
Institute for Computer Science VII
Robotics and Telematics

*Master's thesis*

# TLDR Robot
# Telescopic Linear Driven Rotation Robot — A Locomotion Approach for Spherical Robots

Jasper Zevering

September 2021

| | |
|---|---|
| First reviewer: | Prof. Dr. Andreas Nüchter |
| Second reviewer: | Prof. Dr.-Ing. Sergio Montenegro |
| Advisor: | Dr. Dorit Borrmann |

# Danksagung

Auch wenn diese Arbeit hoffentlich nur einen Zwischenschritt meines akademischen Weges darstellt, möchte ich die Gelegenheit nutzen um Danke zu sagen.

Zu einem möchte ich Dr. Dorit Borrmann danken, die nicht nur diese Arbeit mit mehr Engangement betreut hat als ich es mir jemals wünschen könnte, sondern auch als Vorbild meinen persönlichen Wunsch weiter in der Forschung und Lehre tätig zu sein gefestigt hat. Gleicherweise möchte ich Prof. Dr. Andreas Nüchter danken, der über den gesamte Master mich stets als Ermöglicher begleitet hat und durch Vorleben gezeigt hat, wie Wissenschaft und wertschätzendes Arbeiten sein kann. Beiden möchte ich für die unglaubliche Hilfestellung und Geduld über die letzten zwei Jahre danken.

Auch möchte ich Fabian Arzberger danken, mit dem ich die letzten zwei Jahre mehrer Projekte bezüglich Kugelrobotern umsetzen durfte, mit dem ich das Studium bestritten habe und der mir während dieser Thesis stets zur Hilfe stand.

Anton Bredenbeck und Stefanie Freund danke ich für das Korrekturlesen dieser Arbeit.

Dem technischen Betrieb der Universität Würzburg, inbesondere Herrn Reusch danke ich für die handwerkliche Unterstützung beim Prototypenbau.

Zu guter Letzt möchte ich meiner wundervollen Frau Annika Zevering danken, die mir stets den Rücken freigehalten hat und mit ihren Anregungen zur Arbeit zum defintiv besseren Verständnis beigetragen hat.

## Zusammenfassung

Der Einsatz von Robotern ermöglicht eine risikofreie Erkundung unbekannter und potenziell gefährlicher Umgebungen wie dem Weltraum oder Planetenoberflächen. Unwegsames Gelände stellt jedoch hohe Anforderungen an Roboter, insbesondere an ihr Fortbewegungssystem. Die vorliegende Arbeit untersucht einen neuartigen Antriebsmechanismus für Kugelroboter, der auf Rotation durch teleskopische Linearaktuatoren beruht. Der Ansatz verfolgt dabei das Ziel, bei unbekanntem Terrain möglichst hohe Flexibilität der Fortbewegung zu gewährleisten. Die Vorteile bestehen aus der Vielseitigkeit der Nutzung dieser Linearaktuatoren und den insgesamt bleibenden Vorteiles des Kugelroboters, welche sich möglichst wenig auf ein Terrain spezialisieren und die Payload optimal schützen. Die erarbeiteten Lösungen bieten eine mathematisch-physikalische Systembeschreibung, einfache Algorithmen zur Steuerung und Regelung der Fortbewegung und des Balancierens, eine Dimensionierung der Bauteile und des Roboters, sowie allgemeine Berechnungen zu maximal erreichbaren Leistungsparametern des Roboters. Ein erster Prototyp beweist die grundsätzliche Tauglichkeit des Systems. Zudem werden auch Konzepte für komplexere Regelungen, die ihre Umgebung direkt mit einbeziehen, konzeptuell erläutert.

# Abstract

The usage of robots enables a risk-free exploration of unknown and potentially harsh environments such as space or planetary surfaces. However, rough terrains place high demands on robots, especially on their locomotion system. The present work investigates a novel locomotion approach concerning spherical robots based on rotation using telescopic linear actuators. The approach has been developed with the goal of achieving the highest possible flexibility of locomotion in unknown terrains. Its advantages consist of the versatility of using linear actuators and the capability of the spherical robot to optimally protect the payload, focusing as little as possible on a specific terrain. The developed solutions offer a mathematical-physical system description, simple algorithms for the control and regulation of the locomotion and balancing, and a dimensioning of the components and the robot, as well as general calculations for determining the maximum achievable performance parameters of the robot. The first prototype proves the basic suitability of the system. Yet the results indicate the need for a custom-tailored solution, as a budget version does not fulfill the desired demands. This thesis further explains concepts pertaining to more complex control systems that directly factor in their environment.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Acronyms

| | |
|---|---|
| CDF | Concurrent Design Facility |
| DAEDALUS | Descent And Exploration in Deep Autonomy of Lava Underground Structures |
| ESA | European Space Agency |
| IDU | Internal Drive Unit |
| IMU | Inertial Measurement Unit |
| LiDAR | Light Detection and Ranging |
| NASA | National Aeronautics and Space Administration |
| PID | Proportional-Integral-Derivative (controller) |
| SLAM | Simultaneous Localization and Mapping |
| SMA | Shape Memory Alloy |
| VPIM | Virtual Pose Instruction Map |
| VPIP | Virtual Pose Instruction Plane |
| ZMP | Zero Moment Point |

# List of often used Symbols

This thesis uses a wide variety of symbols. This list provides the often used symbols and general symbols, which are supplemented and further specified by indices in the chapters and sections.

| | |
|---|---|
| $\alpha$ | Angle until which a pole extends for pushing |
| $\beta$ | Angle from which a pole extends for pushing |
| $\gamma$ | Angle at which a poles retraction speed matches the needed retraction speed |
| $\epsilon$ | Angle at which retraction must start for leverage, in order to reach the right extension at $\gamma$ |
| $\epsilon_s$ | Angle at which extension must start for leverage, in order to reach full extension at $\epsilon$ |
| $\epsilon_\phi$ | Error of the roll angle. |
| $\zeta$ | Angle of a specific pole |
| $\zeta^*$ | $0.5\pi - \zeta$ |
| $\eta$ | General angle, not referring to specific pole |
| s $\theta$ | Pitch angle of the robot |
| $\lambda$ | relative extension of a pole needed for contact with VPIP |
| $\mu$ | Friction coefficient |
| $\xi$ | Angle of a slope |
| $\Pi$ | Plane |
| $\tau$ | Torque |
| $\phi$ | Roll angle of the robot |
| $\psi$ | Yaw angle of the robot |
| $\omega$ | Rotation speed of robot |
| $A$ | Constant combining other constants used in the physical representation |
| $a$ | Linear acceleration of robot |
| $a_h$ | Linear acceleration of robot in horizontal direction |
| $a_v$ | Linear acceleration of robot in vertical direction |
| $c_\omega$ | commanded rotation speed |
| $d$ | Distance |
| $d_m^s$ | Distance from middle disc to side disc |
| $e$ | Energy |
| $F$ | Force |
| $f$ | Frequency |
| $g$ | Gravitational acceleration |
| $I$ | Moment of Inertia |
| $k$ | Variable used for substitution |

| | |
|---|---|
| $l$ | Length of a pole |
| $L$ | Line |
| $\boldsymbol{p}$ | Three-dimensional point |
| $m$ | Mass |
| $M$ | Momentum |
| $n$ | Number of poles |
| $r$ | Radius |
| $r_m$ | Radius of the middle disc |
| $r_s$ | Radius of the side disc |
| $s_f$ | Side factor representing each side disc |
| $v$ | Velocity |

# Chapter 1

# Introduction

## 1.1 Motivation

Despite the huge variety of shapes and locomotion systems of robots, only four types are used as baseline for extraterrestrial ground exploration: Robots having mobility based on wheels, tracks, legs, and combinations of them [70]. To date, no planetary surface exploration rover, including those launched by the European Space Agency (ESA) and the National Aeronautics and Space Administration (NASA), has been equipped with a locomotion system different from the above-mentioned ones. All locomotion systems in use employ four to eight wheels and a combination of legs [2, 3]. This validates the widespread practice of using well-known technologies and conservative mechanisms in space missions rather than cutting-edge approaches. However, with the rise of private companies like SpaceX, Boston Dynamics, and Axiom Space, non-public-driven development and research enter the field of space suitable robotics. This, as well as the comprehensive usage of wheeled rovers, leads both agencies to open up the field to new, alternative robotic approaches. Further, the technical development process has shifted from being purely internal to an open, collaborative search of ideas, as indicated by the open calls for ideas and competitions held. Both, NASA and ESA, have adopted this approach for getting inspiration from outside - NASA with its "Call for Ideas" (CFI) [1] and ESA with its SysNova assessment scheme [4]. This approach often leads to solutions, shifting the field of versatile multipurpose robots, often designed as rovers themselves, carrying a payload, to an all-in-one system, wherein every facet of the robots is build around the science requirement. This is the case for the previously mentioned SysNova assessment scheme developing a robot for lunar cave exploration, called "Descent And Exploration in Deep Autonomy of Lava Underground Structures (DAEDALUS)." Designed for lunar cave exploration, this shape of the robot, locomotion system, and all other aspects are built to suit the requirements for exploring moon caves and the restrictions associated with the use of optical and laser scanners. Chapter 2 discusses the details of the DAEDALUS project. It was from this project that the Telescopic Linear Driven Rotation Robot emerged. The robot has its spherical shape due to environmental reasons and mechanism due to the payload.

## 1.2 Outline

The structure of this thesis is as follows: Chapter 2 summarizes the DAEDALUS project, as it is the main reason for the emergence of this locomotion approach and the resulting TLDR robot. It also sums up the main advantages of this locomotion technique along with the general advantages of the spherical shape. Chapter 3 presents the previously used locomotion approaches for spherical robots. It splits up the topic into mechanisms inside the sphere, and extrinsic mechanisms, focusing on the use of linear actuators. Next, Chapter 4 analyzes the locomotion concerning spherical and cylindrical robots using linear telescopic actuators and examines the general usage of linear actuators in round-shaped robots. A general mathematical model is introduced for both the aspects and specific solutions associated with the DAEDALUS specific robot approach. The combination of locomotion and balancing yields, besides naive approaches, an innovative controlling mechanism. This mechanism uses a virtual plane and map directly for controlling. This thesis yields a description and discussion of it, and establishes a first mathematical model. Chapter 5 describes the prototype, which is then evaluated in Chapter 6. The evaluation examines the approaches in the order in which their corresponding theory has been introduced. It proves the algorithms developed from their mathematical representation. Lastly, Chapter 7 concludes all the discussed topics and gives an overview of further research in the field.

# Chapter 2

# Background

The background to the linear telescopic actuator robot is the DAEDALUS project [68]. It was part of a 2020 SysNova study [4] and the Concurrent Design Facility (CDF) later [12], both run by the ESA. The 2020 SysNova study reached out for robotic solutions for exploring and mapping lunar lava caves and tubes. The largely unknown lunar cave environment poses a challenge, leading to huge risks for the robot and equipped sensors. Nonetheless, some aspects of the cave environment are already known, like the presence of very sharp rocks formed due to the lack of erosion by wind or similar factors. This is detrimental for most robot designs currently used in space exploration. Additionally, the considerable temperature range on the moon from 100 K to more than 500 K is very challenging for the robot. As a novel approach, the DAEDALUS project introduced a spherical robot. Figure 2.1 shows the blueprint of the robot, and Figure 2.2 illustrates a representative rendering.

The spherical shell of the robot protects the inner sensors such as the laser scanner, optical sensors, dosimeter, etc., from the described harsh conditions. Unlike wheeled robots, which are mostly designed for specific environments, a sphere functions well in a wide range of possible terrains. A sphere with shock-proof components can safely maneuver through rough tracks, which pose the risk of falling (from rocks or other objects). The mission formulated by the ESA was to descend into the lunar pit and then explore the caves once reaching the ground. An external crane carries out the descend. Therefore, ground exploration requires self-initiated locomotion. Factoring in all requirements, which are addressed in the final non-public report of the CDF study by ESA and in huge parts in the DAEDALUS report [68], leads to a combined locomotion design based on two different approaches.

The first is locomotion by rotation: The DAEDALUS sphere consists of inner and outer structures that are rotatable relative to each other. This was created to meet the need for a 360 ° coverage of the pit while descending, which is only possible either by increasing the number of sensors or making fewer sensors rotate. Due to the use of different sensors (optical and LiDAR (Light Detection and Imaging)) and failure-redundancy, the rotating approach was chosen. For the LiDAR with a horizontal coverage of 38.4 °, the following scanner-to-failure ratios emerge. With two rotating scanners, the robot can still achieve full coverage even if one scanner is out of order. However, any failure in one of nine non-rotating scanners leads to the loss of full coverage. With 18 non-rotating scanners, despite having nine more, only one failure still allows always

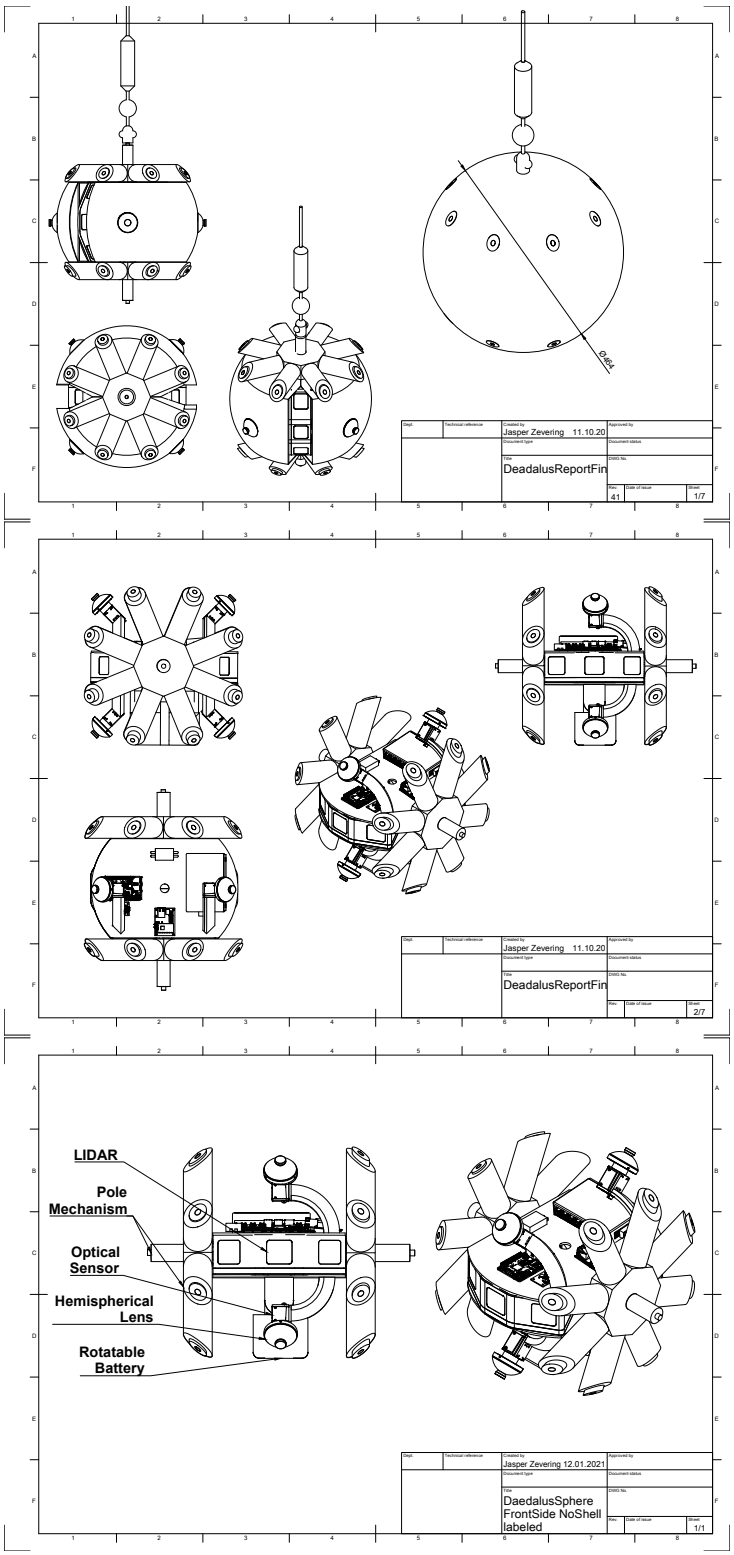



**Figure 2.1:** Blueprint of DAEDALUS with and without coverage.

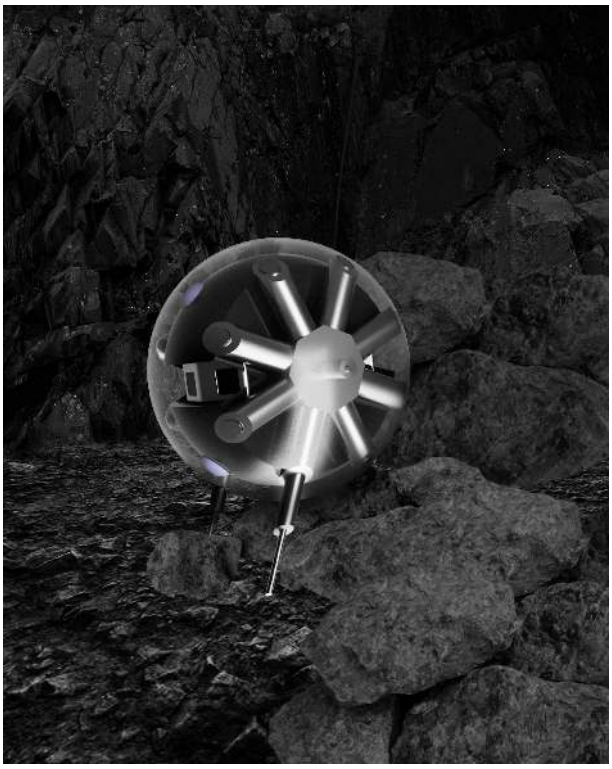


**Figure 2.2:** Rendering of the DAEDALUS sphere.

full coverage. Moreover, with two neighbored failures, full coverage is impossible. Therefore, the mechanism might add additional points of failure but decreases the required amount of scanners drastically. The center of mass of the robot is not at the center of the sphere, but the mass distribution of the outer structure is distributed evenly. Hence, the rotation of the two structures leads to translation, as the outer structure starts rotating, and the inner structure is always pointed toward the ground due to the low center of mass. Chapter 3 explains this principle in more detail. This technique facilitates good locomotion on relatively flat surfaces, but based on this approach, the sphere can only overcome obstacles of less than half its size in theory. Calculations with a realistic mass distribution presented in the DAEDALUS report showed that the sphere can merely overcome obstacles of up to 10 percent of the diameter of the sphere. Now, the second locomotion method is realized by the rods mounted inside the sphere as part of the outer structure. These rods are a result of the following requirements for the DAEDALUS sphere:

- Ability to overcome bigger obstacles than possible by solely utilizing mass distribution: Since the utilization of thrusters is not allowed by the ESA due to their huge environmental impact, another mechanism is required for lifting or pushing the sphere over obstacles.

- Extendable parts in or attached to the sphere for heat regulation.

**Figure 2.3:** Scanning mode of the DAEDALUS sphere. Poles on both sides extend in order to anchor the shell. Then the inner structure holding the LiDAR-scanner rotates, scanning the environment.

- A pole mechanism for pushing the sphere away from the pit wall in the worst-case scenario and the aperture of the pit does not allow a direct descent without touching the wall.
- Transparent/open sides for cameras and LiDARs: Due to the rotating scanners, the line of sight toward the side of the robot needs to be unobstructed, at least in some moments of the rotation.
- Rotation of the inner structure on the ground to ensure full coverage once inside the pit.

Figure 2.1 shows the final design with all the requirements and behaviors carried out by the approach of poles in a star-shaped way. The telescopic rods are extendable to the outside of the sphere, and they lead to rotation by pushing the sphere forward, once on the ground. Furthermore, the rods enable the DAEDALUS sphere to overcome obstacles up to its own size. Besides, the sphere uses the rods for additional tasks, like for heat control by extending them while descending or at the bottom of the pit to enlarge the surface area for radiation. Or, for pushing itself away from the pit walls during the descending phase and also for enabling its scanning mode. In this mode, four rods, two each on the front and rear, extend so that the sphere does not roll in any direction. Using the motors, which connect the inner structure to the outer one, leads to the rotation of the inner structure, while the outer structure does not rotate due to the extended rods. Figure 2.3 depicts this scanning mode. Chapter 4 will further discuss the advantages and disadvantages of the locomotion using extendable telescopic rods.

Combining the rotation of the center of mass and the rods gave DAEDALUS the ability to climb over perpendicular obstacles. A general pole-driven robot does this by rolling in an orientation where one pole is facing straight down, and if the mass of the sphere is perfectly balanced,

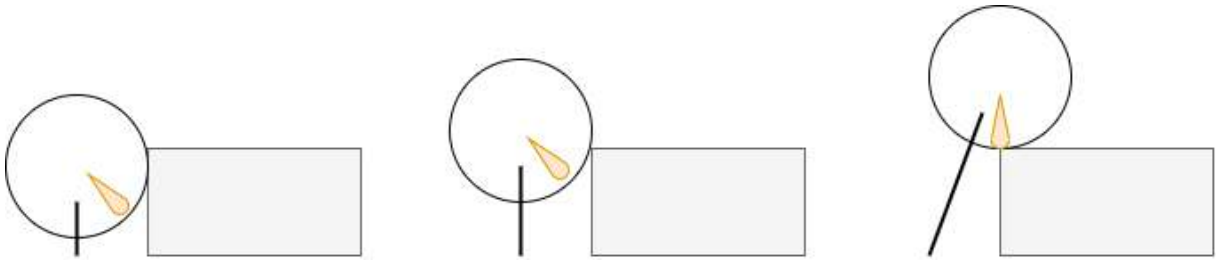


**Figure 2.4:** The concept of combining the internal rotation of the center of mass and rod extension to overcome perpendicular obstacles.

it pushes itself directly up. However, even minor imperfections or the starting orientation not being absolutely perfect leads to the robot falling over. In contrast, DAEDALUS rolls on an obstacle with one rod on each side down, but still avoids this risk. It extends the two rods so that they anchor to the ground. When starting the motors, the internal structure will rotate, while the outer structure remains stationary, just like in scanning mode. The center of mass the lies between the line of the extending poles and the perpendicular obstacle with a slight rotation. When pushed straight up, the sphere will not tip backward as the center of mass is shifted to the side of the obstacle. Figure 2.4 depicts this procedure.

During the DAEDALUS study, the telescopic pole mechanism was handled as a black box and not technically investigated, despite the rods being crucial for the main functions of the sphere. During the CDF study, experts from the ESA and the DAEDALUS team members discussed hydraulic and electric extension possibilities. The hydraulic solution was dismissed due to concerns about the fluid as it needs to be capable of withstanding the huge temperature range on the moon. Consequently, the electric solution was seen as a more straightforward and reliable approach. Chapter 5 will discuss the possible actuators separately from the discussion held during the DAEDALUS study.

The DAEDALUS sphere may not be the proof for the superiority of locomotion by rods. Yet, it is a justification of the concept, as it shows how and why the idea was created, and that one of its major advantage is the versatility of usage besides locomotion. This thesis focuses purely on the usage of rods as a locomotion system for a sphere and excludes all of their other possible uses. This also precludes the combination of the pole-usage with an internal change of center of mass or similar mechanisms like DAEDALUS introduces. A changing center of gravity together with the TLDR approach for spherical robots will be covered in future research.

# Chapter 3

# Related Work

Chapter 2 highlighted the unique requirements for the DAEDALUS project [68], which guided the development of the TLDR approach. The following sections discuss related/previous work on the locomotion of spherical robots.

First, we look into locomotion approaches that use mechanisms inside the sphere. These are the mostly used locomotion techniques and have a long history. Therefore, they are often well studied and have been examined through multiple prototypes.

Second, we present locomotion approaches that use mechanisms exceeding the spherical form and interact more directly with the environment outside the sphere to generate locomotion. Our TLDR approach uses telescopic extending linear actuators, extending from the inside to the outside of the sphere, and interact with the environment by pushing directly into the ground or generating leverage. Therefore we focus in this second part of the related work especially on the usage of linear actuators in this field. This field is much more unexplored, and only a few prototypes deal with this kind of mechanism.

## 3.1 Internal Mechanisms

One of the concepts of spherical robots is found in the patent "508,558" for a spherical toy, filed in 1893 by J.L. Tate [77]. Inside this sphere is a counterweight that will point toward the ground due to gravity. The counterweight is mounted on a middle-axis and is rotatable around it. Also a spring connects it to this axis. Due to the spring, the rotation is loaded by winding the counterweight in one direction. When released, the force of the spring will rotate the weight in the opposite direction. The weight pointing to the ground causes the shell to rotate and thus translate. DAEDALUS and several other spherical robots also use this rotational approach based on the internal rotation of weight. However, most prototypes nowadays use electric actuators to realize a controllable way of locomotion, and they do not require preliminary work like spring loading. Chase and Pandya describe this locomotion by shifting the center of mass in [19] as the governing principle. Also, the DAEDALUS project examined a spherical robot employing this locomotion approach. Figure 3.2 shows the said prototype. It uses its battery as a weight to generate torque. This is done by rotating it around the central axis of the sphere, which is parallel to the ground and perpendicular to the rolling direction. Furthermore, there is a

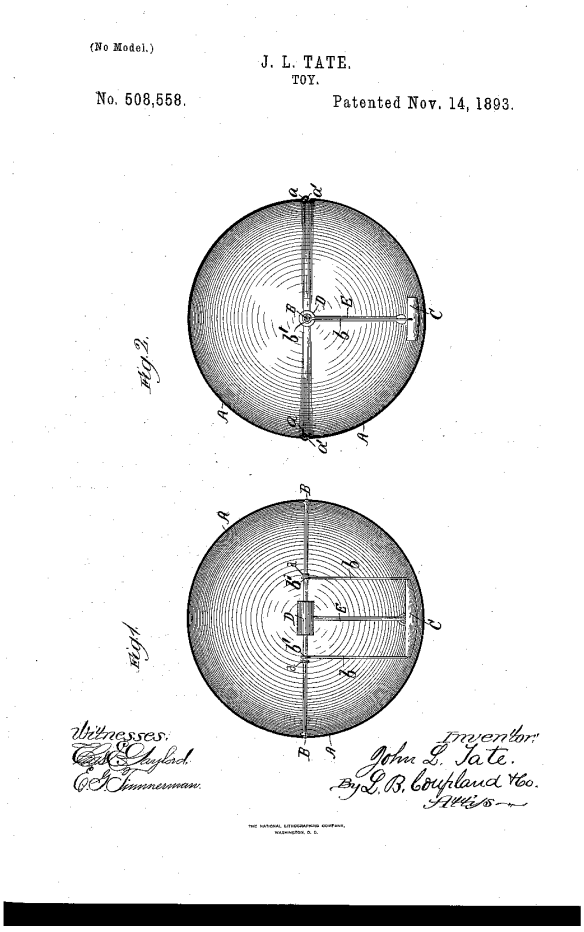



**Figure 3.1:** Patent 508,558 claimed by J.L. Tate [77]. A spherical toy, with an internal, spring-loaded counterweight, which is rotatable around the middle axis.

second actuator, which changes the angle in the rolling direction. This enables the sphere to roll to the side, whereto the weight is shifted. This approach was adopted in the design of the final DAEDALUS sphere [68]. Koshiyama and Yamafuji designed and built a spherical robot in 1993 using this approach [46]. Although its basic idea was rotation by internal weight rotation, the presented construction of the sphere was more complex than most of the here presented prototypes. They investigated basic movements and the equations of motions. Figure 3.3 shows this prototype. More recent robots utilizing this method of locomotion are the BYQ-III [50], and [6], in which Alizadeh and Mahjoob apply this locomotion technique on the water surface, as well as [63] and [54], each providing an application of pendulum-based rotation mechanism. Yang et al. used two completely rotatable pendulums, with one built perpendicular inside the other, whereas all other presented prototypes with two pendulums, limit the second rotation [84]. The evaluation finds no real advantage for the setup if the second inner pendulum is limited to -0.5 rad to 0.5 rad. [52] and [86] used the pendulum approach but split the weight into the left

**Figure 3.2:** Prototype examined during the DAEDALUS study [68]. It uses the concept of rotating internal weight for locomotion. For DAEDALUS, the weight is the battery. Furthermore, the weight is rotatable around the axis in the line of movement. This enables steering.

and right side. Rotating these two sides the same way produces the same effect as if it was one single weight. However, differing the speeds or directions of both weights generates a rotation on place around the axis vertical to the ground. Also, having both sides rotating in the same direction but with a bit of offset enables cornering. Generally, rotating internal weights provides a smooth, well-controllable locomotion but significantly limits the size of obstacles the robot is able to overcome.

Christopher Batten and David Wentzlaf from the Massachusetts Institute of Technology developed a spherical robot in 2001, named Kickbot, that employs the primary approach of rotating an internal structure versus an outer structure for movement [13]. The shell of the robot is split into two halves, with the intersection in line with the direction of the rotation. Both halves are controlled and actuated separately, despite being mostly instructed the same way. This enables Kickbot to perform additional movements and have a different way of steering. The possibility to perform on-place actions by rotating the halves against each other is provided but not explored deeply in the published work. Figure 3.4 shows the prototype. The closer the halves are and therefore the smaller the gap is, the lesser the effect of the rotation on the spot. This is because the rotation is dependent on a small but existing distance between both counter-rotating sides. In an abstract way, this is like there being a robot having two close wheels at its sides. With this, the purpose of the spherical shell, to protect inlaying payloads from environmental influences, is more complicated to achieve than with other approaches because of the two separate sphere halves.

In contrast, the robot Rotundus, created at the Ångström Space Technology Center for Project Mercury (ordered by the ESA) and later developed by the Rotundus AB corporation, was specially designed to withstand a wide variety of environmental influences [24, 36]. Its locomotion system works based only on the rotation of an internal weight, but it was designed to have the durability and ability to maneuver through many environments that may not otherwise be robot-friendly. It is completely sealed from the environment. Figure 3.5 depicts the Rotundus

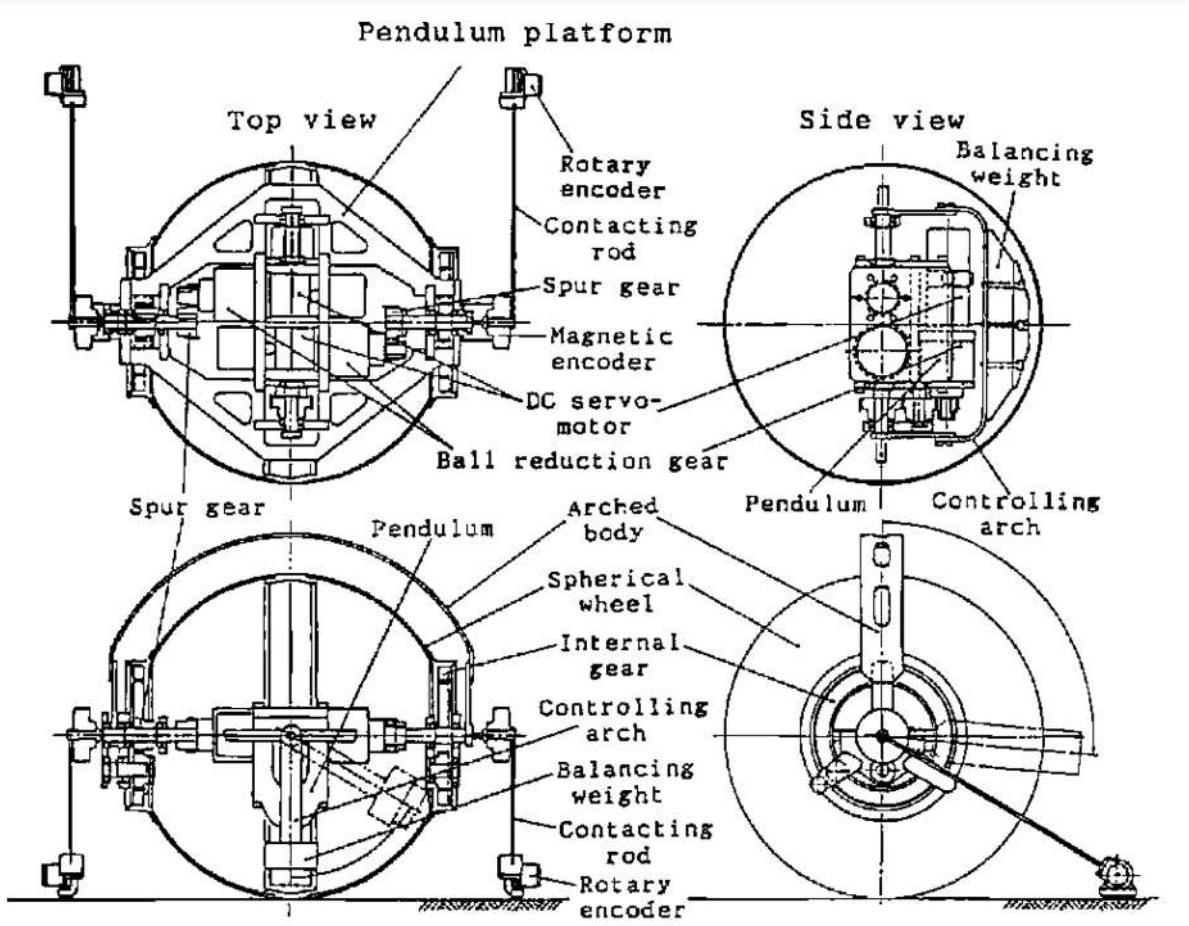



**Figure 3.3:** Prototype developed by Koshiyama and Yamafuji (1993) using the concept internal shifting of weight for locomotion [46].

and its successor GroundBot in different environments.

A second approach for locomotion of spherical robots is the use of internal momentum. This is done in [14] and [58], wherein internal gyroscopes were used to generate overall rotation. Using this technique, obstacles up to the own radius of the sphere can be overcome, although this implies a massive amount of generated momentum. The L.U.N.A. sphere employed a subtype of this approach, the concept of Impulse By Conversation Of Angular Momentum (IBCOAM) with two flywheels placed parallel on both sides of the sphere [87]. This was part of the research for determining suitable locomotion approaches for DAEDALUS. The shortcoming of this approach is the substantial power consumption for a relatively low amount of generated momentum. Moreover, the controllability is limited and only fully achievable with greater efforts than with a simple weight-shifting approach. Figure 3.6 depicts the L.U.N.A. sphere. A noteworthy work using both approaches, or at least parts of them, is the prototype Cyclops, developed by Chemel et al. [20]. Figure 3.7 illustrates this prototype. The inner structure of the prototype is connected to the sphere at two points and is entirely rotatable. Additionally, a rotational mass is mounted

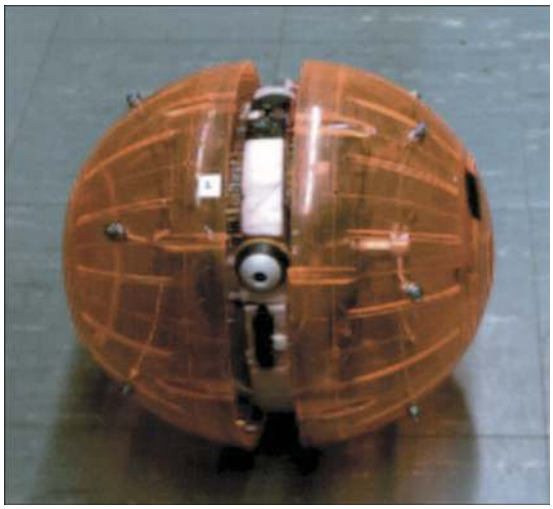


**Figure 3.4:** Kickbot by Christopher Batten and David Wentzlaf [13]. For locomotion it uses two rotatable sphere halves and an internal structure with low center of mass to.

at the bottom of Cyclops. This mass ensures a smooth rotation and controls angular velocity in the yaw angle (axis perpendicular to the ground). Qingxuan et al. go even further in [65] to add a third actuator, rotating the axis of the rotational mass, which they reduce to a flywheel. This enables their robot BYQ-V to use the generated momentum more precisely and in other ways than to just smoothen the locomotion. One such application is the standing mode, where the flywheel is rotated perpendicular to the ground. The generated momentum is then used to turn the main axis of the sphere upwards while rotating the flywheel to stay perpendicular to the ground. This is useful as the BYQ-V has an extendable camera, which is extended out of the main axis in standing mode, enabling a good, unobstructed 360 ° view due to the elevated position.

Another spherical robot design has a more or less independent inner unit in the sphere, called the "hamster-wheel model," developed by Tomi Ylikorpi and Jussi Suomela in [85], or a more formal "Internal Drive Unit (IDU)" by Halme et al. in [28]. The concept is very similar to a hamster running on a wheel, forcing it to rotate, with the difference being that the design incorporates a sphere and technical device instead of a running wheel and hamster respectively. Alves and Dias in [7] used a direct application of this concept. They used a four-wheeled vehicle inside of a sphere, which did not have any technical, electrical, or structural components despite the shell itself. The four wheels of the vehicle were independently controllable and driving them with different rotational speeds resulted in a curved trajectory of the overall sphere. Figure 3.8a shows this prototype.

The prototype showed very good behavior in terms of controllability. It is interesting to note that the vehicle enables the sphere to roll in any direction (not directly but by rolling curves; therefore, it is not holonomic), but the vehicle itself does not have an instantaneous center of curvature (ICC) and therefore, with respect to mobile robot kinematics, is not able to move in curves on flat ground without slip. In contrast to the two previously discussed designs, this kind of robot is energy efficient as the internal vehicle must move to a small degree, leading to the

**Figure 3.5:** Rotundus (left) and the successor GroundBot (right), originally investigated by Ångström Space Technology Center, now by a Swedish company [36][24]. Both use an internal rotatable weight for locomotion. The design focuses on durability in rough terrains. The Groundbot has two acrylic domes on each side for camera sensors.

rotation of the sphere against the rotation of counterweights, which have bigger levers. From the point of view of the internal device, it just rides a slope of a certain degree all the time. Bicchi et al. used the same idea of an internal vehicle, but with unicycle kinematics, using just one actuated wheel. The wheel is fitted onto a round structure with stabilizing wheels, but it is the only actuated wheel. It is rollable around two axes: for the rotation of the wheel itself and for steering. This enables the robot to roll in any direction on the ground without the need for rolling curves. Therefore, it is holonomic. Zhan et al. adopted this unicycle architecture in [89]. The direction of the monowheel is not rotated directly by an actuator but by a wheel, which rolls on the disk at the center. They successfully tested their prototypes, like with the previously mentioned Rotundus, in different rough environments like sand and water. In [22], Chen et al. constructed a sphere as an internal unit inside an outer sphere. Therefore, the inner sphere is actuated and always rolls on the bottom of the outer sphere due to its weight. The rest of the design employs the same principle as the other drive units, with the only difference being in the mechanism of the unit. Another design of the IDU incorporates two parallel wheels. Nguyen et al. implemented this in [61]. Their IDU has in many domains the same dynamics as the unicycle approach. However, instead of an external actuator defining the angle of the rolling direction of the wheel, it is now done by rotating the two wheels in the opposite direction or at different speeds. The electronics and structure are between the two wheels and do not exceed the radius of the wheel. The robot is called Virgo, and Figure 3.8b shows the prototype Virgo 2.5. In [35] and [34], Karavaev et al. introduce a prototype in which the IDU has three Swedish wheels. Figure 3.9 depicts the concept of this prototype. This holonomic IDU leads to an overall holonomic robot. A shortcoming of this was the difficulty to keep movement straight due to the peculiarities of Swedish wheels. Nevertheless, they managed to develop a well-controllable robot with an adapted control system. They state that the implementability of a control system for this kind of IDU relies on the absence of slip between the IDU and shell, as well as between the shell and ground. This is also problematic for other prototypes but especially for this type. This leads to an general problem affecting all non-fixed IDUs. Not

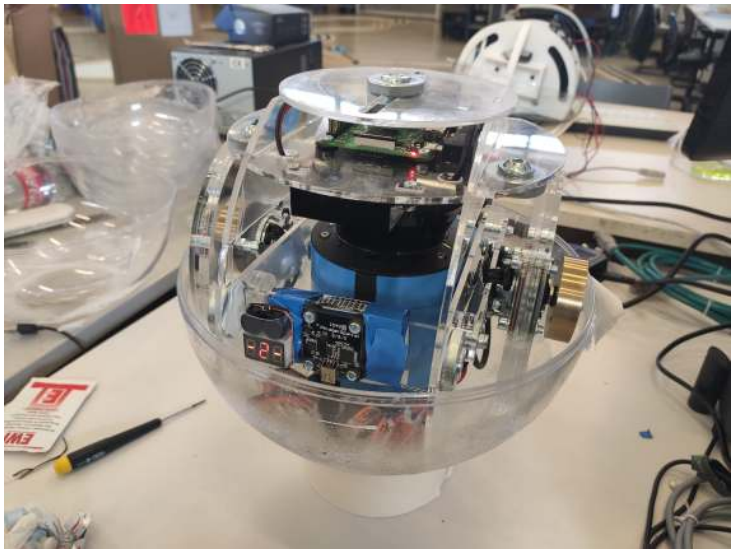


**Figure 3.6:** L.U.N.A. sphere that uses the IBCOAM (impulse by conversation of angular momentum) drive [87]. As the two flywheels on the side start rotating, the overall sphere starts rotating in the other direction due to the conversation of angular momentum.

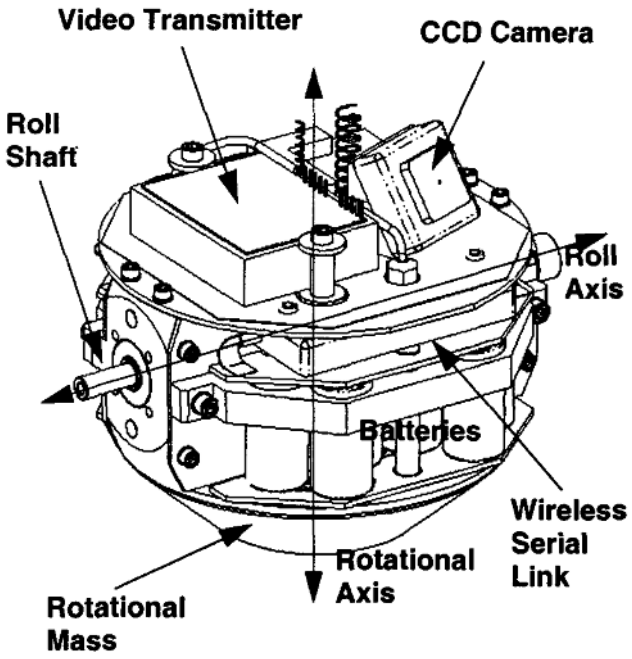
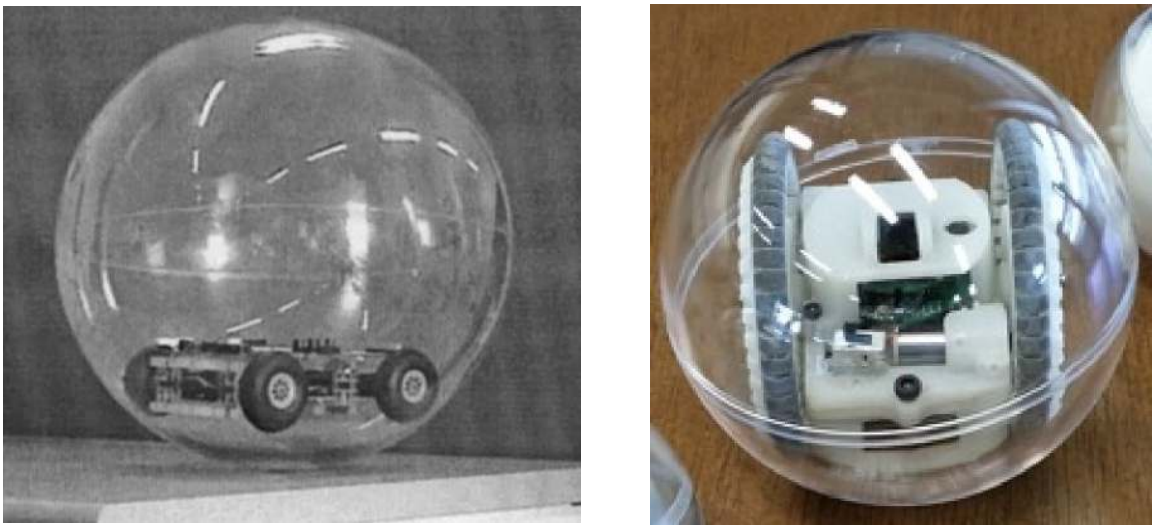



**Figure 3.7:** Prototype Cyclops by Chemel et al. [20].



**(a)** **(b)**

**Figure 3.8:** Prototype made by Alves and Dias (left) [7]; The robot Virgo 2.5 developed by Nguyen et al. (right) [61]. Both used an IDU for adopting the locomotion approach. Alves and Dias used a four-wheeled car like IDU, and Virgo 2.5 uses a two wheeled IDU.

**Figure 3.9:** Concept of Swedish-wheel IDU by Karavaev et al. [35] [34]. The IDU holds three Swedish wheels and is therefore holonomic.

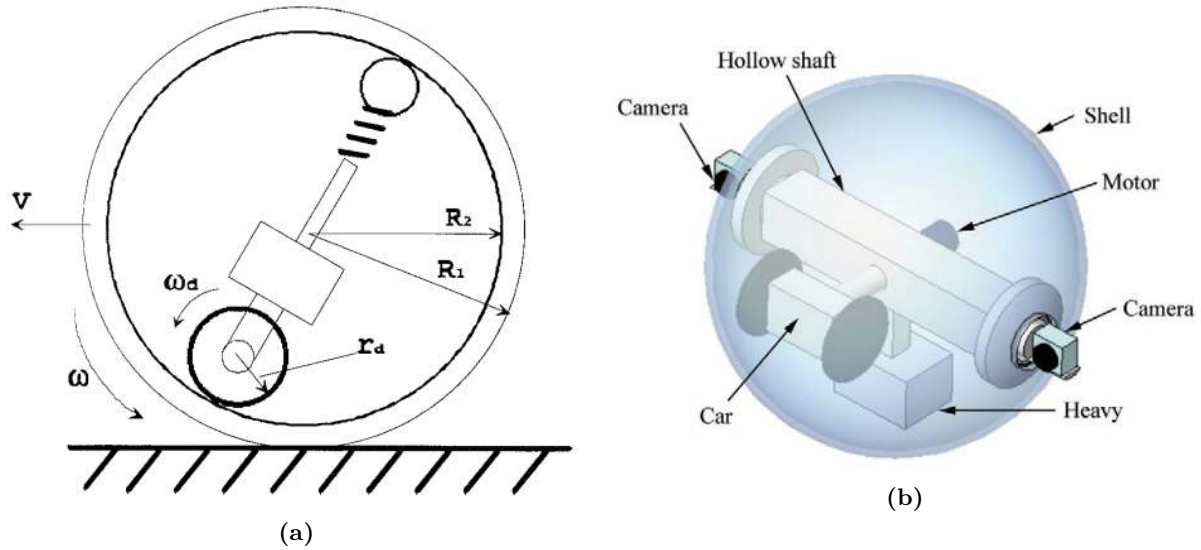
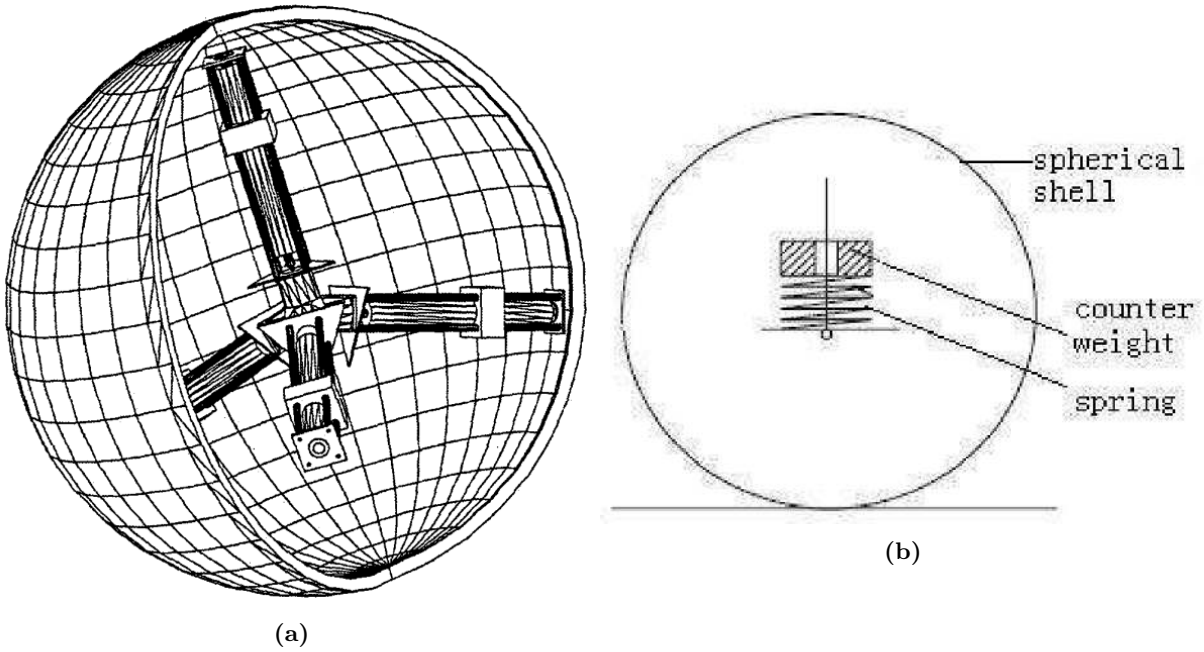



**Figure 3.10:** Left: The concept of the spherical robot with tightened IDU by Halme et al. with mathematical variables describing the setup [28]. Right: The concept of the BHQ-2 robot by Qiang et al. [64].

having any fixed connection to the sphere limits the use of IDU in rough terrains, as falling or steep slopes lead to unwanted behavior. Also, there is a need to have safety mechanisms if the sphere gets stuck. In this case, if the IDU does move forward, it climbs the shell, which does not rotate, and therefore the possibility of a flip is given, which puts the device on its back. For symmetrically built IDUs, this is not a problem, but for non-symmetrical IDUs, it leads to the complete dysfunction of the robot. Figure 3.8a depicts such non-symmetrical IDU proposed by Alves and Dias. One way of overcoming this is by adding mechanisms and/or structures to the IDU so that it is not loose inside the sphere. Halme et al. did this in [28] by adding a pole with a balance wheel on one side to a driving wheel of their IDU, which is inevitability at the bottom of the sphere due to its weight. A spring pushes the balance wheel onto the shell. This tightens the IDU inside the sphere due to the force of the spring while still being able to rotate and therefore drive the sphere. If the sphere gets stuck, the IDU will perform a looping if it is not otherwise prohibited by code, but this does not affect the the overall functionality of robot. Figure 3.10a highlights the concept. The downside to the design is the additional use of space inside the sphere, limiting the possible placement of sensors or other payloads. Furthermore, the additional wheel causes friction, depending on the strength of the spring, which leads to a lesser efficiency of the overall system despite having the same driving part at the bottom. The approach has also been examined for cylindrical robots by Reina et al. in [67]. A symbiosis of this clamped IDU and the internal rotation of weight has been done by Quiang et al. with the robot BHQ-2 in [64]. Figure 3.10b shows the concept of the prototype. A weight is mounted onto a bar at the center of the sphere, which is rotatable. The authors refer to it as "heavy," which we will follow for clarity. Besides rotating this axis by actuators mounted between the shell and the bar, a car device is mounted on the bar perpendicular to the heavy. As the mass of this driving unit is less than the heavy, the heavy will point downward and the driving unit




(a) (b)

**Figure 3.11:** Left: The concept of internal linear weight-shifting of the August robot, developed by Mojabi et al. in [55]. Right: The concept of a spherical robot created by Zhao et al. using the two-mass-one-spring approach [90].

horizontal. As the bar is mounted onto the shell, the driving unit will always be pressed onto the shell. Starting the unit will now lead to the rotation of the bar, which in turn rotates the heavy, thus leading to the rotation of the sphere.

Another approach for achieving the locomotion of a sphere is by internal weight-shifting, not by a rotation but in a linear way. Therefore, weights are mounted on bars, connecting the shell to the center of the sphere. The sphere initiates motion by shifting these weights along the bar, causing the weight distribution to change to one side, resulting in the rotation of the sphere. Mojabi et al. adopted this approach with the prototype of the August robot (2002) in [55], and Mukherjee et al. utilized this in 1999 as the concept of Spherobot in [57]. Figure 3.11a illustrates the concept of the August robot. Lux proposed a method of weight shifting using wires in an evaluation for Jet Propulsion Laboratory of NASA [51]. The payload contains winches that connect it to an anchor point on the shell using wires. By activating the winches, the payload changes position inside the sphere, changing the center of mass and therefore introducing rotation.

