
    else
        error('p0 and p1 must have the same dimension')
    end
elseif max(size(p0))==2
    if max(size(p1))==2
        x0 = p0(1);
        y0 = p0(2);
        x1 = p1(1);
        y1 = p1(2);
        plot([x0;x1],[y0;y1],color);    % Draw a line between p0 and p1

        p = p1-p0;
        alpha = 0.1;  % Size of arrow head relative to the length of the
            vector
        beta = 0.1;  % Width of the base of the arrow head relative to the
            length

        hu = [x1-alpha*(p(1)+beta*(p(2)+eps)); x1; x1-alpha*(p(1)-beta*(p(2)+
            eps))];
        hv = [y1-alpha*(p(2)-beta*(p(1)+eps)); y1; y1-alpha*(p(2)+beta*(p(1)+
            eps))];

        hold on
        plot(hu(:),hv(:),color)  % Plot arrow head
        grid on
        xlabel('x')
        ylabel('y')
        hold off
    else
```

```
            error('p0 and p1 must have the same dimension')
        end
    else
        error('this function only accepts 2D or 3D vector')
    end
```

wheelbase_length.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function l_b = wheelbase_length(par, i)
    l_b = par.cfg.l_s(i) - par.cfg.d(i);
end
```

wheelbase_width_front.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function f_b = wheelbase_width_front(par, i)
    f_b = par.cfg.f_s(i) - par.t;
```

```
end
```

wheelbase_width_rear.m

```
% File created for Lucas Giesen's master thesis (ME51035, 4596102)

% Copyright 2023 Lucas A.M. Giesen
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
% IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
% TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

function w_b = wheelbase_width_rear(par, i)
    w_b = par.cfg.w_s(i) - par.t;
end
```