Zhao et al. used in [90] a modification of the weight-shifting approach. They created a jumping mechanism for locomotion by utilizing the two-mass-one-spring principle. One mass is the sphere itself, and the other is an internal weight. The mathematical system is built and the ground-laying concept of the spring-mass behavior is tested in an experimental setup, but there is no implementation on an actual setup. Figure 3.11b highlights the basic concept.

Lastly, some prototypes and robots use the sphere as the optimal shape for their environment, where on-board instruments need to be sealed and protected. Also, the sphere is one of the best

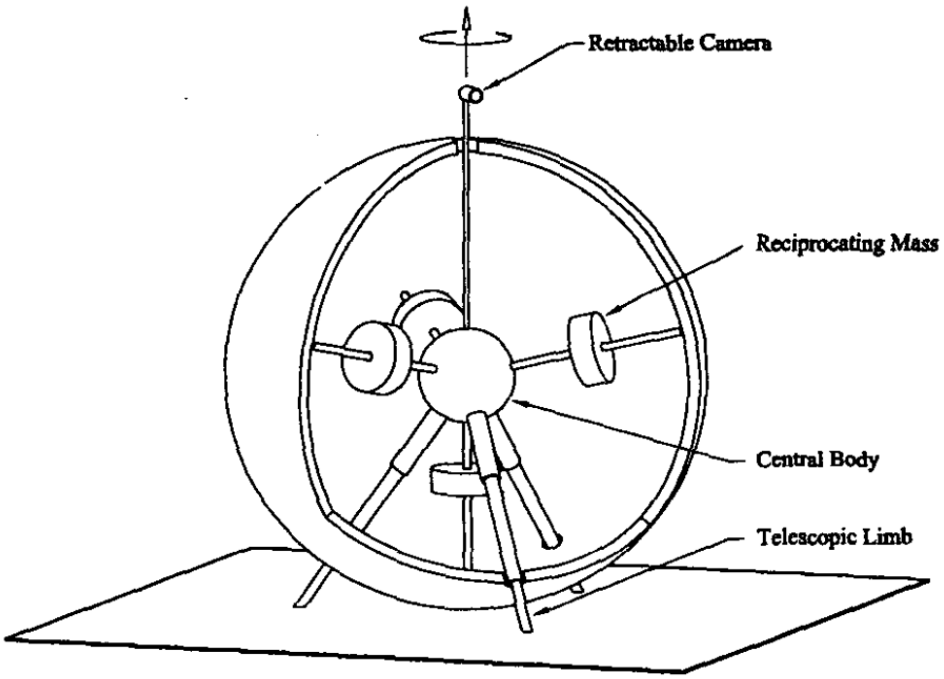



**Figure 3.12:** Spherobot by Mukherjee et al. in [57]; later patented by Mukherjee [56]. It uses internal weight shifting for locomotion and has three telescopic limbs for a standstill mode.

shapes when it comes to aerodynamic efficiency and flexibility for movement. Thus, several projects investigate spheres for underwater usage, but they do not use the sphere for locomotion purposes. Li et al. describe these advantages of the spherical shape for underwater missions but use a rotatable thruster concept, where four rotatable thrusters are mounted onto the sphere [48]. Guo et al. mention the same advantages but use water jet actuating [27]. These and similar robots for other rough terrains all use the spherical shape but are not limited to the spheroid form as their locomotion systems do not rely on it. Therefore, we will not investigate or list more such examples.

## 3.2 Extrinsic Linear Mechanisms

The fields of linear-driven robots are, in comparison with those of spherical robots, rather narrow. We concentrate on such robots, wherein the linear motion of the actuator is directly used for locomotion and not as a replacement for a rotational actuator. The Spherobot of Mukherjee et al. has been described previously because of its locomotion by shifting internal weight. However, it also has linear actuated limbs, which provide stability when the robot is at rest [57]. Figure 3.12 shows the rest position of the Spherobot with extended limbs. The limbs have no other purpose than providing a stationary position and are not meant for locomotion. Also, the Spherobot needs to orientate itself correctly to extend the limbs as they have only a very limited influence on the orientation. When the sphere touches the ground on the shell-surface surrounded by the

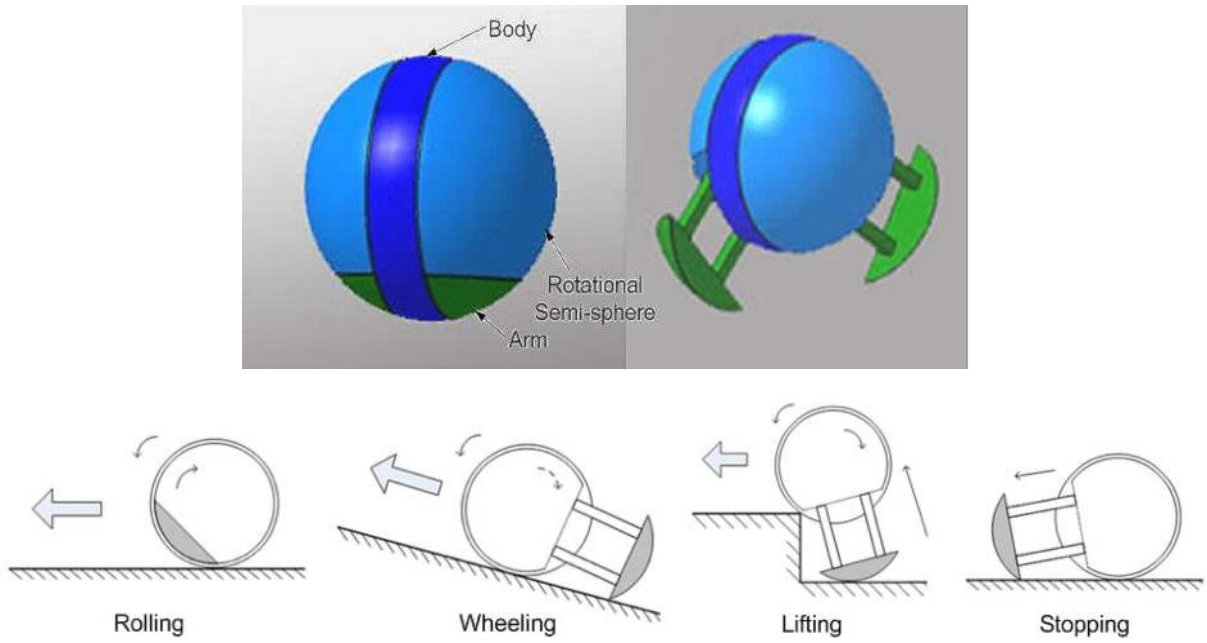
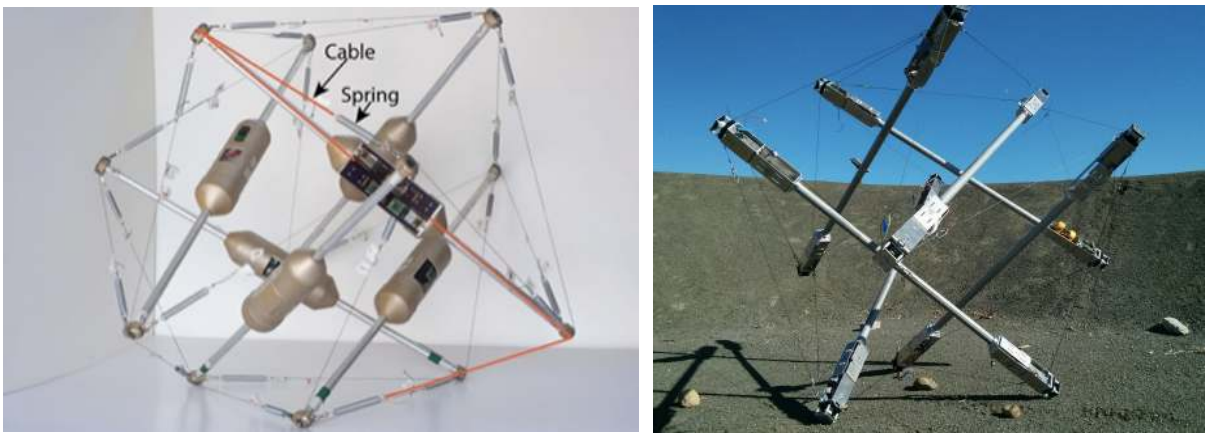



**Figure 3.13:** Above: The concept of Kisbot by Kim et al. [43]. The sphere consists of one center ring and two semispheres on each side. Each hemisphere has an extendable, heavy part integrated, which the authors refer to as "arms." Below: The movement functions of Kisbot.

three limbs, an extension of the limbs leading to stabilization is possible. This is due to the fixed position of the retractable camera, which makes other stabilizable orientations obsolete. Thirteen years later, a publication discussed the design and fabrication of the Spherobot, with Mukherjee as the first author of the original publication. However, this did not include the extendable limbs anymore, and the reasons for nor including them were not stated [78]. Also, the original publication as well as the corresponding patent granted to Mukherjee in 2001 [56] do not hold any information about the actual mechanism of the extension.

Kim et al. in 2010 [43] introduced another prototype using a linear actuator, named Kisbot. It consists of a sphere divided into three parts: one middle ring and two outer semispheres. The rotatable semispheres are mounted onto the middle part and are actuated. They also have a linear extendable part. The upper part of Figure 3.13 shows the concept. For locomotion, it has two driving modes. One is the pendulum-driven rotation. The extendable parts of the hemispheres have an increased mass in comparison with the rest of the hemisphere, therefore shifting the center of mass of the hemispheres toward the extendable parts. As just the inner ring is in contact with the ground, rotating the outer parts will lead to the rotation of the inner part and hence translation. The second mode is the wheeling mode, where both extendable parts are extended, resulting in the overall functioning of the robot like a one-wheel car. The lower left side of Figure 3.13 shows this mode. Nonetheless, the linear extension is also used for locomotion as it allows the sphere to push itself on top of obstacles. Further, extending these parts leads to an abrupt stop, if extended on the side where the robot is rolling towards. This is shown on the lower right side of Figure 3.13. Both driving modes were tested and evaluated.



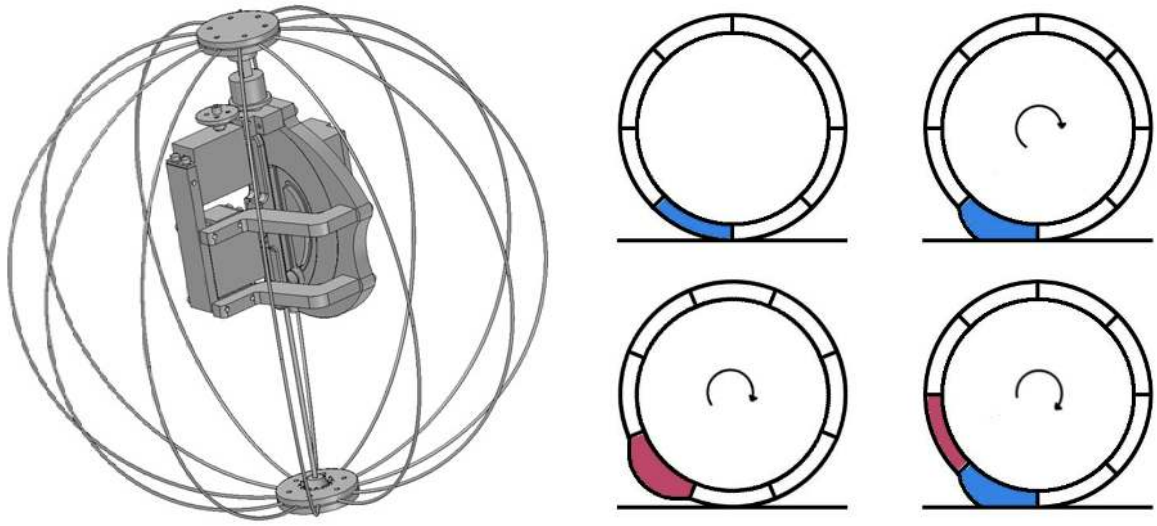

**(a)** **(b)**

**Figure 3.14:** Left: TT-3 prototype[21]. Right: SUPERball of the NASA Innovative Advanced Concepts (NIAC) Program [5]. Both use the concept of tensegrity. Tensioned cables between the rods give the robot its shape. These cables can be conveniently retracted or unrolled to change the shape of the robot.

Regarding lifting and stopping, the concept was only described and not implemented.

In 1961, Fuller introduced the concept of *tensegrity* [25]. As the combination of words "tension" and "integrity" in the word suggests, it deals with the concept of having structures, typically bars, not touching each other, due to the tension of cables between each other. These cables are put at tension to create three-dimensional structures. Robots using linear actuators, and not bars, leading to a complex changing and therefore an eventually moving structure, is a well-examined subgroup of robots [79][71][45]. There have been developments in the field of these spherical robots over the last years. In the research for the NASA Innovative Advanced Concepts (NIAC) Program, SUPERball (the Spherical Underactuated Planetary Exploration Robot Ball) has been developed [5, 16, 69]. The design is tailored for planetary missions and focuses on robustness, flexibility, and cost-efficiency. Figure 3.14b shows the design. It originates from the TT-1, TT-2, and TT-3 prototypes [21]. In the program, they used the tensegrity approach to build a robot that was directly considered a spherical robot, and therefore its dynamics were interpreted with the spherical shape in mind [39]. The TT-3 was especially analyzed on its locomotion and hopping behavior, including hopping not as a side effect but as valid locomotion [40]. Figure 3.14a shows the TT-3. The evaluations showed good, despite being complex, controllability, and limited sensing capabilities [16]. Also, the TT-3 requires huge computational power [5]. Although it is implemented using wires that are curled up and back, the behavior is the same as with linear actuators. There is also a wide range of robots consisting of these linear actuators only, but not in a specific spherical shape like the SUPERball or TT-3 [31, 80]. The basic approach for locomotion for this kind of spherical robot does not often differ from the prototype introduced in this work, but using an actuator influences the whole structure in fundamental ways, making it hard to evaluate single actuator actions.

Armour et al. developed a jumping spherical robot named Jollbot [8], as shown in Figure 3.15a. The basic approach of the robot is the deforming of the sphere, as described previously. Jollbot uses a single linear actuator to perform the deformation, basically flattening



(a) (b)

**Figure 3.15:** Left: Jollbot by Armour et al. [8]. Eight bendable poles are connected to each other at both sides, thus forming a sphere. The motor inside can pull both the connection points together, thereby reshaping the robot. Right: The rolling principle of the prototype made by Wait et al. [81]. The spherical robot consists of multiple cells, which extend and retract to generate rotation. This procedure is visualized in 2D.

out the sphere by pulling two ends toward each other. As its overall shape is more of an oval, the Jollbot is stabilized in that orientation, ensuring the retraction axis is perpendicular to the ground. Extending the mechanism promptly, which is technically achieved by a spring-loaded mechanism, will make the Jollbot jump. A pendulum mechanism performs all further locomotion actions. Despite looking like a completely new approach, its underlying mechanism is again a movement of the center of mass [9]. Sugiyama et al. [75] built a prototype with the same principal of deformation. This robot was not only able to climb steep slopes but also jump as high as twice its diameter. It uses shape-memory alloy (SMA) spokes connected from the center to the soft rubber shell. When applying voltage on one SMA spoke, it contracts, changing the shape of the robot. However, it needs to be said that the robot uses an external power supply. For achieving full autonomy, using an internal supply would add additional weight. It is a robot of small dimensions, having a diameter of 4 cm and weighing 3 g. Despite being designed as a spherical robot solution, its overall behavior and implementation were limited to a cylindrical problem.

Wait et al. also applied deformation for locomotion but with an actual spherical robot [81]. This shell of the robot is segmented into hexagon cells, which separately extend or retract. This enables the robot to move directly in all directions on the ground. To do so, it extends the cells on the opposite side of the intended direction of locomotion. This is visualized in Figure 3.15b. This is similar to our TLDR approach, introduced later, but instead of cells pushing the robot, we use telescopic poles extending from inside the sphere. Additionally, the prototype by Wait et al. uses the shape-changing aspect for the whole robot, besides the direct pushing effect of the

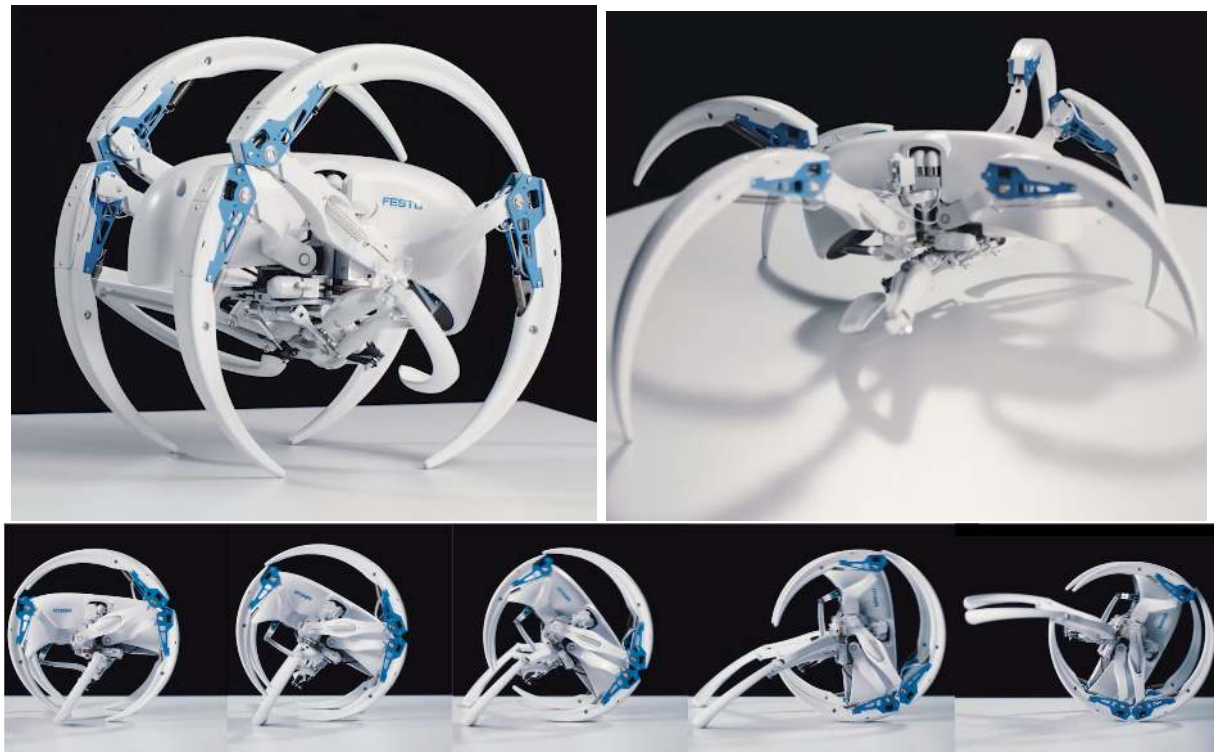


**Figure 3.16:** The BionicWheelBot of the Festo AG & Co. KG [66]. It imitates the flic-flac spider. While rolling, the round structure itself consists of six legs of the robot, and two are used for pushing. While walking, only the six legs are used for the walking itself as the two, which push during rolling, cannot be used due to mechanical restrictions. Above: The Bionicwheelbot in rolling (left) and walking modes (right). Below: The sequence of rolling. Screenshots from [37].

cells. We will later see that this is also done, at least partly, by our prototype. In their paper, they introduce a rough controlling strategy with two criteria for extending a cell. If one is met, the cell extends, otherwise it retracts. For the first proof of concept, this controller works well.

Last, we look into the robot, which seems to be the most similar to our TLDR robot. The company Festo AG & Co. KG created a robot that imitates the so-called flic-flac spider [66]. This spider has eight legs; besides walking, it uses a rolling/jumping movement which is best described by the acrobatic movement known as flic-flac. This spider is imitated by the BionicWheelBot, which also uses its leg setup for walking and rolling. Figure 3.16 depicts the two modes and in detail the rolling movement. The robot uses six of the legs to form a cylinder/sphere-like structure and uses the other two legs to push itself, generating rotation. The pushing legs have a joint that enables the pushing despite the lateral orientation. Nevertheless, this way of pushing for rotation comes very close to pushing with linear actuators. Only one pair of legs is used for pushing, raising the need for enough force generated by the used legs for a full rotation. Also, this leads to a varying rotational speed during one rotation. All this raises the question of why the spider has both modes, despite walking with legs being the more general mode of movement among spiders. Prof. Dr. Rechenberg, the scientific project leader for the BionicWheelBot,

stated in an interview [44], that movement by rotation has two main advantages. First, the increased pace. The spider doubles its speed when using the rotation mode. Second, efficiency. The spider overcomes twice the distance by rolling than by walking until total exhaustion. Due to the mating process and the need for precise movements, e.g., while hunting, the spider relies on walking in addition to rolling. We cannot answer if all this also applies to the BionicWheelRobot, as no further public information about the robot itself is available.

## 3.3 Summary

Multiple locomotion approaches for spherical robots have already been investigated. Most of them use an internal mechanism, which mainly limits the capabilities of the robot in terms of overcoming obstacles and the suitability for uneven terrain. Linear actuators, on the other hand, are not so widespread, with only a few related prototypes and approaches. For the rotation of a spherical robot using a pushing movement by an actuator, there appears to be only one prototype that was actually built, the BionicWheelBot. For this prototype, there exists no detailed public record of its mechanism and control structure. This led to the motivation to provide comprehensive basic calculations for our developed approach since the basic motion sequences and related effects on the robot have not yet been described.

# Chapter 4

# Approach

The overall challenge and goal is to create a stable rolling robot. Therefore, we split the problem into two domains. The first one is to initiate the rolling movement leading to translation, i.e., locomotion. The second one is to handle the instability to the left and right, i.e., balancing. In a broader sense, the stability pertaining to the front and the back needs to be considered, to ensure that the sphere stands still and does not tip over to the front if there are imperfections in the weight. This is part of both the domains, as stability to the front also means no movement frontwards. For this thesis, we consider it as the problem of locomotion. Therefore, the locomotion not only needs to initiate rotation but also stop it.

This thesis refers to the angle of a pole as follows: 0 rad is always downward directed perpendicular to the ground, which is also the gravitational vector, increasing clockwise until reaching $2\pi$ rad from the right side. $2\pi$ rad is referred to again as 0 rad, as we limit the described angle to positive values between 0 rad and $2\pi$ rad.

## 4.1 Movement

This section first covers the basic locomotion idea with rods, followed by the mathematical and physical theory for that. Then the special cases of braking and overcoming slopes and obstacles follow.

The robot can initiate movement using rods in two manners: with and without rotation. The movement without rotation is initiated by pushing the sphere away from the wall or the ground by perfectly balancing it straight upwards. Combining this straight upward movement with a calculated imperfection of the mass distribution of the sphere leads to a fall in the predefined direction, which overall constitutes a locomotion approach. As this exerts huge impact forces on the sphere and does not really control its movement, and as the other approach always depends on a kind of wall, we will not focus on them. Also, the hopping rovers described in Chapter 3 can be developed using rods and lead to movement without relying on rotation. However, this approach needs an internal stabilizing mechanism to control the hops. Figure 4.1 shows the three non-rotating approaches described.

The other sort of movement is by rotation. Here, the rotation of the sphere leads to translation as the friction between the ground and the shell of the robot causes it to roll forward

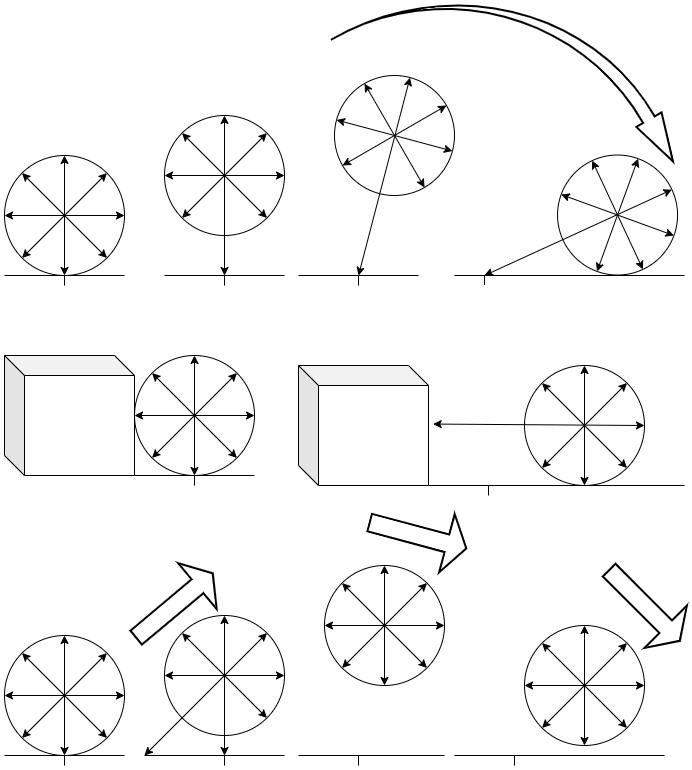


**Figure 4.1:** Locomotion approaches using rods without relying on rotation. From top to bottom: falling, pushing, and jumping.

rather than slipping at the same place. Therefore, the poles are primarily responsible for the rotation and secondarily, for translation. Also, for rotation, there are two main ways to initiate it. The most intuitive is the rotation by pushing against the ground. The rods that lay on the opposite side of the desired direction of rolling extend, which push into the ground and generate torque for the sphere, leading to rotation, as shown in Figure 4.2. For this approach, two parameters need to be found. The first parameter $\beta$ determines from which angle onward the extension begins. $\beta$ must be greater than 0 rad as a pole extending at exactly 0 rad will not create torque but straight force, pushing the sphere upwards. $\beta$ needs to be chosen in such a way that the extension completely leads to torque and not some linear motion. The second one is $\alpha$, the angle until which the robot should extend a rod. Despite reasons due to the control strategy, there are geometrical restrictions for $\alpha$. It cannot be $0.5\pi$ rad or larger as a pole extending between $0.5\pi$ rad and $1.5\pi$ rad never touches the flat ground. Between $1.5\pi$ rad and $2\pi$ rad, the pole extension will work against the desired direction. The possible extended length of the poles also determines the maximum $\alpha$. The larger the chosen $\alpha$, the longer the pole needs to be extended to reach the ground, up till the point where the pole is extended but

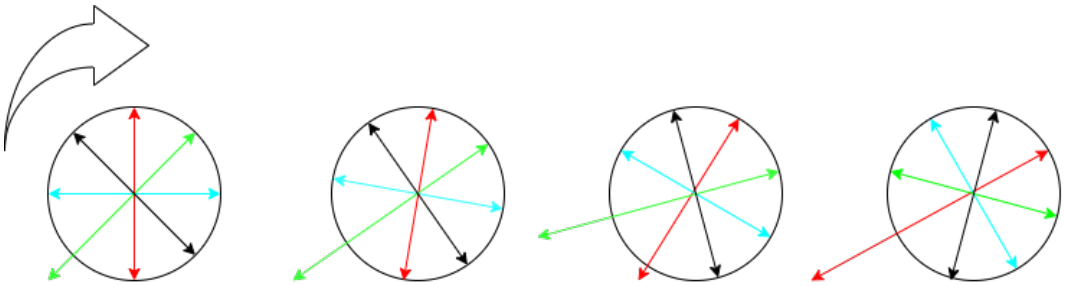


**Figure 4.2:** Initiating the rotation by pushing. The sphere rolls to the right side. The arrows symbolize poles and are colored for better identification over the rotation.

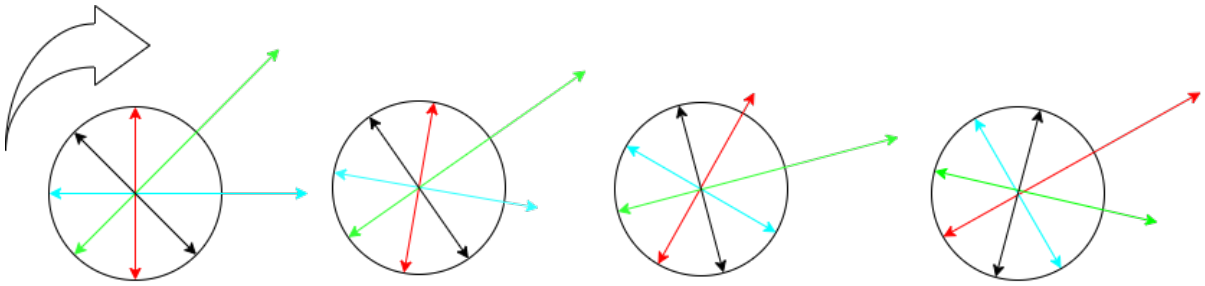


**Figure 4.3:** Initiating rotation by leverage. The sphere rolls to the right side. The arrows symbolize poles and are colored for better identification over the rotation.

not pushing anymore because of the lack of contact with the ground. From that moment on, the pole works against the pushing force due to its leverage. This same behavior can occur if the poles only have one extending speed and a continuous rotation is desired. The steeper the angle to the ground is, the faster the poles need to be extended to cause a constant rotation. When designing such a system with mono-speed rods only, one task is to determine from which angle the extending rod does no longer matches the desired continuity of the rotation speed. Subsection 4.1.2 investigates this mono-speed behavior in more detail.

The second approach of creating torque for the sphere is by leverage. As the poles have a weight, extending them without any contact to the ground provides leverage. Extending the poles between $\pi$ rad and $2\pi$ rad leads to leverage, creating a clockwise rotation that leads to translation. This is shown in Figure 4.3. There are no accurate parameters to be determined as each pole extends over $\pi$ rad and it retracts so that it does not collide with the ground. This relies entirely on the mass of the poles, and with too lightweight poles, the friction becomes more significant than the generated torque. Nevertheless, it is still usable as support for the pushing approach.

Algorithms 1, 2, 3, and 4 show the pseudocode for four different implementations: only pushing, only pushing but ensuring just one pole is extending, only working with leverage, and the combination of pushing and using leverage.

**Algorithm 1:** Push-Only Movement Algorithm

```
foreach pole in poles do
    if α < pole.getAngle() < 0.5π − β then
        extend pole
    else
        retract pole
    end
end
```

**Algorithm 2:** Push-Only Single Movement Algorithm

```
foreach pole in poles do
    if α < pole.getAngle() < 0.5π − β and nextpole.getAngle() > pole.getAngle()
     then
        extend pole
        retract all other poles
    end
end
```

**Algorithm 3:** Leverage-Only Movement Algorithm

```
foreach pole in poles do
    if π < pole.getAngle() <= 1.5π then
        extend pole
    else if pole.getAngle() > 1.5π then
        retract pole (ground avoidance mode)
    else
        retract pole
    end
end
```

**Algorithm 4:** Leverage and Push Movement Algorithm

```
foreach pole in poles do
    if π < pole.getAngle() <= 1.5π or α < pole.getAngle() < 0.5π − β then
        extend pole
    else if pole.getAngle() > 1.5π then
        retract pole (ground avoidance mode)
    else
        retract pole
    end
end
```

### 4.1.1 Mathematical Representation

We will now setup the mathematical representation for the robot and the locomotion of it. We will first investigate the locomotion by pushing an then by leverage.

#### Pushing

Let $l$ be the length of extension of a single rod, $l_{\max}$ be the maximum length of extension possible, and $r$ be the radius of the sphere. Then, $\alpha$ is limited by

$$\alpha \leq \arccos\left(\frac{r}{r + l_{\max}}\right) < \frac{\pi}{2}\,. \tag{4.1}$$

If the angle $\zeta_x$ of a pole is between $\alpha$ and $\beta$, then its length $l_x$ must be extended to

$$l_x = \frac{r}{\cos(\zeta_x)} - r\,, \tag{4.2}$$

in order to touch the flat ground. To obtain the extension speed for a given desired rotational velocity of the sphere, the derivative is formed, leading to

$$\dot{l}_x = \frac{d}{dt}\left(\frac{r}{\cos(\zeta_x)} - r\right) = r \cdot \dot{\zeta}_x \tan(\zeta_x)\sec(\zeta_x)\,, \tag{4.3}$$

where $\dot{\zeta}_x = \dot{\zeta}_1 = ... = \dot{\zeta}_n$ is the same for all poles and is the rotation speed of the sphere. Figure 4.4 shows the speed at which a rod needs to extend for a 1 m radius sphere if it needs to roll at 1 rad/s. Therefore, it can be noted that the maximum possible speed of one pole also limits $\alpha$ and/or the maximum possible rotation speed. By rearranging Equation (4.3) for $\dot{\zeta}_x$ and substituting the rotation speed of the sphere $\omega$ for $\dot{\zeta}_x$ we get

$$\omega = \frac{\dot{l}_x}{r \cdot \tan(\zeta_x)\sec(\zeta_x)}\,. \tag{4.4}$$

For the maximum possible $\omega$, $\omega_{max}$ now applies

$$\omega_{\max} = \frac{\dot{l}_{x,\max}}{r \cdot \tan(\alpha)\sec(\alpha)}\,, \tag{4.5}$$

as $\alpha$ is the maximum possible angle $\zeta_x$. With a lower $\alpha$, a faster $\omega$ can be achieved with the same maximum $\dot{l}_x$. However, with too low $\alpha$, the number of rods touching the ground simultaneously decreases, consequently reducing the applied force to generate torque. Let $n$ be the number of rods and $n_f$ represent the desired number of rods simultaneously touching the ground, then $\alpha$ should be chosen such that

$$\alpha \geq \frac{2\pi}{n} \cdot n_f + \beta\,. \tag{4.6}$$

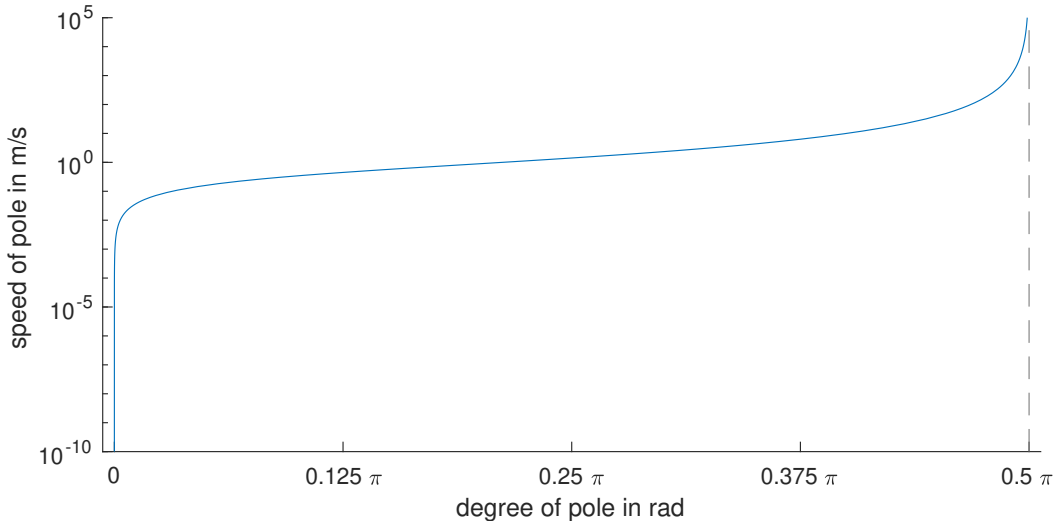



**Figure 4.4:** The speed of a rod needed at a certain angle for a rotation speed of 1 $^{\text{rad}}/_{\text{s}}$ for a sphere of 1 m radius.

### Leverage

For using leverage for locomotion, the only parameter to calculate is when to start retracting the pole. When ignoring the maximum velocity of a pole, we need to ensure that

$$l_x < \frac{r}{\cos(\zeta_x)} - r\,. \tag{4.7}$$

As the premise of multiple poles touching the ground is not given, each pole works separately, extends as soon as possible ($\zeta_x > \pi$ rad), and only needs to take care of itself when retracting. Moreover, the speed of the extension does not directly influence the rotational speed of the sphere, but it achieves faster, greater leverage. Therefore, it is important for poles to extend as fast as possible. One restriction to the retraction is if the $\dot{l}_{x,\max}$ is fast enough to retract the pole without touching the ground but staying close to it. This means the retraction waits until the next moment when the pole would touch the ground. The change of distance between the ground and the shell at a given angle $\eta$ between $1.5\pi$ rad and $2\pi$ rad is therefore given by

$$\dot{l} = \frac{d}{dt}\left(\frac{r}{\cos(\eta)} - r\right) = r \cdot \dot{\eta} \cdot \tan(\eta) \cdot \sec(\eta)\,, \tag{4.8}$$

where $\dot{\eta}$ is $\omega$. Therefore, at the moment when the pole touches the ground, which is at an angle of $2\pi - \arccos\left(\frac{r}{r+l_{x,\max}}\right)$, its (retraction) speed needs to be fast enough to avoid a collision as the nearer the pole gets to $2\pi$ rad, the slower the retraction has to be. The condition for this is

$$\dot{l}_{x,\max} > \left| r \cdot \omega \cdot \tan\left(2\pi - \arccos\left(\frac{r}{r+l_{x,\max}}\right)\right) \cdot \sec\left(2\pi - \arccos\left(\frac{r}{r+l_{x,\max}}\right)\right)\right|\,. \tag{4.9}$$

In this case, the retraction is managed by calculating whether the pole touches the ground in the next moments or not. The specification of the duration of this time period until collision

depends on the calculation speed, the responsiveness of the actuators, etc. If the condition of Equation (4.9) does not hold for the current $\omega$, the retraction needs to be initiated at full speed even if the pole tip is still away from the ground. Therefore, we first find the angle from which the pole is faster than needed and hence capable of retracting just before touching the ground. Let $\gamma$ be the angle at which this is possible, then the Equation (4.8), using $\gamma$ as $\eta$, $\omega$ as $\dot{\eta}$, and $\dot{l}_{x,\max}$ as $\dot{l}$, becomes

$$-\dot{l}_{x,\max} = r \cdot \omega \tan(\gamma) \sec(\gamma)\,. \tag{4.10}$$

Solving this for $\gamma$ and substituting $\frac{-\dot{l}_{x,\max}}{r\cdot\omega}$ with $k$, results in

$$\gamma = \pi + 2\arctan\left(\frac{1}{2}\sqrt{\frac{1}{k^2}+4} + \frac{\sqrt{\frac{1}{k^2} - \frac{4}{k\sqrt{1/k^2+4}} - \frac{1}{k^3\sqrt{1/k^2+4}s}}}{\sqrt{2}} - \frac{1}{2k}\right). \tag{4.11}$$

This represents the angle $\gamma$, the angle from which the pole starts retracting with the ground-avoidance approach, as a function of the maximum possible speed of the pole, the rotation speed of the sphere $\omega$, and the radius of the sphere. If the computational power is limited, this equation is linearizable, but as the working points will differ, this needs to be done for every robot when implemented. To avoid contact with the ground due to slight changes of $\omega$, $k$ is calculated with a safety factor if the implementation shows ground contact on the leverage side. Then, the maximum velocity is assumed a little slower, leading to the retraction speed being faster than needed. As $\dot{l}_{x,\max}$ goes linear into $k$, taking just a percentage of $k$ will have this effect. Therefore, it is advised for practical implementation to use $k = 0.99\frac{-\dot{l}_{x,\max}}{r\cdot\omega}$ where 0.99 reduces the assumed maximum speed and thus leads to a smaller $\gamma$, which will resolve uncertainties in the calculation. Alternatively, $\gamma$ decreases manually, but as the maximum speed and contact point behavior is not linear, this is not advised. Having found $\gamma$, we back-calculated the angle at which the retraction needs to start. Let $\epsilon$ be the angle at which retraction must start, for the pole to be as near to the ground as possible without touching it at the angle $\gamma$, then

$$\epsilon = \gamma - \omega \cdot \frac{l_{x,\max} - \left(\frac{r}{\cos(\gamma)} - r\right)}{\dot{l}_{x,\max}}\,. \tag{4.12}$$

Figure 4.5 visualizes the overall behavior of retraction with focus on the angle $\gamma$, assuming a finite maximum retraction velocity. It is shown that complete retraction at a $\zeta$ of $2\pi$, at is own is an insufficient condition for the collision avoidance, as the blue line fulfills this criterion, but still is above the red line, i.e., collision. With a known $\gamma$, the angle at which retraction must start can be calculated, ensuring always to avoid contact with the ground. Once reaching $\gamma$, it is up to the implementation, if the retraction shall be done at full speed or always have a maximum possible extension at an adapted speed. If the resulting $\epsilon$ is smaller than $1.5\pi$ rad, the initial extension needs to be carried out before reaching $\pi$ rad to achieve at least once full extension before retraction. There will still be a leverage effect as weight is shifted to the right side of the sphere. Figure 4.6a highlights this behavior, of an extension start at an angle smaller than $1.5\pi$ rad.

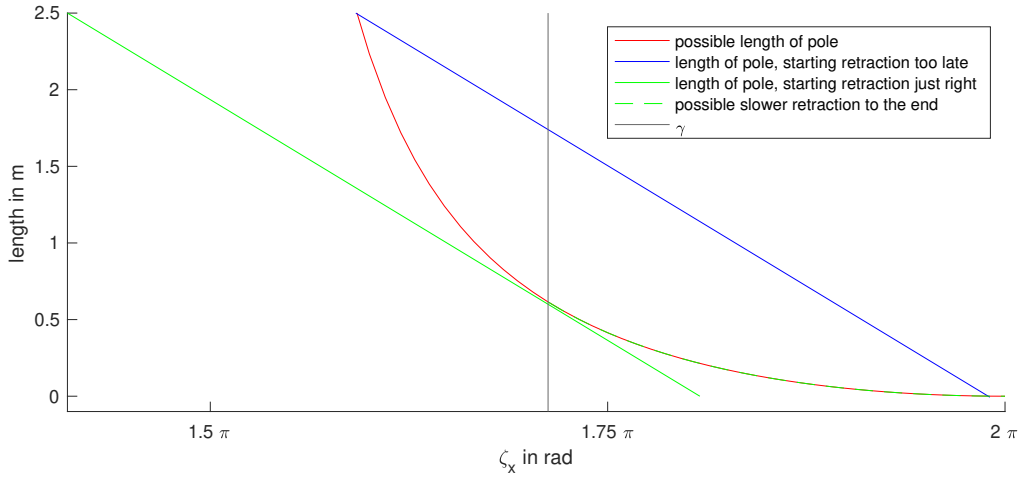



**Figure 4.5:** Visualization of pole retraction when using the leverage approach. The red line symbolizes the allowed length of extension of the pole without touching the ground. If a point lies in the area above this red line, this indicates collision with the ground. The straight lines show the length of the pole when extending and starting too late (blue) or just at the right time (green), at full possible speed. $\gamma$ is the angle at which the required retraction speed becomes slower than the maximum possible retraction speed for the pole. Therefore, the green dashed line shows the possibility of retraction slower to the end. The calculation was done for $\omega = 1\,\mathrm{rad/s}$, a possible retraction speed $\dot{l}_{x,\max}$ of $2\,\mathrm{m/s}$ and assuming a 1 m radius.

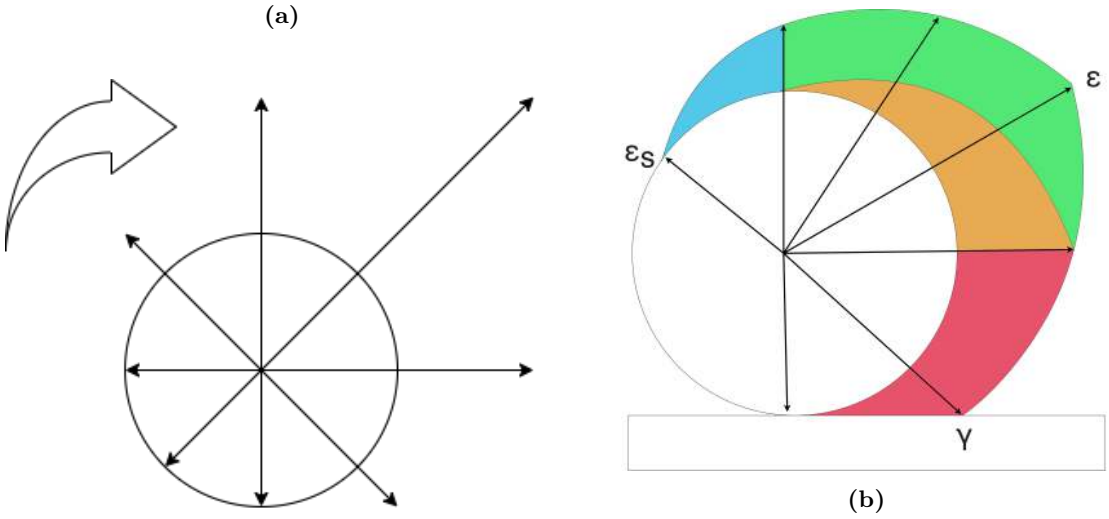



**Figure 4.6:** Left: Leverage approach if $\pi\,\mathrm{rad} < \gamma < 1.5\pi\,\mathrm{rad}$. This leads to the extension starting between 0 rad and $\pi$ rad. This still causes rotation. Right: Visualization of $\epsilon_s$, $\epsilon$, $\gamma$ and the relevant area representing the integration of the overall torque. Blue: Area where torque is generated in the opposite direction than intended. Orange: Area where torque is generated if the extensions start at $\pi$ rad. Full extension is not reached due to the short extension time. Green: Area where torque is generated if the extension starts at $\epsilon_s$, in addition to the orange area. Red: Area that both approaches cover.

Let $\epsilon_s$ be the angle at which extension needs to start to reach full extension once at $\epsilon$, then

$$\epsilon_s = \epsilon - \omega \cdot \frac{l_{x,\max}}{\dot{l}_{x,\max}} \,. \tag{4.13}$$

This raises the question of whether this is counterproductive in comparison to an extension starting at $\pi$ rad, not reaching full extension but at least not creating leverage on the wrong side. The more detailed physical representation of the robot will be derived in Subsection 4.1.3. However, for the evaluation, if the assumption of an $\epsilon_s$ is valid, we anticipate that a single mass point at distance $r(\zeta)$ introduces a torque $\tau$ of

$$\tau = r(\zeta) \cdot (-\sin(\zeta)) \cdot q \,, \tag{4.14}$$

where $q$ is a constant that will be specified in Subsection 4.1.3. Thus, we integrate the torque generated by one pole over the process of one rotation. Figure 4.6b visualizes the described angles $\gamma$,$\epsilon$, and $\epsilon_s$. Also, it visualizes the integration areas where torque is generated. We integrate $\tau$ from Equation (4.14), leading to

$$\tau = \int_0^{2\pi} r(\zeta) \cdot ((-\sin(\zeta))) \cdot q \, d\zeta \,. \tag{4.15}$$

If we start extending at $\pi$, hence ignoring the calculated values for the extension end retraction of $\epsilon_s$ and $\epsilon$, this leads to

$$\tau_\pi = \int_\pi^{2\pi} r(\zeta) \cdot (-\sin(\zeta)) \cdot q \, d\zeta \tag{4.16}$$

$$\begin{aligned} \Leftrightarrow \quad \tau_\pi = &\int_\pi^{1.5\pi} r(\zeta) \cdot (-\sin(\zeta)) \cdot q \, d\zeta + \int_{1.5\pi}^{\gamma} r(\zeta) \cdot (-\sin(\zeta)) \cdot q \, d\zeta \\ &+ \int_\gamma^{2\pi} r(\zeta) \cdot (-\sin(\zeta)) \cdot q \, d\zeta \end{aligned} \tag{4.17}$$

$$\begin{aligned} \Leftrightarrow \quad \tau_\pi = &\int_\pi^{1.5\pi} \dot{l}_{\max} \cdot (\zeta - \pi) \cdot (-\sin(\zeta)) \cdot q \, d\zeta + \int_{1.5\pi}^{\gamma} \dot{l}_{\max} \cdot (\pi - \zeta) \cdot (-\sin(\zeta)) \cdot q \, d\zeta \\ &+ \int_\gamma^{2\pi} (\frac{r}{cos(\zeta)} - r) \cdot (-\sin(\zeta)) \cdot q \, d\zeta \end{aligned} \tag{4.18}$$

Here, we made one simplification, assuming that the retraction starts from $1.5\pi$ rad. This will always lead to less torque than in reality as the decreased speed of the retraction from $\gamma$ will result in an overall less extension in the first part. Therefore, this is a pessimistic estimation. If

we take $\epsilon_s$ as the starting point, also using the $1.5\pi$ assumption, this leads to

$$\begin{aligned}\tau_{\epsilon_s} &= \int_{\epsilon_s}^{\epsilon} r(\zeta)\cdot((-\sin(\zeta)))\cdot q\,d\zeta + \int_{\epsilon}^{1.5\pi} r(\zeta)\cdot(-\sin(\zeta))\cdot q\,d\zeta \\ &\quad + \int_{1.5\pi}^{\gamma} r(\zeta)\cdot(-\sin(\zeta))\cdot q\,d\zeta + \int_{\gamma}^{2\pi} \dot{l}_{\max}(\zeta-\epsilon)\cdot(-\sin(\zeta))\cdot q\,d\zeta \end{aligned} \tag{4.19}$$

$$\begin{aligned}\tau_{\epsilon_s} &= \int_{\epsilon_s}^{\epsilon} \dot{l}_{\max}(\zeta-\epsilon_s)\cdot((-\sin(\zeta)))\cdot q\,d\zeta \\ &\quad + \int_{\epsilon}^{1.5\pi} ((\epsilon-\epsilon_s)\dot{l}_{\max} - \dot{l}_{\max}\cdot(\zeta-\epsilon)))\cdot(-\sin(\zeta))\cdot q\,d\zeta \\ \Leftrightarrow \quad &\quad + \int_{1.5\pi}^{\gamma} \dot{l}_{\max}\cdot(\pi-\zeta)\cdot(-\sin(\zeta))\cdot q\,d\zeta \\ &\quad + \int_{\gamma}^{2\pi} (\frac{r}{cos(\zeta)} - r)\cdot(-\sin(\zeta))\cdot q\,d\zeta\,. \end{aligned} \tag{4.20}$$

The hypothesis is that Equation (4.20), calculating $\tau_\pi$, is less than or equal to Equation (4.18), calculating $\tau_{\epsilon_s}$. Therefore, we set

$$\tau_\pi \leq \tau_{\epsilon_s} \tag{4.21}$$

$$\begin{aligned}&\int_{\pi}^{1.5\pi} \dot{l}_{\max}\cdot(\zeta-\pi)\cdot(-\sin(\zeta))\cdot q\,d\zeta + \int_{1.5\pi}^{\gamma} \dot{l}_{\max}\cdot(\pi-\zeta)\cdot(-\sin(\zeta))\cdot q\,d\zeta \\ &+ \int_{\gamma}^{2\pi} (\frac{r}{\cos(\zeta)} - r)\cdot(-\sin(\zeta))\cdot q\,d\zeta \\ \Leftrightarrow\ &\leq \\ &\int_{\epsilon_s}^{\epsilon} \dot{l}_{\max}(\zeta-\epsilon_s)\cdot((-\sin(\zeta)))\cdot q\,d\zeta + \int_{\epsilon}^{1.5\pi} ((\epsilon-\epsilon_s)\dot{l}_{\max} - \dot{l}_{\max}\cdot(\zeta-\epsilon)))\cdot(-\sin(\zeta))\cdot q\,d\zeta \\ &+ \int_{1.5\pi}^{\gamma} \dot{l}_{\max}\cdot(\pi-\zeta)\cdot(-\sin(\zeta))\cdot q\,d\zeta + \int_{\gamma}^{2\pi} (\frac{r}{\cos(\zeta)} - r)\cdot(-\sin(\zeta))\cdot q\,d\zeta\,. \end{aligned} \tag{4.22}$$

And, if we ignore all angles starting from $1.5\pi$ rad, it results in

$$\begin{aligned}&\int_{\pi}^{1.5\pi} \dot{l}_{\max}\cdot(\zeta-\pi)\cdot(-\sin(\zeta))\cdot q\,d\zeta \\ &\leq \\ &\int_{\epsilon_s}^{\epsilon} \dot{l}_{\max}(\zeta-\epsilon_s)\cdot((-\sin(\zeta)))\cdot q\,d\zeta + \int_{\epsilon}^{1.5\pi} ((\epsilon-\epsilon_s)\dot{l}_{\max} - \dot{l}_{\max}\cdot(\zeta-\epsilon)))\cdot(-\sin(\zeta))\cdot q\,d\zeta\,. \end{aligned} \tag{4.23}$$

Solving this numerically gives a valid range of $1.11\pi < \epsilon \leq 1.5\pi$. Hence, the previously stated hypothesis is false. It is not always beneficial to use an $\epsilon_s$ start rather than the $\pi$ rad start, whereas it is the case if epsilon is greater than $1.11\pi$ rad. Evaluating the right side of Equation (4.23) shows a maximum at $\epsilon = 1.31\pi$ rad. Previously, we simplified the starting point of the retraction as $1.5\pi$ rad, which we need to evaluate as the hypothesis was not confirmed.

However, with the new knowledge of a maximum, we can cut short the long and complicated integration of $\gamma$ by specifying the upper limit for integration as $1.5\pi - \frac{\gamma}{2}$. Starting retraction at $1.5\pi - \frac{\gamma}{2}$ leads to full retraction at $\gamma$. Therefore, this is the absolute worst case, but this time, for the $\epsilon_s$ approach, as the angle until which is integrated on the right side decreases. Therefore, the positive torque also decreases. At the same time, the counterproductive torque on the left side stays the same. Repeating all the previous steps from Equation (4.18) onwards, we get

$$\begin{aligned}
&\int_{\pi}^{1.5\pi-\frac{\gamma}{2}} \dot{l}_{\max} \cdot (\zeta - \pi) \cdot (-\sin(\zeta)) \cdot q \, d\zeta \\
&\leq \\
&\int_{\epsilon_s}^{\epsilon} \dot{l}_{\max}(\zeta - \epsilon_s) \cdot ((-\sin(\zeta))) \cdot q \, d\zeta + \int_{\epsilon}^{1.5\pi-\frac{\gamma}{2}} ((\epsilon - \epsilon_s)\dot{l}_{\max} - \dot{l}_{\max} \cdot (\zeta - \epsilon))) \cdot (-\sin(\zeta)) \cdot q \, d\zeta \, .
\end{aligned} \tag{4.24}$$

Inserting the previously found $\epsilon = 1.31\pi$ rad and numerically solving for $\gamma$ lead to no solution within the logical range of $1.5\pi$ rad to $2\pi$ rad. Therefore, we found a value for $\pi < \epsilon <= 1.5\pi$ rad, which will always be beneficial in comparison to a start of the extension at $\pi$, for any $\gamma$. This leads to an alternative definition of $\epsilon$ as it was initially defined as the angle the retraction needs to start for reaching $\gamma$ in time. With this evaluation, we showed that it is beneficial to always have the maximum extension at a certain $\epsilon$. Note that the said maximum extension does not necessarily refer to $l_{\max}$. It describes that if a full extension is not possible with a start at $\pi$, the extension should start at $\epsilon_s$; therefore, it is not guaranteed that $l_{\max}$ is reached at $\epsilon$, as it cannot be retracted in time before reaching $\gamma$. This leads to the conclusion that we cannot calculate $\epsilon_s$ by just factoring in the time needed for full extension. In fact, we need to calculate an extension of the poles that can be retracted between $\epsilon$ and $2\pi$ rad. From $\epsilon$ to $\gamma$, the retraction happens with full speed and from $\gamma$ to $2\pi$ rad with a reduced speed. There certainly exists a solution for $\epsilon_s$. However, in our opinion, this goes into directions where the benefit for practical implementation does not hold up to the required computational power and the needed precision of all actions. For the prototype introduced later, this is not even possible as there is no feedback on the pole length. Therefore, we conclude that if the combination of pole length, the retraction speed of the pole, and the desired rotation speed are in a relevant range, and the feedback on the exact extension length is given, we implement the adapted $\epsilon$ mechanism. At each calculation cycle, we determine the new $\gamma$; on the basis of this, we find the maximum of Equation (4.24), giving us the optimal $\epsilon$, which is then used to calculate the needed $\epsilon_s$. In the further evaluation, we refer to this step just as "calculate $\gamma, \epsilon$, and $\epsilon_s$." It is incumbent on the actual robot and the requirement if certain described adaptions are made. Therefore, we also ignore these values if the retraction speed is always fast enough to retract in time ($\gamma = 1.5$ rad). For the vast majority of all implementations, this will be sufficient and reduces complexity to a minimum. Algorithm 5 shows the pseudo-code for a push and leverage algorithm, taking all these parameters into account.

The code presented in Listing A.1 simulates the extension of a single rod for given parameters. The $\omega$ is assumed to be held constant as a force simulation is too high depending on the actuators and behaviors, which would go far beyond the scope of this thesis. Subsection 4.1.3 contains the derivation of general forces and their interaction. Therefore, the $\omega$ can be adjusted, even

**Algorithm 5:** Leverage and Push Movement Algorithm with detailed boundaries and limitations.

```
while true do
    calculate γ, ϵ and ϵ_s
    foreach pole in poles do
        predict ζ with measured ω and commanded c_ω
        if β ≤ ζ ≤ α then
            extend pole
        else if retraction speed > maximum retraction speed needed then
            if ζ > π and no contact to ground then
                extend pole
            else
                retract pole (ground avoidance mode)
            end
        else if ζ > γ then
            retract pole (ground avoidance mode)
        else if (ζ > ϵ_s or ζ > π) and ζ > α then
            extend pole
        else
            retract pole
        end
    end
end
```

during execution, but we assume that the leverage and force applied are enough to generate this $\omega$. The resulting output includes two figures. The first one is a time-based Cartesian coordinate system, where the $x$-axis is the elapsed time, and the $y$-axis shows the extension of the pole, the maximum possible extension at a particular moment, and the angle of the pole at that moment. The second one is an angle-based polar coordinate system. The angle in the plot represents the angle of the pole. It is not time-dependent. Figure 4.7 explains the output figure, and Figure 4.8 shows the exemplary results.

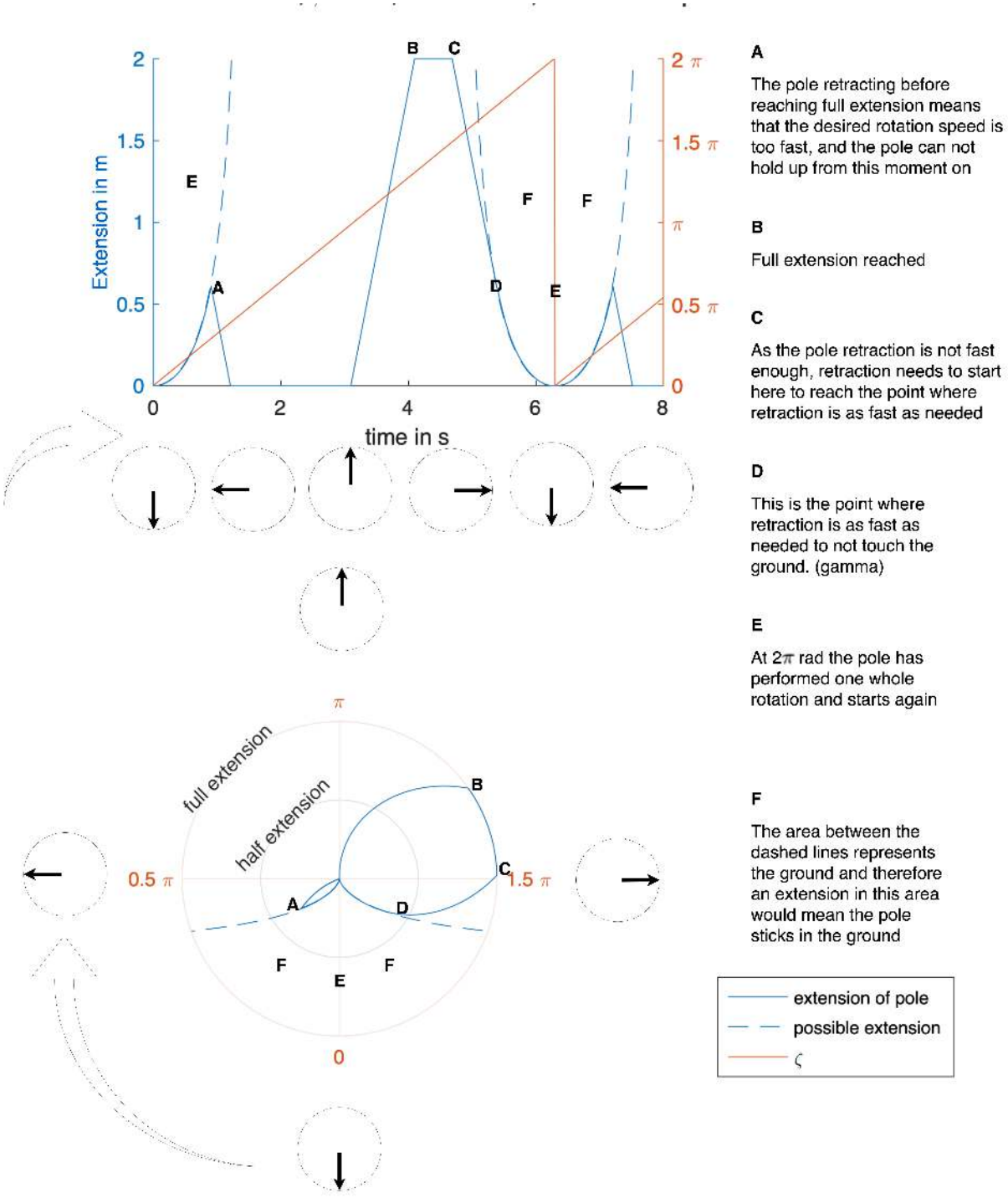



**Figure 4.7:** Explanation of the output of the pole-simulation. The shown movement is one rotation with the combined push and leverage approach. Left: Cartesian coordinate time-based system. The $x$-axis is the elapsed time, and the $y$-axis shows the extension of the pole (blue line), the maximum possible extension at a particular moment (dashed blue line), and the angle of the pole at that moment (orange line). Right: Polar coordinate angle-based system. The angle in the plot represents the angle of the pole (orange). The extension is shown for the specific angle (blue line). It does not show information on time.

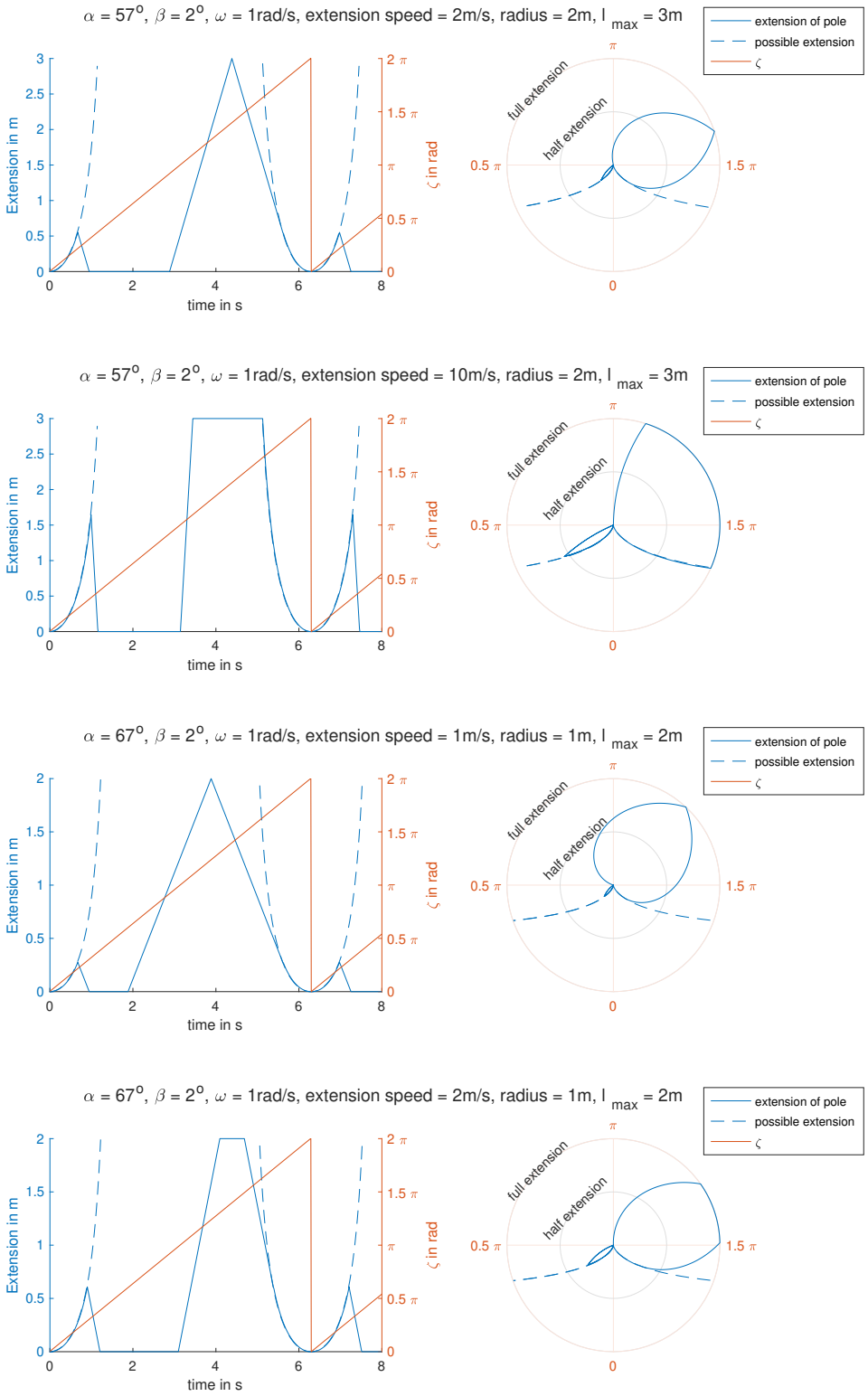



**Figure 4.8:** Simulation of the extension of one specific rod with the stated parameters, while using the push and leverage approach. The $\omega$ was held constant.

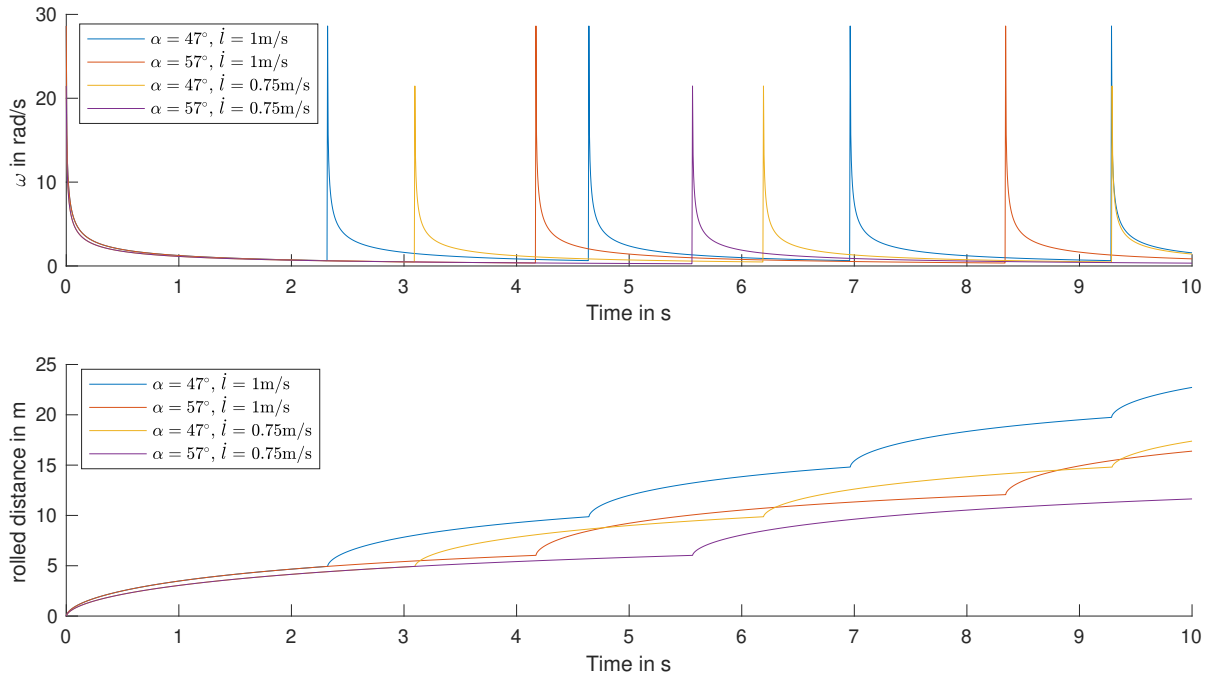



**Figure 4.9:** Above: The rotation speed of a sphere, assuming a 1 m radius, for different $\alpha$ and different constant $\dot{l}_x$. $\beta$ is chosen 2°. Below: The rolled distance of the same sphere setup. Due to the constant $\dot{l}_x$, this visualizes the mono-speed problem. The same extension speed of the pole at small angles near $\beta$ results in faster rotations of the sphere than for poles at larger angles towards $\alpha$. Therefore, the spikes of high rotation speeds occur each time a pole starts extending at $\beta$. As the angle of the pole increases until $\alpha$, the rotation speed decreases.

### 4.1.2 Mono-Speed Problem

As shown in Equation (4.5), the maximum achievable rotation speed depends on the maximum possible extension speed of the rod and the chosen $\alpha$, which again depends on the desired number of rods pushing simultaneously and the chosen $\beta$. All this implies a continuously controllable speed of the individual rods.

Nevertheless, this assumption might not be valid for the technical realizations of a TLDR robot. The simple versions of electric as well as linear hydraulic motors often do not have speed control. With increasing $\zeta$, the needed speed and acceleration of the rod is not always possible. It is impossible to have more than one mono-speed-rod actively pushing, as the next rod needed to extend faster than the one nearest to 0 degrees. Also, a constant $\omega$ is not possible with a finite $n$. Considering Equation (4.4) for a constant $\dot{l}_x$, and the changing rotation speed of the sphere shown in Figure 4.9, results in a rolled distance, as shown in the same figure.

These calculations are based on the simplified idea of a rod extending, leading directly to rotation. This means the rod has an infinite amount of force. However, this does not apply to real-world actuators. Motors can touch the ground and start trying to extend themselves,

applying force. If this force is not sufficient to overcome the moment of inertia of the sphere, there will be no extension. Depending on the actual actuator, this might or might not be a problem that is damaging to the motor. Subsection 5.2.1 highlights this discussion in detail. This behavior gives the mono-speed approach the possibility of more than one pole actively touching the ground to generate torque. The second pole is not needed for the rotation if one pole has enough power to cause the rotation of the whole sphere, at such a fast pace that the next pole is not even able to extend into the ground. This is especially the case if the initial pushing angle is relatively small. Therefore, the following assumptions are always from the point of view that one pole is not enough to generate a fluent rotation of the sphere with an acceptable speed. Looking at Figure 4.4, one can notice a plateau in the middle with a rather steep pitch at the beginning and end.

In the first part, it is rather unlikely that the rod will have enough force to achieve rotation. This may lead to the aforementioned damage. Approaching $0.5\pi$, the needed velocity of pole approaches infinity, which is not realistic and delays or slows down the rotation. Therefore, the range from approximately $0.01\pi$ rad to $0.4\pi$ rad is chosen for this paper as a useful range for extending mono-speed poles. The difference between them still has exponential growth, as Figure 4.4 has a logarithmic scale, but after $0.4\pi$ rad, this only gets steeper. For the leverage approach, the evaluation for mono speed is more straightforward. There still exists a $\gamma$ at which the speed of the pole $\dot{l}_x = \dot{l}_{x\_\max}$ matches the exact speed to avoid the ground. Therefore, there is still an $\epsilon$ at which retraction needs to start at full speed (the only available speed) to reach $\gamma$. Also, the angle $\epsilon_s$ still exists to get a full extension at $\epsilon$, and if $\epsilon_s > \pi$ rad, the extension after $\pi$ rad is done anyways at full speed. Therefore, only the retraction after $\epsilon$ differs from the non-mono approach. Figure 4.10 depicts the difference between the mono-speed behavior and the stepless one. To display the impact on the used scale, we set the stimulation frequency from 100 Hz to 20 Hz. It has a kind of aliasing effect as the continuous change of the possible length must be matched with just the one available speed. This means the desired length must be matched with possible changes of $\pm\frac{\dot{l}_{x\_\max}}{f}$, where $f$ is the frequency of calculation or of the communication to the actuator. It still happens at faster frequencies, but the errors are smaller and hence not as visible as with lower frequencies.

Figure 4.9 represents the change in the rotation speed. This causes problems for the calculation of retraction speed for the leverage approach if we assume the simplified idea that the extension of a rod directly leads to rotation. To show this, we extended the simulation not to have a constant $\omega$, but rather, the $\omega$ was calculated by rotation speed resulting from the rod extension. The acceleration of $\omega$ was limited, as even with receiving a high force from the poles, there is a physical limitation in terms of acceleration. Therefore, the simulation code of Listing A.1 is extended by the function shown in Listing A.2. Figure 4.11 provides the result for a limitation of $1\,{}^{\mathrm{rad}}/_{\mathrm{s}^2}$ and $10\,{}^{\mathrm{rad}}/_{\mathrm{s}^2}$.

It becomes evident that higher possible rotation accelerations of the sphere result in problems as the changing $\omega$ leads to changing $\gamma$ and $\epsilon$. At some moments, the $\omega$ is so high that the pole starts extending as it has already reached or passed $\epsilon$. As the $\omega$ slows down, $\epsilon$ grows. Therefore, the pole that was extending just before, starts retracting since its angle is now a smaller $\epsilon$. This behavior is noted in the simulation with a maximum $\dot{\omega}$ of $10\,{}^{\mathrm{rad}}/_{\mathrm{s}^2}$. The simulation with a maximum $\dot{\omega}$ of $1\,{}^{\mathrm{rad}}/_{\mathrm{s}^2}$ does not show this behavior as $\omega$ does not reach such high values due to

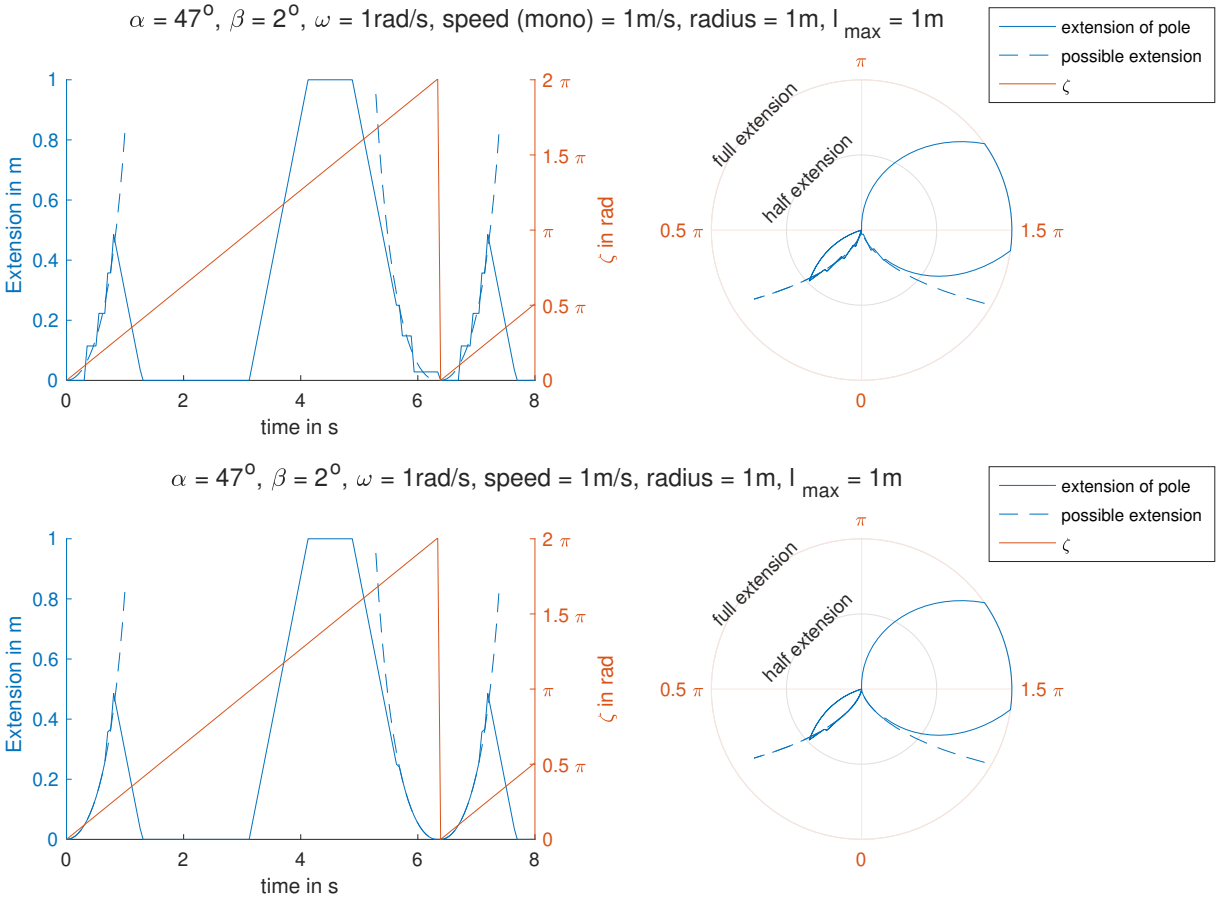



**Figure 4.10:** Simulation of the extension of one specific rod with the stated parameters, slowed down from 100 Hz to 20 Hz, for an actuator with mono speed (above) and a stepless controllable one (below). The push and leverage approach is used. The mono-speed pole shows a stepped behavior, whereas the variable speed matches the possible extension rather well.

the limited acceleration. To solve this problem, there are several avoidance mechanisms:

- If a higher $\beta$ is chosen, the problematic high $\omega$ is not generated.
- Measuring for the robot the fastest possible $\omega$ and calculating $\gamma$ and $\epsilon$ with this value.
- If the behavior is fully unknown, ensuring the pole is fully retracted at the angle at which a full extension has ground contact, avoids the whole problem.
- An on-board simulation calculates all the angles at which the pole does not achieve the required velocity.

Overall, higher possible accelerations lead to problems for robots that are relatively lightweight in comparison to their possible extension power. Then, the avoidance mechanism of this behavior needs to be implemented. Also, the ground on which the robot moves influences this behavior, as we will see in the evaluation.

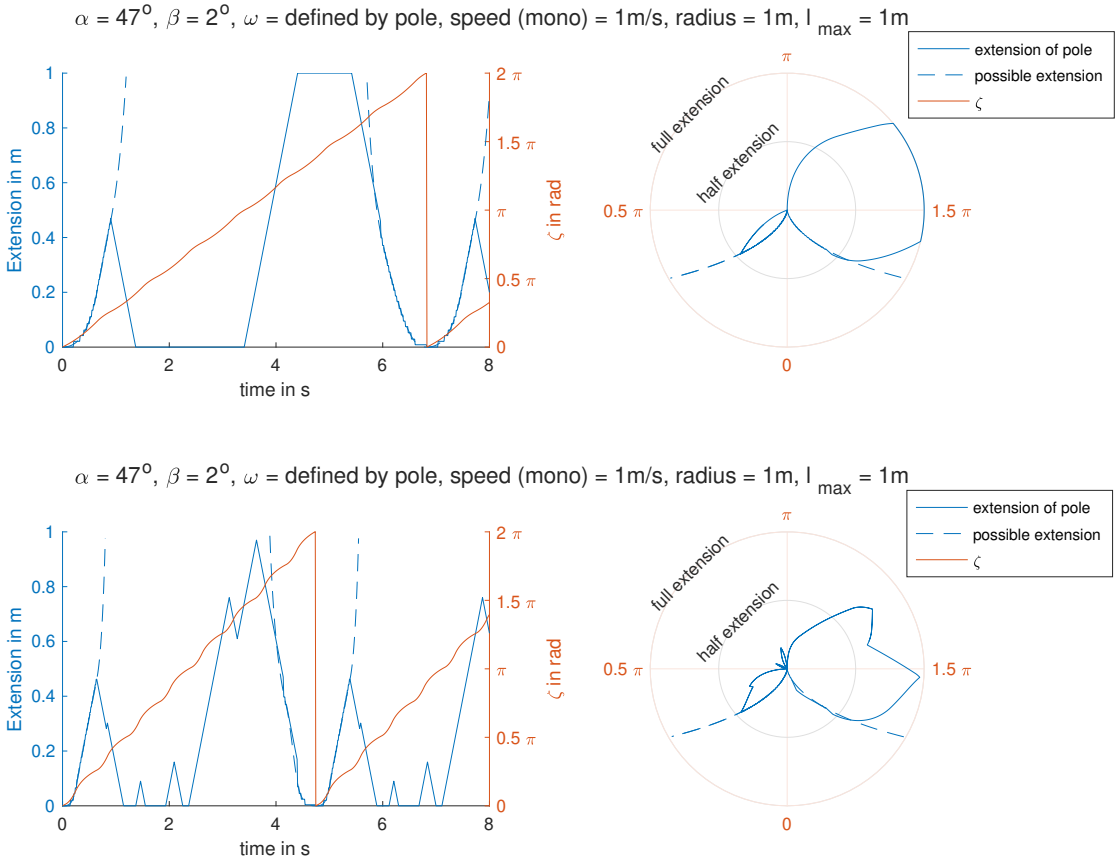



**Figure 4.11:** Simulation of the extension of one specific rod with the stated parameters, with $\omega$ calculated based on the extraction of the poles. This assumes an infinite force of the poles. The push and leverage approach is used. $\dot{\omega}$ is limited to $1\,^{\mathrm{rad}}/_{\mathrm{s}^2}$ (above) and to $10\,^{\mathrm{rad}}/_{\mathrm{s}^2}$(below). With a too high allowed $\omega$s, $\epsilon$ and $\epsilon_s$ change so fast such that the angle of the pole is larger than $\epsilon_s$ at one moment and smaller in the next, leading the short extension and retraction spikes.

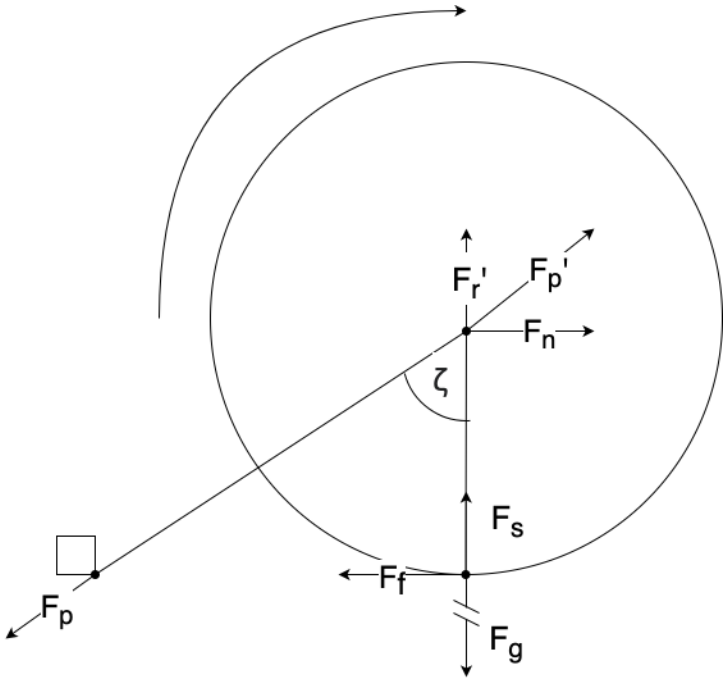



**Figure 4.12:** Force evaluation of the pushing approach with no slip due to an obstacle.

### 4.1.3 Physical Representation

Having described the geometrical constraints regarding the basic structure in Subsection 4.1.1, we now want to evaluate the fundamental physical interactions of the TLDR robot. This evaluation is based on [15, 29, 38, 62], and [30], which mostly deal with the integration of friction into the analysis of a rolling spheroid. This physical evaluation will help identify the limitations of the prototype and its requirements. In the following, we will evaluate the pushing approach with and without slip as well as the leverage approach in terms of their physical representation.

#### Pushing with no slip

First, we look at the simplest case: pushing with no slip at the pole end. Here, we assume an obstacle at the end of the actively pushing pole. This prohibits every slip of the pole. Figure 4.12 shows the simplified forces in this case. We focus on the direct force interaction of the pole and the robot. Forces like air resistance, sinking in due to the softness of the ground, etc., are neglected. As the pole pushes with $\boldsymbol{F}_P$ against the ground and non-moving obstacle, the counterforce $\boldsymbol{F}_p^{'}$ acts at the middle of the robot and has the two components $\boldsymbol{F}_r$ and $\boldsymbol{F}_n$, which can be calculated by

$$\boldsymbol{F}_p^{'} = F_p^{'} \cdot \begin{bmatrix} \sin(\zeta) \\ \cos(\zeta) \end{bmatrix} = \begin{bmatrix} F_r^{'} \\ F_n \end{bmatrix} . \tag{4.25}$$

The vertical $\boldsymbol{F}_r$ is directly countered by the gravity $\boldsymbol{F}_g$. The difference between $\boldsymbol{F}_g$ and $\boldsymbol{F}_r$ is compensated by the structural force $\boldsymbol{F}_s$. If $\boldsymbol{F}_r$ is larger than $\boldsymbol{F}_g$, the sphere starts to accelerate

in the vertical direction. The horizontal force $\boldsymbol{F}_n$ is countered by the frictional force $\boldsymbol{F}_f$. Let $\mu_{rs}$ be the sum of the friction coefficient $\mu_s$ of the robot and surface, and the rolling friction coefficient $c_r$, which depends on how much the specific structure sinks into the surface due to its own weight. Then, $\boldsymbol{F}_f$ is defined as

$$\boldsymbol{F}_f = \mu_{rs} \cdot (-\boldsymbol{F}_n)\,. \tag{4.26}$$

Note that $\mu_{rs}$ is chiefly determined experimentally. We will often assume it to be 1, which means a good grip with no slip. The difference between $\boldsymbol{F}_f$ and $\boldsymbol{F}_n$, which is always in the direction of $\boldsymbol{F}_n$ as $\mu_{rs}$ is between zero and one, leads to the translation of the sphere. Let $a_{\text{direct}}$ be the acceleration of this direct translation without translation, and m be the mass of the robot, then

$$a_{\text{direct}} = \frac{F_n - F_f}{m} = \frac{F_n \cdot (1 - \mu_{rs})}{m} \tag{4.27}$$

Let $\boldsymbol{\tau}_f$ be the torque generated by $\boldsymbol{F}_f$, and $r$ be the position vector of the acting point of $\boldsymbol{F}_f$, then

$$\boldsymbol{\tau}_f = \boldsymbol{r} \times \boldsymbol{F}_f \tag{4.28}$$

$$\tau_f = r_m \cdot F_f\,. \tag{4.29}$$

Let $I$ be the moment of inertia of the sphere and $\dot{\omega}$ the acceleration of the angular velocity, then

$$\dot{\omega} = \frac{\tau_f}{I}\,. \tag{4.30}$$

$I$ depends on the mass, shape, and mass distribution of the robot. For a massive sphere, this is $\frac{2}{5}mr^2$, but the length of extension of the pole needs to be considered as it changes the moment of inertia, even in this case. Also, if the robot is not built perfectly balanced, $I$ has different values for each rotation axis. Therefore, it actually is a tensor, i.e., 3x3 matrix. With mechanical simulations or experiments, this tensor is determinable. We will refer to it as general $I$.

The initiated rotation and the translation by sliding cause the rolling of the robot. The translation consists of not only the direct linear slip but also the rotation-initiated translation in the form of

$$a_{\text{rotation}} = 2\pi \cdot r_m \cdot \dot{\omega}\,. \tag{4.31}$$

This leads to the overall system. Let $a$ be the overall acceleration of the robot in the vertical direction (so $a_{direct} + a_{rotation}$), $\dot{\omega}$ the angular acceleration, and $m$ the mass of the robot, then

$$\begin{bmatrix} a \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{\sin(\zeta) \cdot F_p \cdot (1-\mu_{rs})}{m} + 2\pi \cdot r_m^2 \cdot \mu_{rs} \cdot \sin(\zeta) \cdot F_p \cdot I^{-1} \\ r_m \cdot \mu_{rs} \cdot \sin(\zeta) \cdot F_p \cdot I^{-1} \end{bmatrix}\,. \tag{4.32}$$

For the case where there is no slip, $\mu_{rs} = 1$, so we get

$$\begin{bmatrix} a \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 2\pi \cdot r_m^2 \cdot \sin(\zeta) \cdot F_p \cdot I^{-1} \\ r_m \cdot \sin(\zeta) \cdot F_p \cdot I^{-1} \end{bmatrix} = \begin{bmatrix} 2\pi \cdot r_m \cdot \dot{\omega} \\ r_m \cdot \sin(\zeta) \cdot F_p \cdot I^{-1} \end{bmatrix}\,. \tag{4.33}$$

Note that as $\dot{\omega} = \ddot{\zeta}$, this is a second-order nonlinear differential equation. For the evaluation, we assume $\frac{r_m \cdot F_p}{I}$ to be constant and refer to it as constant $A$. Solving it analytically gives the

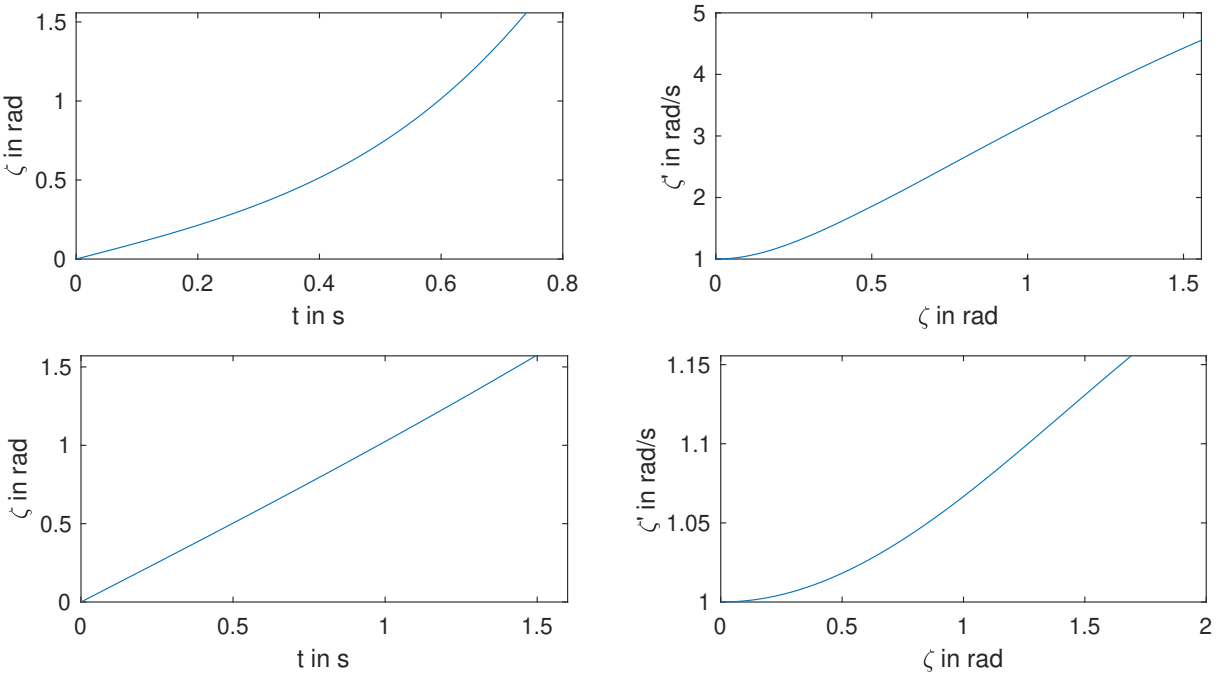



**Figure 4.13:** Analytical solution of Equation (4.33). Above: Solution for $A = 10$. Below: Solution for $A = 0.15$.

result shown in Figure 4.13, once for a high factor $A = 10$ and once for the factor $A = 0.15$, which is the magnitude of the later-introduced prototype.

As expected, with an increasing $\zeta$, the angular velocity also raises. The plot cannot be seen as a direct representation of the actual $\zeta$ as this implies an obstacle that is at the assumed distance, which moves with the extending pole and counters the force perfectly. This is why we assumed a linear $\omega$ for the mathematical representation. However, even if that case occurs, the resulting angular velocity has a limit, defined by the pole length and extension speed. Taking the speed and $\zeta$ limitations of Equations (4.8) and (4.7) and comparing them with the solution of the force equation show that the theoretical introducible speed as per the force calculation exceeds what is geometrically possible. Figure 4.14 visualizes this.

The larger $\zeta$ becomes, the slower the maximum $\dot{\zeta}$ can be up to the point where the maximum possible length of the pole $l_{\max}$ limits the reachable $\theta$. In the shown calculation, the configuration with the fastest $\dot{l}$ does indeed cut the force calculation at the vertical transition to 0 rad, which means that in this case, the length was the limit. Combining the geometrical and physical behaviors shows the overall problem of the system. These both influences do not just limit each other or reinforce each other. The symbiosis of both leads to a system that is hard to predict. The following are two exemplary reasons for it:

- A pole jumps over the ground until it finds an obstacle or reaches a certain level of bending.
- A rotation that leads to an extension of the pole in the air without ground contact initiates leverage in the other direction than intended.

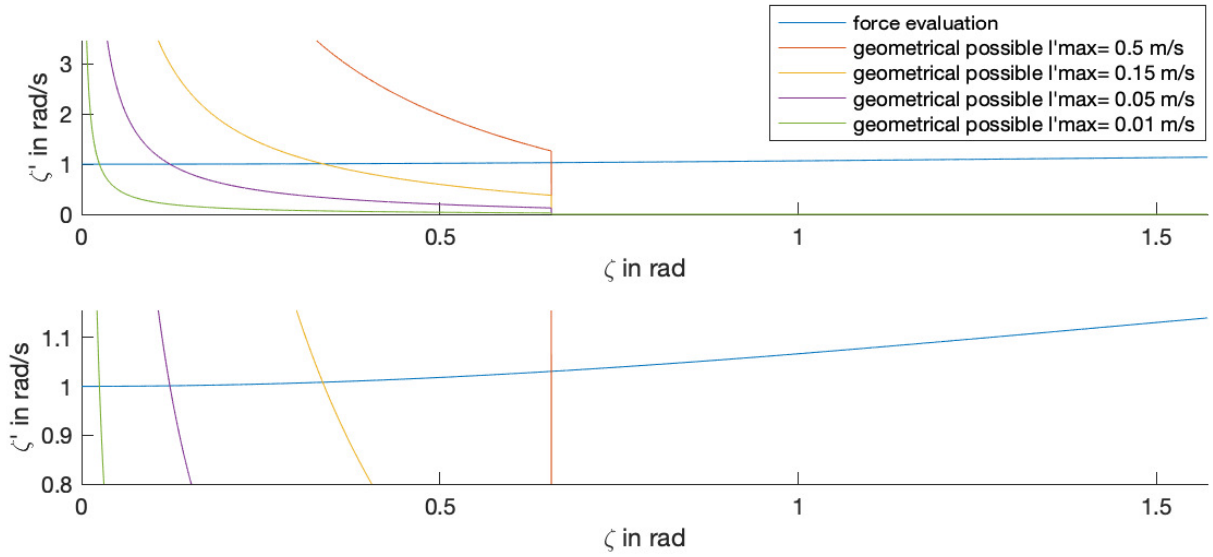



**Figure 4.14:** Analytical solution of Equation (4.33) with $A = 0.15$ and the geometrical possible $\dot{\zeta}$s with a radius of 0.4 m and a maximum length of 0.1 m. The force solution is limited by either the extension speed or the sheer fact that the pole length is not long enough to reach $\zeta$

Because of these, the starting $\omega$ of each pole at $\zeta = \alpha$ will always be different.

Overall, this thesis will not provide a complete solution to the whole system. It focuses on evaluations for general limitations and dimensioning components. Therefore, the further evaluations and their resulting differential equations will not be solved as the acceleration at a given $\zeta$ is the only value of interest.

### Pushing with slip

The subsequent evaluation is for the situation in which there is no obstacle at the end of the pole. This not only changes the counterforce from the pole applied on the sphere but also another component, the lever. This does not mean it is the leverage approach, where the poles are extended on the side of the robot towards which it is to roll. This is evaluated separately in Subsubsection 4.1.3. However, we consider a ground with no friction so that the pole will extend and touch the ground; however, rather than that the applied force being directly countered by the ground, the pole slips to the side. This is comparable to a person trying to push himself/herself while standing on ice only using a pole with a big flat surface. When trying to push himself/herself, the pole will slide over the ice surface, but the person will not move. The sphere generates no translation on a friction-less surface, but it has the ability to initiate rotation. Figure 4.15 illustrates the working forces for this. Like before, the starting point is the force of the pole $\boldsymbol{F}_p$. It pushes into the ground at the angle $\zeta$, which splits it up into parts, the force in the slip direction $\boldsymbol{F}_s$ and the force toward the ground $\boldsymbol{F}_r$. $\boldsymbol{F}_p$ splits as

$$\boldsymbol{F}_p = F_p \cdot \begin{bmatrix} \cos(\zeta) \\ \sin(\zeta) \end{bmatrix} = \begin{bmatrix} F_r \\ F_s \end{bmatrix} . \tag{4.34}$$

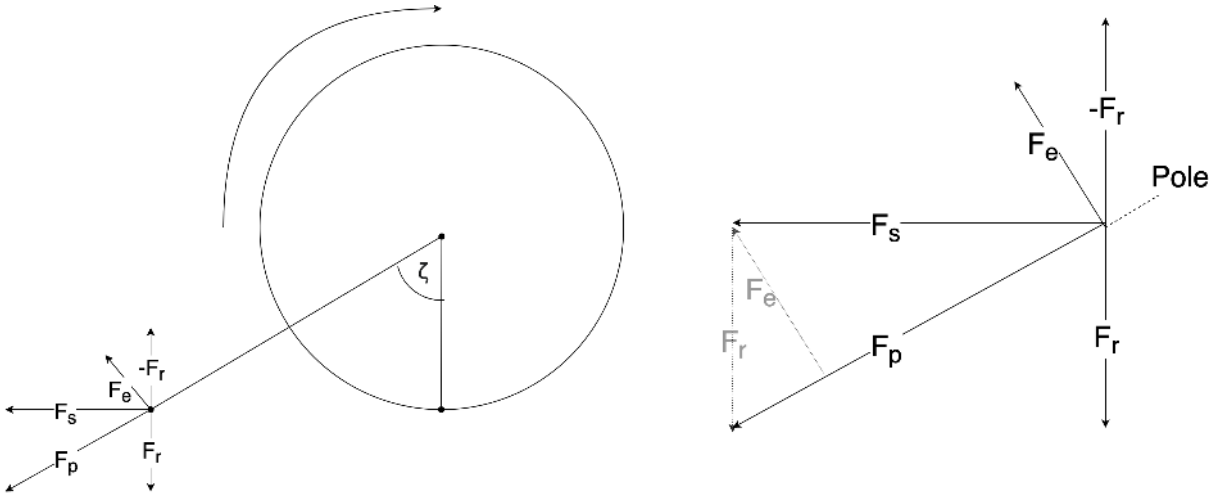



**Figure 4.15:** Force evaluation of the pushing approach with a complete slip of the poles.

Like with the obstacle, $\boldsymbol{F}_r$ is countered by the ground itself, producing $-\boldsymbol{F}_r$. The difference is the arising $\boldsymbol{F}_s$, which was previously countered by the obstacle, hence becoming $\boldsymbol{F}_n$, but it now has no counterpart. This creates an acceleration of the pole tip in this direction. However, the point is part of a whole structure, and because of this, $\boldsymbol{F}_s$ results in the lever force, perpendicular to the line of the pole, $\boldsymbol{F}_e$. The amount is determined by

$$F_e = \cos(\zeta) \cdot F_s \tag{4.35}$$

Let $\tau_e$ be the torque generated by force $\boldsymbol{F}_e$, then

$$\boldsymbol{\tau}_e = \boldsymbol{r} \times \boldsymbol{F}_e \tag{4.36}$$

$$\tau_e = (l(\zeta) + r_m) \cdot F_e \,. \tag{4.37}$$

Let $I$ be the moment of inertia of the sphere and $\dot{\omega}$ the acceleration of the angular velocity, then

$$\dot{\omega} = \frac{\tau_e}{I} = (l(\zeta) + r_m) \cdot F_e \cdot I^{-1} \,. \tag{4.38}$$

For $l(\zeta)$, we can use Equation (4.2). This leads to

$$\begin{bmatrix} a \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 \\ (\frac{r_m}{\cos(\zeta)}) \cdot F_e \cdot I^{-1} \end{bmatrix} . \tag{4.39}$$

In this case, there is neither a $a_{\text{direct}}$ nor a $a_{\text{rotation}}$ due to the lack of grip at the bottom of the sphere, causing full rotation and no translation. The next evaluation is the combination of the two previous systems (with and without friction), introducing a variable friction force. Figure 4.16 depicts this concept. The horizontal force $\boldsymbol{F}_s$, which arises from $\boldsymbol{F}_p$, is now countered by the frictional force $\boldsymbol{F}_f$, resulting in the friction between the pole end and the ground. This

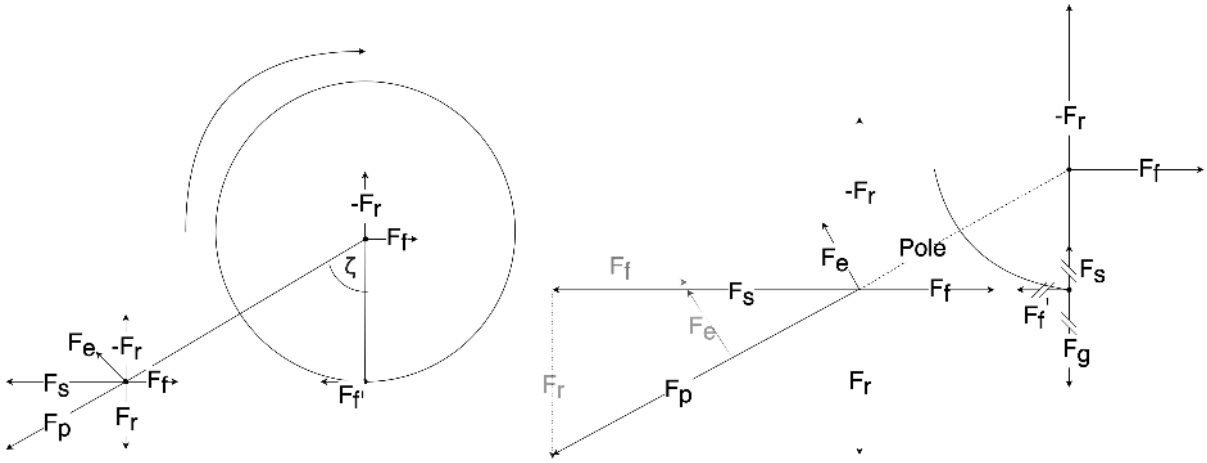



**Figure 4.16:** Force evaluation of the pushing approach with variable friction applied at the pole ends.

force does not necessarily counter $\boldsymbol{F}_s$ completely; it depends on the static friction coefficient $\mu_{s\text{Pole}}$. Therefore

$$F_f = \mu_{s\text{Pole}} \cdot F_s = \mu_{s\text{Pole}} \cdot \sin(\zeta) \cdot F_p \,. \tag{4.40}$$

The lever force $F_e$ at the end of the pole is now determined from the combination of both leading to

$$F_e = \cos(\zeta) \cdot (F_s - F_f) = \cos(\zeta) \cdot F_s(1 - \mu_{s\text{Pole}}) \,. \tag{4.41}$$

The force $\boldsymbol{F}_f$ also acts at the middle of the sphere, just like in the evaluation with the obstacle. Similarly, it is countered by the frictional force at the contact point of the sphere and ground $\boldsymbol{F}'_f$. This depends now on $\mu_r s$ as the rotational resistance is applied here, which leads to

$$F'_f = \mu_{rs} F_f = \mu_{rs} \cdot \mu_{s\text{Pole}} \cdot F_s \,. \tag{4.42}$$

The difference $F_f - F'_f$ results in $a_{\text{direct}}$ like in Equation (4.27),

$$a_{\text{direct}} = \frac{F_f - F'_f}{m} = \frac{(1 - \mu_{rs}) \cdot \mu_{s\text{Pole}} \cdot F_s}{m} \tag{4.43}$$

Also, $a_{\text{rotation}}$ applies like in Equation (4.31). However, in this case, the $\omega$ differs as there is not only the torque $\tau_f$ from Equation (4.29) or $\tau_e$ from Equation (4.29) but in fact both. Let $\tau_{fe}$ be the overall torque working on the sphere. Using Equations 4.41 and 4.42, we get

$$\tau_{fe} = \tau_{f'} + \tau_e = F'_f \cdot r_m + F_e \cdot \frac{\cos(\zeta)}{r_m} \tag{4.44}$$

$$= \mu_{rs} \cdot \mu_{s\text{Pole}} \cdot F_s \cdot r_m + \cos(\zeta)^2 \cdot F_s(1 - \mu_{s\text{Pole}}) \cdot \frac{1}{r_m} \,. \tag{4.45}$$

This then produces a change in angular velocity,

$$\dot{\omega} = \frac{\tau_{fe}}{I} \,. \tag{4.46}$$

In this case, $a_{\text{rotation}}$ is not merely taken as the surpassed scope of the robot as the whole length is not directly translated to the ground. As $\mu_{rs} = 0$ means just slipping with no rotation being transferred to translation, and $\mu_{rs} = 1$ means that the whole scope is also reached as translation, we take the friction coefficient as a linear transition between these two points. This is an approximation as the real relation between the friction coefficient, rolling resistance coefficient, and the generated horizontal linear acceleration from rotation may not be completely linear. Nonetheless, for the precision aimed at this point, it is still approximated as linear [23]. Therefore, $a_{\text{rotation}}$ becomes

$$a_{\text{rotation}} = \mu_{rs} 2 \cdot \pi \cdot r_m \cdot \frac{\tau_{fe}}{I} \tag{4.47}$$

$$= \mu_{rs} \cdot 2\pi \cdot r_m^2 \cdot F_s \cdot I^{-1} \cdot (\mu_{rs} \cdot \mu_{s\text{Pole}} \cdot + cos(\zeta)^2 (1 - \mu_{s\text{Pole}}) \cdot \frac{1}{r_m^2}) \tag{4.48}$$

The overall system is the combination of the acceleration of translation ($a$) and rotation $\dot{\omega}$, which give

$$\begin{bmatrix} a \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} a_{\text{rotation}} + a_{\text{direct}} \\ \frac{\tau_{fe}}{I} \end{bmatrix} = \tag{4.49}$$

$$\begin{bmatrix} \mu_{rs} \cdot 2\pi \cdot F_s \cdot I^{-1} \cdot (\mu_{rs} \cdot \mu_{s\text{Pole}} \cdot r_m^2 + cos(\zeta)^2 (1 - \mu_{s\text{Pole}}) \cdot + \frac{(1-\mu_{rs}) \cdot \mu_{s\text{Pole}} \cdot F_s}{m} \\ \mu_{rs} \cdot \mu_{s\text{Pole}} \cdot F_s \cdot r_m + \cos(\zeta)^2 \cdot F_s (1 - \mu_{s\text{Pole}}) \cdot \frac{1}{r_m} \cdot I^{-1} \end{bmatrix}. \tag{4.50}$$

Inserting the Equation (4.34) leads to

$$\begin{bmatrix} a \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \sin(\zeta) \cdot F_p \cdot \left( \mu_{rs} \cdot 2\pi \cdot I^{-1} \cdot (\mu_{rs} \cdot \mu_{s\text{Pole}} \cdot r_m^2 \cdot + cos(\zeta)^2 (1 - \mu_{s\text{Pole}})) + \frac{(1-\mu_{rs}) \cdot \mu_{s\text{Pole}}}{m} \right) \\ \sin(\zeta) \left( \mu_{rs} \cdot \mu_{s\text{Pole}} \cdot F_p \cdot r_m + \cos(\zeta)^2 \cdot F_p \cdot (1 - \mu_{s\text{Pole}}) \cdot \frac{1}{r_m} \cdot I^{-1} \right) \end{bmatrix}. \tag{4.51}$$

For now, we only looked at the horizontal component of $a$ as the gravitational force counters the vertical force. There certainly exists the possibility that the vertical force exceeds the gravitational force. In that case, a vertical, linear acceleration will take place, driven by the resulting force. The resulting vertical force is not considered negative as the force difference between the vertical initiated and gravitational force is countered by the structural force $F_s$. Let $a_{\text{v}}$ be the vertical acceleration of the robot and $\Theta(x)$ the unit step size function, then

$$a_{\text{v}} = \frac{\Theta(F_R - F_G)(F_R - F_G)}{m} = \frac{\Theta(\cos(\zeta) \cdot F_p - m \cdot g)(\cos(\zeta) \cdot F_p - m \cdot g)}{m}$$

$$= \Theta(\cos(\zeta) \cdot F_p/m - g)(\cos(\zeta) \cdot F_p/m - g)\,. \tag{4.52}$$

Referring to $a$ in Equation (4.51) as $a_{\text{h}}$ as it is the horizontal acceleration and inserting Equation (4.52) give

$$\begin{bmatrix} a_{\text{v}} \\ a_{\text{h}} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \Theta(\cos(\zeta) \cdot F_p/m - g)(\cos(\zeta) \cdot F_p/m - g) \\ \sin(\zeta) \cdot F_p \cdot \left( \mu_{rs} \cdot 2\pi \cdot I^{-1} \cdot (\mu_{rs} \cdot \mu_{s\text{Pole}} \cdot r_m^2 \cdot + cos(\zeta)^2 (1 - \mu_{s\text{Pole}})) + \frac{(1-\mu_{rs}) \cdot \mu_{s\text{Pole}}}{m} \right) \\ \sin(\zeta) \left( \mu_{rs} \cdot \mu_{s\text{Pole}} \cdot F_p \cdot r_m + \cos(\zeta)^2 \cdot F_p \cdot (1 - \mu_{s\text{Pole}}) \cdot \frac{1}{r_m} \cdot I^{-1} \right) \end{bmatrix}, \tag{4.53}$$

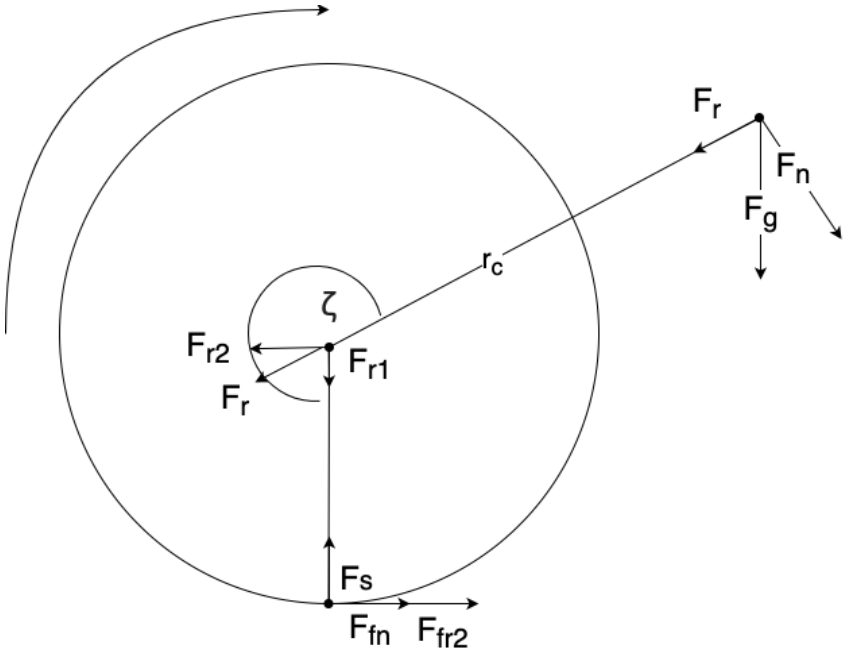



**Figure 4.17:** Force evaluation of the leverage approach.

which is the final solution for the introduced model using the pushing approach for locomotion. We can derive the following trivial behaviors:

- Setting $\mu_{rs}$ and $\mu_{s\text{Pole}}$ to one, i.e., full friction and no slip, gives the Equation (4.33).
- Setting $\mu_{rs}$ and $\mu_{s\text{Pole}}$ to zero, i.e., no friction and only slip, gives the Equation (4.39).
- At a $\zeta$ of 0 rad, there is neither rotation nor horizontal translation but only vertical movement as we push straight upwards.
- $\mu_{rs}$ and $\mu_{s\text{Pole}}$ play a crucial role as they influence most parts of the final system representation as square.

### Leverage

The last part is the physical evaluation of the leverage approach. Figure 4.17 shows the acting forces. All extended poles are reduced to a point mass at which the gravitational force $\boldsymbol{F}_g$ acts. This force is split up into a normal force $F_n$, perpendicular to the axis of midpoint and center of mass point (with only one pole, this axis is identical with the pole axis), and the force $F_r$, perpendicular to it. The splitting is done as follows:

$$\boldsymbol{F}_g = F_g \cdot \begin{bmatrix} -\cos(\zeta) \\ -\sin(\zeta) \end{bmatrix} = \begin{bmatrix} F_r \\ F_n \end{bmatrix} . \tag{4.54}$$

Let $\boldsymbol{\tau}_n$ be the torque, initiated by $\boldsymbol{F}_n$, and $\boldsymbol{r}_c$ be the vector from the center of the sphere to the single mass point of the poles, then

$$\begin{aligned} \boldsymbol{\tau}_n &= \boldsymbol{r}_c \times \boldsymbol{F}_n \\ \tau_n &= r_c \cdot F_n \,. \end{aligned} \tag{4.55}$$

However, this is not the only acting torque. $\boldsymbol{F}_r$ acts at the midpoint of the sphere and is split into $\boldsymbol{F}_{r1}$ and $\boldsymbol{F}_{r2}$ by

$$\boldsymbol{F}_r = F_r \cdot \begin{bmatrix} -\cos(\zeta) \\ -\sin(\zeta) \end{bmatrix} = \begin{bmatrix} F_{r1} \\ F_{r2} \end{bmatrix} . \tag{4.56}$$

$F_{r2}$ introduces a frictional force $F_{f_r 2}$ at the bottom of the sphere, which adds another torque to the sphere. Let $\tau_{fr2}$ be the force introduced by the frictional force $F_{f_r 2}$, then

$$\tau_{fr2} = r_m \cdot F_{fr2} = r_m \cdot \mu_{rs} \cdot F_{r2} = r_m \cdot \mu_{rs} \cdot \sin(\zeta)\cos(\zeta) F_g = r_m \cdot \mu_{rs} \cdot \frac{\sin(2\zeta)}{2} F_g \,. \tag{4.57}$$

The difference between the two torques is the resulting torque $\tau_r$,

$$\begin{aligned} \tau_r &= \tau_n - \tau_{fr2} = r_c \cdot (-\sin(\zeta)) \cdot F_g - r_m \cdot \mu_{rs} \cdot \sin(\zeta)\cos(\zeta) F_g \\ &= -F_g \left( r_c \cdot \sin(\zeta) + r_m \cdot \mu_{rs} \cdot \sin(\zeta)\cos(\zeta) \right) \end{aligned} . \tag{4.58}$$

We can prove that the leverage algorithm produces no torque in the opposite direction of the intended movement because of the following:

- $r_c > r_m > 0$
- $\pi > \zeta < 2\pi$ leads to $\sin(\zeta) < \sin(\zeta)\cos(\zeta)$ and $\sin(\zeta) < 0$
- $0 \geq \mu_{rs} \leq 1$.

Therefore, $r_c \cdot \sin(\zeta)$ will be negative and always be smaller than $r_m \cdot \mu_{rs} \cdot \sin(\zeta)\cos(\zeta)$, which leads with the multiplication of $-F_g$ to an overall positive torque. The initiated rotational acceleration is therefore given by

$$\dot{\omega} = \frac{\tau_r}{I} = -\sin(\zeta) \cdot F_g \cdot I^{-1} \cdot (r_c + r_m \cdot \mu_{rs} \cdot \cos(\zeta)) \,. \tag{4.59}$$

$F_{r1}$ is directly countered by the structural force of the robot. The part of $F_{r2}$, which is not countered by the friction force, causes direct horizontal translation. This acceleration is negative, as the acceleration is pointed in the opposite direction then intended.

$$a_{h\text{direct}} = -\frac{F_{r2} - \mu_{rs} F_{r2}}{m_{\text{robot}}} = -\frac{\sin(\zeta)\cos(\zeta) F_g (1 - \mu_{rs})}{m_{\text{robot}}} = \sin(\zeta)\cos(\zeta) g (\mu_{rs} - 1) \cdot \frac{m_{\text{lever}}}{m_{\text{robot}}} \tag{4.60}$$

Also, the rotation initiates a horizontal translation, depending on $\mu_{rs}$ like in Equation (4.48), with

$$a_{h\text{rotation}} = \mu_{rs} \cdot 2\pi \cdot \omega = \mu_{rs} \cdot 2\pi \cdot (-\sin(\zeta)) \cdot F_g \cdot I^{-1} \cdot (r_c + r_m \cdot \mu_{rs} \cdot \cos(\zeta)) \tag{4.61}$$

With the leverage approach for locomotion, there is no possibility for a vertical acceleration, setting $a_v$ to zero. The moment of inertia $I$ is again taken as constant, as with the pushing approach, for simplification. Depending on the configuration of the robot and the weight and length of the poles, the $I$ changes during the movement process in non-negligible ways. In these cases, $I$ needs to be made dependent on $\zeta$, thus making it the function $I(\zeta)$. We will stay on $I$ for further evaluation. With this, we set up the overall system:

$$\begin{aligned}\begin{bmatrix} a_\mathrm{v} \\ a_\mathrm{h} \\ \dot{\omega} \end{bmatrix} &= \begin{bmatrix} 0 \\ \sin(\zeta)\cos(\zeta)\cdot g\cdot(\mu_{rs}-1)\cdot\frac{m_\mathrm{lever}}{m_\mathrm{robot}} + \mu_{rs}\cdot 2\pi\cdot(-\sin(\zeta))\cdot F_g\cdot I^{-1}\cdot(r_c + r_m\cdot\mu_{rs}\cdot\cos(\zeta)) \\ -\sin(\zeta)\cdot F_g\cdot I^{-1}\cdot(r_c + r_m\cdot\mu_{rs}\cdot\cos(\zeta)) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ \sin(\zeta)\cdot g\cdot m_\mathrm{lever}\cdot\left(\cos(\zeta)(\mu_{rs}-1)\cdot\frac{1}{m_\mathrm{robot}} - \mu_{rs}\cdot 2\pi\cdot I^{-1}\cdot(r_c + r_m\cdot\mu_{rs}\cdot\cos(\zeta))\right) \\ -\sin(\zeta)\cdot F_g\cdot I^{-1}\cdot(r_c + r_m\cdot\mu_{rs}\cdot\cos(\zeta)) \end{bmatrix}. \end{aligned} \tag{4.62}$$

It is evident that $\dot{\omega}$ is always positive (ignoring the trivial solutions for $r_c = r_m = 0$, etc.) if $\zeta$ is between $\pi$ rad and $2\pi$ rad. For the horizontal velocity, determining this is not trivial as for a $\zeta$ between $\pi$ rad and $2\pi$ rad, $a_{h\mathrm{rotational}}$ and $a_{h\mathrm{direct}}$ are in opposite directions. Therefore, we evaluate

$$\begin{aligned} &0 \leq a_\mathrm{h} \\ &0 \leq \sin(\zeta)\cdot g\cdot m_\mathrm{lever}\cdot\left(\cos(\zeta)(\mu_{rs}-1)\cdot\frac{1}{m_\mathrm{robot}} - \mu_{rs}\cdot 2\pi\cdot I^{-1}\cdot(r_c + r_m\cdot\mu_{rs}\cdot\cos(\zeta))\right). \end{aligned} \tag{4.63}$$

We reduce $\sin(\zeta)$ as it is always negative for the evaluated range of $\zeta$, and $g\cdot m_\mathrm{lever}$ is also always positive, leading to

$$\begin{aligned} &0 \geq \cos(\zeta)(\mu_{rs}-1)\cdot\frac{1}{m_\mathrm{robot}} - \mu_{rs}\cdot 2\pi\cdot I^{-1}\cdot(r_c + r_m\cdot\mu_{rs}\cdot\cos(\zeta)) \\ &\Leftrightarrow \mu_{rs}\cdot 2\pi\cdot I^{-1}\cdot(r_c + r_m\cdot\mu_{rs}\cdot\cos(\zeta)) \geq \cos(\zeta)(\mu_{rs}-1)\cdot\frac{1}{m_\mathrm{robot}}. \end{aligned} \tag{4.64}$$

If we set $\mu_{rs} = 1$, Equation (4.64) becomes

$$2\pi\cdot I^{-1}\cdot(r_c + r_m\cdot\cos(\zeta)) \geq 0\,.$$

Considering that $2\pi\cdot I^{-1} > 0$ and that $r_c$ is $r_m$ plus a length $r_x$ dependent on the pole length and the mass distribution of the poles, we further simplify:

$$\begin{aligned} &(r_c + r_m\cdot\cos(\zeta)) \geq 0 \\ \Leftrightarrow &\cos(\zeta) \geq \frac{-r_c}{r_m} \\ \Rightarrow &\cos(\zeta) \geq -1 > \frac{-(r_m + r_x)}{r_m} \\ \Rightarrow &\cos(\zeta) \geq -1\,. \end{aligned} \tag{4.65}$$

This statement will always be true. Therefore, Equation (4.63) is always fulfilled for $\mu_{rs} = 1$. For the proof that the direction is not always in the direction of desired movement, we use Equation (4.64) and set $\mu_{rs} = 0$, which gives

$$\begin{aligned} 0 \cdot 2\pi \cdot I^{-1} \cdot (r_c + r_m \cdot 0 \cdot \cos(\zeta)) &\geq \cos(\zeta)(0-1) \\ 0 &\geq -\cos(\zeta) \cdot \frac{1}{m_{\text{robot}}} \,. \end{aligned} \tag{4.66}$$

This is true for $\zeta \geq 1.5\pi\,\text{rad}$, as $\frac{1}{m_{\text{robot}}} > 0$. For $\zeta > 1.5\pi\,\text{rad}$, $F_{r2}$ changes its sign, leading to acceleration in the intended direction, therefore an overall acceleration in the opposite direction is not possible. Still, this leaves the range of $\pi\,\text{rad} < \zeta < 1.5\pi\,\text{rad}$. Therefore, we conclude that not for every value of $\mu_{rs}$, $a_{\text{h}}$ goes in the intended direction of rotation.

Rearranging Equation (4.64) for $\mu_{rs}$, and solving numerically the worst case of $\zeta$ gives $\zeta = \pi\,\text{rad}$. This means at $\zeta = \pi\,\text{rad}$ we need to evaluate which $\mu_{rs}$ is required to always have translation int the right side by leverage. After the rearrangement of Equation (4.64) and inserting $\zeta = \pi\,\text{rad}$, we get

$$\mu_{rs} \geq \frac{2\pi \cdot I^{-1} \cdot m_{\text{robot}} \cdot \left( \sqrt{\frac{(2\pi \cdot I^{-1} \cdot m_{\text{robot}} \cdot r_c)^2 + 2 \cdot 2\pi \cdot I^{-1} \cdot m_{\text{robot}} \cdot r_c + 4 \cdot 2\pi \cdot I^{-1} \cdot m_{\text{robot}} \cdot r_m + 1}{(2\pi \cdot I^{-1} \cdot m_{\text{robot}})^2}} + r_c \right) + 1}{2 \cdot 2\pi \cdot I^{-1} \cdot m_{\text{robot}} \cdot r_m} \,. \tag{4.67}$$

Using the moment of inertia of a solid ball around the rolling axis, assuming a straight path, $I = \frac{2}{5} \cdot m_{\text{robot}} \cdot r_m^2$ and some values in the same order of magnitude as the later-introduced prototype $r_m = 0.4$, $r_c = 0.9$, $m_{\text{robot}} = 25$, and $m = 0.1$, leads to a minimum $\mu_{rs}$ of 0.012. This means that for a guaranteed translation in the desired direction at all times, we need a friction coefficient and rolling resistance coefficient of 0.012 in total, which is in the magnitude of a ground of ice.

So we conclude that if the sphere experiences no slip, the leverage approach will always generate translation in the intended direction. For each individual prototype, there exists an $\mu_{rs} < 1$ for the same statement, but we are only able to make a worst-case assumption for it.

### 4.1.4 Braking

For a comprehensive description of the movement of the TLDR robot, we need to discuss the action of braking. Braking is achieved like all other locomotion actions with the pushing or leverage approach. To simplify the description of the side of the poles, we will assume, for this subsection, a rotation to the left, which is braked by poles on the right.

#### Braking by leverage

The leverage approach is rather straightforward as it shares its critical states or actions with the movement by leverage, which Subsections 4.1.1 and 4.1.3 describe. In the same way, the leverage generates rotation using torque for locomotion; it decreases this rotation by torque in the other direction. Therefore, we need to adapt the concept of Algorithm 3 as the original algorithm considers no problem when extending poles on the opposite side of the commanded

rotation. When braking, the angle direction changes as the transition from 0 rad to $2\pi$ rad is always in the direction of the commanded rotation. Depending on the implementation, the algorithm performs this by changing every $\zeta$ (referred to in the pseudo-code as pole.getAngle()) to $2\pi - \zeta$ and taking this as its own braking algorithm, or by defining the angles depending on the commanded rolling direction. The retraction needs to occur between $\pi$ rad and $1.5\pi$ rad. Retracting at full speed is sufficient to avoid collisions as the duration of extension on the other side of the sphere has been longer. Thus, the adapted Algorithm 3 for braking is shown with Algorithm 6. It simplifies as there are only two cases, one for extending, one for retracting, unlike Algorithm 3 that has three.

**Algorithm 6:** Leverage-Only braking Algorithm

```
foreach pole in poles do
    if pole.getAngle() >= 1.5π then
        extend pole (ground avoidance mode)
    else
        retract pole
    end
end
```

Here begins the discussion of angle limits, just like with the leverage movement algorithm. There is the possibility to start retraction at angles smaller than $1.5\pi$ rad, because the overall torque is on the right side, even if some poles have not finished their retraction on the left side. Even a limit of $\pi$ rad leads to braking. This is valid as the extension on the right does not always happen at full speed (referred to in Algorithm 6 as "ground avoidance mode"). Therefore, retraction at full speed on the left side will always have an overall less leverage. According to the same argument, there exists an angle $\epsilon_b$ between $\pi$ rad and $1.5\pi$ rad at which the retraction can start and still reach full retraction at $\pi$ rad.

The discussion, when retracting at this angle or one that leads to a $l \neq 0$, is the same as for the leverage movement if it is conducive to extend poles at an $\epsilon_s$ that is less than $\pi$ rad. We will not further evaluate this but will use the conclusion derived from Equation (4.24). Therefore, there exists an angle at which it is always beneficial to start the retraction rather than ensuring a full retraction at $\pi$ rad, under the condition that full extension is not possible before reaching $1.5\pi$ rad. Let $\epsilon_b$ be this angle and $\gamma_b$ the angle at which the extension is as fast as the possible extension speed without causing ground collision, then

$$\epsilon_b = \gamma_b - \omega \cdot \frac{l_{\max} - (\frac{r_m}{\cos(\gamma_b)} - r_m)}{\dot{l}_{\max}} \,. \tag{4.68}$$

The angle $\gamma_b$ is calculated like $\gamma$ in Equation (4.11),but using the extension speed rather than the retraction speed. In this case, there is no need for calculating an angle like $\epsilon_s$ as it here refers to the angle on the left side at which the retraction is finished. This angle exists but is obsolete as the retraction at either $1.5\pi$ rad or $\epsilon_b$ ensures collision avoidance. This leads to Algorithm 7, for braking with leverage, taking all boundaries into consideration. For this algorithm, there is no condition if the retraction or extension speed is faster than needed as there is no extension

**Algorithm 7:** Leverage-only braking algorithm with detailed boundaries and limitations.

```
Loop
    calculate γ_b an ε_b
    foreach pole in poles do
        predict ζ with measured and set ω
        if ζ ≥ γ_b then
            extend pole (ground avoidance mode)
        else if ζ > ε_b or ζ ≥ 1.5π then
            extend pole
        else
            retract pole
        end
    end
EndLoop
```

slow enough to fulfill this requirement, ignoring the trivial case of $\dot{l}_{\max} = 0$ and assuming the extension is as fast as the retraction.

### Braking by pushing

Next, we focus on braking by the pushing approach. For this, the poles on the left side extend so that they will be in contact with the ground, opposing the ongoing rotation. There are three main approaches to be discussed:

- Soft brake: Braking slowly with changing pole speeds.
- Hard brake without lift-off: Extending poles before ground contact but not retracting them. The impact does not lift the robot of the ground.
- Hard brake with lift-off: Extending poles before ground contact but not retracting them. The impact does lift the robot off the ground to a maximum angle.

The advantage of the hard brakes is the possible massive deceleration compared with that achieved with the soft brakes. Moreover, the impact on the sphere and payload are accordingly huge. The hard brake has an even bigger impact with lift-off as the sphere will also fall backward, leading to a second impact. For the soft brake, the pole needs to extend until the point where the retraction is faster than the needed retraction once the touchpoint is reached. This was exactly the definition of $\gamma$, which gave Equation (4.11). However, rather than using it to avoid ground contact by having a superior retraction speed, we use it to force this contact in a controlled manner. Equation (4.3) gives us the required retraction speed at a certain angle $\zeta$. An extension near or at 0 rad does not have much impact as the force will just lift the sphere up. Therefore, we utilize the previously used angle $\beta$ until which we will retract. Between $\beta$ and 0 rad, the pole will retract at full speed, not contributing to the braking. Hence, we need to specify how much one pole within its $\gamma$ to $\beta$ range slows down the overall $\omega$. We refer to this

value as $\Delta\omega$. Let $\omega_{\text{start}}$ be the $\omega$ at the $\gamma$ of a specific pole, $\omega_{\text{end}}$ the desired $\omega$ at $\beta$ of that same specific pole, and $\dot{l}_{sb}(\zeta)$ the pole speed at angle $\zeta$ for soft braking, then

$$\dot{l}_{sb}(\gamma) = r_m \cdot \omega_{\text{start}} \cdot \tan(\gamma) \cdot \sec(\gamma) \qquad \dot{l}_{sb}(\beta) = r_m \cdot \omega_{\text{end}} \cdot \tan(\beta) \cdot \sec(\beta)\,. \tag{4.69}$$

It is certainly up to the requirements of the specific robot to adapt to the transition between these two $\omega$s; we will assume a desired linear deceleration. Yet, the linear deceleration of $\omega$ does not lead to a linear change in the retraction speed. Let $\gamma(\omega_x)$ be the $\gamma$ for the specific $\omega_x$ and not the current $\omega$, then $\dot{l}_{sb}(\zeta)$ is

$$\dot{l}_{sb}(\zeta) = r_m \cdot \left(\omega_{\text{end}} + \frac{\omega_{\text{start}} - \omega_{\text{end}}}{\frac{\zeta-\beta}{\gamma(\omega_{\text{start}})-\beta}}\right) \cdot \tan(\zeta) \cdot \sec(\zeta)\,. \tag{4.70}$$

This is the solution for speed-controlled poles, assuming they have been extended at $\gamma$ to the maximum possible length. This speed control does not hold true for poles between $\gamma$ and $\beta$ at the moment the braking is commanded. Let $l_{sb}(\zeta)$ be the required length at a specific $\zeta$ between $\gamma(\omega_{\text{start}})$ and $\beta$ for braking, then

$$\begin{aligned} l_{sb}(\zeta) &= \int_{\gamma(\omega_{\text{start}})}^{\zeta} r_m \cdot (\omega_{\text{end}} + \frac{\omega_{\text{start}} - \omega_{\text{end}}}{\frac{\zeta-\beta}{\gamma(\omega_{\text{start}})-\beta}}) \cdot \tan(\zeta) \cdot \sec(\zeta)\, d\zeta \\ &= \frac{1}{2}(r_m \cdot \zeta \cdot \tan(\zeta) \cdot \sec(\zeta))^2 \left(\frac{(\gamma(\omega_{\text{start}}) - \beta) \cdot (\omega_{\text{start}} + \omega_{\text{end}})}{\zeta - \beta} + \omega_{\text{end}}\right)^2 \\ &\quad - \frac{\gamma(\omega_{\text{start}})^2}{2} + \frac{r}{(cos(\gamma(\omega_{\text{start}})))}\,. \end{aligned} \tag{4.71}$$

Now, we evaluate the question of a lift-off as there needs to be a speed of the robot that cannot be compensated in one period of braking (which translates to $\omega_{\text{end}} = 0\,^{rad}/_{\text{s}}$), as the sphere will take this pole as a lever and start accelerating vertically, like an athlete pole-vaulting. We will not evaluate this for the worst case with a friction coefficient of $\mu_{\text{pole}}$ of 1. Figure 4.18 illustrates the occurring forces during braking. As the real system is highly complex and depends on the bending of the pole, the consistency of the ground, etc., we simplify it for the lift-off evaluation. We further assume that until the evaluated moment, the braking worked as intended. Therefore, the braking force $F_b$ is generated by the deceleration achieved by the reduction of $\omega$, as

$$F_b = m \cdot \dot{v} = m \cdot r_m \cdot \dot{\omega}\,. \tag{4.72}$$

The force $F_\omega$ is countered by the ground and is likely to cause a rotation backward depending on the poles. However, as we consider the worst for the lift-off, we assume full countering of $F_\omega$. As $F_b$ is not parallel to the pole, but the pole is the axis along which the force is transferred, it initiates a force perpendicular to $F_b$, $F_l$. $F_l$ is given by

$$F_l = \tan(\zeta) \cdot F_b\,. \tag{4.73}$$

$F_l$ is countered by the gravitational force $F_g$. To get the maximum possible braking $\omega$, we set

$$\begin{aligned} F_l &= F_g \\ \tan(\zeta) \cdot F_b &= m \cdot g \\ \tan(\zeta) \cdot m \cdot r_m \cdot \dot{\omega} &= m \cdot g\,. \end{aligned} \tag{4.74}$$

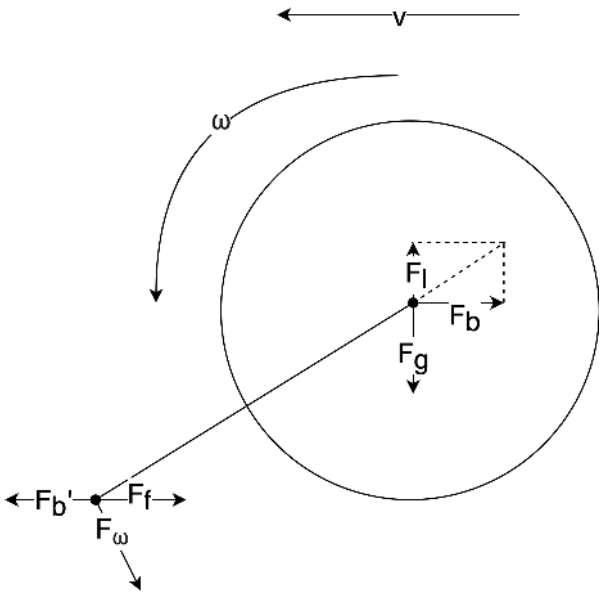



**Figure 4.18:** Forces during braking when rolling onto a pole.

Considering $t$ as the elapsed time between $\omega_{\text{start}}$ and $\omega_{\text{start}}$ and taking into account that we designed the braking for the linear deceleration of $\omega$, we write

$$\dot{\omega} = \frac{\omega_{\text{start}} - \omega_{\text{end}}}{t} = \frac{\omega_{\text{start}} - \omega_{\text{end}}}{(\gamma(\omega_{\text{start}}) - \beta) \,/\, \left(\frac{\omega_{\text{start}} - \omega_{\text{end}}}{2}\right)} = \frac{(\omega_{\text{start}} - \omega_{\text{end}})^2}{2 \cdot (\gamma(\omega_{\text{start}}) - \beta)} \,. \tag{4.75}$$

Inserting this into Equation (4.74) gives

$$\tan(\zeta) \cdot r_m \cdot \frac{(\omega_{\text{start}} - \omega_{\text{end}})^2}{2 \cdot (\gamma(\omega_{\text{start}}) - \beta)} = g \,. \tag{4.76}$$

Solving this for $\omega_{\text{end}}$, looking at the last angle of relevance, which is $\zeta = \beta$, and ignoring the negative solution, we get

$$\omega_{\text{end}} = \sqrt{\frac{2 \cdot g \cdot (\gamma(\omega_{\text{start}}) - \beta)}{\tan(\beta) \cdot r_m}} - \omega_{\text{start}} \,. \tag{4.77}$$

This is the maximum possible $\omega_{\text{end}}$ to avoid lift-off. If we want to brake completely in one period, $\omega_{\text{end}}$ needs to be $\leq 0\,{}^{rad}/_{\text{s}}$. This holds if

$$\sqrt{\frac{2 \cdot g \cdot (\gamma(\omega_{\text{start}}) - \beta)}{\tan(\beta) \cdot r_m}} \leq \omega_{\text{start}} \,. \tag{4.78}$$

If needed, one can solve this equation for $\omega_{\text{start}}$ to get the maximum possible braking speed for the minimum possible $\beta$. So far, we carried out the evaluation of the slow brake. For the hard brake, we again start by using Equation (4.74). It is impossible to specify the $\dot{\omega}$ without exact

knowledge of the robot and, more specifically, its poles. Therefore we set $\dot{\omega} = \frac{\omega_{\text{start}}}{t}$, where the time constant t of the complete deceleration is highly dependent on the robot. For $\zeta$, we can set $\arccos(\frac{r_m}{r_m + l_{\max}})$, as the maximum possible $\zeta$ is beneficial for braking as the lift-off is made more unlikely. Yet, this leads to increased stress on the poles. We also need to consider the rotation, which is stopped, as the reduction is not continuous as in the soft brake, but abrupt. This is only of relevance if the pole gets stuck in the ground, as otherwise, a bounce in the other direction counters the rotation. So, we assume the worst case that the overall rotational energy $e_{\text{rot}} = 0.5 \cdot I\omega^2$ is completely transformed into kinetic energy at the right angle $e_{\text{kin}} = 0.5 \cdot m \cdot v^2$. Simplifying the structure to a single mass point on a massless pole produces a moment of inertia $I$ of $(r_m + l_{\max})^2 \cdot m$. Setting $e_{\text{rot}} = e_{\text{kin}}$, we get

$$\begin{aligned} 0.5(r_m + l_{\max})^2 \cdot m \cdot \omega^2 &= 0.5 \cdot m \cdot v^2 \\ v &= (r_m + l_{\max}) \cdot \omega \,. \end{aligned} \tag{4.79}$$

Therefore, besides $F_l$, there is the worst case by rotational force $F_{wcr}$, defined by

$$F_{wcr} = m \cdot \frac{v}{t} = \frac{(r_m + l_{\max}) \cdot \omega_{\text{start}}}{t} \cdot m \,. \tag{4.80}$$

Overall, the hard stop without lift-off is possible if

$$\begin{aligned} F_l + F_{wcr} &\leq F_g \\ \tan(\arccos(\frac{r_m}{r_m + l_{\max}})) \cdot r_m \cdot \frac{\omega_{\text{start}}}{t} + \frac{(r_m + l_{\max}) \cdot \omega_{\text{start}}}{t} &\leq g \\ \frac{\omega_{\text{start}}}{t} \left( \tan(\arccos(\frac{r_m}{r_m + l_{\max}})) \cdot r_m + (r_m + l_{\max}) \right) &\leq g \\ \frac{\omega_{\text{start}}}{t} \left( (l_{\max} + r_m) \cdot \left( 1 + \sqrt{1 - \frac{r_m^2}{(r_m + l_{\max})^2}} \right) \right) &\leq g \,. \end{aligned} \tag{4.81}$$

Solving this for the maximum brakable $\omega$ leads to

$$\omega \leq \frac{g \cdot t}{(l_{\max} + r_m) \cdot \left( 1 + \sqrt{1 - \frac{r_m^2}{(r_m + l_{\max})^2}} \right)} \tag{4.82}$$

For the hard stop with the acceptance of a lift-off, we take the energy conversation principle to evaluate the brakable speed. For the worst case, we simplify that all kinetic energy and rotation energy is converted to potential energy. Due to loss at nearly every point, this is a pessimistic maximum approach. We quickly evaluate this as

$$\begin{aligned} e_{\text{rotation}} + e_{\text{kinetic}} &= e_{\text{pot}} \\ 0.5 \cdot I \cdot \omega^2 + 0.5 \cdot m \cdot v^2 &= m \cdot g \cdot h \,. \end{aligned} \tag{4.83}$$

To ensure that the robot will always fall back and never tip over the perpendicular point, we set $h$ to the maximum possible $r_m + l_{\max}$. This gives

$$0.5 \cdot \frac{I}{m} \cdot \omega^2 + 0.5 \cdot v^2 < g \cdot (r_m + l_{\max}) \,. \tag{4.84}$$

We again simplify the structure to a single mass point at a massless pole, producing a moment of inertia $I$ of $m \cdot r_m^2$. Also, $v = r_m \cdot \omega$ is another simplification as we assume full friction and therefore direct translation from rotation to transition. Putting this into Equation (4.84) and solving for $\omega$ leads to

$$\begin{aligned} 0.5 \cdot r_m^2 \cdot \omega^2 + 0.5 \cdot (r_m \cdot \omega)^2 &< g \cdot (r_m + l_{\max}) \\ \omega &< \sqrt{\frac{g \cdot (r_m + l_{\max})}{r_m^2}} \,. \end{aligned} \tag{4.85}$$

This is the maximum brakable $\omega$ if lift-off is allowed. We need to stress that it is very likely that there will be no lift-off but a bounce and/or abrupt change of rotational direction. After all, hard stops are seen more as emergency procedures or as an option for robots wherein the actual controllability is not important. Sometimes, it is simply the only available option in robots. This is the case with robots using mono-speed poles. The lack of pole-retraction speed control makes it impossible to perform a soft stop. Therefore, hard stops are the only usable braking procedure for the prototype.

### 4.1.5 Slopes and Obstacles

One reason the DAEDALUS project included a pole mechanism was because of obstacles and its assumed better capability to climb slopes and overcome obstacles. Therefore, we analyze parameters restricting and influencing these capabilities.

#### Obstacles

First, we examine perpendicular obstacles, as the forces are rather simple for that kind of setup. The gravitational force $F_G$ pulls the sphere down, and the Force $F_p$ of the extending pole pushes the sphere up. If $F_G < F_p$, we are able to push the sphere directly upwards. In theory, as there are no forces to the front and back of the sphere, it pushes itself straight up without touching the obstacle. DAEDALUS uses an internal weight that can be shifted to the front for ensuring that it does not fall backward (this was already shown in Figure 2.4). In practice, we can just have a minimum angle of the pole. This introduces a frictional force $F_f$ due to the movement and a direct force due to the gravitational force in the direction of the wall $F_w$ due to the angle of the pushing. Let $\gamma$ be the friction coefficient between the surface of the sphere and wall, then

$$F_f = \gamma \cdot F_w \,. \tag{4.86}$$

Let $\zeta$ be the angle of the extending pole, then

$$F_w = F_G \cdot \tan(\zeta) \,. \tag{4.87}$$

So, to initiate climbing, we need

$$F_p > \sqrt{(F_G + F_f)^2 + F_w^2} = \sqrt{(F_G + \gamma \cdot F_G \cdot \tan(\zeta))^2 + (F_G \cdot \tan(\zeta))^2} \,. \tag{4.88}$$

Figure 4.19 illustrates the forces. So, with the $\zeta = 0\,\text{rad}$, Equation (4.88) reduces to the trivial

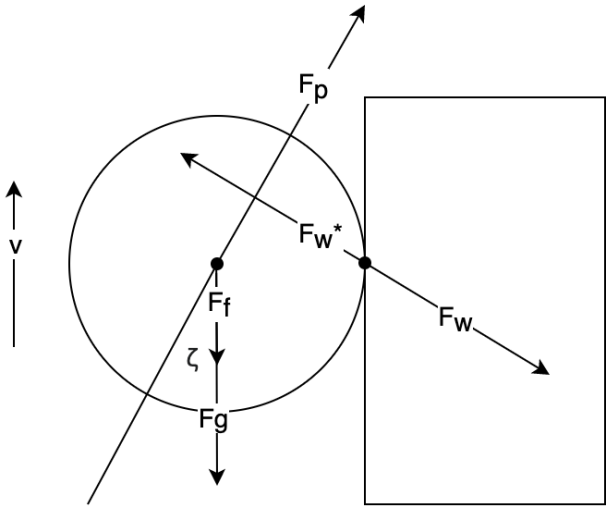



**Figure 4.19:** Illustration of force for climbing a perpendicular obstacle.

$$F_p > F_G\,. \tag{4.89}$$

With a larger $\zeta$, the pushing of the pole will increasingly start to push directly against the wall. Equation (4.88) represents this by the tan, which increases with larger $\zeta$. The infinite value at $\tan(0.5\pi)$ represents a direct pushing against the wall, which cannot cause the sphere to be raised. Therefore, the angle $\zeta$ is crucial. The question of a minimum $\zeta$ depends on whether the leverage algorithm can be used or not. With the leverage algorithm, the minimum $\zeta$ simplifies as the sphere can be pushed with a $\zeta$ of 0 rad. Once the sphere reaches the top, it will just tip onto the obstacle. Figure 4.20 depicts this concept.

The transition from the perpendicular upwards pushing to the translation to the right of $0.5 \cdot r_m$ shows a small change of the angle. Yet, in contrast with the evaluation of a small angle while pushing upwards, the obstacle does slip under the sphere, therefore supporting its weight. We ignore this angle as there are two options.

First, the sphere has been balanced in such a perfect way that it does not touch the wall, which is highly unlikely. In that case, the vertical pole extends until the sphere is completely above the obstacle. Then the robot initiates a tip over to the side by extending one pole, leading to a shift in the center of mass in that direction.

Second, the sphere already has a minimal angle to stay close to the wall. In this case, a complex transition takes place. With every new length, $\zeta$ changes, the direction and amount of force from the wall to the sphere change towards a supporting character, and the needed resulting force compensate gravity increases with $\zeta$. At the beginning the required force is the same as with a higher obstacle at the same position, and at the end it is almost zero, because the obstacle compensates the gravity. The analytical solution for this would go far beyond the scope of this thesis. However, we limit the amount as the needed force does not exceed the force with the same maximum $\zeta$ while still climbing. Let $\zeta_{\text{end}}$ be the angle of the pole on top of the obstacle, $\zeta_{\text{start}}$ the angle at the start of the transition, $h$ the height of the obstacle, and $r_m$ the radius of the sphere, then

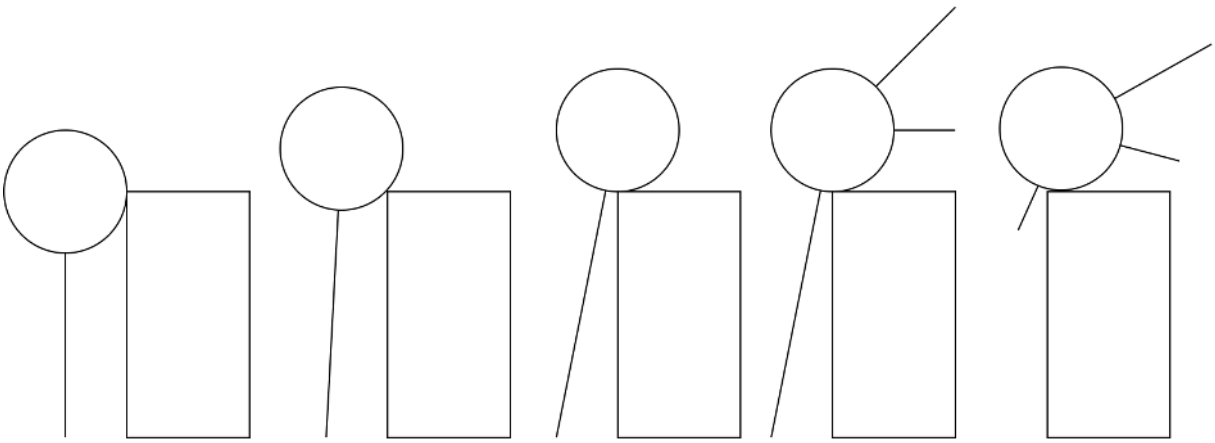


**Figure 4.20:** The sequence of pushing the sphere up with the leverage algorithm. An initial $\zeta$ as low as 0 rad is valid for this procedure.

$$\zeta_{\text{end}} = \arctan\left(\frac{\tan(\zeta_{\text{start}}) \cdot h + r_m}{h + r_m}\right) = \arctan\left(\frac{\tan(\zeta_{\text{start}}) \cdot h/r_m + 1}{h/r_m + 1}\right). \tag{4.90}$$

Let $\zeta_\Delta$ be the change of $\zeta$ due to the movement onto the top of the obstacle, then

$$\zeta_\Delta = \arctan\left(\frac{\tan(\zeta_{\text{start}}) \cdot h/r_m + 1}{h/r_m + 1}\right) - \zeta_{\text{start}} \tag{4.91}$$

The calculation for this is included in the Appendix B.1. With a $\zeta_{\text{start}}$ of 0, $\zeta_\Delta$ becomes

$$\zeta_\Delta = \arctan\left(\frac{1}{h/r_m + 1}\right). \tag{4.92}$$

To get the maximum possible $\zeta_\Delta$, we need to lower the ratio between $h$ and $r_m$. The lowest realistic ratio for this evaluation is $h/r_m = 1$, as its whole purpose is to narrow the maximum extra force required when pushing the sphere over the top, where the endpoint needs less force than the starting point. Therefore, a smaller ratio than 1 does not make sense. Considering this, we get a maximum $\zeta_\Delta$ of 0.464 rad. So, taking this as additional $\zeta$ in Equation (4.88) and a normal $\zeta = 0$ rad that leads to $F_p > F_G$, we get an additional needed force of

$$F_{\text{additional}} = F_G - \sqrt{(F_G + \gamma \cdot F_G \cdot \tan(0.464))^2 + (F_G \cdot \tan(0.464))^2}\,. \tag{4.93}$$

For a friction coefficient of 0.2, the extra force needed is smaller than 21 %. As mentioned previously, this is the absolute maximum, which ignores every change of $F_w$, and we assume that it is much lower or in fact always lower than the force required for pushing upwards. So, if the robot is capable of movement by the leverage approach and able to balance itself perfectly, the poles need a power of $F_G$. If it is not capable of doing so, Equation (4.93) provides the maximum extra power required. If the leverage approach is not possible or available, the whole estimation becomes different. This is because the rotation starts on top of the obstacle, which

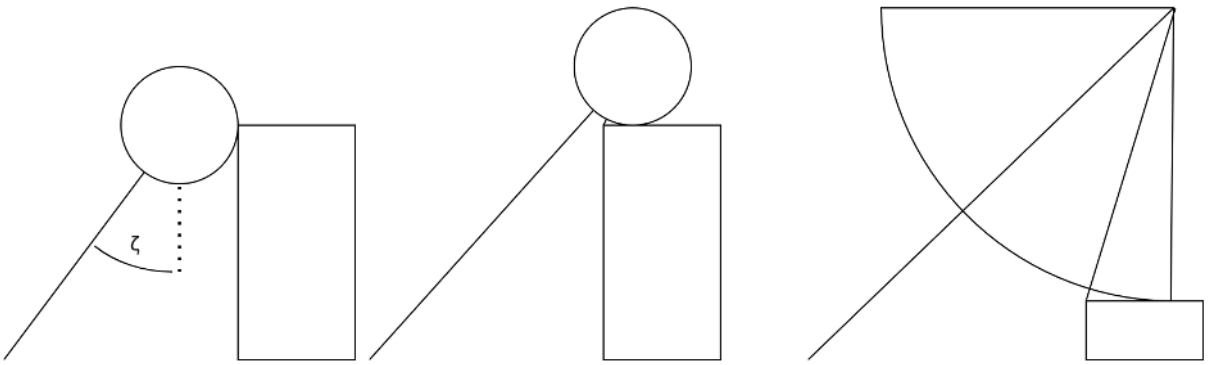



**Figure 4.21:** The sequence of pushing the sphere up with the pushing algorithm. The initial $\zeta$ cannot be 0 rad.

requires a certain angle. Figure 4.21 shows this concept. Although climbing is possible with $\zeta = 0\,\text{rad}$, the locomotion at the top requires a certain angle for the next pole to push the sphere for locomotion. The pole intended for locomotion needs to exceed the $\beta$ value, which is $> 0\,\text{rad}$. The angle between the poles is fixed by the number of poles, $n$. Let $\zeta_{\text{end}}$ be the last required $\zeta$ for overcoming the obstacle and the next pole being able to start locomotion, then

$$\zeta_{\text{end}} \geq \beta + \frac{2\pi}{n}\,. \tag{4.94}$$

However, the actual ascension angle is much steeper at the beginning, as the point of the pole touching the ground is the same, but the sphere is lower. This initial angle $\zeta_{\text{bottom}}$ depends on the height of the obstacle $h$ and the radius of the sphere $r_m$. Figure 4.22 depicts the variables and their placement. Let $x$ be the distance between the touchpoint of the pole and the line perpendicular to the ground that touches the sphere at one point, then

$$\zeta^*_{\text{bottom}} = \arctan(\frac{x + r_m}{r_m}) \tag{4.95}$$

and

$$\zeta^*_{\text{end}} = \arctan(\frac{x + 2 \cdot r_m + \tan(\beta) \cdot r_m}{h + r_m})\,. \tag{4.96}$$

Solving Equation (4.96) for $x$ leads to

$$x = \tan(\zeta^*_{\text{end}}) \cdot (h + r_m) - 2 \cdot r_m - \tan(\beta) \cdot r_m \tag{4.97}$$

and putting it in Equation (4.95) gives

$$\zeta^*_{\text{bottom}} = \arctan\left(\frac{\tan(\zeta^*_{\text{end}}) \cdot (h + r_m) - 2 \cdot r_m - \tan(\beta) \cdot r_m + r_m}{r_m}\right) \tag{4.98}$$

$$\zeta_{\text{bottom}} = 0.5\pi - \arctan\left(\tan(\zeta^*_{\text{end}}) \cdot (h/r_m + 1) - 1 - \tan(\beta)\right)\,. \tag{4.99}$$

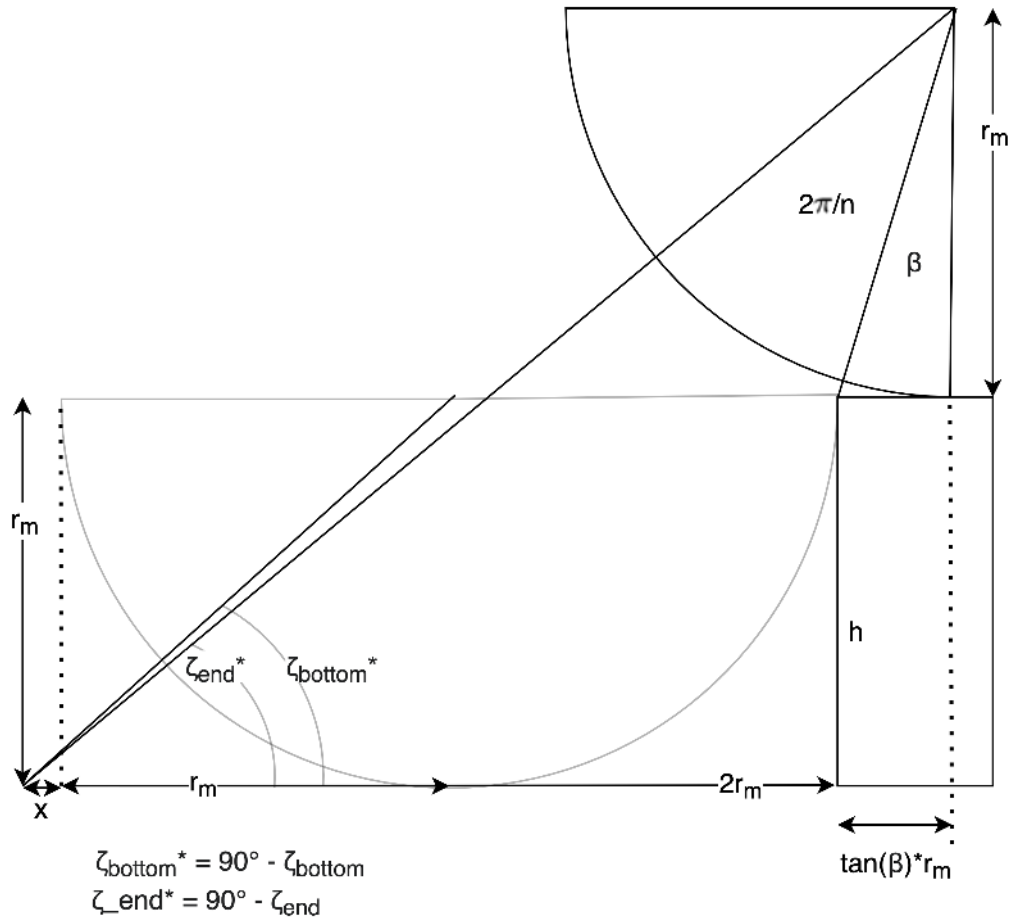



**Figure 4.22:** Illustration of variables for calculating the needed $\zeta$ to overcome obstacles without using the leverage approach atop the obstacle.

Taking the minimum possible value of the original Equation (4.94) and putting it in Equation (4.99), it becomes

$$\zeta_{\text{bottom}} = 0.5\pi - \arctan\left(\tan(0.5\pi - \frac{2\pi}{n} - \beta) \cdot (h/r_m + 1) - 1 - \tan(\beta)\right) . \tag{4.100}$$

To reduce the the minimum possible $\zeta_{\text{bottom}}$, we can

- increase the number of rods, $n$, to have smaller angles between the poles,
- decrease $\beta$ by using stronger actuators,
- increase the radius $r_m$ (this also increases the needed pole length), or
- reduce the obstacle height the robot wants to overcome.

If the calculated $\zeta_{\text{bottom}}$ is larger than $\frac{2\pi\,\text{rad}}{n}$, then there exists a pole where $\theta$ is smaller than $\zeta_{\text{bottom}}$ but larger than 0 rad. Therefore, it starts the pushing at the bottom, and once reaching higher points, the other pole takes over. If there lie multiple poles between the pole needed for achieving $\theta_{\text{end}}$ and the starting pole, it is not beneficial in terms of maximum needed force, to change the working pole to a pole between those two. The first one has the steepest angle, which cannot possibly become negative (falling backward), if it is capable of the starting configuration. So, the first pole does the main part of the pushing, and the original pole provides the right angle at the end. It is important to note that the first pole needs to start at a certain $\theta$, ensuring the right angle for the end position. So, the starting angle $\theta_{\text{BottomMain}}$ for a pole is given by a modified version of Equation (4.100)

$$\zeta_{\text{bottomMain}} = 0.5\pi - \arctan\left(\tan(0.5\pi - \frac{2\pi}{n} - \beta)\cdot(h/r_m + 1) - 1 - \tan(\beta)\right) \pmod{\frac{2\pi}{n}}. \tag{4.101}$$

If there is one pole that can do the main extension before the other pole takes over at almost the top of the obstacle, it does influence the $\zeta$ at which a pole must be able to push the sphere upwards, which in turn influences the needed force. Here starts the same problem as with the leverage approach: The transition from the upward movement to the sideways movement is complicated and highly dependent on, for instance, materials. Again, we only consider the maximum possible forces and therefore look at the maximum possible angle, ignoring it while pushing against the wall or at the top of the obstacle. Even without the possibility of the second pole, the maximum angle is dependent on the ratio of $h$ and $r_m$, $\beta$, and the number of rods. As all four variables interact with each other, we define the maximum angle as

$$\zeta_{\max} = \max\left(\left(0.5\pi - \arctan\left(\tan(0.5\pi - \frac{2\pi}{n} - \beta)\cdot(h/r_m + 1) - 1 - \tan(\beta)\right) \pmod{\frac{2\pi}{n}}\right), \left(\frac{2\pi}{n} + beta\right)\right). \tag{4.102}$$

Then, $\zeta_{\max}$ is used with Equation (4.88) to calculate the force required for a pole to enable perpendicular obstacle climbing. Until now, we did not take the length of the pole $l$ into account, limiting the maximum obstacle size. For both approaches (leverage and pushing), the end position defines the minimum needed $l$, $l_{\text{minObs}}$. $l_{\text{minObs}}$ is given by

$$l_{\text{minObs}} = \frac{h + r_m}{cos(\zeta_{\text{end}})}. \tag{4.103}$$

For push locomotion, this reduces to

$$l_{\text{minObs}} = \frac{h + r_m}{\cos\left(\frac{2\pi}{n} + \beta\right)}. \tag{4.104}$$

For leverage locomotion, this gives

$$l_{\text{minObs}} = \frac{h + r_m}{\cos\left(\arctan\left(\frac{\tan(\zeta_{\text{start}})\cdot h/r_m + 1}{h/r_m + 1}\right)\right)}. \tag{4.105}$$

Equation (4.104) does lead us to the minimum number of poles for the pushing algorithm. To get a non-negative length, we need at least

$$n > \frac{2\pi}{0.5\pi - \beta} \geq 4 \tag{4.106}$$

poles. The minimum number of poles for the leverage algorithm to overcome perpendicular obstacles is not based on Equation (4.105) but on the nature of the leverage locomotion. For overcoming obstacles, two poles are enough, as one pushes with sufficiently small $\zeta_{\text{start}}$ and the other extends to generate force over leverage. However, two poles are not enough for a valid, self-starting leverage robot as it requires at least three poles.

### Slopes

Next, we evaluate the possibility of TLDR robots climbing slopes. We will evaluate the minimum force required for remaining stationary at the slope and not a further system description as this would just lead to a bulkier version of Equations 4.53 and 4.62, whereas the needed force of the poles for certain slope degrees receives bigger interest. We take the initial evaluation of the force shown in Figure 4.16 for pushing and Figure 4.17 for leverage and pair it with the force evaluation of the standard physics problem of a cylinder rolling down a slope. Figure 4.23 shows the resulting system for a slope with angle $\xi$. On the slope, the gravitational force of the robot itself is no longer neglectable, nor does it counter trivial forces. It is split into a force horizontal to the plane $f_h$ and a normal force $F_n$. It is split as follows:

$$\boldsymbol{F}_g = F_g \cdot \begin{bmatrix} \sin(\xi) \\ \cos(\xi) \end{bmatrix} = \begin{bmatrix} F_h \\ F_n \end{bmatrix} . \tag{4.107}$$

$F_n$ is now the force countered by the structural force of the robot and ground and therefore not of interest for this evaluation. $F_h$ initiates a counterforce due to the friction between shell and ground, $F_{fh}$, given by

$$\boldsymbol{F}_{fh} = \mu_{rs} \cdot \boldsymbol{F}_h . \tag{4.108}$$

With $\mu_{rs} = 1$, the sphere rolls down the slope with no slip, with 0, it just slips down the slope without rotation. To have a standstill, both need to be countered completely. For the pushing algorithm (marked blue in Figure 4.23), these forces are the resulting lever force $F_e$ and the force $F_r$, which act at the center of the robot. In this case, $F_{fh}$ arises from the difference between $F_r$ and $F_h$. There are now two options. First, counter $F_h$ completely by $F_r$. This will lead to a $F_{fh} = 0$; therefore, this is enough for the robot for this slope. However, this is no complete balance as there is still a component $F_e$ that initiates torque and hence rotation. This means that the calculated force is enough but not the possible minimum. The second one is to counter $F_h$ not completely but just enough that the resulting counterforce $F_{fh}$ is countered by $F_e$, and the difference of both initiates enough rotation to compensate for the direct initiated horizontal acceleration from the resulting force at the midpoint, which is there as $F_h$ is not countered completely at the midpoint. Practically, this means that the sphere is always rolling on the same spot. The first one is a rather quick evaluation. Note that here it is important to use an adapted $\zeta_\xi$ as the absolute pitch of the robot defines $\zeta$, but for the equations to still hold, we

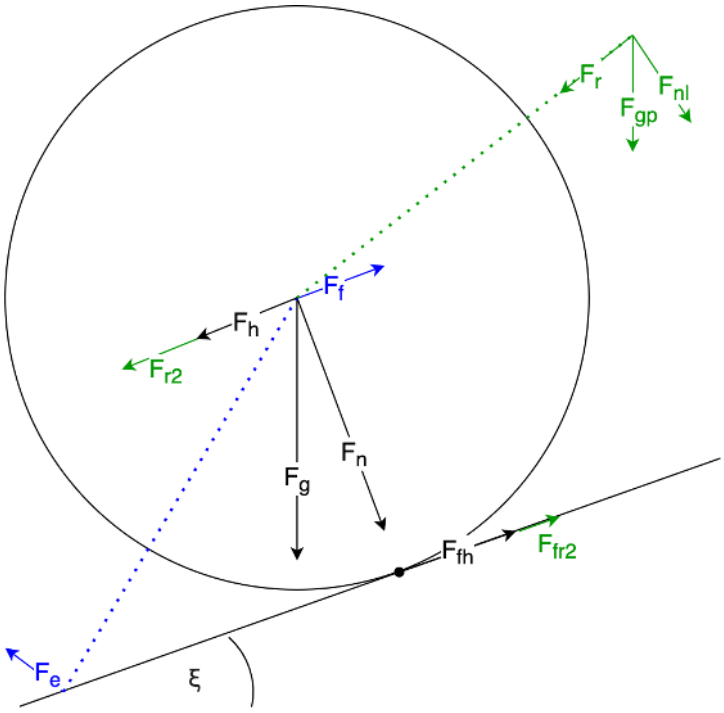



**Figure 4.23:** Force evaluation of a TLDR robot on a slope. The blue parts are the forces arising from the pushing approach, while the green parts denote those from the leverage approach.

need to evaluate from the point of view of the slope (askew to the actual ground). This means the original $\zeta$ is enlarged by $\xi$; therefore, we define $\zeta_\xi = \zeta + \xi$. We now use Equation (4.40), defining $F_f$, and set it to counter $F_h$. Solving for the force of the pole $F_p$ leads to

$$\begin{aligned} F_h &= F_f \\ F_g \cdot \sin(\xi) &= \mu_{s\text{Pole}} \sin(\zeta_\xi) \cdot F_p \\ F_p &= \frac{m \cdot g \cdot \sin(\xi)}{\mu_{s\text{Pole}} \cdot \sin(\zeta_\xi)} \end{aligned} \tag{4.109}$$

It is trivial that $\mu_{s\text{Pole}}$ needs to be greater than 0 as there is no possibility for climbing a perfect slippery slope. Solving Equation (4.109) for $\xi$ with $\zeta = 0.25\pi\,\text{rad}$ gives

$$\xi = 2 \cdot \arctan\left( \frac{\sqrt{2}\left( \sqrt{\left(\frac{F_p \cdot \mu_{s\text{Pole}}}{m \cdot g}\right)^2 - \sqrt{2}\left(\frac{F_p \cdot \mu_{s\text{Pole}}}{m \cdot g}\right) + 1} + 1 \right)}{\left(\frac{F_p \cdot \mu_{s\text{Pole}}}{m \cdot g}\right)} \right). \tag{4.110}$$

For the later-introduced exemplary prototype (assuming perfect grip, $F_p = 24.5\,\text{N}$, $m = 25\,\text{kg}$), this leads to a maximum slope of $\xi = 0.071\,\text{rad} = 4.1\,\text{deg}$. Just for reference, at the lunar surface

with a gravitational acceleration of $1.61\,{}^{m}/_{s^2}$, the same setup is able to climb slopes of a maximum $\xi = 0.643\,\text{rad} = 36.8\,\text{deg}$. Ivanova et al. evaluate the maximum possible slope for pendulum-driven robots in [33]. Their example uses a robot with the values $m = 2\,\text{kg}$ and$r = 1\,\text{m}$. The rest of the parameters is not of interest to our evaluation. Still, their pendulum drive is an optimized realistic setup that is also realizable as our prototype in the low-budget sector. They estimate the maximum slope as $0.125\,\text{rad} = 7.16\,\text{deg}$. If we apply the TLDR robot to the same mass and radius and take realistic pole strengths between 1 N and 2 N, we get a maximum slope between $0.076\,\text{rad} = 4.35\,\text{deg}$ and $0.167\,\text{rad} = 9.57\,\text{deg}$. For the pole mechanism in general, there is no real general slope limit, as hydraulic poles weighing less than 0.5 kg easily bring up 300 N. Inserting such huge power leads to maximum slopes of $0.5\pi\,\text{rad}$, which means these poles are capable of pushing the robot straight up, like described for perpendicular obstacles, at least in a purely mathematical sense.

After all this, we look at the previously described second possibility to define the maximum slope, where dynamically, the downwards acceleration is compensated by the rotation using leverage. This will result in a lower needed force of the pole, maintaining the same maximum slope. However, having a continuous moving system for maintaining the position is not recommended, as we suggest classifying a slope as climbable if the robot can stop and scan in the middle without having a complex situation to control. Still, in this case, we have a system of two equations, one regarding the torque and the other regarding the $F_h$. The unchanging position defines the system, therefore

$$\begin{aligned} a_{\text{rotation}} - a_{\text{direct}} &= 0 \\ \mu_{rs} \cdot 2\pi \cdot r_m \cdot \dot{\omega} - (F_h - F_f)/m &= 0 \\ \mu_{rs} \cdot 2\pi \cdot r_m \cdot \dot{\omega} - (\sin\xi \cdot F_g - \mu_{s\text{Pole}} \cdot \sin(\zeta_\xi) \cdot F_p)/m &= 0\,. \end{aligned} \tag{4.111}$$

With the knowledge of Subsection 4.1.3, we calculate $\dot{\omega}$ with the differences of both torques $\tau_{fh}$ and $\tau_e$, leading to

$$\begin{aligned} \dot{\omega} &= \frac{\tau_e - \tau_{fh}}{I} \\ \dot{\omega} &= (F_e \cdot r_{\text{PoleEnd}} - (F_h - F_f) \cdot \mu_{rs} \cdot r_m) \cdot I^{-1} \\ \dot{\omega} &= \left( cos(\zeta_\xi)^2 \cdot F_s(1 - \mu_{s\text{Pole}}) \cdot \frac{1}{r_m} - (\sin\xi \cdot F_g - \mu_{s\text{Pole}} \cdot \sin(\zeta_\xi) \cdot F_p) \cdot \mu_{rs} \cdot r_m \right) \cdot I^{-1}\,. \end{aligned} \tag{4.112}$$

Putting this in Equation (4.111) gives

$$\begin{aligned} &\mu_{rs} \cdot 2\pi \cdot r_m \cdot \left( cos(\zeta_\xi)^2 \cdot \sin(\zeta_\xi) \cdot F_p(1 - \mu_{s\text{Pole}}) \cdot \frac{1}{r_m} - (\sin\xi \cdot F_g - \mu_{s\text{Pole}} \cdot \sin(\zeta_\xi) \cdot F_p) \cdot \mu_{rs} \cdot r_m \right) \cdot I^{-1} \\ &- (\sin(\xi) \cdot F_g - \mu_{s\text{Pole}} \cdot \sin(\zeta_\xi) \cdot F_p)\frac{1}{m} = 0 \end{aligned} \tag{4.113}$$

Solving this for $F_p$ leads to

$$F_p = \frac{F_g \cdot \sin(\xi) \cdot \csc(\zeta_\xi)(2\pi \cdot m \cdot r_m^2 I^{-1} \cdot \mu_{rs} + 1)}{2\pi \cdot m \cdot \mu_{s\text{Pole}} \cdot r_m^2 \cdot I^{-1} \cdot \mu_{rs} - 2\pi \cdot m \cdot (\mu_{s\text{Pole}} - 1) \cdot I^{-1} \cdot \cos(\zeta_\xi)^2 \cdot \mu_{rs} + \mu_{s\text{Pole}}}\,. \tag{4.114}$$

Evaluating the case of no slip ($\mu = 1$), this reduces to

$$\begin{aligned} F_p &= \frac{F_g \cdot \sin(\xi) \cdot \csc(\zeta_\xi)(2\pi \cdot m \cdot r_m^2 I^{-1} + 1)}{2\pi \cdot m \cdot r_m^2 \cdot I^{-1} + 1} \\ \Rightarrow F_p &= F_g \cdot \sin(\xi) \cdot \csc(\zeta_\xi) \\ \Rightarrow F_p &= m \cdot g \frac{\sin \xi}{\sin(\zeta_\xi)}\,, \end{aligned} \tag{4.115}$$

leaving us with the same result as Equation (4.109) with no slip. Because every summand in the denominator has at least $\mu_{rs}$ or $\mu_{s\text{Pole}}$ as multiplication, evaluating the case of absolute slip ($\mu = 0$) reduces to non-defined behavior as the denominator becomes 0. Therefore, with a perfect slippery slope, there is no chance of climbing.

Next, we evaluate the maximum slope of the leverage approach, marked green in Figure 4.23. The evaluation focuses on the pure use of the leverage approach, not a combination like in the aforementioned perpendicular obstacle evaluation. We will not derive the forces from scratch as it already has been done in Section 4.1.3 with Figure 4.17. $F_{nl}$ and $F_r$ arise from the gravitational force $F_{gp}$. Let $\tau_{nl}$ be the torque initiated by $F_{nl}$, and $r_c$ be the distance between the midpoint of the sphere and the point of combined leverage by the poles, then

$$\tau_{nl} = r_c \cdot F_{nl} = r_c \cdot (-\sin(\zeta_\xi)) \cdot F_{gp}\,. \tag{4.116}$$

$F_r$ is split up, and $F_{r1}$ is directly countered, but $F_{r2}$ adds up with $F_h$, introducing a linear acceleration and/or frictional force, depending on the $\mu_{rs}$. Let $\tau_s$ be the torque, introduced by the forces $F_{fh}$ and $F_{fr2}$ on the shell of the robot, and using Equation (4.57) for $F_{fr2}$, then

$$\tau_s = (F_{fh} + F_{fr2}) \cdot r_m = (F_h + F_{r2})\mu_{rs} \cdot r_m = (\sin(\xi) \cdot F_g + \frac{\sin(2\zeta_\xi)}{2} \cdot F_{gp}) \cdot \mu_{rs} \cdot r_m\,. \tag{4.117}$$

In contrast with the pushing locomotion, it is not possible to solve this in two ways. The force we can counter directly is the torque $\tau_s$ that is countered by $\tau_{nl}$. However, this does not counter the linear, direct acceleration. Therefore, the initiated rolling motion due to the difference of $\tau_s$ and $\tau_{nl}$ needs to compensate for this direct acceleration $a_{\text{direct}}$. As they are in the opposite direction, we can put

$$\begin{aligned} &a_{\text{direct}} = a_{\text{rotation}} \\ &\Rightarrow \frac{(F_h + F_{r2}) - (F_h + F_{r2}) \cdot \mu_{rs}}{m} = \frac{\tau_{nl} - \tau_s}{I} \cdot 2\pi \cdot r_m \cdot \mu_{rs} \\ &\Rightarrow (F_h + F_{r2}) \cdot \frac{1 - \mu_{rs}}{m} = \frac{r_c \cdot (-\sin(\zeta_\xi)) \cdot F_{gp} - (F_h + F_{r2})\mu_{rs} \cdot r_m}{I} \cdot 2\pi \cdot r_m \cdot \mu_{rs} \\ &\Rightarrow \frac{1 - \mu_{rs}}{m} = \left(r_c \cdot (-\sin(\zeta_\xi)) \cdot \frac{F_{gp}}{(F_h + F_{r2})} - r_m \cdot \mu_{rs}\right) \cdot I^{-1} \cdot 2\pi \cdot r_m \cdot \mu_{rs} \\ &\Rightarrow \frac{1 - \mu_{rs}}{m} = \left(\frac{r_c \cdot (-\sin(\zeta_\xi)) \cdot F_{gp}}{(\sin(\xi) \cdot F_g + \frac{\sin(2\zeta_\xi)}{2} \cdot F_{gp})} - r_m \cdot \mu_{rs}\right) \cdot I^{-1} \cdot 2\pi \cdot r_m \cdot \mu_{rs} \\ &\Rightarrow \frac{1 - \mu_{rs}}{m} = \left(\frac{-r_c}{(\frac{\sin(\xi)}{\sin(\zeta_\xi)} \cdot \frac{m}{m_p} + cos(\zeta_\xi))} - r_m \cdot \mu_{rs}\right) \cdot I^{-1} \cdot 2\pi \cdot r_m \cdot \mu_{rs}\,. \end{aligned} \tag{4.118}$$

At this point, it is already clear that for $\mu_{rs} = 0$, there exists no solution. Solving this equation for the needed mass of the mass point of the combined poles leads to

$$m_p = -\frac{m \sin(\xi) \csc(\zeta_\xi) \left(I^{-1} \cdot 2\pi \cdot r_m^2 \cdot \mu_{rs}^2 - \mu_{rs} + 1\right)}{I^{-1} \cdot 2\pi \cdot r_c \cdot m \cdot r_m \cdot \mu_r s + \cos(\zeta_\xi) \left(I^{-1} \cdot 2\pi \cdot r_m^2 \cdot \mu_{rs}^2 - \mu_{rs} + 1\right)} \,. \tag{4.119}$$

Setting $\mu_{rs} = 1$ and $\zeta_\xi = \frac{5}{4}\pi\,\mathrm{rad}$, representing the maximum needed $m_p$ without slip, the equation simplifies to

$$m_p = \frac{2 \cdot m \cdot r_m \cdot \sin(\xi)}{\sqrt{2} r_c - r_m} \,. \tag{4.120}$$

We can form this equation that it relies on the ratio of mass between the sphere and lever mass point and on the ratio between the radius and the distance to this mass point.

$$\frac{m_p}{m} = \frac{2 \cdot \sin(\xi)}{\sqrt{2} \frac{r_c}{r_m} - 1} \,. \tag{4.121}$$

Therefore, we find the maximum $\xi$ with

$$\xi = \arcsin\left(\frac{m_p(\sqrt{2}\frac{r_c}{r_m} - 1)}{2 \cdot m}\right) \,. \tag{4.122}$$

With parameters roughly describing the later-introduced prototype ($m = 25\,\mathrm{kg}, m_p = 0.4\,\mathrm{kg}, r_m = 0.4\,\mathrm{m}$, and$r_c = 0.85\,\mathrm{m}$), we get a $\xi = 0.016\,\mathrm{rad} = 0.92\,\mathrm{deg}$. With angles of this magnitude, the geometrical constraints can be ignored. However, with steeper slopes, one needs to consider that $r_c$ might be limited as it touches the ground. This is also not a problem as long as $\frac{3}{2}\pi - \zeta \geq \xi$. In this case, the line of the pole is parallel to the slope. However, as we defined it as the combination of all poles, this also does not hold. Therefore, this evaluation needs to be done for every leverage approach-configuration of the specific robot itself.

## 4.2 Balancing

Balancing ensures a constant angle to the sides while rolling forward and remaining stationary. As this is a conservation of a certain condition, closed-loop control is adopted for this problem, rather than an open-loop algorithm like for the locomotion. As this is a vast field and, like most closed-loop controlling systems, very dependent on parameter tuning, we will focus on the problem taking the limitations of our used prototype into account. This means there is no feedback on the actual extension and mono-speed extension.

### 4.2.1 General algorithm

First, we identify all the variables. The output of the system is the measured roll. The input is the desired roll that will always be 0 rad in our case as there is (for now) no reason wanting askew rolling by the poles. Therefore, the measured output is also the error input for the controller. At first sight, there are two possible system inputs: the extension length of a particular pole or the velocity. Due to the lack of real feedback on the length of a pole, the only way for estimating

it is to integrate the time when it was extended or retracted, like it is done in the locomotion approach. This leads to adjusting a predicted length of the pole that does not necessarily match the actual one. Nevertheless, it works as an askew pose is corrected by increasing the length on one side and decreasing it on the other. Still, there is an actual length of each pole at which the roll is 0 rad. Let $r_m$ be the radius of the sphere and $r_s$ the radius of the disc on which the actuators are mounted; then, the length $l$, needed at a certain angle $\zeta$, is given by

$$l_{\text{balance}} = \frac{r_m}{\cos \zeta} - r_s \,. \tag{4.123}$$

Of course, this yields for $\zeta$ between $0.5\pi$ rad and $1.5\pi$ rad. This leads to a minimum $l_{\text{balance}}$ of $r_m - r_s$, showing that controlling the assumed length to control roll is not logical as there is a theoretical optimal extension. Thus, the task is rather to control the pole to actually reach this specific extension for its $\zeta$ at that moment. So, the only useful controllable parameter is the extension and retraction speed as the actual extension is either too short despite the estimated length being correct, or longer than the estimation. This leads to the pole velocity being the system output, and rather than controlling the roll by the theoretical length of the pole with the help of speed, we control the roll directly using speed. So, for poles with variable speed, the speed itself can be taken as system input, which gives a huge variety of controllers that are theoretically able to solve the problem. In our case, with the mono-speed problem, we only can extend, retract, or hold. This comes with simplifications, such as the omitting of an anti-windup system, as there is no unlimited increasing or decreasing value, just the three states with no further changeable value. However, the knowledge of the existence of $l_{\text{balance}}$ does, even with this controller, help. For most controllers, an error of 0 corresponds to no change of the system input as the desired value is reached. In our case, we still want the extension/retraction of a pole until its $l_{\text{balance}}$ is reached because otherwise, every action includes the extension of at least $r_m - r_s$ distance until $l_{\text{balance}}$. This is most likely to be unstable for a finite $\dot{l}_{\max}$. Therefore, the controller needs to ensure the theoretical extension to the contact point with the ground. At a standstill, the controller can do this for the poles on both sides of the sphere (seen in rolling direction).

In Section 4.3, we will find a compromise between movement and rolling. Nevertheless, reducing the poles needed to control is always beneficial as it necessitates fewer interactions between them and ensures fewer conflicts with moving or other actions. Therefore, we define the minimum number of poles as two because two poles together with the bottom of the sphere form, if not in line, a triangle, theoretically giving the sphere the capability for stabilization. The two poles do not need to have the same pole number on both sides, but this does ensure that the three points are not on a line, when excluding a $\zeta$ of 0 deg. Figure 4.24 visualizes this.

Combining two rods on two sides of the sphere for balancing is possible if they do not have the same extension. However, for a rolling motion, the same extension for two chosen poles on both sides will happen periodically. Taking two poles with the same $\zeta$, one on each side, gives the advantage that, with the exception of 0 rad, they always provide a balanceable configuration. It is definitely possible to use more than two poles, but this gives more room for errors. So, we try to minimize the number of used rods for balancing, especially if we add movement. Therefore, we introduce $\alpha_b$ and $\beta_b$, where $\alpha_b \geq \beta_b$, $\beta_b > 0$ rad, and $\alpha_b$ are smaller than the touchpoint angle. Also, $\alpha_b$ needs to be chosen so that at least one pole is always within the boundaries

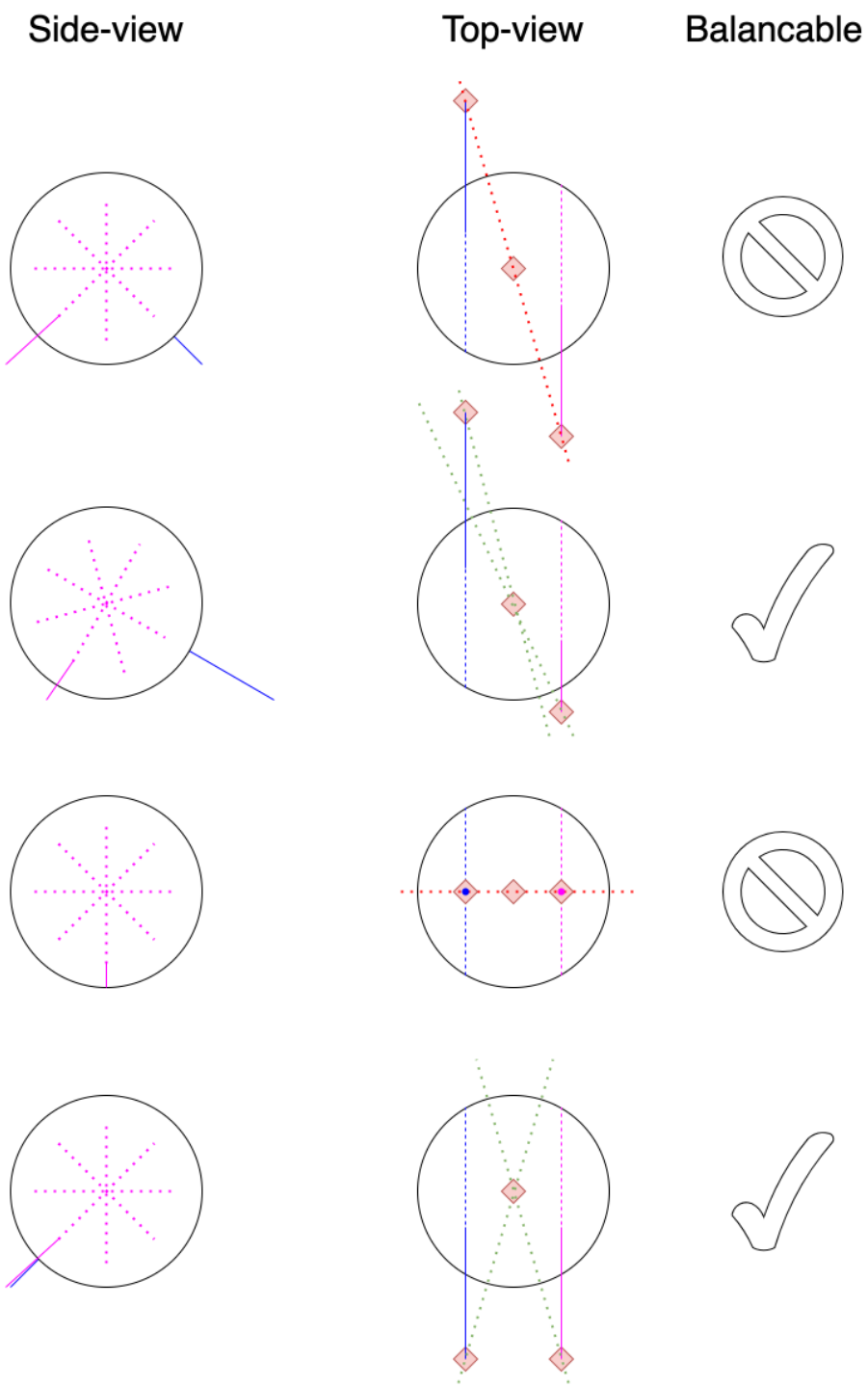



**Figure 4.24:** Visualization of combinations of rod positions relative to the middle and if they are balanceable. They are so if the three contact points to the ground do not build a line. The pink and blue lines represent the two side discs with the corresponding poles. If they are continuous, they represent an extended pole. The red squares show the contact points of the shell and poles with the ground. The green dotted lines indicate that no line fits all points. The red dotted line shows a line that crosses all contact points.

to be activated for balancing. This was not necessarily required with locomotion, as once it started, the momentum kept the rotation alive. Altogether, this leads to

$$\arccos\left(\frac{r}{r+l_{\max}}\right) \geq \alpha_b - \frac{2\pi}{n_{\text{poles}}} \geq \beta_b > 0\,. \tag{4.124}$$

The rod pair, which is between both values, is extended. For a standstill, $\alpha_b$ and $\beta_b$ are mirrored on the axis perpendicular to the ground. This way, extension to balance the sphere, will not lead to the rotation of the overall sphere in a way that only extending poles at one side do. For $\alpha_b$ and $\beta_b$, many argumentations used for $\alpha$ and $\beta$ for the locomotion apply. $\beta_s$ shall not be chosen too small as the extending will not have that much impact near 0 rad. Also, $\alpha_b$ shall be chosen appropriately for the stability of the pole. This means that for very flexible poles, a balancing at full extension will lead more to the bending of the rods if the robot tips over and less to stabilization. It is important that this two-pole approach assumes a perfectly balanced sphere as a starting point. It is not enough to only have three points simultaneously touching the ground, but all three need to be under the load of the sphere. This means all points marked in Figure 4.24 are points on which the mass of the sphere is applied, and only in this case, are the assumptions valid.

If the robot tips to one side, even slightly, the pole on the opposite side may or may not touch the ground, but in both cases, it is not bearing any weight of the robot. Therefore, the stable three-point system becomes an unstable two-point system. Figure 4.25 highlights this. This slight change of $\phi$ is a problem for the three-point approach. As all three points need to be stressed with the weight, the center of mass needs to stay in the triangle build by the three points when looking from the top of the robot. If the center of mass crosses the imaginary line between two points, the remaining point that lays on the other side of this line, then the center of mass will no longer support the weight but merely touch the ground without pressure. This leads to a tip over along the crossed axis. With three points, there is a triangle in which the center of mass can lay and change without endangering stability. Unfortunately, the robot is supposed to be well-balanced regarding its center of mass. Therefore, the center of mass lies on, or very near toward, the point of contact of the shell itself. Hence, slight shifts to one of the sides will lead to exceeding the borders of the triangle. With five points, the center of mass cannot exceed any of the outer lines between the four touchpoints of the pole. If the center of mass shifts to the left or right, the diagonal lines are crossed and therefore two points are no longer loaded with weight, which leaves three points and therefore still a balance system. Figure 4.26 shows this overall behavior.

We see that with five points, the green area, which represents the possible area of the center of mass, is much larger than with three points. However, this implies a center of mass near the shell, which is not suggested and technically highly unlikely. Therefore, we focus on the case that the robot has the center of mass at the center of the sphere. This leaves the three-point setup with only one stable roll, 0 rad, as with any other roll, the center of mass lies to the left or right of the original point.

However, the five-point setup leaves a line of possible positions and hence a band of stabilizable $\phi$. With all this taken into account, Algorithm 8 shows an algorithm to stabilize the robot. If the evaluation shows a rattling behavior, the error between the desired roll $\phi$ and the actual $\phi$, $\epsilon_\phi$ needs to be extended by a hysteresis factor. For positive error, lowering it with a

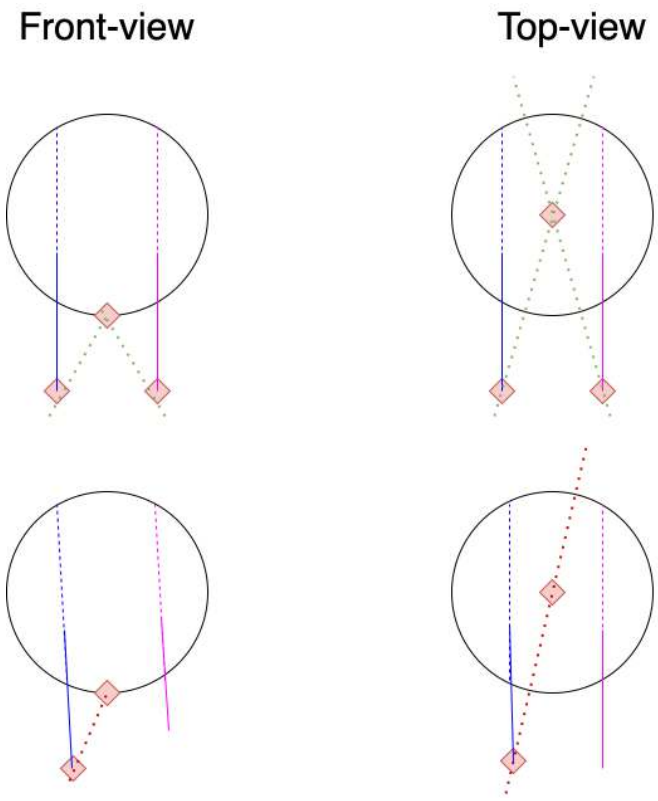



**Figure 4.25:** Visualization of the change of three-supporting points to two-supporting points system if $\phi$ changes slightly. The pink and blue lines represent the two side discs with the corresponding poles. If they are continuous, they represent an extended pole. The red squares show the contact points of the shell and poles with the ground. The green dotted lines indicate that no line fits all points. The red dotted line shows a line that crosses all contact points.

factor if it has been exceeded and increasing it with a factor if it has been undercut, will create a hysteresis. For negative error, it is done the other way round.

As in theory, well-met $l_{\text{balance}}$ do not allow any $\epsilon \neq 0$; we suggest handling every extension of the stabilize algorithm as one without the actual length change of the pole as it is likely to push into the ground. If there is no measurement of the actual length but just an estimation from integrating time, this extension will lead to huge miscalculations. This leads to a retraction after the $\phi$ is back near 0 rad as the theoretical length is far extended due to the last seconds of extension, and the pole will try to reach $l_{\text{balance}}$. This is met in the pseudo-code in general by not retracting but just holding if $\epsilon$ is small but $l > l_{\text{balance}}$. In the later-used real code run on the prototype (shown in Appendix A.4), this is done by the implementation of a state of the pole called "ExtensionFruitless," which extends a pole but prohibits the integration of the length estimation.

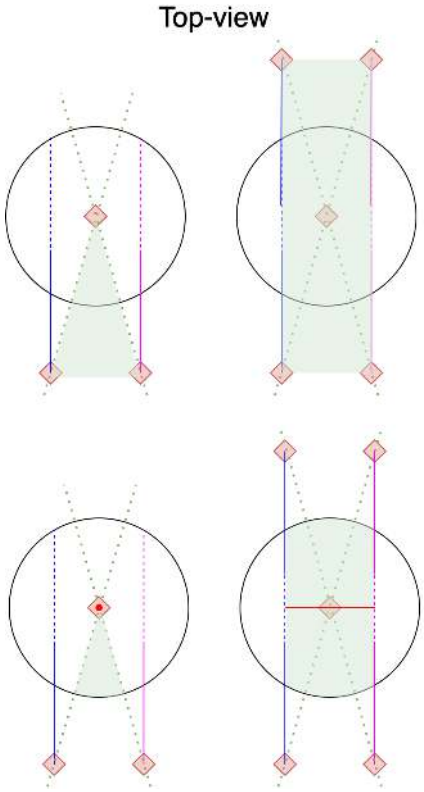



**Figure 4.26:** Visualization of the change of three supporting points to two supporting points system if $\phi$ changes slightly. Above: Theoretically possible area for the center of mass. Below: Logically possible center of mass, as it is within the sphere. The red line/dot shows the stable place(s) for the center of mass if it is located in the middle of the sphere and the sphere is rolled from the mid-position to an askew position. The pink and blue lines represent the two side discs with the corresponding poles. If they are continuous, they represent an extended pole. The red squares show the contact points of the shell and poles with the ground. The green dotted lines shows the imaginary line crossing the contact points of the poles and the one of the shell.

### 4.2.2 Limitations

The overall behavior of mono-speed poles limits the balancing capabilities of the robot, as a three-point controller with hysteresis is not well-suited for complex controlling strategies [10]. However, not only the speed but also the position and arrangement of the pole inside the sphere leads to overall limitations. Therefore, there exists a $\phi_{\text{catastrophic}}$ at and after which no pro-active stabilization using poles is possible. This $\phi_{\text{catastrophic}}$ starts at the angle at which the points touching the ground from poles of one side and the sphere itself build a line. Figure 4.27 visualizes this behavior. Let $r_m$ again be the radius of the sphere, $r_s$ the radius of the disk where the pole is mounted, and $\phi_{\text{catastrophic}}$ the angle at which no pro-active balancing is possible, then

$$\phi_{\text{catastrophic}} = \arccos\left(\frac{r_s}{r_m}\right). \tag{4.125}$$

**Algorithm 8:** Balancing Algorithm

```
foreach pole in poles do
    if β_b ≤ ζ ≤ α_b then
        if ε_φ > threshold then
            extend pole
        else if ε_φ < −threshold then
            retract to r_m − r_s
        else
            if l < l_balance then
                extend pole
            else
                hold pole
            end
        end
    else if 2π − β_b ≤ ζ ≤ 2π − α_b then
        if ε_φ > threshold then
            extend pole
        else if ε_φ < −threshold then
            retract to r_m − r_s
        else
            if l < l_balance then
                extend pole
            else
                hold pole
            end
        end
    else
        retract pole
    end
end
```

If we do not use a spherical shell but rather use an oval one or just disks, as the initial prototype in this thesis, there is a distance $d_m^s$ between the middle and the side discs. For a sphere, this distance is

$$d_{m\text{Sphere}}^s = \tan\left(\arccos\left(\frac{r_s}{r_m}\right)\right) \cdot r_s \,. \tag{4.126}$$

Let $d_m^s$ be the variable distance between the middle and the pole disc, then

$$\phi_{\text{catastrophic}} = 0.5 \cdot \arcsin\left(\frac{d_m^s}{0.25 \cdot r_m}\right). \tag{4.127}$$

Appendix B.2 provides the derivation and the visualization. Equation (4.127) does not depend on $r_s$. This is due to the variable length of the pole. Therefore, a smaller $r_s$ has the same $\phi_{\text{catastrophic}}$ as a larger one, as the extending poles will touch the ground on the same line. All these calculations assume an evenly distributed mass as well as no bending of the poles. Bending

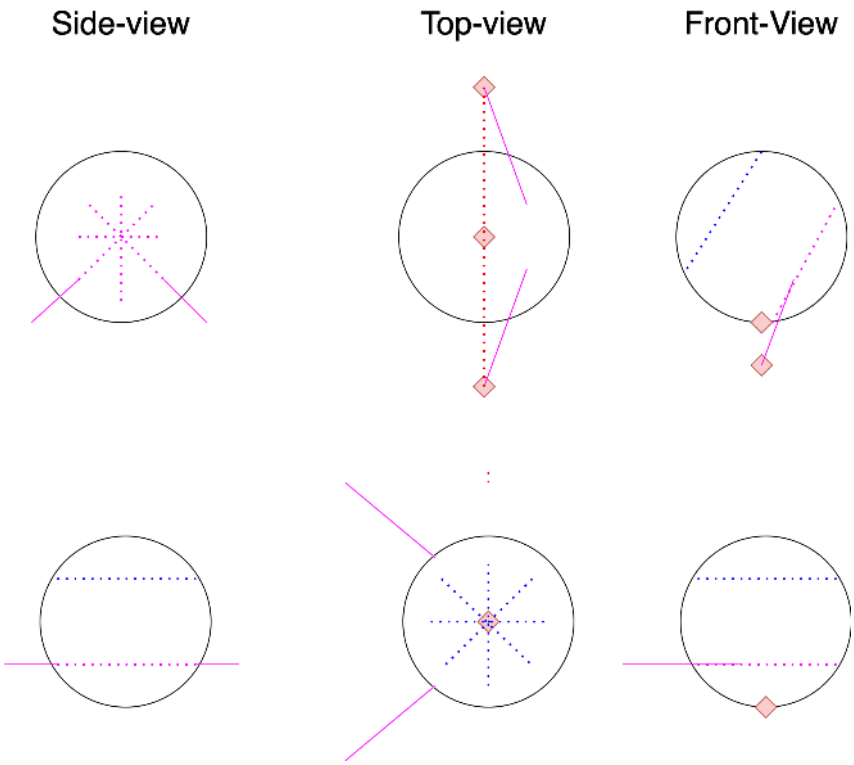



**Figure 4.27:** Above: Visualisation of the $\phi_{\text{catastrophic}}$, where the touchpoints of the poles on one side build a line with the touchpoint of the sphere. Below: Further escalation to the point that the pole does not even touch the ground. The pink and blue lines represent the two side discs with the corresponding poles. If they are continuous, they represent an extended pole. The red squares show the contact points of the shell and poles with the ground. The red dotted line shows a line that crosses all contact points.

influences the $\phi_{\text{catastrophic}}$ tremendously. Figure 4.28 shows the influence of bending. To avoid the whole problem, a hardware-based solution is to add a side pole to the sides of the robot, perpendicular to the other poles. This pole is supposed to push the robot from its side back to at least the $\phi_{\text{catastrophic}}$. This is possible with a short extension but with enough force and speed to generate momentum or with a long enough pole. With a too short pole, even with a fast and powerful extension, there might be an angle at which the main poles have no chance to stabilize the robot, while the side rod does not reach the ground. The length of the side pole of a general robot is given by

$$l_{\text{side}} = \frac{0.5 \cdot r_m}{\tan\left(\phi_{\text{catastrophic}}\right)} - d_m^s\,, \tag{4.128}$$

so, with Equation (4.127)

$$l_{\text{side}} = \frac{0.5 \cdot r_m}{\tan\left(0.5 \cdot \arcsin\left(\frac{d_m^s}{0.25 \cdot r_m}\right)\right)} - d_m^s\,. \tag{4.129}$$

For the sphere, we assume space between the side disc and the shell, which we not include in the extension length of the pole.

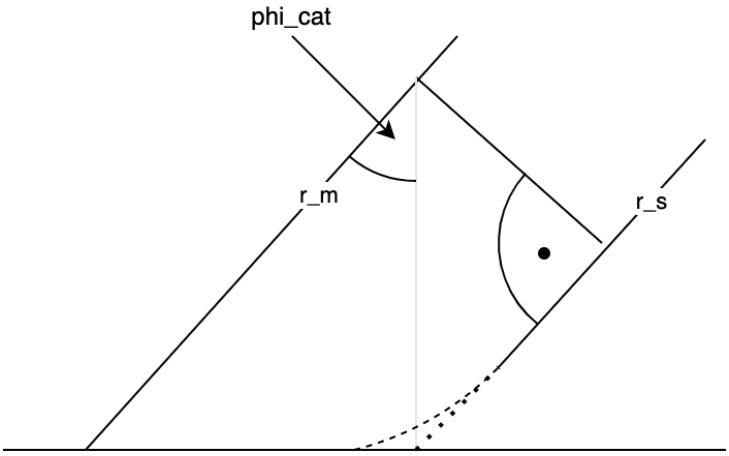



**Figure 4.28:** Influence of pole bending on the $\phi_{\text{catastrophic}}$. The dotted line is the pole without bending, reaching exactly the grey line, and is therefore on the edge of $\phi_{\text{catastrophic}}$. The dashed line includes the bending of the pole that is not able to stabilize the robot.

Let $l_{\text{sideSphere}}$ be the minimum possible extension needed for a side pole of a spherical robot to turn the sphere to $\phi_{\text{catastrophic}}$, then

$$l_{\text{sideSphere}} = \frac{0.5 \cdot r_m}{\sin\left(\phi_{\text{catastrophic}}\right)} - 0.5 \cdot r_m \,. \tag{4.130}$$

Inserting Equation (4.125), this becomes

$$l_{\text{sideSphere}} = \frac{0.5 \cdot r_m}{\sin\left(\arccos\left(\frac{r_s}{r_m}\right)\right)} - 0.5 \cdot r_m = \frac{0.5 r_m}{\sqrt{1 - \left(\frac{r_s}{r_m}\right)^2}} - 0.5 \cdot r_m \,. \tag{4.131}$$

Appendix B.3 contains the illustration of the variable for the derivation. For our implementation, the $\phi_{\text{catastrophic}}$ does not have a direct influence on the code or evaluation as we neither have side poles to compensate for this, nor want to initiate any other $\phi$ than 0 rad. If the spherical robot needs to have some kind of askew mode, this becomes more relevant as it limits the maximum possible $\phi$. Also, with existing side poles, any $\phi > \phi_{\text{catastrophic}}$ is a non-negotiable indication for their use as the only chance for pro-active re-balancing. Without such an explicit askew mode and without a side pole, we only use it to detect irreversible tip-overs.

### 4.2.3 Comparison of disc- and sphere-setup

We switched between the usage as disc-robot, cylinder, and sphere for locomotion, as it does not differ regarding its pitch behavior. For balancing, we get different behaviors with these shapes. The behavior of the stable areas for the center of mass is the same, and the calculation of the $\phi_{\text{catastrophic}}$ was done for both, as they got direct different values of angles. However, for $\dot{\phi}$, there are also differences. This does not lead to any angles at which a disc robot is not stable, but a sphere is. Still, taking the extending speed of the actuators into account, there are cases wherein

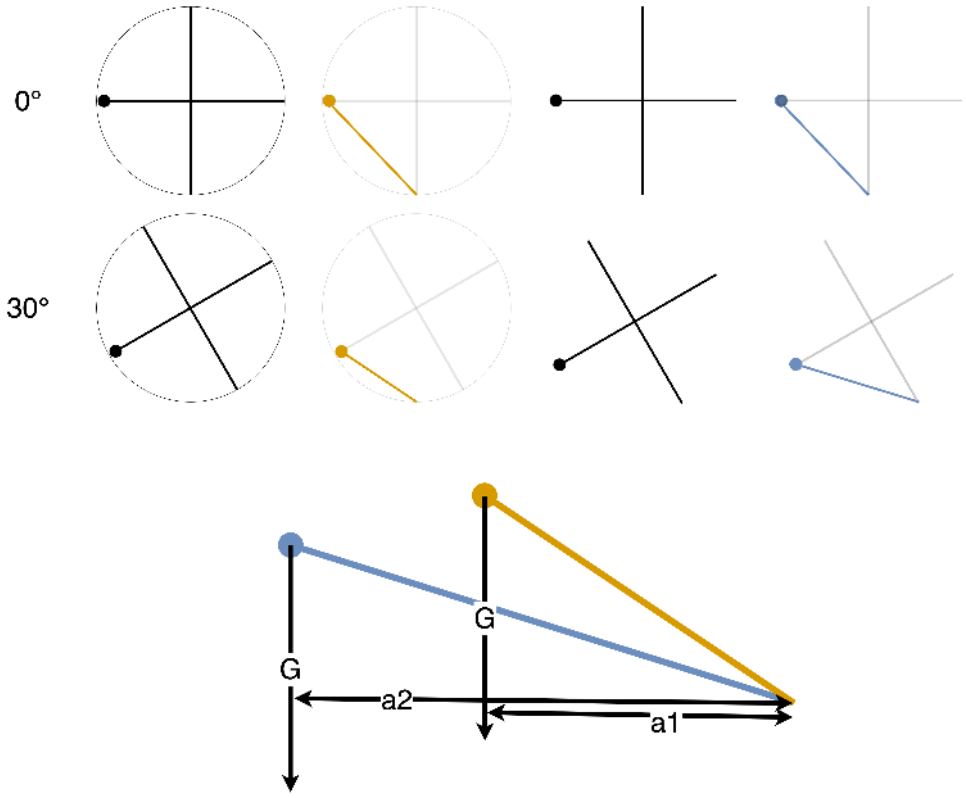



**Figure 4.29:** Reducing a massless sphere and massless single disc prototype with one point mass on the left side to a tick with a point mass at 0 deg and 30 deg. The yellow parts represent the reduction of the sphere, the blue of the disc setup. G indicates the gravitational vector on the structures.

the velocity and acceleration of $\phi$ determine if the setup is balanceable. The hypothesis is that with a sphere, $\dot{\phi}$ and $\ddot{\phi}$ are generally slower than with a disc robot.

We reduce the sphere robot to a massless sphere with one point mass on one side for this evaluation. We also reduce the disc robot to a massless structure with one point mass. The structure contains a middle disc of the size of the sphere. For simplification, we assume infinite small side discs, which are not relevant as we focus on the passive behavior without any pro-active influence. To evaluate the momentum the point mass generates, which will lead to a change of $\phi$, we reduce both robots at every single moment to a massless stick with the weight on it. This is valid in terms of momentum because it is defined by $M = F \cdot a$, where $a$ is the position vector perpendicular to the force vector, and $F$ is the applied force. This means that the only relevant points are the position of the point mass, the position of the point touching the ground, and the applied force [53]. Reducing the robots in the described way changes none of this. However, doing so reveals that they reduce to the same structure at a vertical start, but once $\phi$ changes, they differ, as the sphere continuously changes the point on the sphere that touches the ground. As we roll to the side of the point mass, except at the start, $a$ is always smaller than with the reduced disc structure, and the angle will always be smaller. Figure 4.29 shows this circumstance. The same applied force with a smaller angle (steeper) and shorter $a$

distance will lead to less torque. This again results in smaller $\ddot{\phi}$ and therefore smaller $\dot{\phi}$, which may allow the slow and weak actuators to interfere at a certain $\phi$, at which they may not be able to with a disc setup. The conclusion here is that if a setup is balanceable as a disc setup, the same setup with a spherical shell would also be balanceable. The same argumentation and behavior apply for spheroids, as the touchpoint also migrates nearer to the point mass if it tips. This holds true for the exact same setup (same weight, same actuators, size, etc.). For example, this does not hold true for our prototype as a disc prototype is placed inside a spherical shell. The weight of the shell is not included in the disc setup. Therefore, the moment of inertia is bigger as an overall larger mass is given. In the extreme case, a disc structure with minimal weight is placed inside a shell that has an infinite weight, not evenly distributed, which will not be stoppable by actuators with finite force, showing that the requirement of the same weight conditions of both structures is essential. The friction of the sphere with the ground does have an influence, and further research needs to address it for a precise analytical evaluation. For the dimensions and materials of our prototype, it becomes neglectable [28].

The last point regarding the sphere versus disc robot issue is rather obvious, but we need to mention it for the sake of completeness. The previous evaluation used a point mass at the outer point of the robot, which leads to no configuration where there is no momentum, so there is no zero moment point (ZMP) near the starting configuration. Of course, this is highly unlikely, and both robot types are very likely to be built very symmetrically regarding weight. This results in the analysis of changes of $\phi$, which are introduced by external force and not an internal imbalance of the weight. As the nature of the sphere, if the weight is right at the center of the sphere, an external change of $\phi$ will not cause further escalation as with the changed angle the center of mass is still in the middle and does not initiate any resulting force. This means there is a ZMP, and the stable region for it is infinitely large. When rotating a sphere, the point nearest to the ground (automatically the touchpoint) will always lay in the middle. Therefore, there is an infinite stable area. For a disc setup, this however does not hold true. With a large middle disc, changing $\phi$, the point nearest to the ground and touching it is still the same. However, it does not lie under the center of mass anymore. Therefore, the gravitational force attacking at the center of mass is not parallel to the axis of the touchpoint and center of mass. As a result, the structure no longer compensates the whole gravitational force, which is now directed to the side and leads to acceleration. Figure 4.30 shows this in the same manner as in Figure 4.29 but with centered mass. The real prototypes are likely to have an imbalance in them, and therefore the resulting dynamics will be somewhere between both evaluations. However, the stable area of the ZMP for the disc setup will always be smaller than the one for the sphere [74].

## 4.3 Combination of Balancing and Movement

### 4.3.1 Individual approach

After evaluating balancing and locomotion separately, we want to discuss the combination of both. There is a clear priority inclination between both parts, as balancing is crucial for the overall mission concerning the possibility of full mission failure if we exceed certain angles (calculated as $\phi_{\text{catastrophic}}$). Therefore, the primary objective needs to be controlling $\phi$, and the secondary should be to control $\omega$. The straightforward approach is either a balance or locomo-

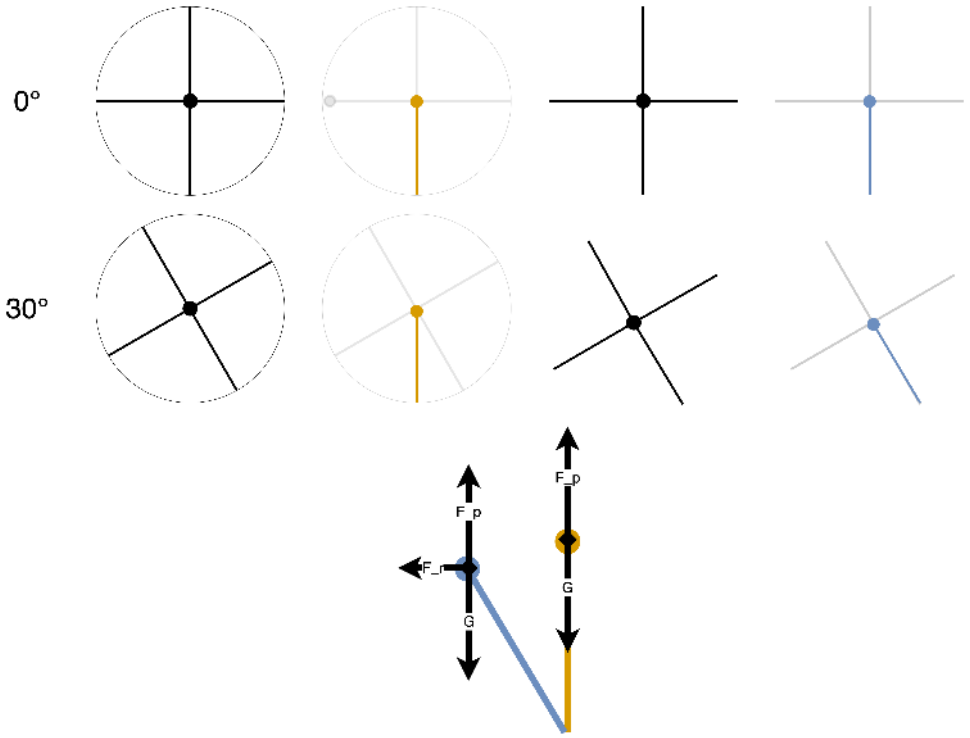



**Figure 4.30:** Reducing a massless sphere and massless single disc prototype with one point mass in the middle to a tick with a point mass at 0 deg and 30 deg.The yellow parts represent the reduction of the sphere, the blue of the disc setup. $G$ is the gravitational force, $F_p$ is the force by the stick, and $F_r$ is a non-compensated force to the side.

tion approach. We define an accepted error of phi $\epsilon_\phi$ and start the locomotion until this error is exceeded, at which moment the balancing algorithm starts. Also, this might sound ineffective, but we need to keep in mind that a mass-balanced spherical robot is a stable system. For a disc-based robot, this will not work as the balancing is likely to occur all the time due to the instability of the system. Further, this is highly ineffective as there exists the possibility of balancing while moving. However, for this kind of stable sphere robot, we can take Algorithm 8, which balances the robot, and adopt it. We extend the two cases that have no direct influence on the balancing and allow an overwriting by the locomotion algorithm (which can be any of the algorithms described in Section 4.1; we amploy Algorithm 5 as it has leverage and pushing, and Algorithm 7 for braking). In addition, we limit the use of the rods on the side facing the desired locomotion direction to the standstill case and the emergency case. The emergency case refers to an uncontrolled tip-over. We determine if this tip-over occurs or if it is just the commanded rotation with a faster $\omega$ than the intended $c_\omega$. This together leads to Algorithm 9 for mono-speed poles.

**Algorithm 9:** Balancing and locomotion for mono-speed algorithm

```
foreach pole in poles do
    if β_b ≤ ζ ≤ α_b then
        if ε_φ > threshold then
            extend pole
        else if ε_φ < −threshold then
            retract to r_m − r_s
        else
            if l < l_balance then
                extend pole
            else
                if ω > 0 then
                    use Algorithm 5
                else
                    use Algorithm 7
                end
            end
        end
    else if 2π − β_b ≤ ζ ≤ 2π − α_b and 0 < c_ω ≤ ω then
        if ε_φ > threshold then
            extend pole
        else if ε_φ < −threshold then
            retract to r_m − r_s
        else
            if l < l_balance then
                extend pole
            else
                hold pole
            end
        end
    else
        if ω > 0 then
            use Algorithm 5
        else
            use Algorithm 7
        end
    end
end
```

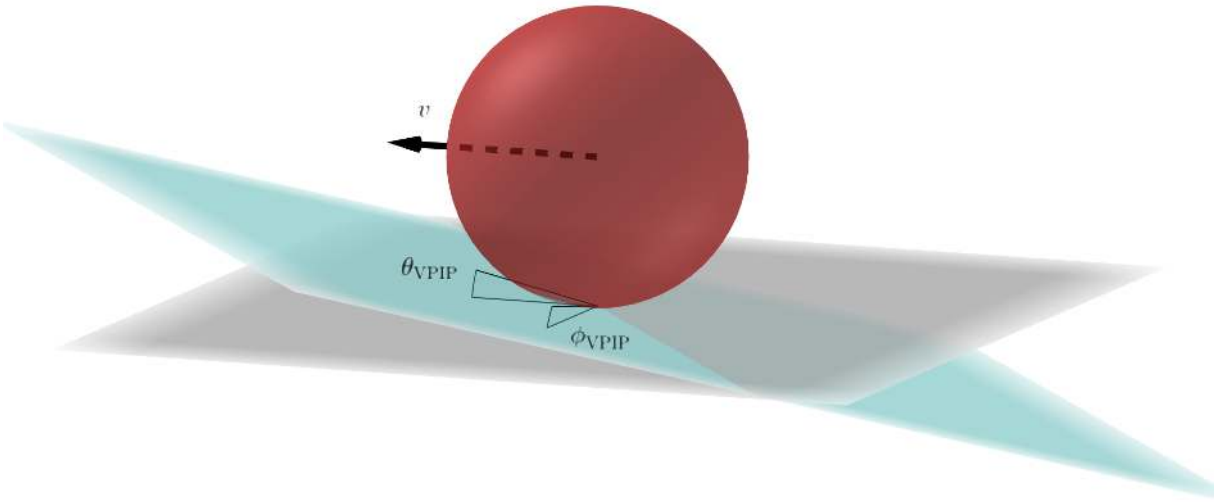



**Figure 4.31:** Visualization of VPIP (blue) and the describing angles $\phi_{\text{VPIP}}$ and $\theta_{\text{VPIP}}$. The robot (red sphere) rolls into the $v$ direction. The grey plane shows the actual ground.

### 4.3.2 Virtual Pose Instruction Plane (VPIP)

For poles with variable speeds, we can derive a more detailed algorithm. Rather than combining the separately evaluated instructions of locomotion and balancing, we want to evaluate the state of the poles based on the need for both of them. We therefore control the poles by absolute length. This brings us from the evaluation in 2D, as was done separately for balance and locomotion, to an evaluation in three dimensions. Also, we will leave out the field of evaluations where it does not matter if it was a spheroid or sphere robot as this evaluation is specific to spherical robots. If the poles are only speed-controllable, this can be done by the derivation of the length. We, therefore, introduce the Virtual Pose Instruction Plane (VPIP). The VPIP is a plane that determines the lengths of poles. The poles extend and retract to the length that they lay theoretically on the VPIP. The VPIP is fixed to the robot at the same point where the robot touches the ground. The plane itself has no shape and hence no relevant rotation about the axis perpendicular to the real ground. Therefore, only two angles describe this plane: $\phi_{\text{VPIP}}$ and $\theta_{\text{VPIP}}$. They have the same direction as the internal $\phi$ and $\theta$ of the robot. Figure 4.31 illustrates VPIP. We define the rolling direction as the $y$-axis, the $z$-axis perpendicular to the flat ground passing the midpoint of the robot, and the $x$-axis to complete the right-hand system. Let the contact point of the robot $\boldsymbol{r}_0$ to the ground be always at $(0, 0, 0)$, and $\boldsymbol{n}$ be the normal vector on the plane, then, the VPIP $\Pi_{\text{VPIP}}$ is defined by

$$\Pi_{\text{VPIP}} = \boldsymbol{n} \cdot (\boldsymbol{r} - \boldsymbol{r}_0) = \begin{bmatrix} \tan(\phi_{\text{VPIP}}) \\ \tan(\theta_{\text{VPIP}}) \\ 1 \end{bmatrix} \cdot \frac{1}{\sqrt{\tan(\phi_{\text{VPIP}})^2 + \tan(\theta_{\text{VPIP}})^2 + 1}} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0$$

$$\leftrightarrow \tan(\phi_{\text{VPIP}}) \cdot x + \tan(\theta_{\text{VPIP}}) \cdot y + z = 0\,. \tag{4.132}$$

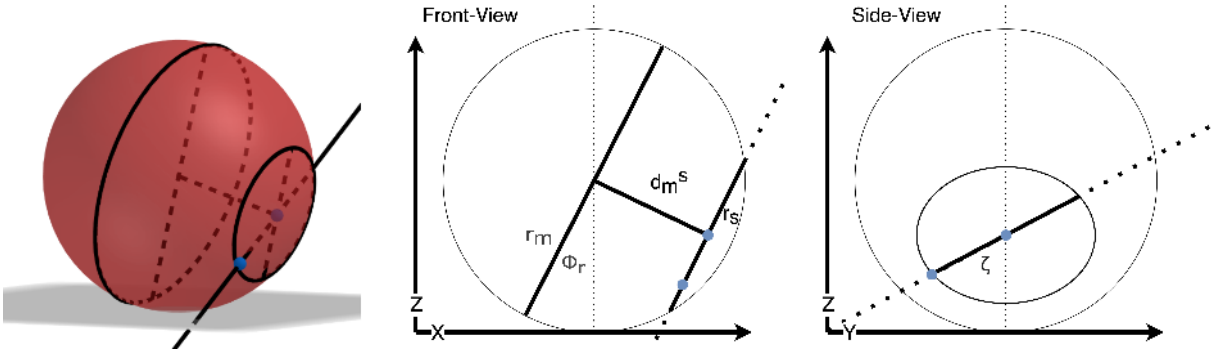



**Figure 4.32:** Illustration of pole line for VPIP evaluation. The blue points are used to calculate the line.

The idea is to not change the pole length to touch the ground but the VPIP. Therefore, if we tilt the VPIP, as shown in Figure 4.31, the poles on the left front side (seen into velocity direction) retract, on the rear right side extend. For the other directions (front right, rear left), we can not state general actions for every pole, as it depends on their $\zeta$.

To find the needed length of a pole to touch the VPIP, we need to find the equation of a line of a pole in this representation. Therefore, we define the point $\boldsymbol{p}_m$ in the middle of the side disc and the point $\boldsymbol{p}_t$ as the tip of the fully retracted pole. Figure 4.32 shows the two points in the three-dimensional view as well as the two two-dimensional views for easier calculation. We derive from this figure that

$$\boldsymbol{p}_m = \begin{bmatrix} \cos(\phi_r) \cdot d_m^s \\ 0 \\ r_m - \sin(\phi_r) \cdot d_m^s \end{bmatrix}, \tag{4.133}$$

and with that, we can derive $\boldsymbol{p}_t$ as a combination of the point $\boldsymbol{p}_m$ and the vector from $\boldsymbol{p}_m$ to $\boldsymbol{p}_t$. Let $\boldsymbol{d}_{mt}$ be the vector between $\boldsymbol{p}_m$ to $\boldsymbol{p}_t$, then

$$\boldsymbol{p}_t = \boldsymbol{p}_m + \boldsymbol{d}_{mt} = \boldsymbol{p}_m + \begin{bmatrix} -\cos(\zeta) \cdot r_s \cdot \sin(\phi_r) \\ -\sin(\zeta) \cdot r_s \\ -\cos(\zeta) \cdot r_s \cdot \cos(\phi_r) \end{bmatrix}. \tag{4.134}$$

A more detailed derivation of this is attached in the Appendix B.4. $d_m^s$ is not variable in the case of a sphere but defined by

$$d_m^s = \sin\left(\arccos\left(\frac{r_s}{r_m}\right)\right) \cdot r_m \,. \tag{4.135}$$

for both sides of the pole discs, there needs to be an adaption. We do this by mirrororing $\boldsymbol{p}_m$ at the main disc plane. Therefore, we define $\boldsymbol{p}_{ml}$ as the $\boldsymbol{p}_m$ for the middle of the left disc and $\boldsymbol{p}_{mr}$ of the right (seen in the rolling direction). This leads for $\boldsymbol{p}_{ml}$ to

$$\boldsymbol{p}_{ml} = \begin{bmatrix} -\cos(\phi_r) \cdot d_m^s \\ 0 \\ r_m - \cdot \sin(\phi_r) \cdot d_m^s \end{bmatrix}, \tag{4.136}$$

To work out an overall solution, we give the left side the number $-1$ and the right side 1. We will refer to this as side factor $s_f$ and write the general $\boldsymbol{p}_m$ as

$$\boldsymbol{p}_m = \begin{bmatrix} s_f \cos(\phi_r) \cdot d_m^s \\ 0 \\ r_m - s_f \cdot \sin(\phi_r) \cdot d_m^s \end{bmatrix}, \tag{4.137}$$

Now, we calculate the line of a specific pole, which goes through the point $\boldsymbol{p}_m$ with the direction $\boldsymbol{d}_{ms}$. Let $\zeta$ be the angle of the pole, $\phi_r$ the actual roll angle of the robot, $r_s$ the radius of the disc inside the spherical robot wherein the poles lie, $r_m$ the radius of the sphere, $d_m^s$ the distance between the midpoint and the side disc of the poles (which is non-variable and set due to the fixed $r_m$ and $r_s$, which will be specified later), and $s_f$ the side factor; then, the line of this pole $L_p(\zeta, \phi_r, s_f)$ is defined by

$$\begin{aligned} L_p(\zeta, \phi_r, s_f) &= \left\{ (x,y,z) | \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} p_m^x \\ p_m^y \\ p_m^z \end{bmatrix} + \lambda \begin{bmatrix} d_{mt}^x \\ d_{mt}^y \\ d_{mt}^z \end{bmatrix} | \lambda \in \mathbb{R} \right\} \\ L_p(\zeta, \phi_r, s_f) &= \left\{ (x,y,z) | \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} s_f \cdot \cos(\phi_r) \cdot d_m^s \\ 0 \\ r_m - s_f \cdot \sin(\phi_r) \cdot d_m^s \end{bmatrix} \right. \\ &\quad \left. + \lambda \begin{bmatrix} -\cos(\zeta) \cdot r_s \cdot \sin(\phi_r) \\ -\sin(\zeta) \cdot r_s \\ -\cos(\zeta) \cdot r_s \cdot \cos(\phi_r) \end{bmatrix} | \lambda \in \mathbb{R} \right\}. \end{aligned} \tag{4.138}$$

We use $\lambda$ as an indication of the extended length as we have the midpoint of the pole disc as a fixed point for the line and then $\lambda$ times the gradient of the pole. So, if we normalize the vector of the gradient, $\lambda$ represents the actual length of the pole but including $r_m$. Therefore, we write $\lambda$ as $\frac{r_s + l}{r_s + l_{\max}}$. This leads to a mathematical possible negative $l$ (if the original $\lambda$ is 0) and $l$ larger than $l_{\max}$. In these cases, we need to consider the practical limitations of the pole. The negative extension means full retraction only; larger extensions than possible are just full extensions. For this representation of $\lambda$, the vector that it scales needs to be normalized. This

leads to

$$L_p(\zeta,\phi_r,s_f)=\left\{(x,y,z)|\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}p_m^x\\p_m^y\\p_m^z\end{bmatrix}+\lambda\frac{\boldsymbol{d}_{mt}}{||\boldsymbol{d}_{mt}||}|\lambda\in\mathbb{R}\right\}$$

$$L_p(\zeta,\phi_r,s_f)=\left\{(x,y,z)|\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}s_f\cdot\cos(\phi_r)\cdot d_m^s\\0\\r_m-s_f\cdot\sin(\phi_r)\cdot d_m^s\end{bmatrix}\right.$$
$$+\frac{r_m+l}{r_m+l_{\max}}\cdot\begin{bmatrix}s_f\cdot\cos(\phi_r)\cdot d_m^s-\cos(\zeta)\cdot r_s\cdot\sin(\phi_r)\\-\sin(\zeta)\cdot r_s\\r_m-s_f\cdot\sin(\phi_r)\cdot d_m^s-\cos(\zeta)\cdot r_s\cdot\cos(\phi_r)\end{bmatrix}$$
$$\left.\cdot\left|\left|\begin{bmatrix}s_f\cdot\cos(\phi_r)\cdot d_m^s-\cos(\zeta)\cdot r_s\cdot\sin(\phi_r)\\-\sin(\zeta)\cdot r_s\\r_m-s_f\cdot\sin(\phi_r)\cdot d_m^s-\cos(\zeta)\cdot r_s\cdot\cos(\phi_r)\end{bmatrix}\right|\right|^{-1}|l\in\mathbb{R}\right\}. \tag{4.139}$$

Note that we set $l\in\mathbb{R}$ despite the physical limitation is from $r_m$ to $r_m+l_{\max}$, as this does not imply if the pole needs to be fully extended or retracted in cases of no solution for the intersection with the VPIP.

Now, we evaluate the intersection of $L_p(\zeta,\phi_r,s_f)$ and $\Pi_{\text{VPIP}}$. This is the point where the poles need to extend to. Therefore, we set the components of Equation (4.138) into Equation (4.132) and get

$$\Pi_{\text{VPIP}}=L_p(\zeta,\phi_r,s_f) \tag{4.140}$$
$$\tan(\phi_{\text{VPIP}})\cdot L_p(\zeta,\phi_r,s_f)_x+\tan(\theta_{\text{VPIP}})\cdot L_p(\zeta,\phi_r,s_f)_y+L_p(\zeta,\phi_r,s_f)_z=0$$
$$l=\frac{(-\tan(\phi_{\text{VPIP}})\cdot s_f\cdot\cos(\phi_r)\cdot d_m^s-r_m+s_f\cdot\sin(\phi_r)\cdot d_m^s)\cdot||d_{mt}||\cdot r_s+l_{\max}}{\tan(\phi_{\text{VPIP}})\cdot d_{mtx}+\tan(\theta_{\text{VPIP}})\cdot d_{mty}+d_{mtz}}-r_s\,. \tag{4.141}$$

With $l$, one could find the corresponding $(x,y,z)$ of the intersection by putting $l$ into Equation (4.139). However, for our evaluation, $l$ is the wanted variable for controlling the poles to lie on the VPIP; therefore, we refer to this as $l_{\text{VPIP}}$. Appendix (B.1) contains the fully inserted equation. This step needs to be done for every step for all poles. For most cases, we reduce the amount by just ignoring all poles between $0.5\pi$ rad and $1.5\pi$ rad, as it needs particular, extreme cases for them to interact. The overall control mechanism always controls the poles that their $l$ matches $l_{\text{VPIP}}$ and controls $\theta_{\text{VPIP}}$ and $\phi_{\text{VPIP}}$. For the angles of the VPIP, we will not derive a full control system as this exceeds the scope of this thesis and will be addressed in further research, but we give a short overview of the expected behavior. $\phi_{\text{VPIP}}$ needs to compensate the $\epsilon_\phi$ and therefore is in the same direction as the roll of the robot $\phi$. So, if the robot falls to the right, the VPIP will lower the right and rise to the left side. This leads to an extension of the poles on the right side and a lowering of the pole on the left, which is the right action for countering the tip to the right. The idea is to control in a closed-loop manner the $\phi_{\text{VPIP}}$ on the basis of $\epsilon_\phi$. In the same manner, $\theta_{\text{VPIP}}$ is controlled by $c_\omega$. Lowering the VPIP on the back and rising it on

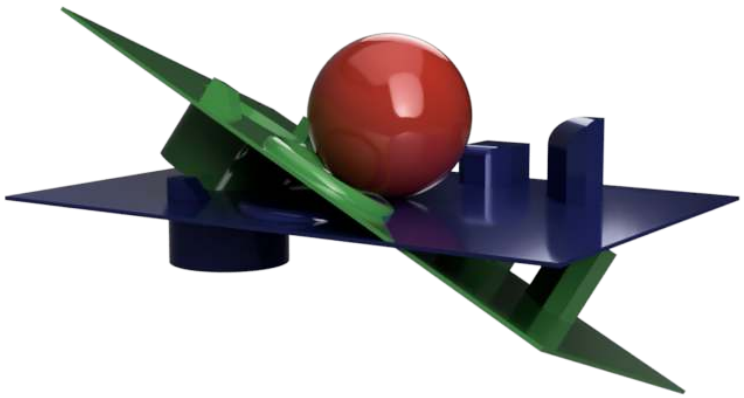


**Figure 4.33:** Virtual Pose Instruction Map (VPIM) visualization. Green: VPIM. Blue: original map. Red: robot.

the front of the robot will lead to retraction of poles on the front and extension of poles on the back, leading to locomotion to the front. As the $\zeta$ change during this process, $l_{\mathrm{VPIP}}$ changes for the specific poles, with the actual VPIP staying the same. Therefore, a $\theta_{\mathrm{VPIP}} \neq 0\,\mathrm{rad}$ leads to a continuous motion. With increasing $c_\omega$, $\theta_{\mathrm{VPIP}}$ can be chosen steeper. This is also done using a closed-loop controller. As the later-introduced prototype has mono-speed poles, we cannot evaluate the results of VPIP.

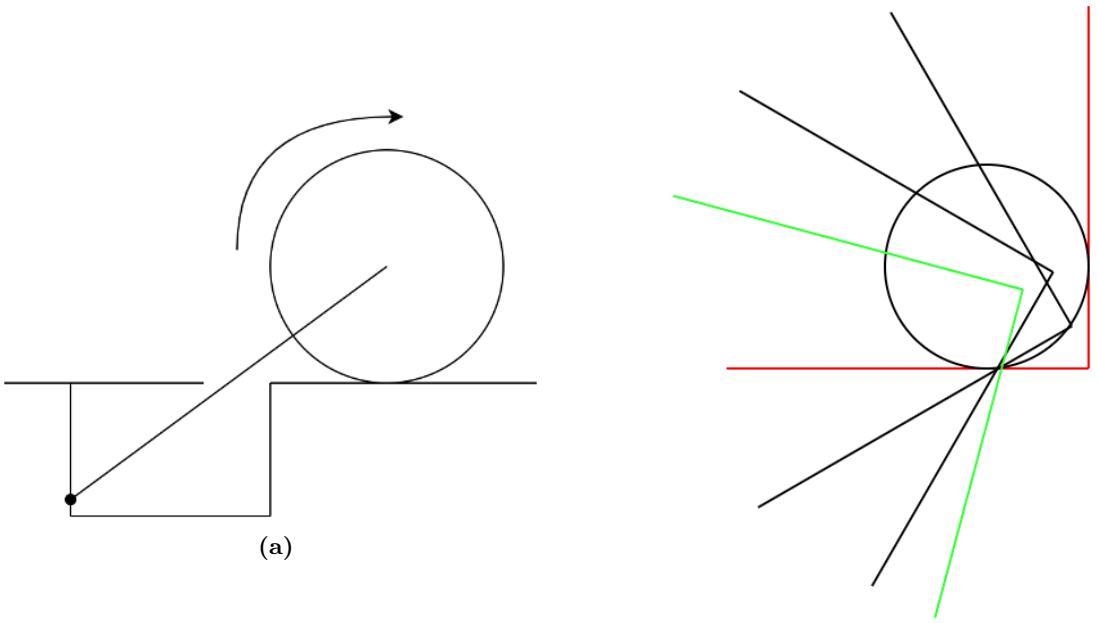
### 4.3.3 Outlook on Virtual Pose Instruction Map (VPIM)

We want to look into possible further steps and problems regarding VPIP. First, we want to take the environment into account and therefore widen the application to not only a plane but a whole map of the environment structure. We introduce the Virtual Pose Instruction Map (VPIM), where the plane is replaced by the actual map, which is again changed in tilt and pitch, $\phi_{\mathrm{VPIM}}$ and $\theta_{\mathrm{VPIM}}$. The overall behavior stays the same as with VPIP. Therefore, we calculate the intersection of the map and the poles, like in Equation (4.140), but with the Map $M_{\mathrm{VPIM}}$ instead of the plane $\Pi_{\mathrm{VPIP}}$. Figure 4.33 shows the visualization of VPIM. This raises the question of the origin of the map. Of course, it can be a predefined map, but there needs to be a sensor for a localization algorithm. Also, a map generation in situ, as Chapter 2 introduces it as idea of the DAEDALUS sphere, is possible. Here, the robot houses at least one LiDAR scanner (light detection and ranging). The scans are done by dedicated phases of scanning or continuous scanning during locomotion. With the generated point cloud, the map of the immediate environment of the robot can be calculated. This procedure has already been investigated and optimized [83] [11] [59]. This map is then rotated by a rotation matrix generated based on $\phi_{\mathrm{VPIM}}$ and $\theta_{\mathrm{VPIM}}$. Several open points need to be investigated. We list some

open points and problems without claiming completeness:

- Computational power: There are many point cloud-to-map optimization algorithms, as well as simultaneous localization and mapping (SLAM) algorithms for localization in a given map. These also work on embedded systems. However, generating maps is often not considered on the robot side and, most often, not in near real time. Using VPIM, this becomes necessary to work in nearly real time, as for locomotion, the map needs to be generated for direct usage.
- Foresighted actioning: There are several situations wherein an extension of a pole may be possible and indicated, but further rotation and movement lead to entrapment. Figure 4.34a shows this schematically. Here, with a slow $\dot{l}_{\max}$, the pole will get stuck once rotation proceeds, despite being valid in the instantaneous evaluation.
- Anti-windup and extreme scenarios: Figure 4.34b depicts the case of a perpendicular object in front of the robot. It takes nearly a $\theta_{\text{VPIM}}$ of $0.5\pi$ rad for the right poles to extend. As we assume the changes are non-rapid (especially $\theta_{\text{VPIM}}$), this transition will take a long time. During this time, the sphere pushes itself onto the obstacle until finally lifting off. This transition may be harmful to the robot; therefore, there is the need for adaptions in extreme cases like perpendicular objects. This targets the anti-windup mechanism, as we do not want to change $\theta_{\text{VPIM}}$ too quickly, since the whole system may not be able to move faster or the transition takes longer. Still, we also do not want to push for an extended period against the obstacle until finally starting to push. Also, there is the problem that a $\theta_{\text{VPIM}}$ of $0.5\pi$ rad in the shown case will only lead to an intersection of a pole at $\zeta = 0$ rad, but as the tip lies directly on the VPIM, it will not extend. Therefore, the procedure of vertical acceleration is not given without modifications of VPIM.
- Linear shift: The problem with the perpendicular obstacle shows the problem with all intended linear, non-rotational transitions of the robot with the VPIM approach, as it is designed for just the rotation movement. One solution is to add linear shifts to the VPIM. For the obstacle example given in Figure 4.34b, a shift of the whole map downwards leads to an extension of the rods at the bottom of the sphere, resulting in the intended climbing of the perpendicular obstacle.
- Scope: The mandatory characteristics for a robot to be controllable by the VPIM as well as VPIP need to be defined to know for which kind of robots they are suitable for.
- Map requirement: The required quality of the map in order to get a certain quality of controlling needs to be defined. This brings information about the possible usable sensors and algorithms for map creation and optimization.

Figure 4.35 shows the overall cycle. This process of acquiring a point cloud (which also includes the optimization), generating the map out of the point cloud, the rotation of the map, and the calculation of the intersection of each pole with the VPIM, need to be optimized to the speed of the overall control frequency of the rods. Each step has a lot of potential for optimization and needs to be adapted to the specific needs of the VPIM control approach. This



**(a)** **(b)**

**Figure 4.34:** Illustration of two open points of VPIM. Left: Foresighted actioning. As the robot rolls to the right side, the pole will get stuck in the pit if the retraction speed is not fast enough. Right: Extreme scenario of perpendicular obstacle. The red line shows the start-VPIM, which is tilted further until it is the green line. Only when reaching the green setup, VPIM will lead to successful actions of the poles.

also gives the opportunity for a symbiosis of payload data with the robot itself, like mentioned in the introduction in Section 1.1. The point clouds and maps produced for the scientific return of a mission can be used for controlling, even if it is likely to be in a reduced or simplified version. This symbiosis can also work in the other direction. Due to this combination, there is not one LiDAR each for scientific return and the robot, but two same sensors, leading to redundancy for both applications.

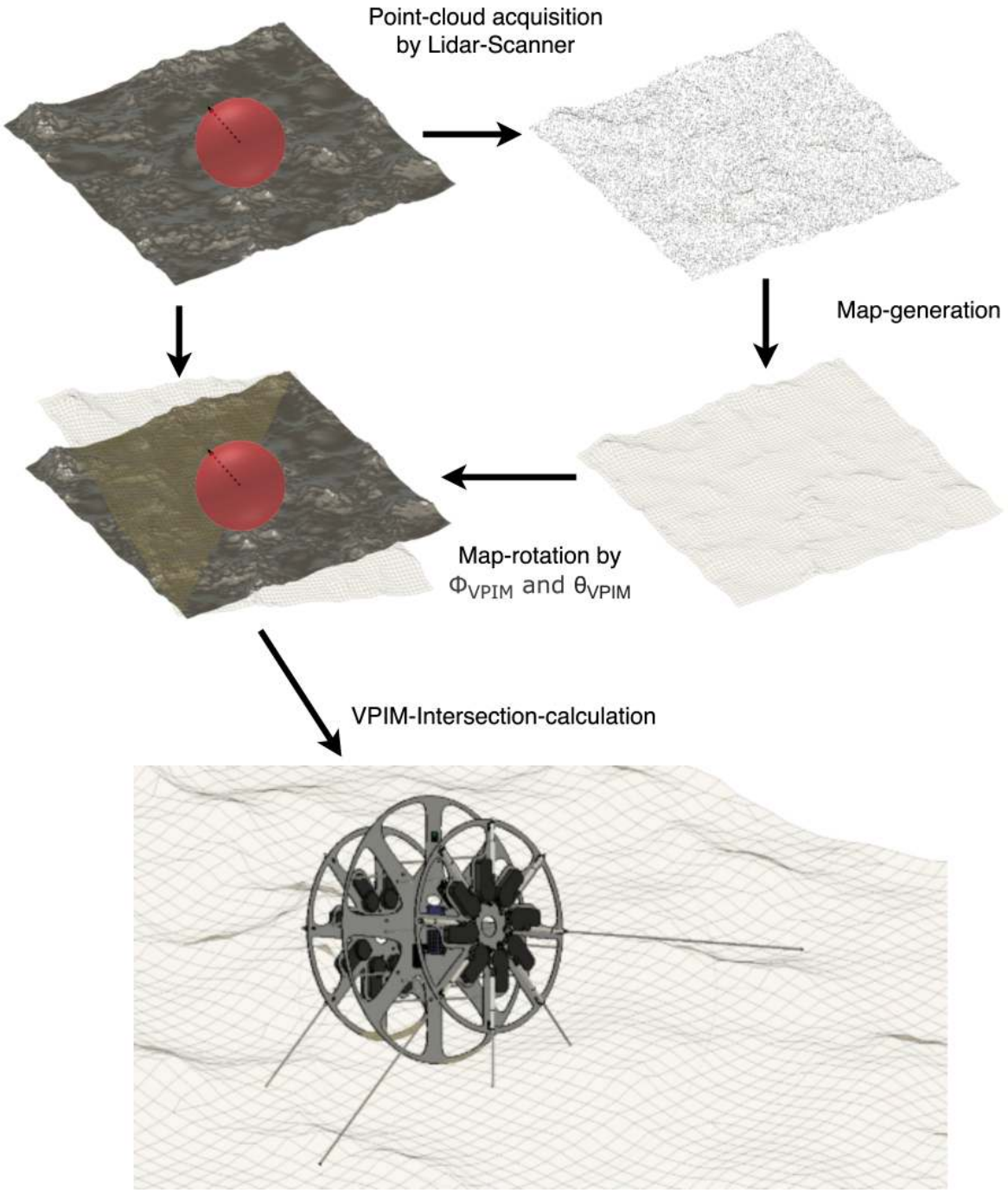



**Figure 4.35:** Cycle of the calculation of VPIM.

# Chapter 5

# Prototype

## 5.1 System

### 5.1.1 Number of Rods and Extensions

One point to be considered for a TLDR robot is the number of rods and the extensions used per rod. This is a pure practical issue, as all formulas and evaluations in the previous chapters just assumed a certain pole length, which is larger than the radius of the side disc, but it is not how this is practically achieved. The number of extensions and number of used rods need to be analyzed in combination as they are highly dependent on each other. For this evaluation, the rods are numbered and referred to as shown in Figure 5.1. The first one is always the one with the smallest $\zeta$ and then increase the pole-number counterclockwise. This definition certainly does not hold for rotating the overall structure as the same rod would change its number, but it is sufficient for this evaluation and the calculations.

The number of rods is mostly defined by the capability of the configuration for locomotion. For forward locomotion using pushing, rod two must rotate the sphere right to the point where the extension of rod one leads to rotation. This is theoretically possible with every configuration with more than four rods. With four rods, even with the infinite extension of rod two, rod one

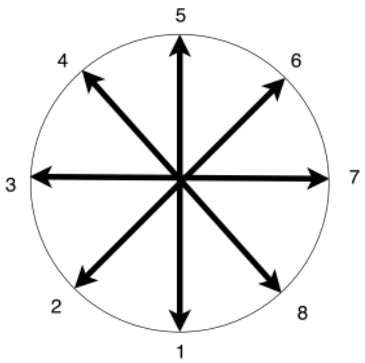



**Figure 5.1:** Number assignment of rods.

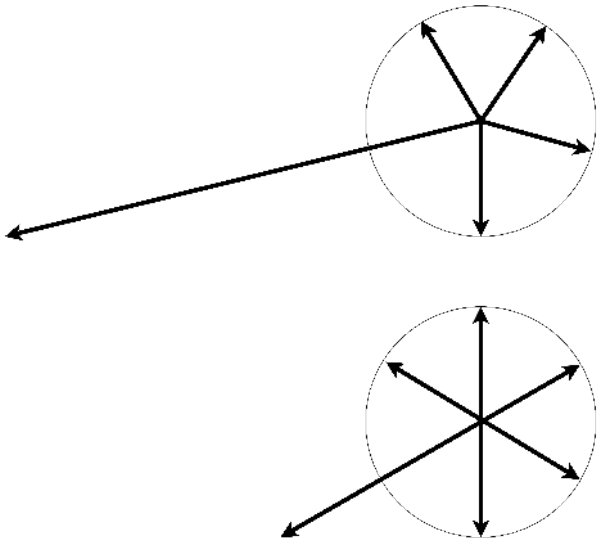


**Figure 5.2:** Different extensions are needed for five- and six-rod configurations to get rod one to point down.

is just at $\zeta$ of 0 rad. Therefore, five is the minimum rod number to theoretically enable a pure pushing behavior. Although there is the possibility of rotation at a specific speed so that the existing momentum will rotate the sphere the needed amount to start extending rod 1. For a start, where the momentum is 0, this is not possible.

Figure 5.2 shows this behavior using a five- and a six-rod configuration. Chapter 4 describes the need for an angle $\beta$, as 0 rad is not enough for extending. Let n be the number of rods, and $\beta$ the angle at which rod 1 starts its extension, then

$$\beta_{\max} = 0.5\pi - \frac{2\pi}{n}\,, \tag{5.1}$$

which gives a maximum $\beta$ of 0.314 rad. If we take the maximum length $l_{\max}$ of a pole into account, using Equation (4.2), we get

$$\beta_{\max} = \arccos\left(\frac{r}{l_{\max} + r_s}\right) - \frac{2\pi}{n}\,. \tag{5.2}$$

We will proceed with the knowledge of the minimum five rods, as the prototype introduced in the next section fulfills the requirement with its $l_{\max}$.

For leverage, Section 4.1 already found three poles to be sufficient for a theoretically possible leverage locomotion. This was, as with two poles, the scenario of both pointing vertically, one upwards and one downwards, which does not allow the start of the leverage approach. With three poles, there is no such case.

The rods are not only used for locomotion on flat ground but also to overcome obstacles. For the best case (rod 1 points down), there is no difference between different rod configurations as the extension of rod one leads to climbing. More interesting is the worst case, where rod one is

| A\B | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 0 % | 4 | 2 | 2 | 1 | 1 | 1 |
| 20 % | 5 | 3 | 2 | 2 | 2 | 1 |
| 40 % | 6 | 3 | 3 | 2 | 2 | 2 |
| 60 % | 7 | 4 | 3 | 3 | 2 | 2 |
| 80 % | 8 | 5 | 4 | 3 | 3 | 3 |
| 100 % | 9 | 5 | 4 | 3 | 3 | 3 |

**Table 5.1:** The number of extensions needed to be capable with B rods per side disc to overcome obstacles with A percent of its radius. A = 0 is the necessary number of extensions for a configuration to just roll on flat ground. Calculated for a diameter of 1 m, an overall length of the actuator of 0.40 m, a non-extendable part per rod of 0.1 m, and a non-extendable part per extension of 0.01 m. The side disc is shifted 25 cm from the middle disc.

a minimum angle before rod 8 is able to start pushing. Subsection 4.1.5 provides the calculation of the exact lengths required; not the length of the extended pole to fulfill the requirement but rather the number of telescopic extensions needed is of interest. More extensions lead to more complexity and instability of the single rods. However, the points of failure also increase with more actuators. Therefore, the overall goal is to find a good compromise between extensions per pole and number of poles. Table 5.1 shows the needed number of extensions for a given number of rods to overcome certain obstacle heights. We assume a non-extendable part of a rod of 0.1 m and a overall size of one actuator 0.4 m, which is a good radio. The line of a height of 0 % therefore is the needed extension for just implementing pushing locomotion. The same is done for the used actuator of our prototype, with an overall size of 0.35 m and a non-extendable part of 0.23 m, which is a rather bad ratio. Table 5.2 shows this. Table 5.1 and 5.2 give a rough

| A\B | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 0 % | 11 | 5 | 4 | 3 | 3 | 2 |
| 20% | 13 | 7 | 5 | 4 | 4 | 3 |
| 40% | 16 | 9 | 7 | 5 | 5 | 4 |
| 60% | 19 | 10 | 8 | 7 | 6 | 5 |
| 80% | 21 | 12 | 9 | 8 | 7 | 6 |
| 100% | 24 | 14 | 11 | 9 | 8 | 7 |

**Table 5.2:** The number of extensions needed to be capable with B rods per side disc to overcome obstacles with A percent of its radius. A = 0 is the necessary number of extensions for a configuration to just roll on flat ground. Calculated for a diameter of 1 m, an overall length of the actuator of 0.35 m, a non-extendable part per rod of 0.23 m, and a non-extendable part per extension of 0.01 m. The side disc is shifted 25 cm from the middle disc.

overview over the range of extensions, as the first is a good ratio of length per extension to the overall length and the second a bad one, but both belonging to a existing range of real actuators. So overall, we can decrease the number, if there is a sufficient stable way for multiple extensions in one rod. It may be considered that an even number of rods allows mechanisms of coupling opposite rod motors, as this is also an approach for lowering the number of needed actuators. For our prototype, we used eight actuators with maximum five extensions.

### 5.1.2 Complete Design

The design of the prototype approximates the shape of the sphere by three discs. On both the outer discs, eight linear telescopic actuators are mounted. Struts connect those discs to the middle discs, where all further electronics are mounted. Figure 5.3 shows the design as blueprint, whereas Figure 5.4 illustrates the actual prototype without shell and Figure 5.5 shows it with shell.

The disc design leads to an aggravated tilting behavior and, therefore, makes balancing more challenging. For testing different approaches for locomotion, a modified version of the prototype of Figure 5.6 was designed. As the evaluation of which rods to use and when to extend, would be overshadowed by the extensions and retractions for balancing; the initial evaluation of different movement methods is done by a cylindrical design rather than spherical design. Therefore, the radius of the middle plate was reduced to the same radius as both side plates. All three plates now touch the ground simultaneously, and there is no need for stabilizing to the left or right. To ensure the same rolling behavior and same forces, the original middle plate and the shrank have the same volume and hence the same mass. This is achieved by reducing the amount of cut-out material. The electrical components stay the same. Therefore, the overall mass stays the same. Most importantly, for the IMUs, all components stay at the same position relative to the center of the sphere as no point lies between the smaller and the original radius. Figure 5.7 shows the real setup of the prototype. Table 5.3 lists all the used components.

## 5.2 Electric components

The electrical components are the same for both designs. A Raspberry Pi 3 microcontroller is the main processing unit. It estimates the pose of the prototype based on three Phidget 1044 inertial measurement units (IMUs). Due to the striving toward space-suitable solution, only the accelerometer and gyroscope of the IMUs are used, and the magnetometer is not used. Therefore, it is not a problem to place the IMUs near quickly voltage changing parts. The algorithm used to estimate the pose of a spherical robot with three IMUs is described in Two relay boards with each 16 relays switch the power and signal line of each of the 16 actuators on and off. They switch the 12 V of the main power supply directly to both lines of the actuator. The Subsection 5.2.1 describes the function principle of the actuator. The relay boards themselves are also powered directly by 12 V. As the Pi runs on 5 V, a step-down converter is put between the main source and the Pi. The signal for switching the relay boards needs to be 5 V for the used optocouplers used on the relay board. The GPIO pins of the Raspberry Pi provide only 3.3 V, an Arduino mega 2560 Board is used to control the relay boards, as the Arduino switches 5 V over its GPIOs. The communication between both the microcontrollers is done by serial communication, using a USB

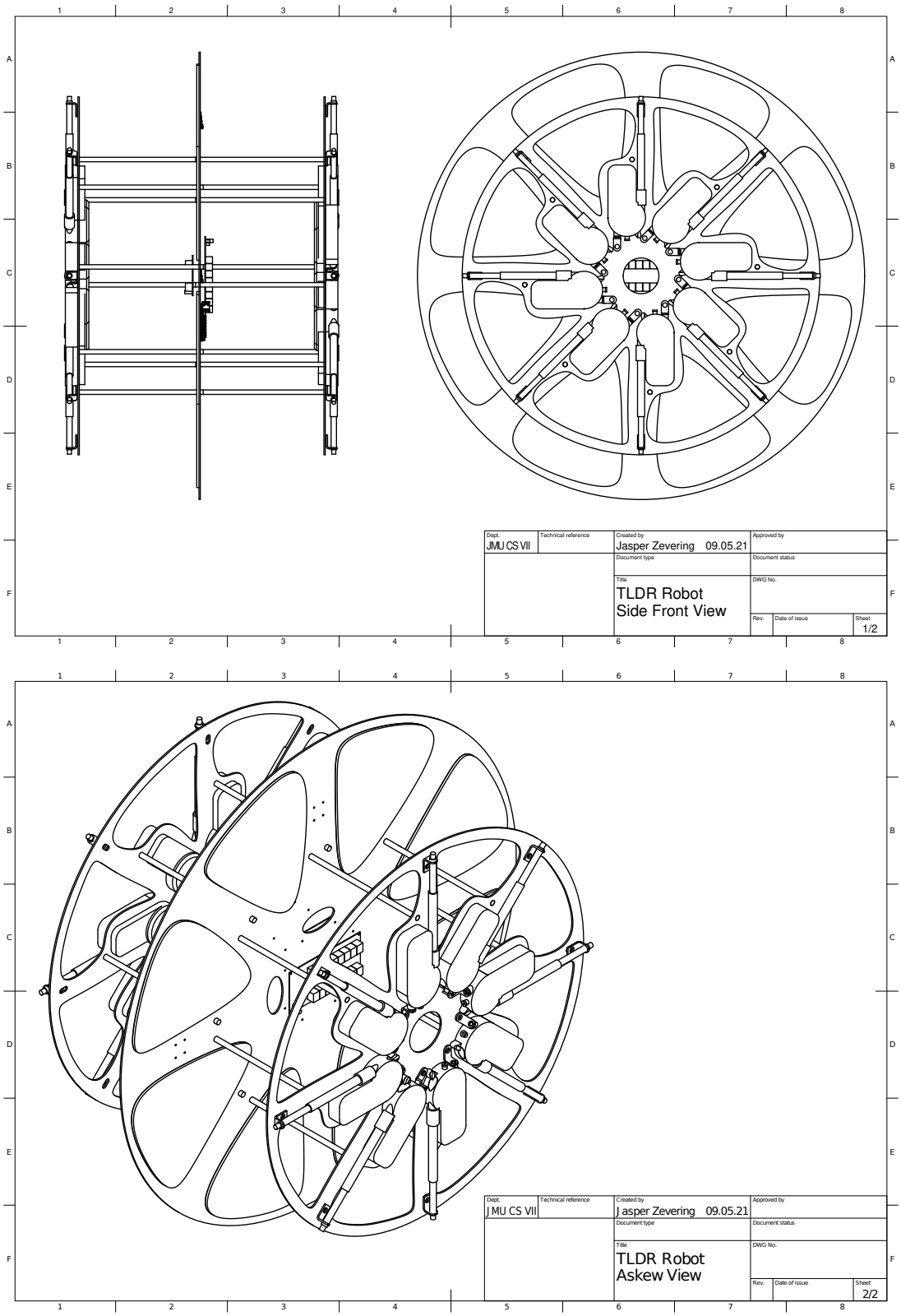



**Figure 5.3:** Blueprint of the prototype without the spherical shell.

**Table 5.3:** Used components for the Prototype. Small or generic parts like installation terminals or usb cables are excluded.

| Component Name | Quantity | Input Voltage |
|---|---|---|
| Controlling | | |
| Raspberry Pi 3 B+ | 1 | 5 V |
| Arduino Mega 2560 | 1 | 5 V/12 V |
| Phidget 1044 IMU | 3 | 5 V |
| USB Hub (4x) | 1 | 5 V |
| Actuator | | |
| Electrical Motorantenna Universal (Solvig) | 16 | 12 V |
| 16-Relais Module 12V with Optocupppler (AZDelivery) | 2 | 12 V |
| Power | | |
| V-Mount Battery 190 Wh | 2 | 14.4 V |
| V-Mount Battery Plate | 2 | - |
| 12 V Stabilizer with 6 A output | 2 | 8 V-40 V |
| 5 V converter | 1 | 12 V |
| Structure | | |
| Side Disc | 2 | - |
| Middle Disc | 1 | - |
| Disc-connection screws M10 280mm | 16 | - |
| Spherical shell half | 2 | - |

connection. This gives a more versatile solution for further prototypes. Appendix A.3 holds the used code for communication on the Arduino side . A baud rate of 9600 ensures communication with a low error rate. The Arduino sends an acknowledgment message after every successfully received message and an error message after corrupted messages. However, as it is a closed system without huge communication distances or wireless transmission, the communication is generally rather simple. Figure 5.8 shows the connection schema of the components. For the built prototype, the power is provided by standard V-mount batteries. They are used because of their standardized mounting system, which provides fast-changing batteries and a tight physical connection when mounted, which is important with a rotating robot. As V-mounts provide 14.4 V, and to protect all components, a voltage stabilizer is put between the battery and the 12 V distribution point.

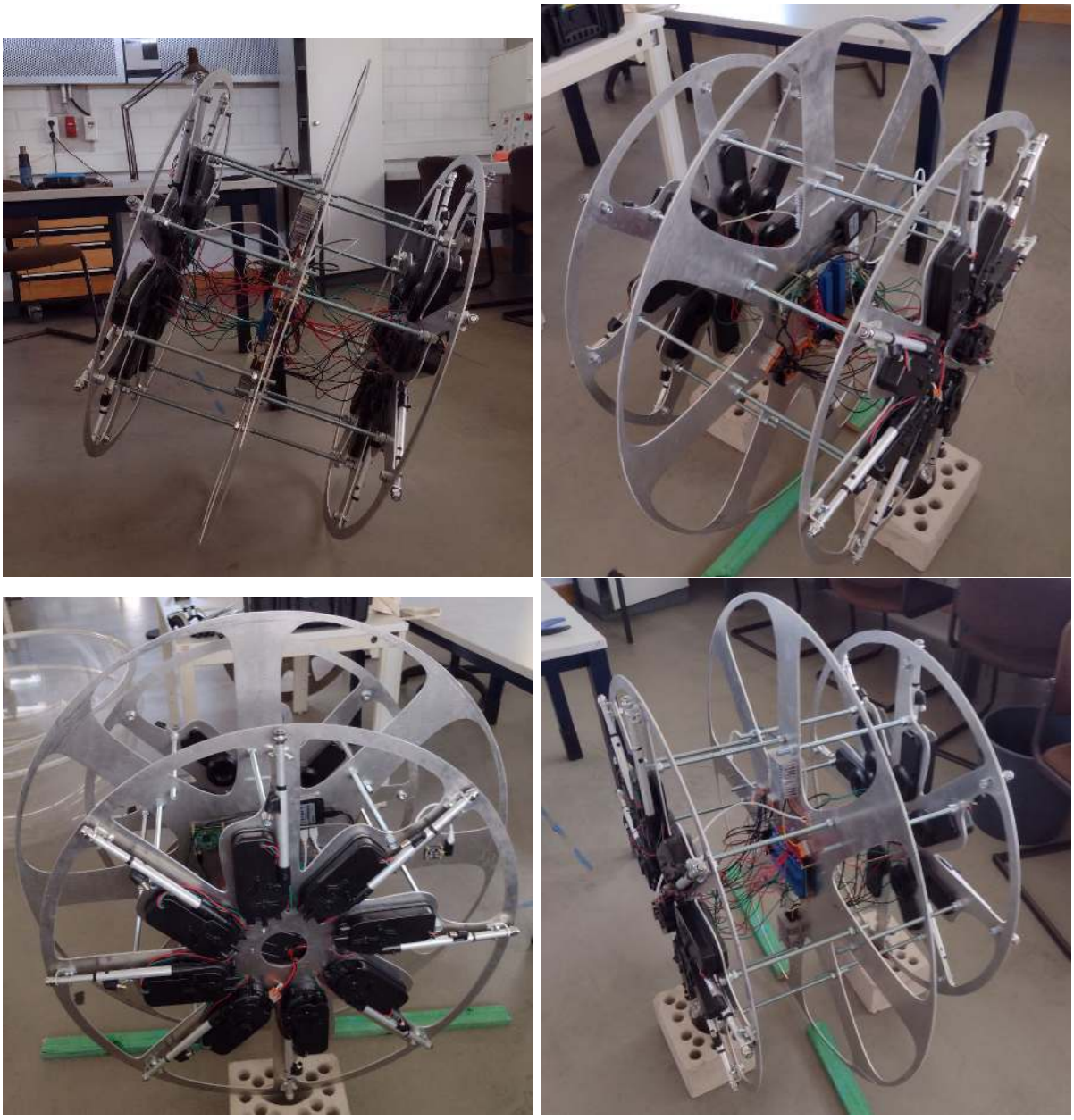


**Figure 5.4:** Photos of the Prototype without shell.

**Figure 5.5:** Photos of the Prototype with shell.

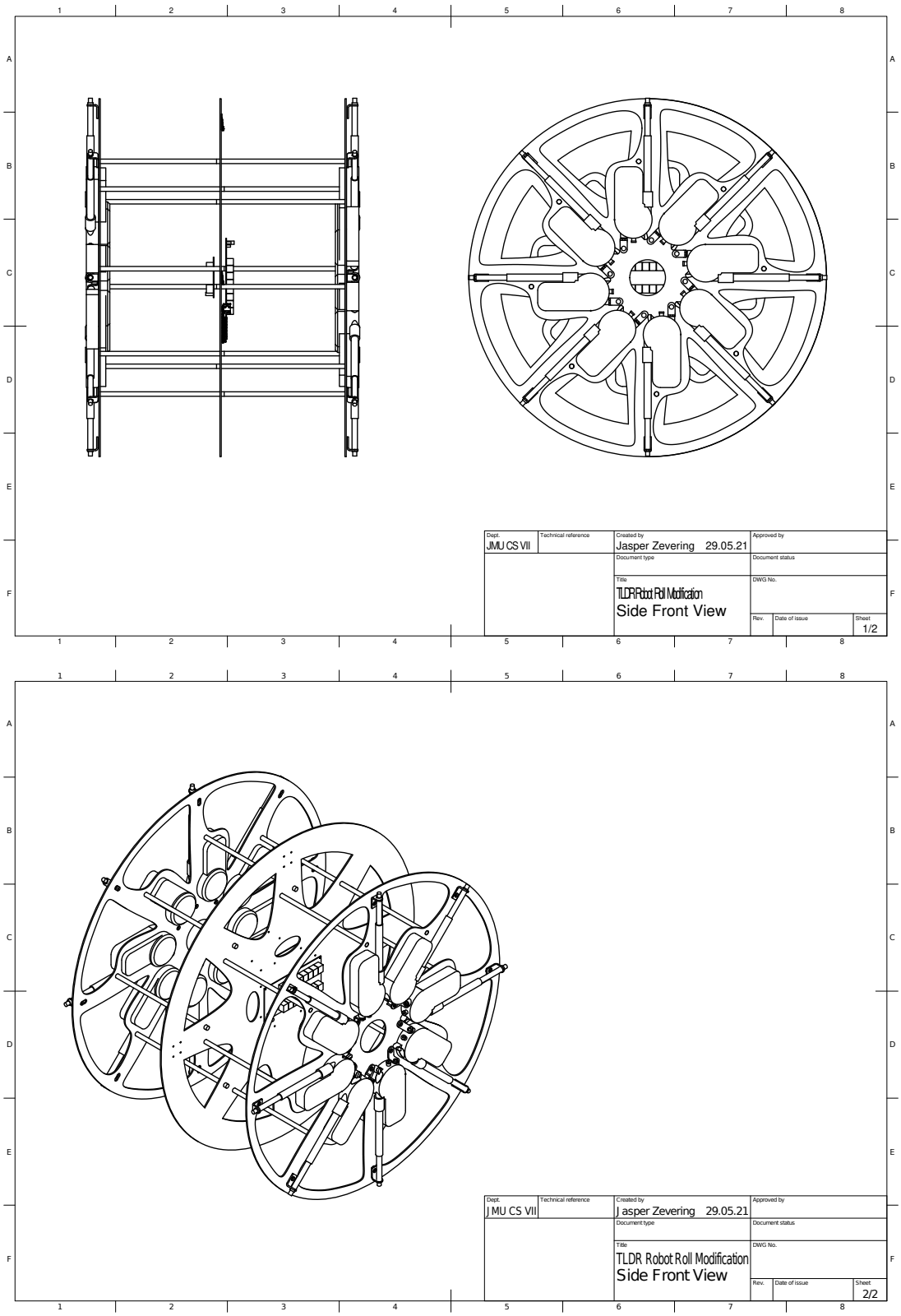



**Figure 5.6:** Blueprint of the prototype with the rolling modification without the spherical shell.

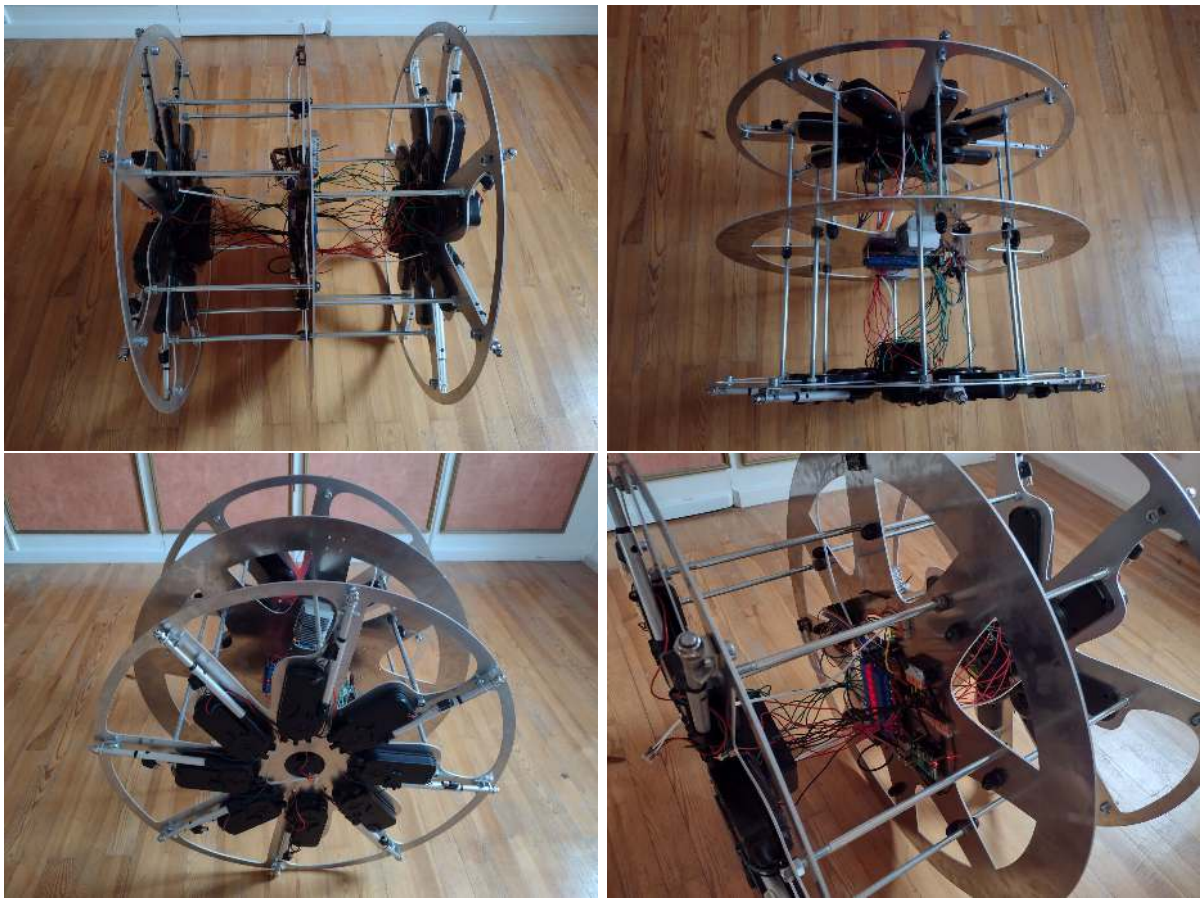


**Figure 5.7:** Photos of the Prototype with the small middle disc for the rolling tests.

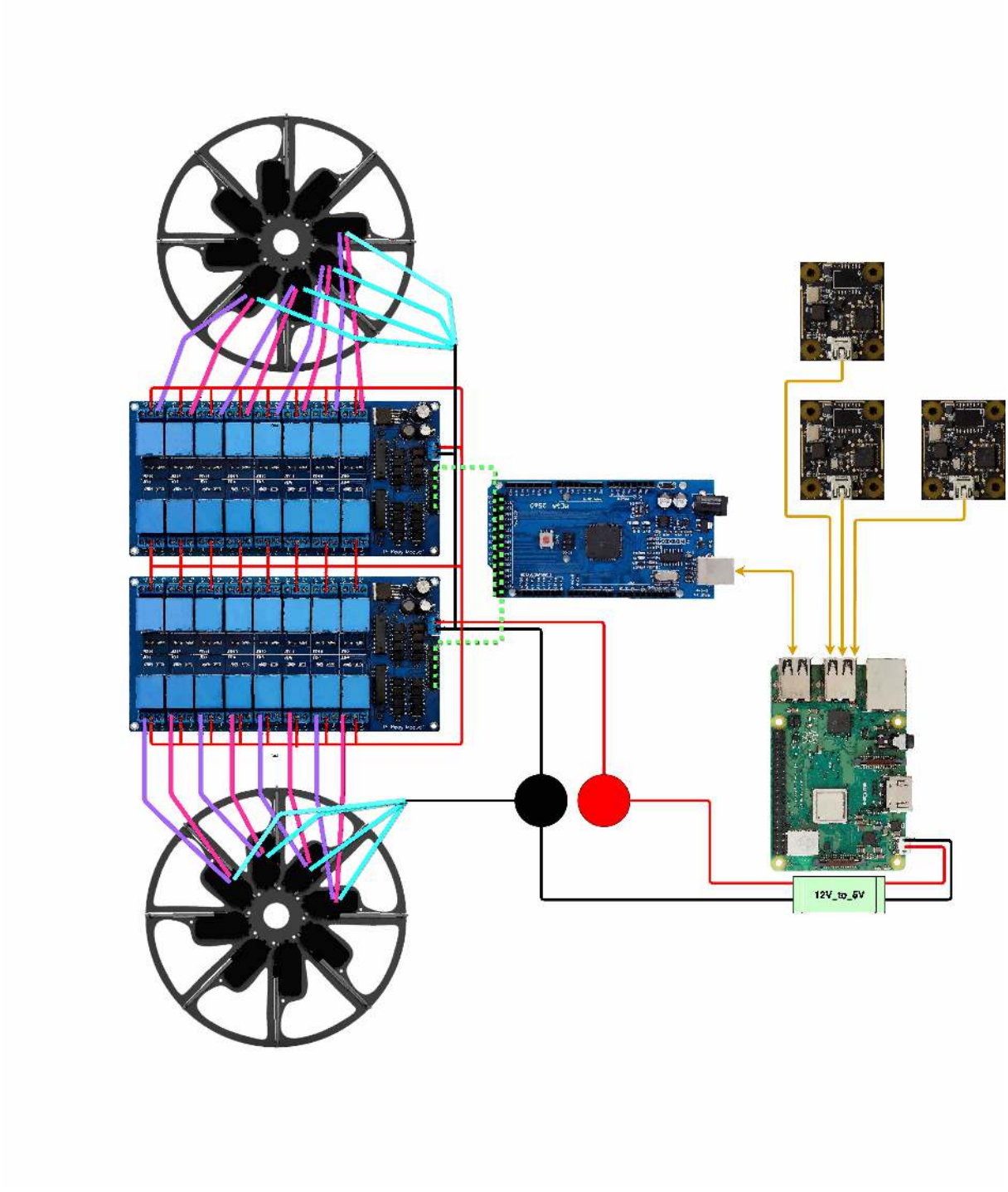



**Figure 5.8:** Diagram of general electrical and data link between components. Yellow: USB connection. Black and Red circle: 12 V voltage supply. Green dotted line: GPIO connection.

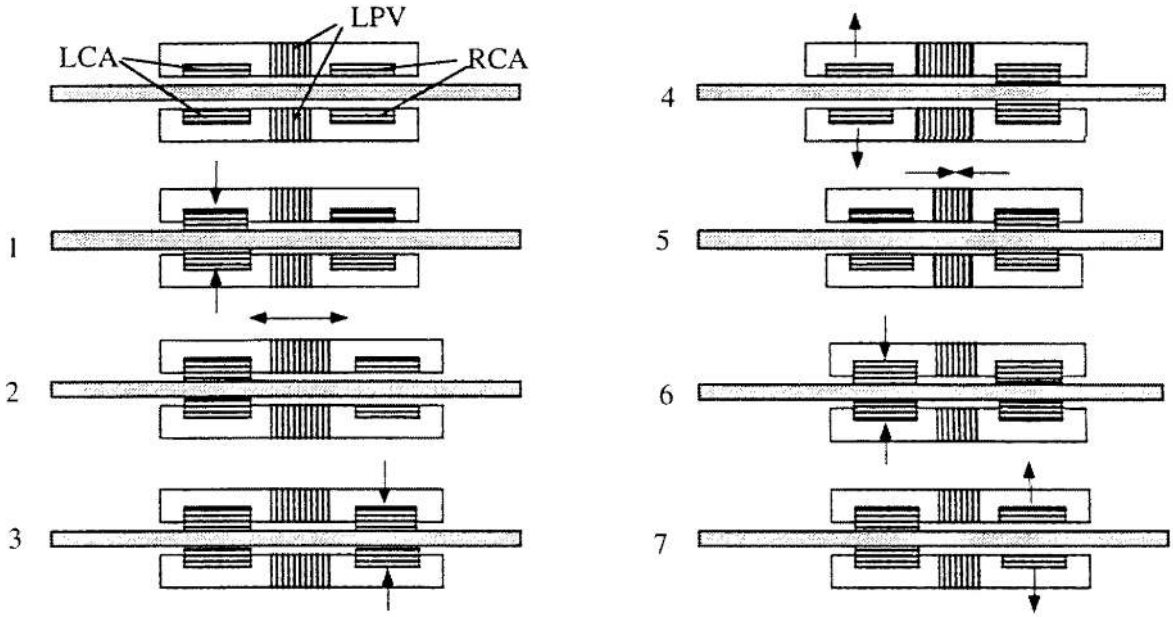



**Figure 5.9:** Concept of the inchworm motor, using the piezoelectric effect [60]. LCA, LPV, and RCA are three contractible and expandable units. With the shown sequence of contracting, releasing, and expanding of the three units, the pole is shoved to the right side.

### 5.2.1 Actuator

The TLDR prototype holds 16 linear telescopic actuators. In principle, there are multiple suitable actuator types, as there are only three absolute necessary requirements. First, the speed of extension and retraction needs to meet the requirements for the desired rotation speed, for which Subsection 4.1.1 contains the calculation. The needed speed decreases by forgoing high omegas or changing the $\alpha$ and $\beta$ values, but there will always be a requirement of minimum available speeds. Secondly, the strain of an actuator. It depends on the requirements of the robot whether it needs to push itself straight up, or other procedures requiring a certain force of an actuator. Third, the required maximum possible extension must be reached, which leads in all configurations calculated in Subsection 5.1.1 to the need for some telescopic extension, as the required length of extension exceeds the length of the fully retracted actuator. The first two basic requirements may be achievable with just the right scaling of an actuator. However, the third requirement excludes some actuators, which basic working principle does not allow telescopic extensions and mostly focuses on small, sometimes ultrasonic, movements like magnetostriction-based motors [41][82][42]. Others focus on slow but powerful movements like mechanicochemical actuators [76]. Others just have not been investigated for telescopic linear use like piezo motors (piezoelectric actuators) [60][73]. The piezo motors have high power, good controllability, are constructed very small, and provide a sufficient extension and retraction speed for this kind of robot. There are different kinds of implementations. Figure 5.9 shows one of them, the inchworm motor.

Permanent magnet motors or magnetostriction-based motors that do not focus on ultrasonic

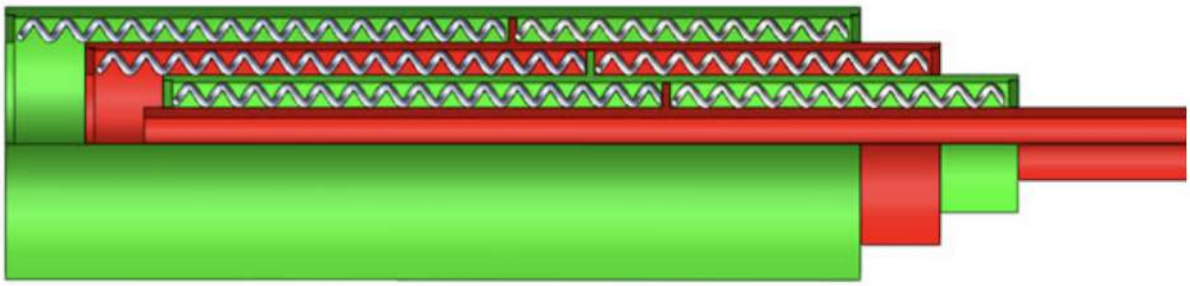


**Figure 5.10:** Concept of o SMA telescopic linear actuator by Spaggiari et al.[72]. The coils shrink if put voltage on and expand otherwise. Two coils connect each extension stage (green and red complexes) to the next stage. By shrinking one and expanding the other, the stage extends or retracts.

speed also do not have developed specific telescopic mechanisms [26]. Besides these hard requirements, there are further requirements on a more practical basis. Of course, the actuators need to be available as there are very promising linear telescopic actuator approaches. However, they are still in the conceptual stages, like SMA-based linear telescopic actuators [72], or the beforementioned telescopic piezo motors. The SMA-based approach is a promising concept, which is designed to produce high torques and fast speeds. Also, the dimensions the concept is for match the dimensions of current spherical robots. Figure 5.10 shows this concept.

Also, the cost-effectiveness needs to be kept in mind, as in the low-cost segment, linear motors are often more expensive than rotating motors since there is just a small field of low-cost linear actuators [17]. Also, side effects often occur, for instance, magnetostriction-based motors often impact other sensors due to the strong magnetic field [47]. This leaves the three most widespread technologies: hydraulic, pneumatic, and direct electric linear actuators. Hunter et al. describe in [32] the difference of possible force of hydraulic, pneumatic, and electric linear actuator as a ratio of 1000 (hydraulic), 35 (pneumatic) to 1 (electromagnetic). Of course, this is always highly dependent on the actually used components, but it gives a rough overview of the relationship. Hydraulic and pneumatic telescopic cylinders both have the option of single-acting and double-acting. This means that either the pressure of the system pushes out the cylinder, and the retracting happens by external forces, like the loading area of a dump truck pushing back, or there are two inlets for pressure, one extending and one retracting. As there is no continuous external force, ensuring retraction, the pneumatic does not do single-acting for the TDLR robot. For the hydraulic, the retraction of a single-acting cylinder is possible with the pump rotating backwards. This leads to each cylinder having its own pump or complex structures. Also, an initial test with low-cost hydraulic telescopic linear actuators shows that the retraction only works unreliably as the cylinder gets caught in itself, as the structure is not built for internal low pressure for retraction. Regardless of whether operated hydraulically or pneumatically, double-acting telescopic cylinders exceed the price of electric actuators by a huge margin. Therefore, we chose the electric telescopic linear actuator for the prototype. Figure 5.11 shows the chosen actuator, made by the company Solvig. It consists of a coil, which is to a certain degree flexible, wound up on a rotational motor. Rotating the motor leads to unrolling the coil, and it is shoved into a telescopic cylinder, which is then pushed outwards. Like

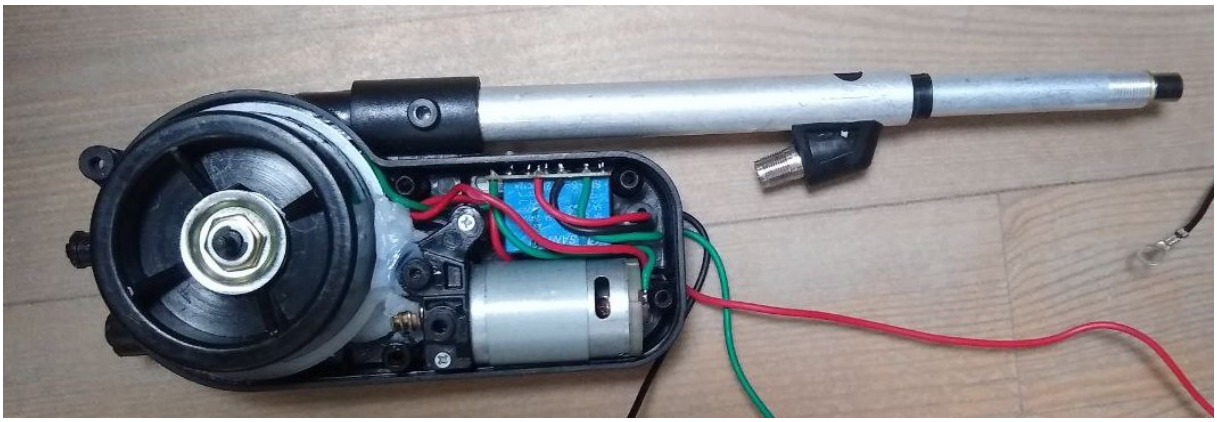


**Figure 5.11:** Linear telescopic antenna motor manufactured by Solvig.

**Table 5.4:** Data-sheet of the linear actuator by Solvig. The * marked values are measured, all others are from the manufacture data-sheet.

| Parameter | Value |
|---|---|
| Maximum Extension | 95 cm |
| Number of Extensions * | 8 |
| Dimensions | 35 cm x10 cm x5 cm |
| Voltage | 12 V |
| Power consumption while extension * | 10 W |
| Extension Speed * | 0.15 m/s |
| Retraction Speed * | 0.15 m/s |

the most used, screw-based linear motion of electrical linear motions, other electrical approaches are limited in speed and the possible number of extensions. The chosen mechanism is, in its basic principle, extendable to any number of extensions. It just needs to be ensured that the telescopic sleeve has enough tension. Table 5.5 lists the properties of the actuator. Each actuator has three input cables. Ground (black), power (red), and signal(green). The ground is connected to the negative terminal of the battery. Power and Signal are switched independently by the relays. The corresponding behavior of the actuator is shown in Table 5.4. The actuators provide no feedback for their current status. Therefore, the only way of estimating the length of the extension is integrating the extension speed over time. The powering of the actuator

**Table 5.5:** Behaviour of Actuator by Solvig with activation of Signal and Power line.

| | Signal high | Signal low |
|---|---|---|
| Power high | Extension | Retraction |
| Power low | Hold | Hold |

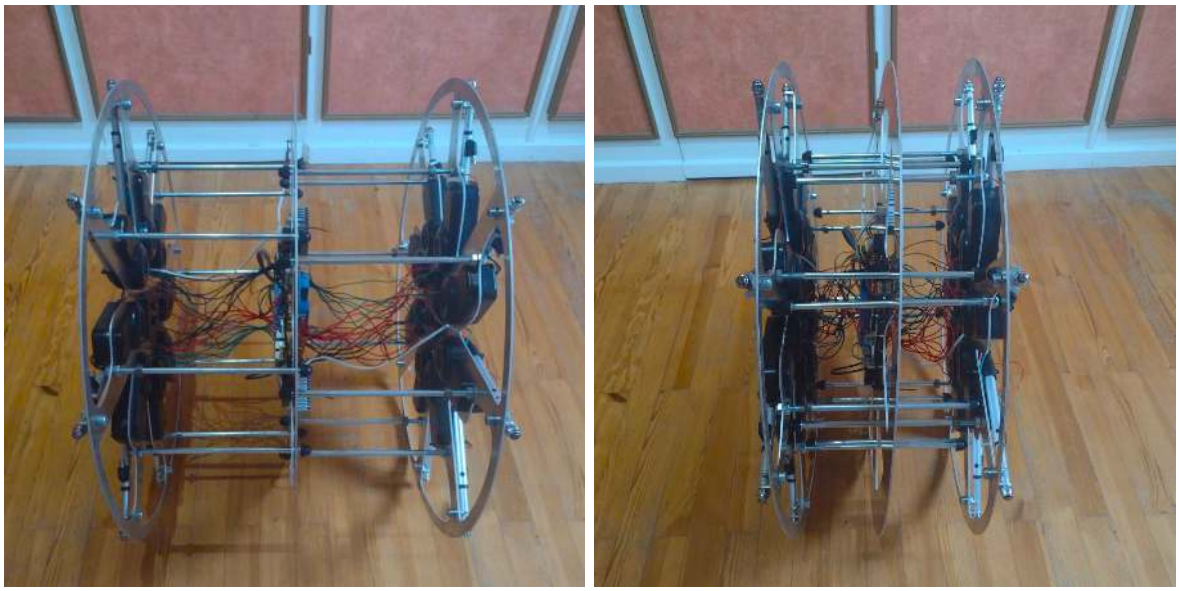


**Figure 5.12:** Prototype in the normal configuration (left) and collapsed configuration for transport(right).

without actual extension is problematic. This is the case if the actuator tries to extend into the ground. The software-based estimation of the length will then miscalculate the length. This overall behavior leads to the unreliability of the estimated length. Therefore, the evaluations in Chapter 6 will only show linear extension, even if no actual extension happened. Without further proof, measuring the resistance of each pole or voltage when applied did not help estimate the current length of a pole. The measured values did not correspond directly to the length, and the poles themselves differed from unit to unit significantly.

### 5.2.2 Structure

The overall weight of the robot is 37.85 kg (inner structure 20.70 kg and shell 17.15 kg). The weight of all 16 motors together is 12.24 kg. The weight of the inner structure is the same for the both versions. The aluminum discs of 3mm provide a good ratio of stability and mass. The three discs are connected by 16 M10 screws with quick-releases on the middle disc side on the screw part between middle and side discs. This enables a quick fold-up for transportation. Figure 5.12 shows this collapsed state.

The variable fixation also enables the fast construction and installation of further middle discs with different sensors and/or dimensions. Later, they were replaced by self-securing nuts for better stability. The discs themselves have cut-out areas to reduce weight. Due to the lack of sufficient mounting options of the actuators, the shape of the main part of the actuator is cut out of the side disc, and the actuator is pushed into it with force. To ensure the right positioning of the rod itself, the end of the guide tube of the actuator is mounted to the side disc. The structure is put into the spherical shell, consisting of two halves. As the discs fit the inner diameter of the shell, there is no need for further fixation since there are no huge internal

forces.

### 5.2.3 Pose-estimation

The prototype uses only IMUs as sensors for pose estimation. This raises problems regarding the position-estimation, as IMUs have no absolute reference for the position in contrast with their orientation-estimation, which has the gravitational vector and magnetometer as absolute reference. And double integration of the accelerometer, which leads in theory to the position, leads to exponential error addition and therefore an unusable position estimation. The spherical robot is a special case, as here, the rotation speed and translation speed correspond to each other as rolling with $\omega$ leads to a translation of $\omega \cdot r$. This is only for spherical/cylindrical robots. Based on this assumption, pose estimation is possible with just IMUs if the $z$-axis is fixed. [88] contains an algorithm and further explanation. For our evaluations, we focus primarily on $\omega$ and $\theta$, which do not need a position estimation. As later, the goal is to use VPIM for controlling, there exists a map and sensor, which a SLAM algorithm then uses for pose estimation. This is more precise then the pure IMU-based evaluation and provides an absolute reference. The IMU is a possible further input to a filter and enables interpolating between SLAM-estimated poses. This all leads to better map generation from which also VPIM will in turn benefit [49].

# Chapter 6

# Evaluation

## 6.1 Locomotion

### 6.1.1 Push Only

The first experiment for evaluation is using the push-only algorithm (Algorithm 1). The parameters of the algorithm are $\beta = 2\,^\circ$ and $\alpha = 57\,^\circ$ Due to the eight rods, an alpha of $47\,^\circ$ is the push-only-single algorithm (Algorithm 2) and is evaluated separately. Figure 6.1 shows the result for a flat, sealed ground with a low grip.

The extension itself is only an estimation as there is no real feedback providing the length of a certain pole. Therefore, the theoretical length of the pole is computed using the extension and retraction time. The flat ground experiments show huge perturbations with a maximum $\omega$ of about $0.4\,\mathrm{rad/s}$. This is due to a slip each pole experiences when it is in contact with the ground. In other words, the pole bends as its tip slips over the ground until the bend is big enough so that further bending requires more force than initiating rotation. Figure 6.2 depicts this behavior where the pole slips over the ground without leading to rotation. Having rather big obstacles (stones etc.) or having just ground with increased grip (outdoor, rough asphalt, etc.) does not allow, or at least minimizes, such behavior. At some point, the pole tip gets canted, even if only minimal. Therefore, the same experiment was performed on the ground with wooden floorboards and on the gravelly ground. Figure 6.3 shows the result for the wooden floorboard, whereas Figure 6.4 shows the result for gravelly ground .

The sliding behavior of the poles does not happen on the grounds with more grip, as there is no or only minimal sliding of the poles. Figure 6.5 shows this for the gravelly surface. The rotation starts with the pole end only moving minimal in comparison to Figure 6.2, where much more slip does not provide any rotation. Additionally, the minimal change of position of the pole end happens with a relatively quick jump, in contrast with the slow, continuous slide on the flat ground . Therefore, the overall speed is higher with way fewer perturbations. The achieved $\omega$ is for the wooden floorboard at about $0.7\,\mathrm{rad/s}$ and for the gravelly at about $1\,\mathrm{rad/s}$. The remaining small oscillation is a result of the mono-speed problem, which Subsection 4.1.2 described. The problem for both surfaces with high grip is the repeatability of the experiments, as the description of a wooden floorboard and gravelly is imprecise in terms of its grip. For the flat, sealed ground, the grip itself is also not well defined, but at least all geometrical aspects are

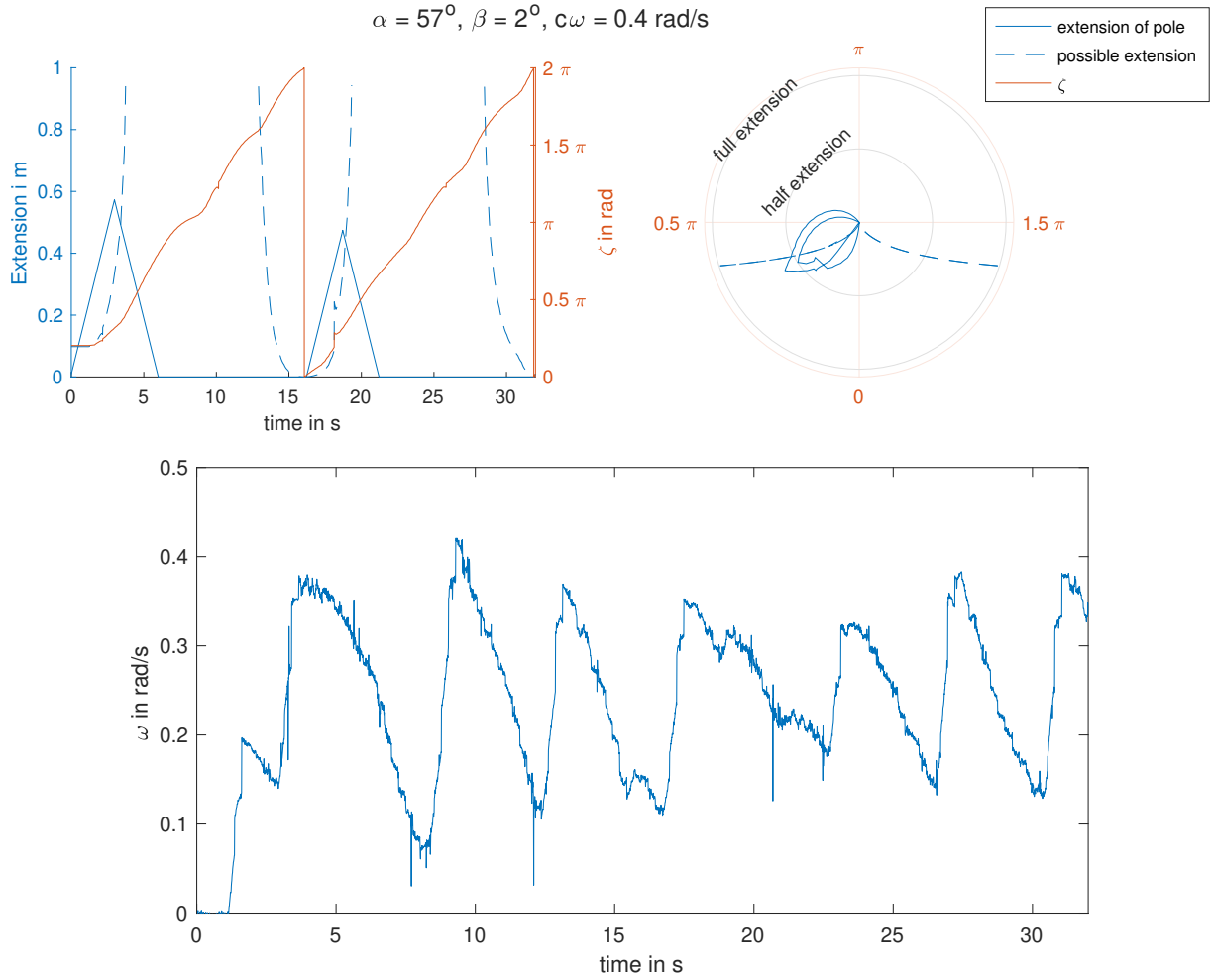



**Figure 6.1:** Robot on a hard, flat surface with less grip. Above: Extensions of prototype with push-only approach with $\alpha$=57 ° and $\beta$=2 °. The prototype starts out of a standstill. Below: measured $\omega$ of the same experiment.

defined since it is just flat with no obstacles. The quantitative description of grip, the coefficient of friction, does rely on the measured object, and its measurement is difficult for moving, in this case rotating, objects [18]. As prototypes are likely to use different actuators, the coefficients of friction will not be measured with the pole tips of this prototype. The huge differences in the behavior, such as the different $\omega$, or the different strength of oscillation, are less the result of the grip itself. The surface touching the ground is minimal, and therefore the frictional force is minimal. The behavior is due to the missing sliding of each pole as they run into unevenness or obstacles on the ground. Further, a smoother surface is beneficial for the leverage algorithm, where no pole pushes into the ground. Therefore, all further general experiments are performed on this ground. A larger $\alpha$ of 67 ° for all experiments does not impact the reached $\omega$ as the pole does not touch the ground anymore.

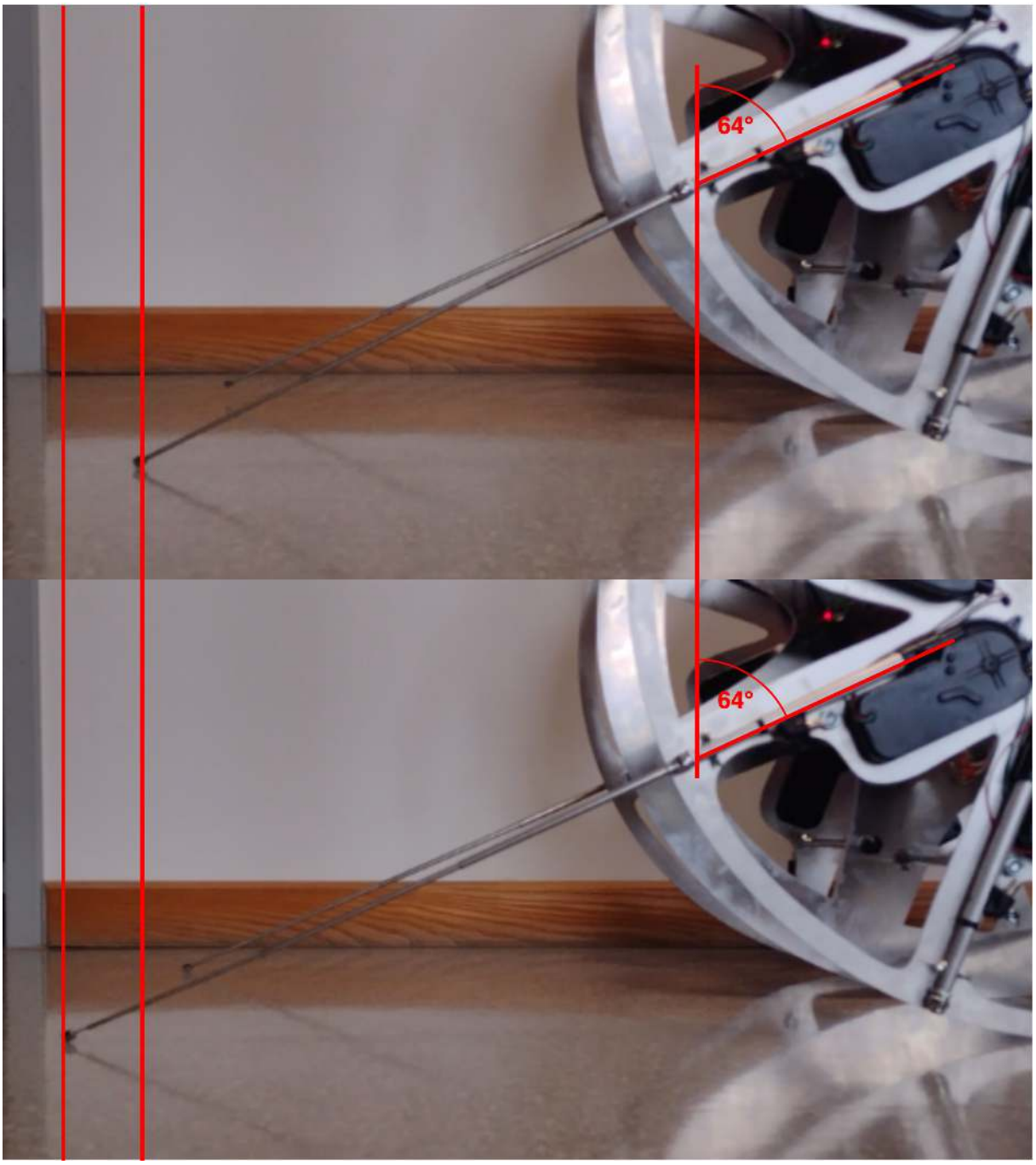



**Figure 6.2:** Demonstration of the slip and bend behavior of the rods on flat, sealed ground. The poles extend and bend, but no rotation is initiated.

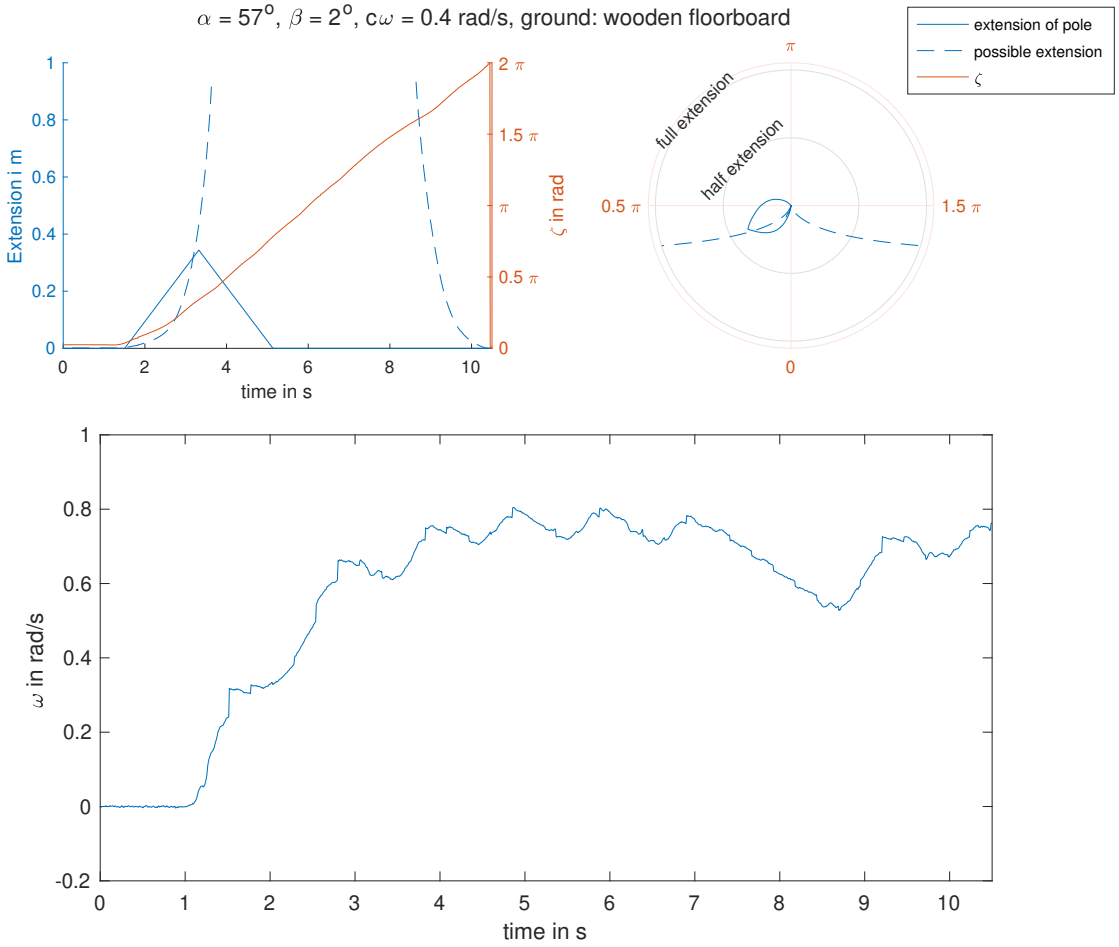



**Figure 6.3:** Robot on wooden floorboard surface. Above: Extensions of prototype with push only approach with $\alpha$=57 ° and $\beta$=2 °. The prototype starts out of a standstill. Below: measured $\omega$ of the same experiment.

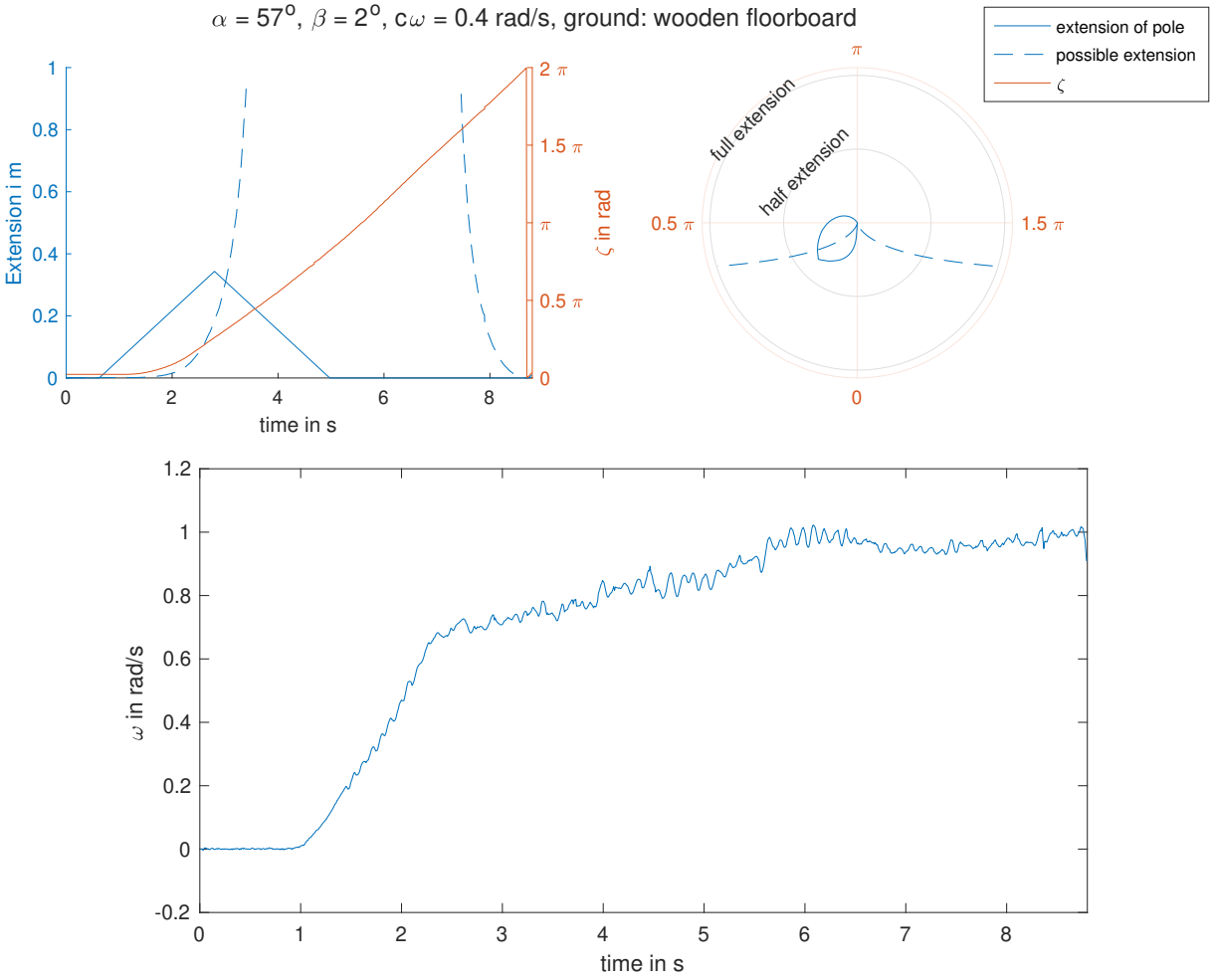



**Figure 6.4:** Robot on a gravelly surface. Above: Extensions of prototype with push only approach with $\alpha = 57\,^\circ$ and $\beta = 2\,^\circ$. The prototype starts out of a standstill. Below: measured $\omega$ of the same experiment.

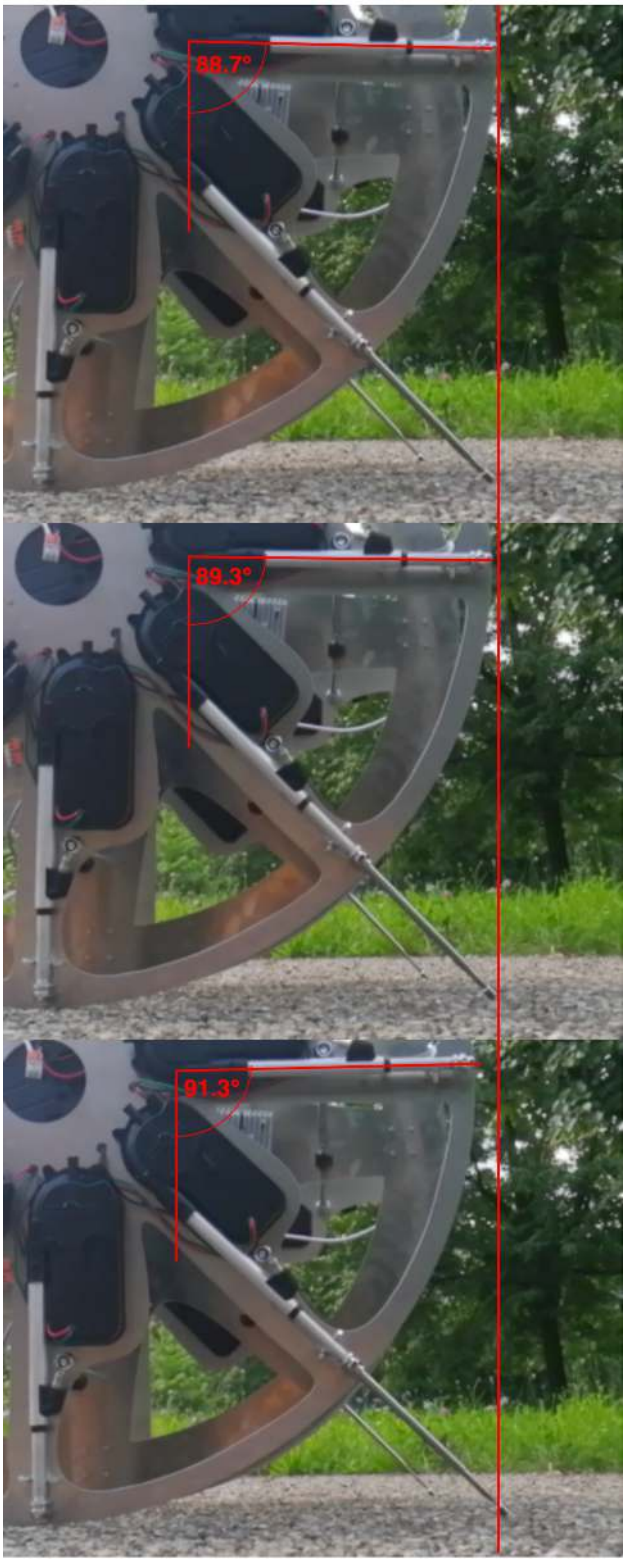



**Figure 6.5:** Demonstration of the slip and bend behavior of the rods on gravelly ground. As the poles extend, the angle of the robot changes.

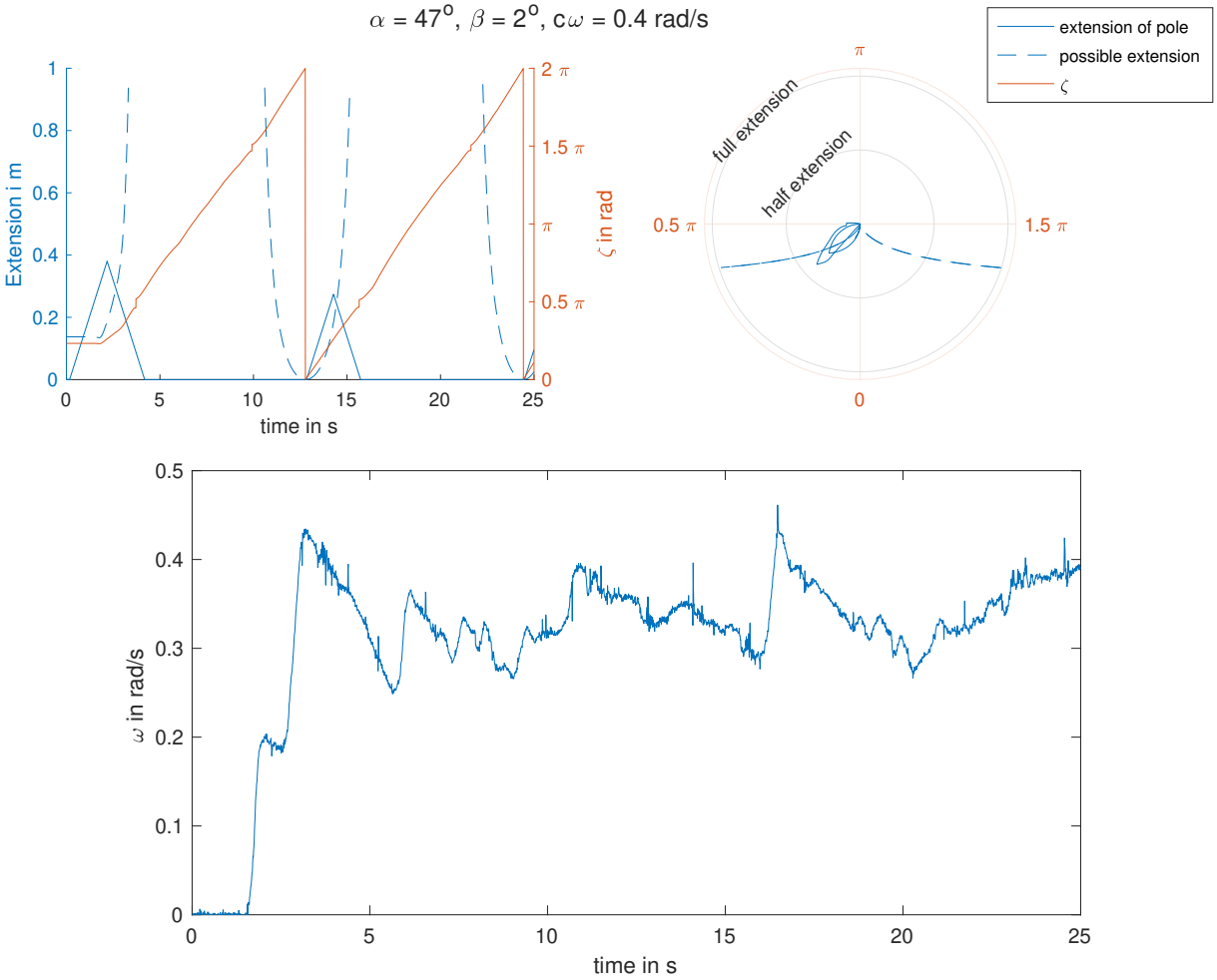



**Figure 6.6:** Robot on a hard, flat surface with less grip. Above: Extensions of prototype with push only approach with $\alpha$=47 °and $\beta$=2 °. This corresponds to a single rod use at once. The prototype starts out of a standstill. Below: measured $\omega$ of the same experiment.

### 6.1.2 Push Only Single

Using an $\alpha$=47 ° and $\beta$=2 °, leads to the push-only-single algorithm (Algorithm 2). Figure 6.6 shows the results for a extension between 2 ° and 47 ° and Figure 6.7 shows an extension between 0 ° and 45 °. We see that the one with $\beta$=2 ° has significantly fewer perturbations than that with $\beta$=0 °. This is the behavior described before: the extension at 0 ° does not lead to a rotation but does lead to internal tension, which increases as the actuator pushes without extensions. If the $\zeta$ angle increases, the pole extends with the previously built internal tension. Therefore, the acceleration is too fast for the extension of the pole, and it does extend without contact with the ground. This does lead to the oscillation seen in the plotted $\omega$. Increasing both angles to an extension between 5 ° and 50 ° leads to the behavior shown in Figure 6.8. In this case, a big oscillation occurs, which decreases rapidly. The overall speed is slower, which is the

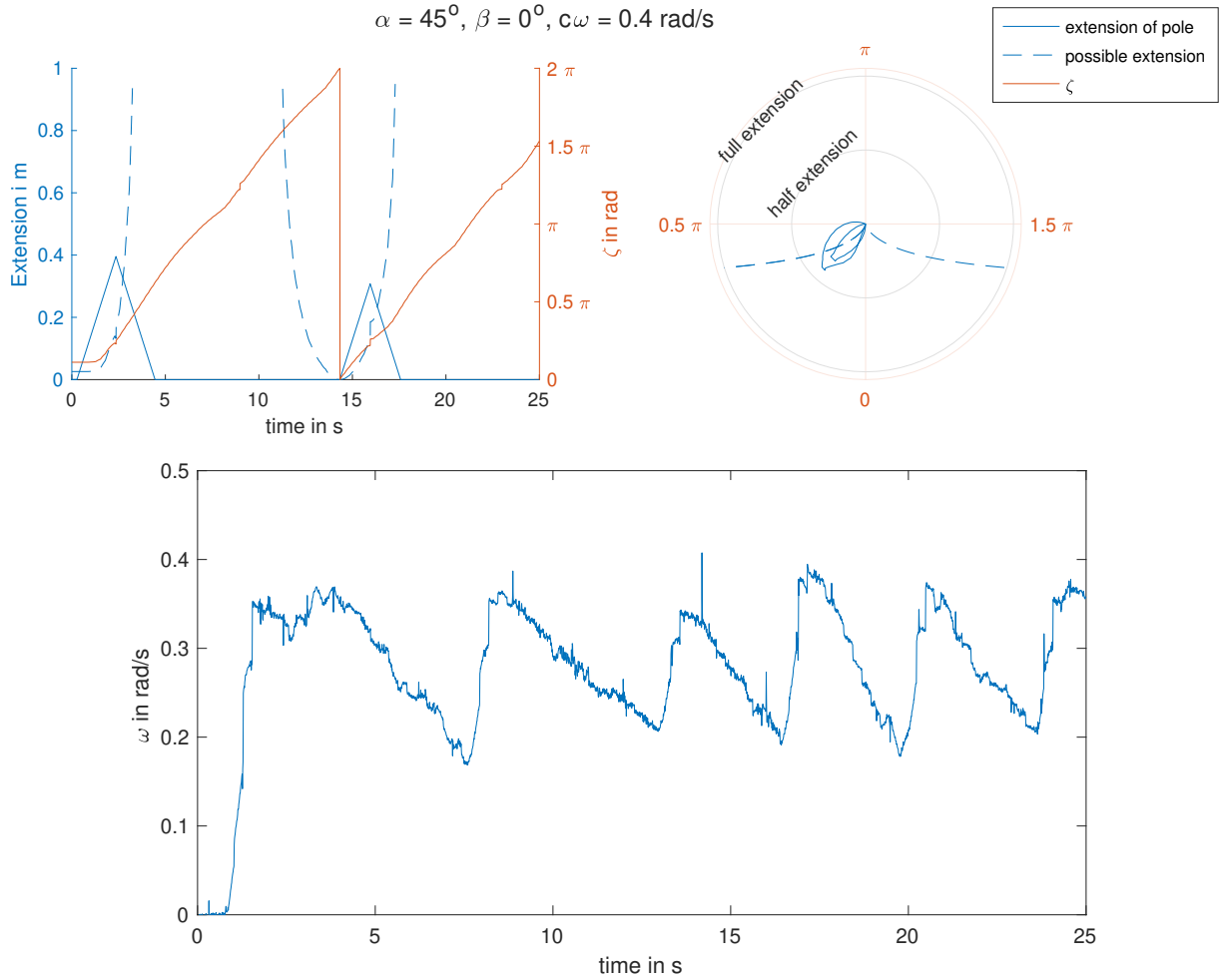



**Figure 6.7:** Robot on a hard, flat surface with less grip. Above: Extensions of prototype with push only approach with $\alpha$=45 ° and $\beta$=0 °. This corresponds to a single rod use at once. The prototype starts out of a standstill. Below: measured $\omega$ of the same experiment.

behavior Figure 4.9 showed before. This shows that choosing $\beta$ either too large or too small is counterproductive. Therefore, we take 2 ° as value for further experiments. The purpose of this single-rod use approach was to provide a solution for mechanically restricted systems. As our prototype does provide individual actuators and has no limitation of simultaneous extensions, this special case of the pushing approach will not be further evaluated as its own approach, but will just be referred to as a push-only approach with these specific values.

### 6.1.3 Leverage Only

For the TLDR prototype, the extension of the poles on the side to which the prototype rolls, generates enough torque to lead to rotation. However, when at a pitch of 0 rad, extending the pole with $\zeta$ of $\frac{5}{8}\pi$ rad or $\frac{3}{2}\pi$ rad on its own is not enough. Figure 6.9 shows the behaviour when

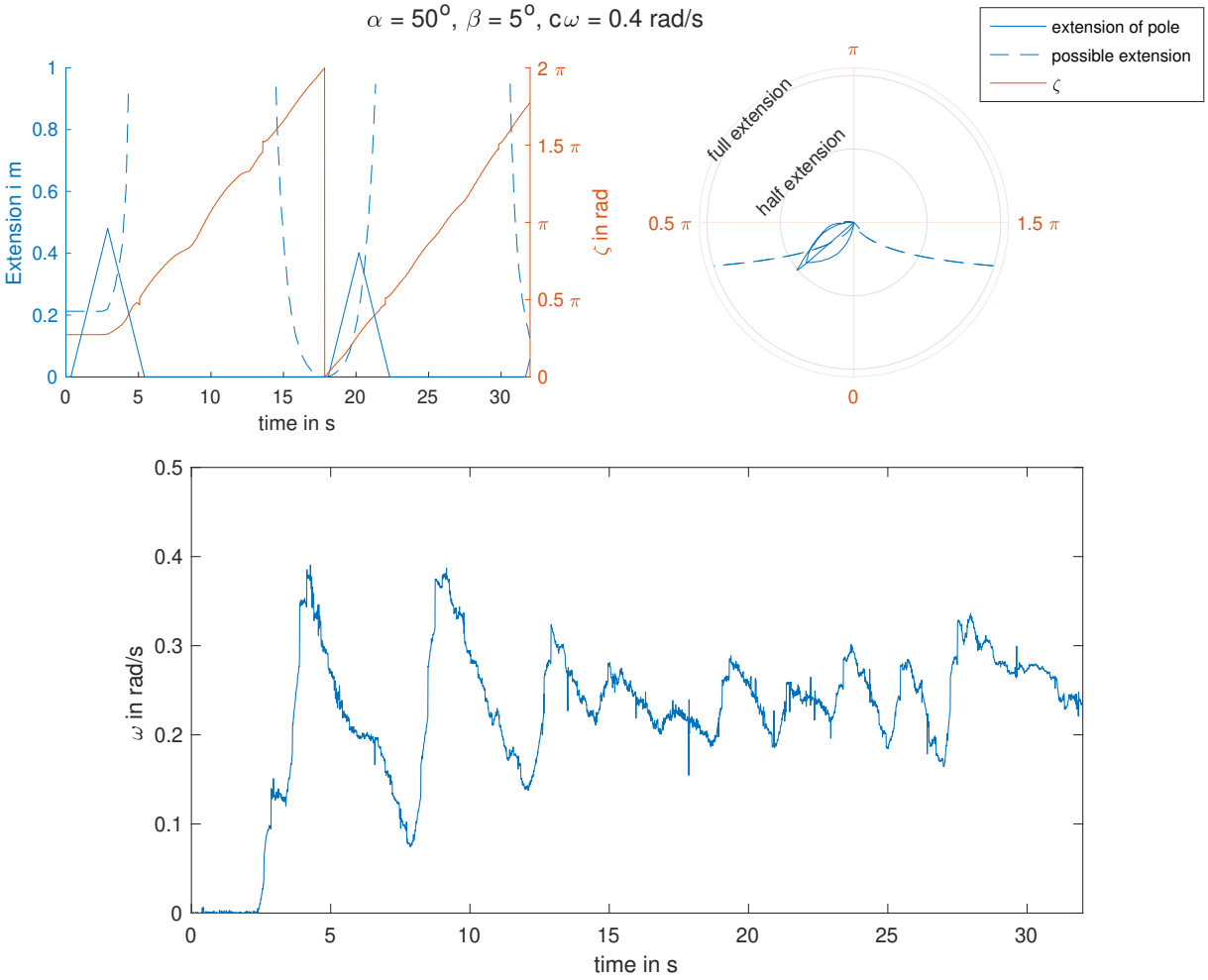



**Figure 6.8:** Robot on a hard, flat surface with less grip. Above: Extensions of the prototype with push-only approach with $\alpha$=50 ° and $\beta$=5 °. This corresponds to a single rod use at once. The prototype starts out of a standstill. Below: measured $\omega$ of the same experiment.

the pole at $\frac{5}{8}\pi$ rad is at full extension, and the one at $\frac{3}{2}\pi$ rad is extended. We see that only at nearly full extension the rotation of the prototype starts. This leads to a problem as the $\gamma$ is smaller than $1.5\pi$ rad. Just ignoring this leads to an uncontrollable movement of rolling back and forth. The prototype rolls onto its extended poles, which bend to a certain angle at which they begin bending back to the original position, causing rotation in the opposite direction. The backward rotation will slow down and turn into a forward rotation, but with an already retracted pole, this specific pole will now retract enough not to block the further rotation. Unfortunately, the next one will. Figure 6.10 shows this behavior.

This does not mean the leverage approach is not suitable for this prototype. If the rotation of the prototype is initiated manually, the leverage approach maintains the rotation, respecting the calculated $\gamma$, $\epsilon$, and $\epsilon_s$, but this is only if the (too) fast acceleration is taken into account. This results in the solutions, which Section 4.1 discussed: using a simulation or implementing

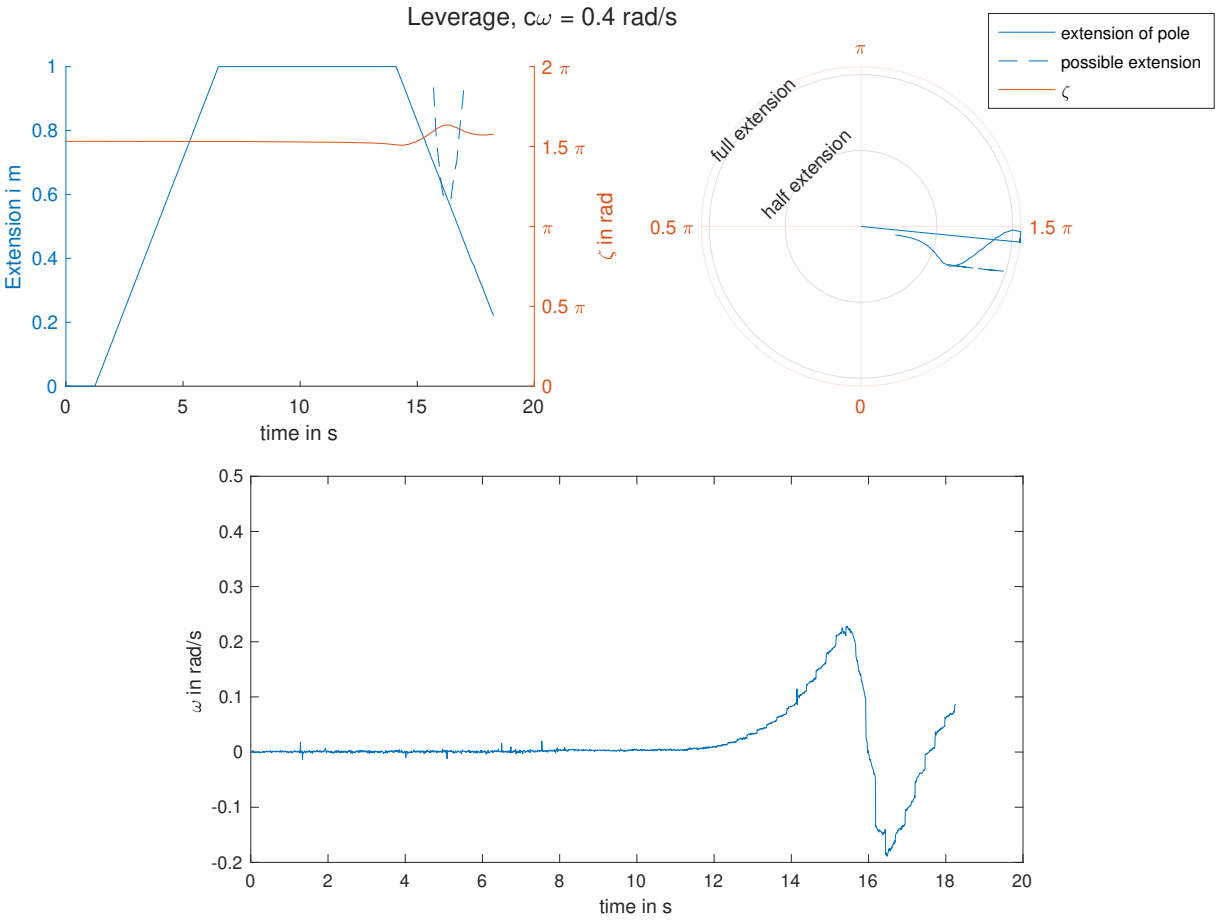



**Figure 6.9:** Leverage approach with a slow start. The pole at about $\frac{3}{2}\pi$ rad is shown, while the one at $\frac{5}{8}\pi$ rad is fully extended all the time. The rotation starts very slow. The $\omega$ grows exponentially, which is too fast for the retraction of the pole, leading to it contacting the ground at about 16 seconds, which leads to a rotation backward.

empirical boundaries, Both lead the approach of calculating exact angles values to absurdity. In contrast with the pushing approach, the leverage approach is not suitable as a standalone locomotion approach. Therefore, this prototype uses leverage only as an addition to the push approach if the robot does not generate enough torque by pushing.

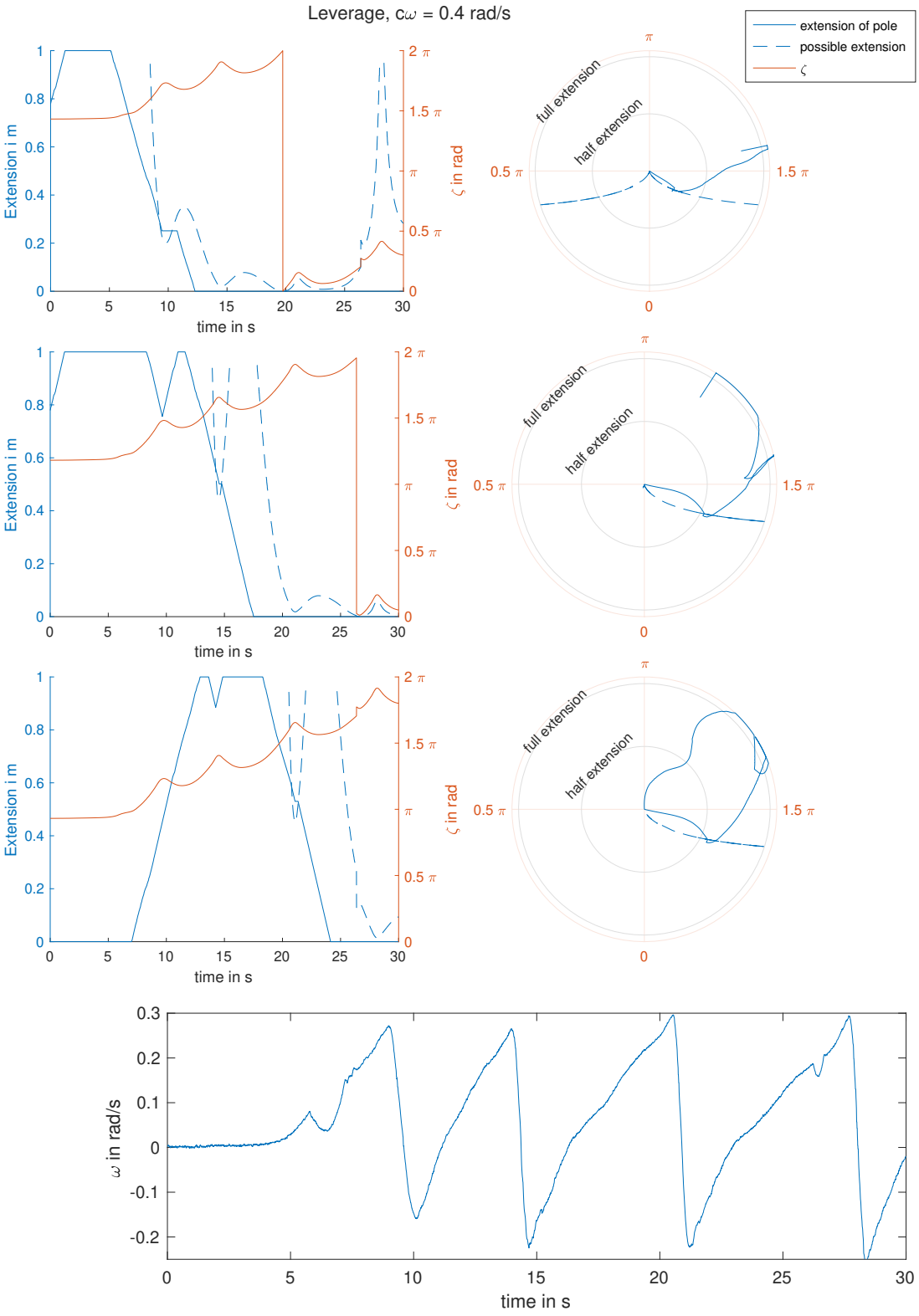



**Figure 6.10:** Leverage approach with bouncing behavior. Each visualized pole leads to one bounce and hence negative rotation. The first hits the ground at 9 s, the second at 14 s, and the third at 22 s.

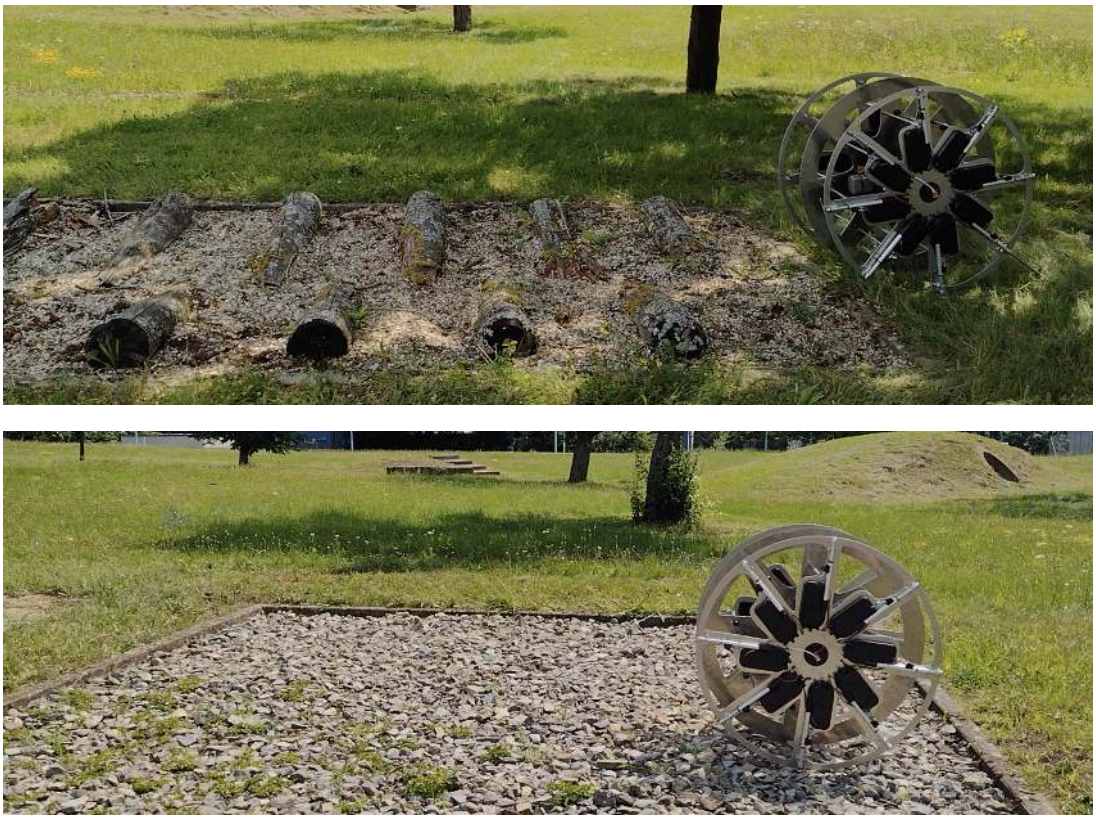


**Figure 6.11:** Conducted outdoor experiments. Above: grass and for additional experiment trunks as obstacles. Below: stones. All experiments were unsuccessful.

### 6.1.4 Push and Leverage

Due to the extremely fast acceleration with the leverage-only approach, the evaluation of the push and leverage approach together was unsuccessful. The same behavior as with the leverage-only algorithm occurs with initial tests, i.e., pole retraction is not fast enough once acceleration starts. Together with the pushing, the actuators experience more force than during the leverage-only approach; thus, the pushing increased the bending of the poles even further. This leads to irreversible damage to two poles that needed to be replaced. Therefore, no measured data is available, and the approach was not investigated further on flat ground. The experiment was performed for uneven ground/outdoor experiments with rougher terrain than a hallway or street as the acceleration is much more damped due to the softer underground. All tests were unsuccessful as no rotation was initiated at all, despite using leverage and pushing approaches simultaneously. Figure 6.11 shows the conducted experiments with no further data as no rotation was initiated.

Further experiments tested the capability to climb slopes.

We used a ramp with increasing tilt to perform climbing attempts on different slopes. First experiments were performed with $\alpha = 57\,^\circ$ and $\beta = 2\,^\circ$. However, on the ramp, there is a gap between the pole of $\zeta = 0\,^\circ$ and the ramp due to its inclination. This means an extension at $0\,^\circ$

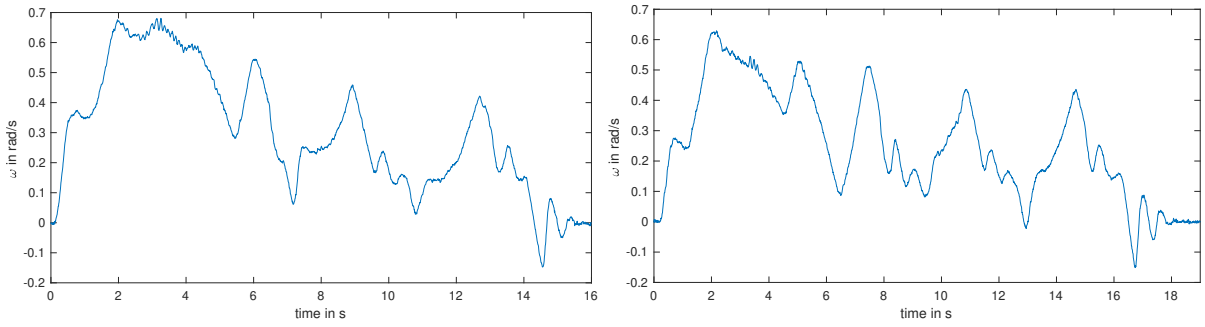



**Figure 6.12:** $\omega$ of the slope test for pushing only with $\alpha = 57\,^\circ$ and $\beta = 2\,^\circ$(left) and $\alpha = 67\,^\circ$ and $\beta = 0\,^\circ$.

does lead to rotation. Also, $\alpha$ was increased as $\omega$ goes towards $0\,\mathrm{rad/s}$ at the end, which enables even high $\alpha$ to still touch the ground. Figure 6.12 shows the difference between both values of $\alpha$ and $\beta$. The similar behavior of both is noteworthy, as the same changes in values lead to huge differences in the flat evaluation in Subsection 6.1.1. The rolled distance on the slope with increasing inclination was the same, and both achieved an inclination of $2.5\,^\circ$. Therefore, for the further evaluation of the slope, we took the theoretically superior $\alpha = 67\,^\circ$ and $\beta = 0\,^\circ$. The starting position has a crucial impact on the achieved inclination as it impacts the end position at a certain inclination, which determines if a new pole starts extending or not. Therefore, after this experiment, the starting position was empirically optimized, and therefore the final evaluation shows higher achievable inclinations. Figure 6.13 shows the results from using only leverage only, only pushing, and both together.

Of course, a full distance at the given inclination would result in more accurate data of the robot than with a raising inclination of one track. Still, we want to focus on the relative behavior and achieved values between the approaches, rather than on accurate absolute values for this specific prototype. Leverage only achieved $1.3\,^\circ$. Pushing $3.6\,^\circ$ and Pushing and Leverage combined achieved $3.7\,^\circ$, which is in the range of the theoretical calculated values provided in Subsection 4.1.5, which estimated for leverage $0.92\,^\circ$ and for pushing $4.1\,^\circ$.

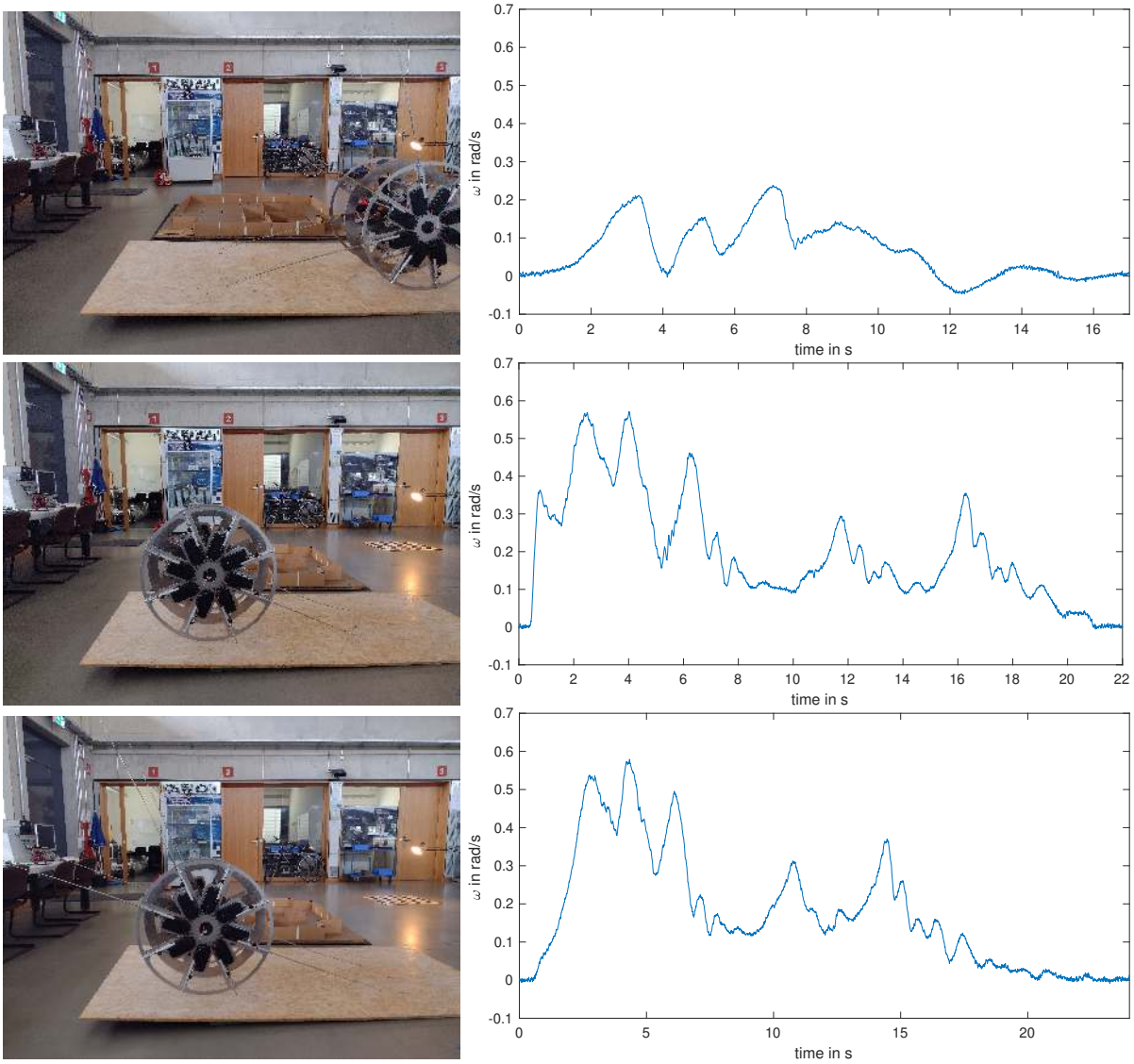



**Figure 6.13:** Slope experiments. The robot rolls up a ramp with increasing slope. The slope is then measured with a water level at the place where the robot stops. Top: Leverage only locomotion. Achieved inclination: 1.3 °. Middle: Pushing only locomotion. Achieved inclination: 3.6 °. Bottom: Pushing and Leverage locomotion. Achieved inclination: 3.7 °.

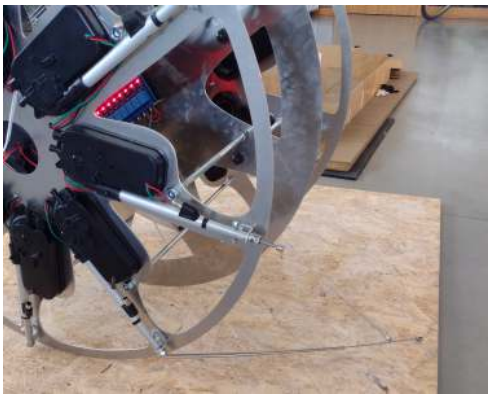


**Figure 6.14:** Irreversible damage after a roll onto extended poles.

### 6.1.5 Braking

First tests of braking showed the incapability of the poles to support the weight of the robot, even with slow rotation speeds. As this leads to irreversible damage (see Figure 6.14), no further experiments are conducted regarding braking.

### 6.1.6 Friction

To test the influence of the friction of the pole with the ground $\mu_{\text{Pole}}$, rubber surfaces are attached to the pole ends. Figure 6.15 shows the pole ends and the corresponding $\omega$ on the same flat ground.

With rubber, the overall rolled distance is less, and the data is noisier. This is counter-intuitive to the assumption that more grip leads to better rotation. This is explained by the small oscillations, which always occur when the robots exactly roll over one pole ($\zeta = 0$). The robot is raised minimally, expending energy that would otherwise be used for transversal motion. Therefore, the experiment needs to be conducted again with the overall robot in the spherical shape. Other factors like balancing may influence this, but an evaluation is not possible with the given prototype.

## 6.2 Balancing

The first short evaluation regarding balancing is the one targeting the three, and five-point approach explained in Section 4.2 with Figure 4.25 and 4.26. Figure 6.16 shows the result. Further experiments with a three-point stabilizing approach will only be conducted with a spherical shell, as in this case, the tip-over is usable in interaction with locomotion.

Therefore, the following evaluations are with respect to the basic balancing of the same disc-size robot with the middle disc lifted. Figure 6.17 provides the result of stabilizing the $\phi$ of the robot.

The $\phi$ is kept constant quite well. After initial fast changes between the states, the robot gets to a pose at which it has balanced itself. It is noteworthy that this stable pose has the

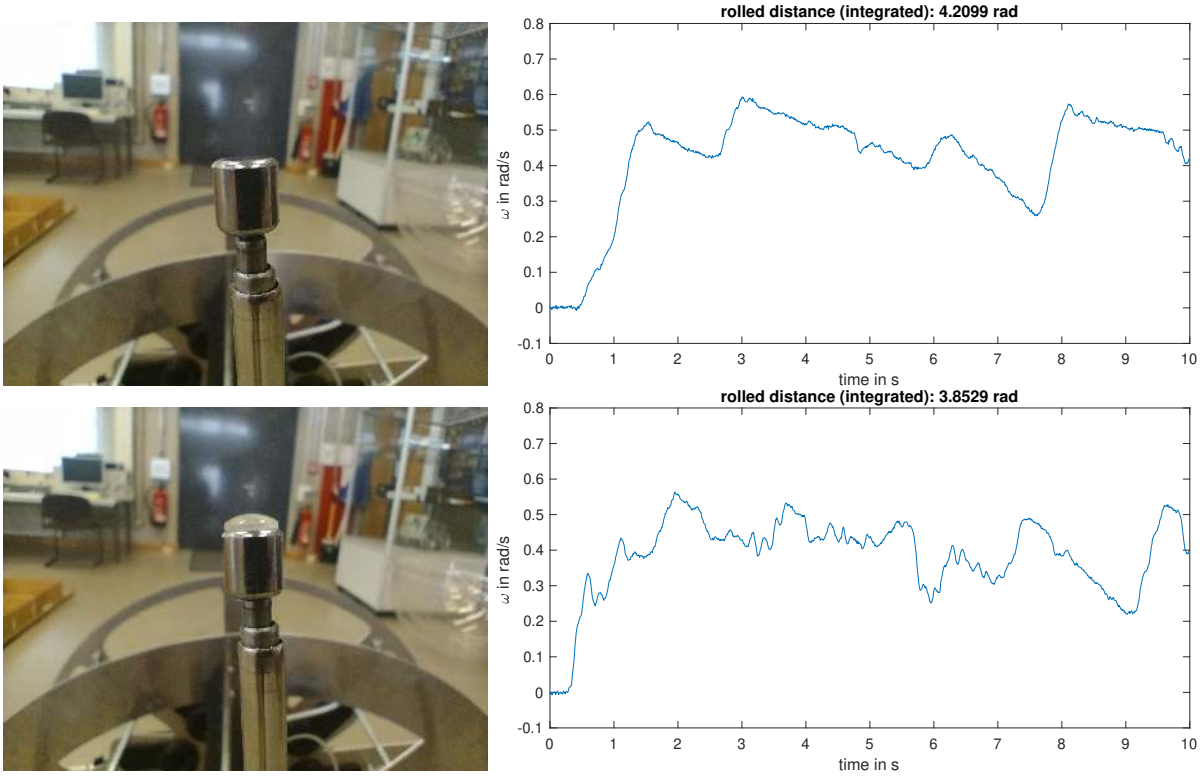



**Figure 6.15:** Friction experiment on the cylindrical (small middle disc) prototype. Above: plain pole ends. Below: with rubber attached.

pole at state "Hold," and not extending all four poles continuously, which also has the possibility to bring a stable pose but also of damaging the actuators, and is more power-consuming. As stated in Section 4.2, it is crucial to maintain the poles, which are at the moment responsible for balancing due to their $\theta$, at $l(\theta) = l_{\text{balance}}(\theta)$, even if $\epsilon_\phi$ is small, to keep a fast interfering actuator if $\epsilon_\phi$ becomes too big. For the same reason, the retraction shall never undercut $r_m - r_s$. Figure 6.18 shows what happens if these values are undercut.

The $\phi$ does not exceed the geometrical possible $\phi_{\text{catastrophic}}$, nor the limit $\phi$ given by the strength of the poles. However, it does oscillate between $0.04\pi$ rad and $-0.03\pi$ rad. The initial peak, initiated by a manual push, is intercepted at $0.05\pi$ rad. This was done using a smaller $r_m$ than the real value, which leads to a smaller $l_{\text{balance}}(\theta)$ and a smaller $r_m - r_s$. Figure 6.19 shows the middle position, as well as both extreme points of the oscillation. As a stable $\phi$ is the optimum and shows the capability of the actuators to stabilize the robot, we will not perform further experiments regarding balancing on the small prototype. We re-evaluate this case with the real spherical robot. For the balance experiment, the result of the evaluation we expect the outcome to be nearly the same.

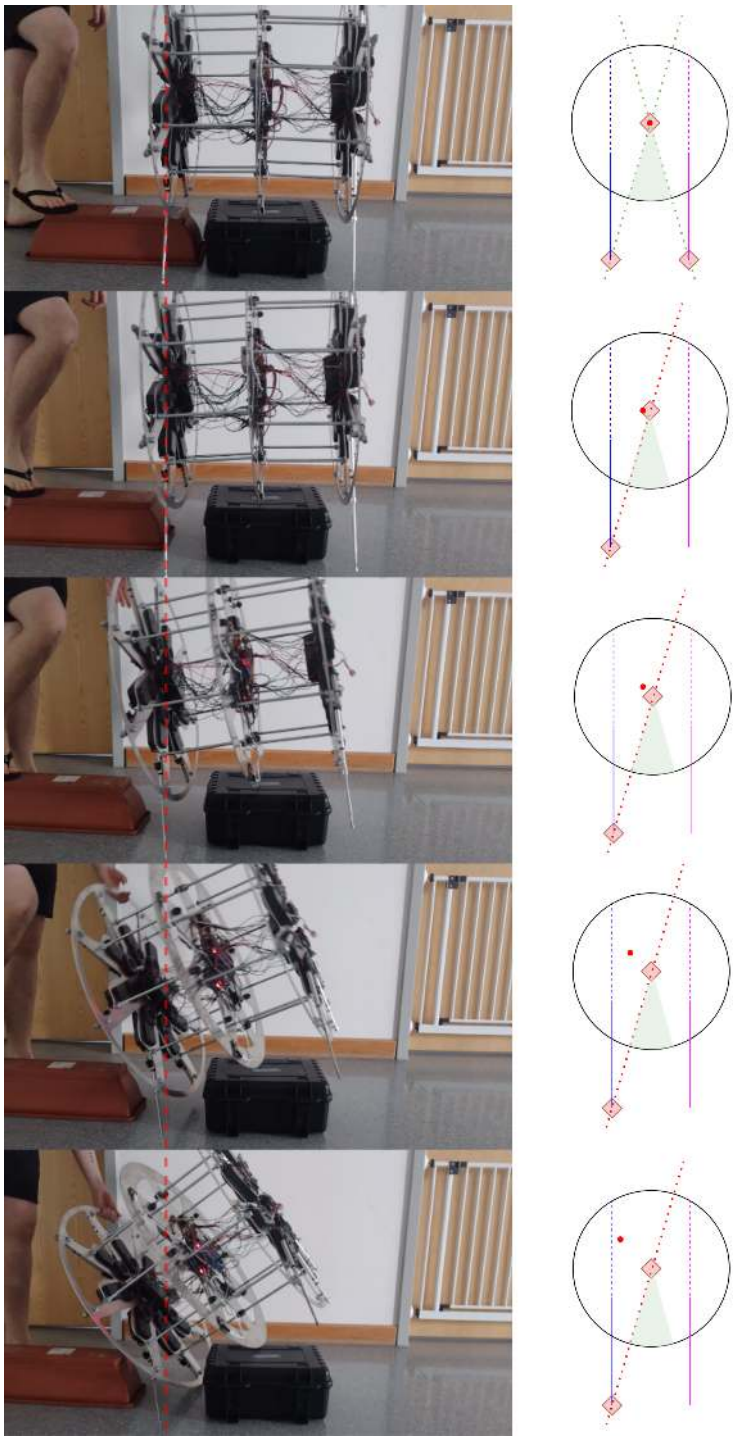


**Figure 6.16:** Tip-over with three-point stabilization. The red dashed line, crossing all five pictures, shows that the touch-point stays at the same position. Therefore sliding of the pole is not a problem. The abstraction shows the contact points of the poles and the shell with the ground (red square), the center of mass (red dot), and the stable area for the center of mass (green area). The pink and blue lines represent the two side discs with the corresponding poles. If they are continuous, they represent an extended pole. The green dotted line indicates that no line crosses all contact points. The red dotted lines if one line crosses all contact points.

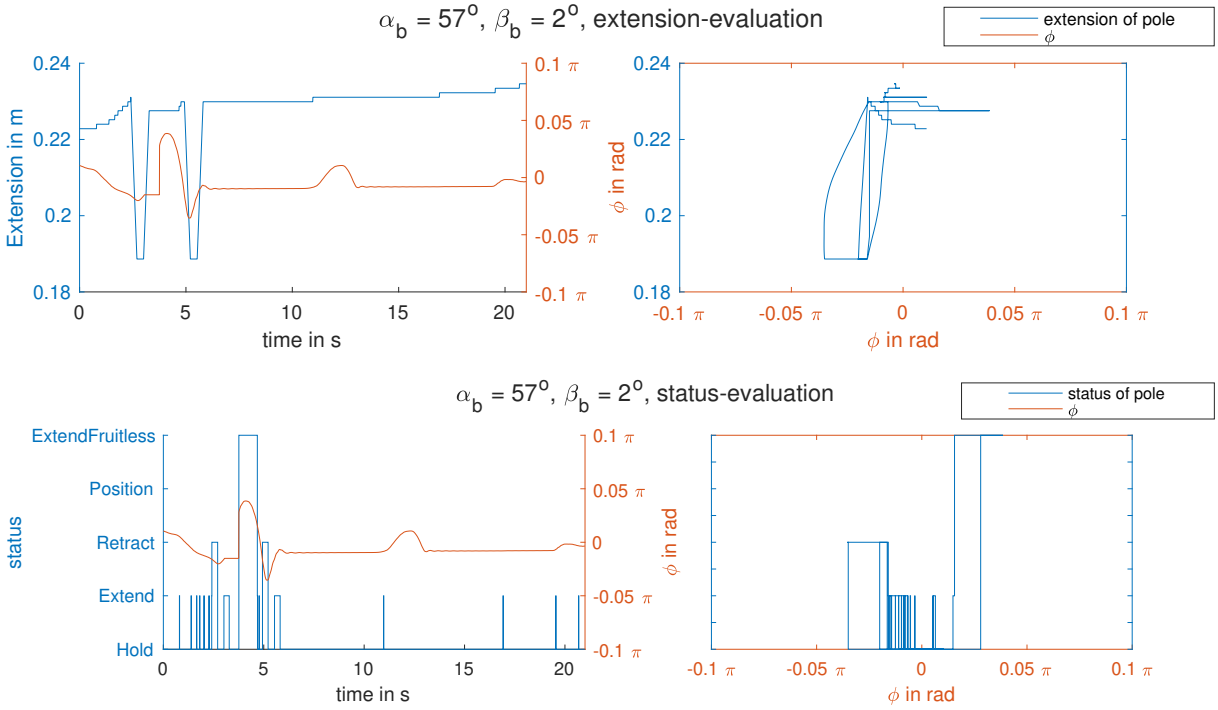



**Figure 6.17:** Balancing evaluation without spherical shell. Above: Estimated length. Below: Status of the pole. Both: $\phi$.

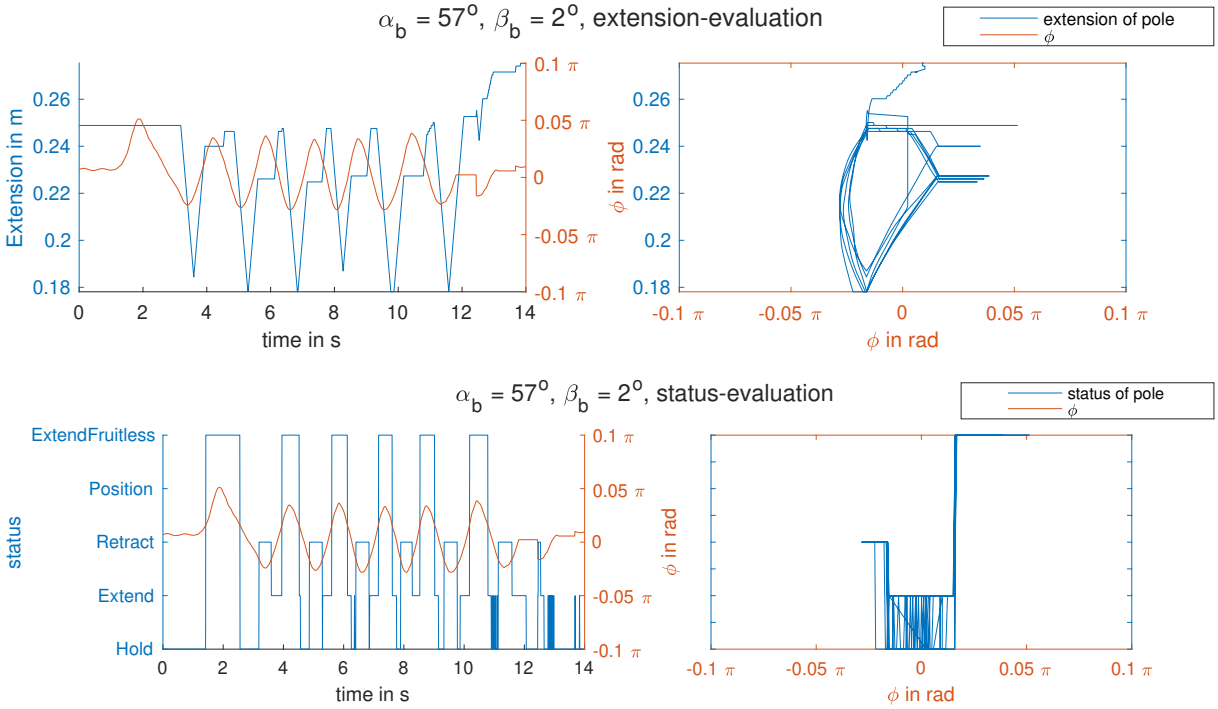



**Figure 6.18:** Balancing evaluation. Above: Estimated length. Below: Status of the pole. Both: $\phi$. For this experiment, the $l_{\text{balance}}$ was calculated too short due to a too-small $r_m$.

**Figure 6.19:** Evaluation of balancing with too small evaluated $l_{\text{balance}}(\theta)$ and $r_m - r_s$. Left: minimum $\phi$. Middle: optimal $\phi$. Right: maximum $\phi$.

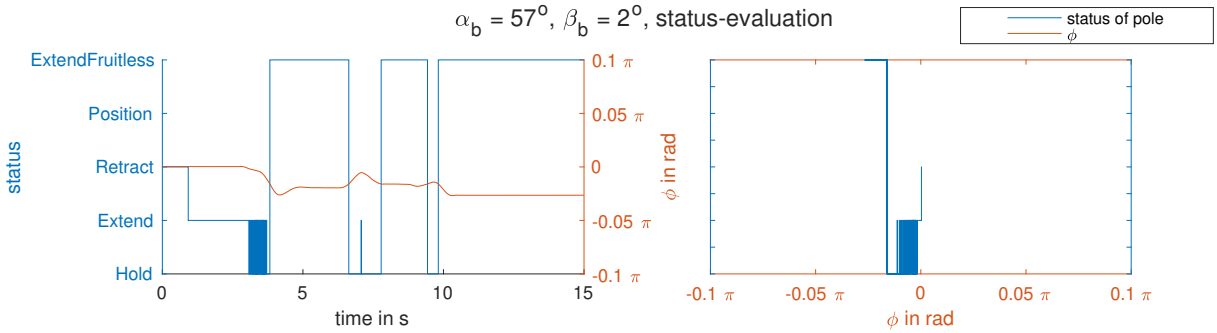



**Figure 6.20:** Balancing evaluation of the complete system with no locomotion.

**(a)** **(b)**

**Figure 6.21:** Left: Leverage approach used with the full sphere prototype. No rotation was initiated. Right: End-status of the push-only locomotion of the full robot with shell. The poles do not have enough force to generate rotation at this point.

## 6.3 Complete System

We made all previous evaluations with the cylindrical prototype. The complete system with the spherical shell shows bad to no controllability and overall unusable behavior. We will discuss the reasons for this, but not re-evaluate each previous experiment.

The only successful demonstration is the pure balancing. Figure 6.20 shows the results of a balancing procedure. The roll error is kept under $0.05\pi$ rad.

For locomotion, we again differ between push-only, leverage-only, and push and leverage approach. Using only leverage does not lead to any movement of the sphere, as the weight is significantly increased compared with the smaller prototype due to the shell. Therefore, the extended poles do not generate enough torque. For the sake of completeness, Figure 6.21a shows the non-changing state. The push-only approach leads to a slow rotation but only until the point where a certain $\phi$ is reached at the same time where one rod is more extended than its pendant on the other disc and starts bending. In this case, the rotation does not proceed

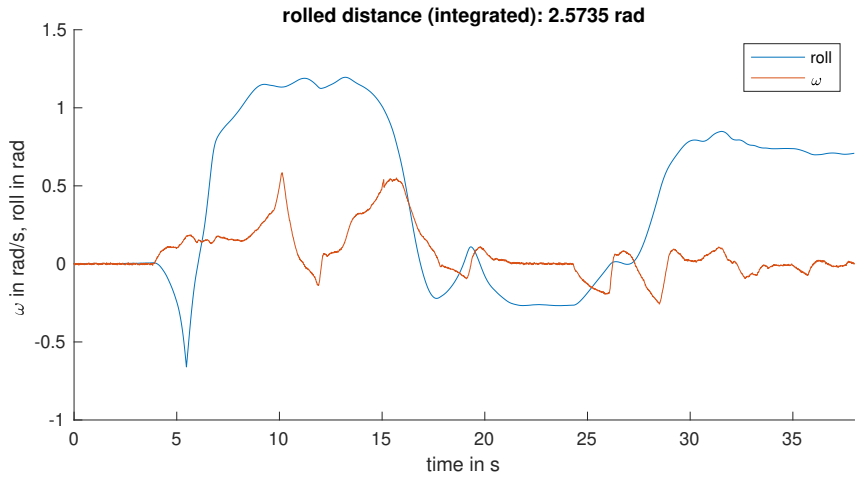



**Figure 6.22:** Push only approach with the full spherical prototype.

as the rods on the lower side do not have enough power to push against the one on the high side. Therefore, the force of those more extended poles leads to more roll and it then stalls at a certain point. Figure 6.21b shows this state. It does not depend on a certain roll angle but more on the described certain case, which in all test runs occurred within the first rotation, but at different roll angles. To stress this, Figure 6.22 shows the $\omega$ and roll angle. The roll angle at the end, at which the stall happens, is not the maximum as the robot reached higher values before, which are empirically confirmed.

With pushing and leverage, this point is overcome as enough torque is generated. Unluckily, this leads to further rotation in the direction in which the push-only experiments stall and lead to a complete roll onto this side, way over the $\phi_{\text{catastrophic}}$. This also interacts with the outstanding parts on the side of the sphere, which are used to tighten the two sphere halves. However, even without the rim, this leads to unusable, uncontrollable behavior. Figure 6.23 shows the sequence of these events.

These four evaluations, the working balancing, the not strong enough pushing algorithm and leverage algorithm, and the strong enough but overturning push and leverage approach, lead to the evaluation of balancing with push and leverage locomotion. Figure 6.24 and Figure 6.25 show two test runs of the experiment.

The balancing prevents the roll from getting out of control, but in both cases, after less than 0.25 rad, they get to a point where the systems stall. However, in contrast to the push only, this is an active situation. The sphere starts pushing itself into a not acceptable roll. Therefore the balancing algorithm retracts the pole on the higher side. The pole on the lower side still pushes, as the balancing algorithm as well as the locomotion instructs this. The upper parts of the evaluations show this pole. It tries to extend constantly in the second half of the experiments. However, this extension is not enough to initiate rotation. As the pole on the other side extends again to bring up enough force to start the rotation, it just pushes the sphere again into the roll angle. In the first run (Figure 6.24), this leads to a constant oscillation of roll and pitch. This is not a clear oscillation in the second run(Figure 6.25) but still the same sequence. Therefore, the ground laying problem of the push-only algorithm is that two poles on a tipped over side do


**Figure 6.23:** Push and Leverage locomotion approach for the full robot with shell on flat ground.

not have enough power to start rotation and turn the sphere upwards again sufficiently.

As the small prototype showed an improved behavior on gravelly ground as the poles transmit force more directly in comparison with the flat ground, the full sphere is also tested on the same ground. With the full sphere, there is no real improvement. The push and leverage algorithm with stabilization still has not enough power to start rotation and stalls. Figure 6.26 shows this state. With stabilization, the same oscillating sequence as on the flat ground happens, leading to the same shortcomings of the prototype.

Nevertheless, the uneven ground outside, together with the imbalance of the spheres shell, gives the opportunity to place and orient the sphere empirically by hand such that just minimal force is enough to start rotation. Therefore, the resulting test run gives no generally valid statement for the behavior of the sphere but gives an outlook of possible behavior if the poles have enough strength. Right after starting the rotation, the acceleration is too fast for the poles, so they extend without ground contact. The imbalance leads to a left curve of the robot, which is uncompensable as at the crucial moment, no rod can perform stabilization. The sphere leaves the $\phi_{\text{catastrophic}}$ range shortly after but has performed a yaw change of nearly $0.5\pi$ rad. This shows that the poles on the front of the rotation need to slow down too fast accelerations and speeds as no stabilization is possible otherwise. Figure 6.27 shows the sequence of described events. In picture three of the figure, the pole responsible for the stabilization is visible, extending into the air with a roll angle, at which stabilization is difficult. This stresses the point of the

possible usage of VPIP, described in Subsection 4.3.2, as it uses poles in the front and back of the rotation. This requires variable speed poles.

The last point that the outdoor experiments showed is the damage to the poles increases. When extending but there is no rotation of the robot, they do not glide over the ground as they anchor into the ground. This leads to more damage to the poles. Figure 6.28 shows two initially undamaged poles after approximately 30 seconds of the oscillating state of the stabilization and locomotion test on the gravelly ground. We did not discern this kind of damage on the flat ground experiments.

## 6.4 Summary

The ground-laying algorithms and approaches for locomotion work if evaluated isolated from stabilizing on the cylindrical robot. For the cylindrical disc prototype, this is the case for all three approaches (leverage only, push, only, push and leverage). The rotation speed is rather variable due to the flexibility of the rods and the lack of variable extension and retraction speeds. Also, the rotation acceleration is highly dependent on the chosen ground. The used poles show a low acceptance of external force applied to them, which leads to breaking the poles once the sphere rolls onto extended poles. In some cases, the poles break just by their own extending force.

The basic balancing algorithm works for the unstable disc setup and for the full spherical setup. Here, the pole strength is enough to provide controllability of the roll angle. Therefore, the end-roll angle is stable and not oscillating.

The evaluation of the locomotion for the full spherical setup shows no movement of the leverage-only approach as the weight is significantly higher with the spherical shell than for the disc setup. With the push-only approach, the sphere pushes itself into a stall position. When adding the leverage approach, the prototype overcomes the stall position but goes directly into a non-compensable roll, leading to the overall failure of the locomotion.

Using the stabilization and push and leverage approach avoids this non-compensable roll angle. Still, as the poles on a single side are not strong enough to lead to rotation, the robot oscillates at a certain state, as pushing on the higher (due to roll) side is needed for locomotion but leads to too large roll.

Therefore, we assume the following points as the main reasons for unsuccessful implementation of the full spherical prototype in order of significance :

- Too low power of the poles.
- Imperfect shape and mass distribution of the shell.
- Too high mass of the whole robot.
- Too high flexibility of the poles.
- Lack of feedback of extension of poles.

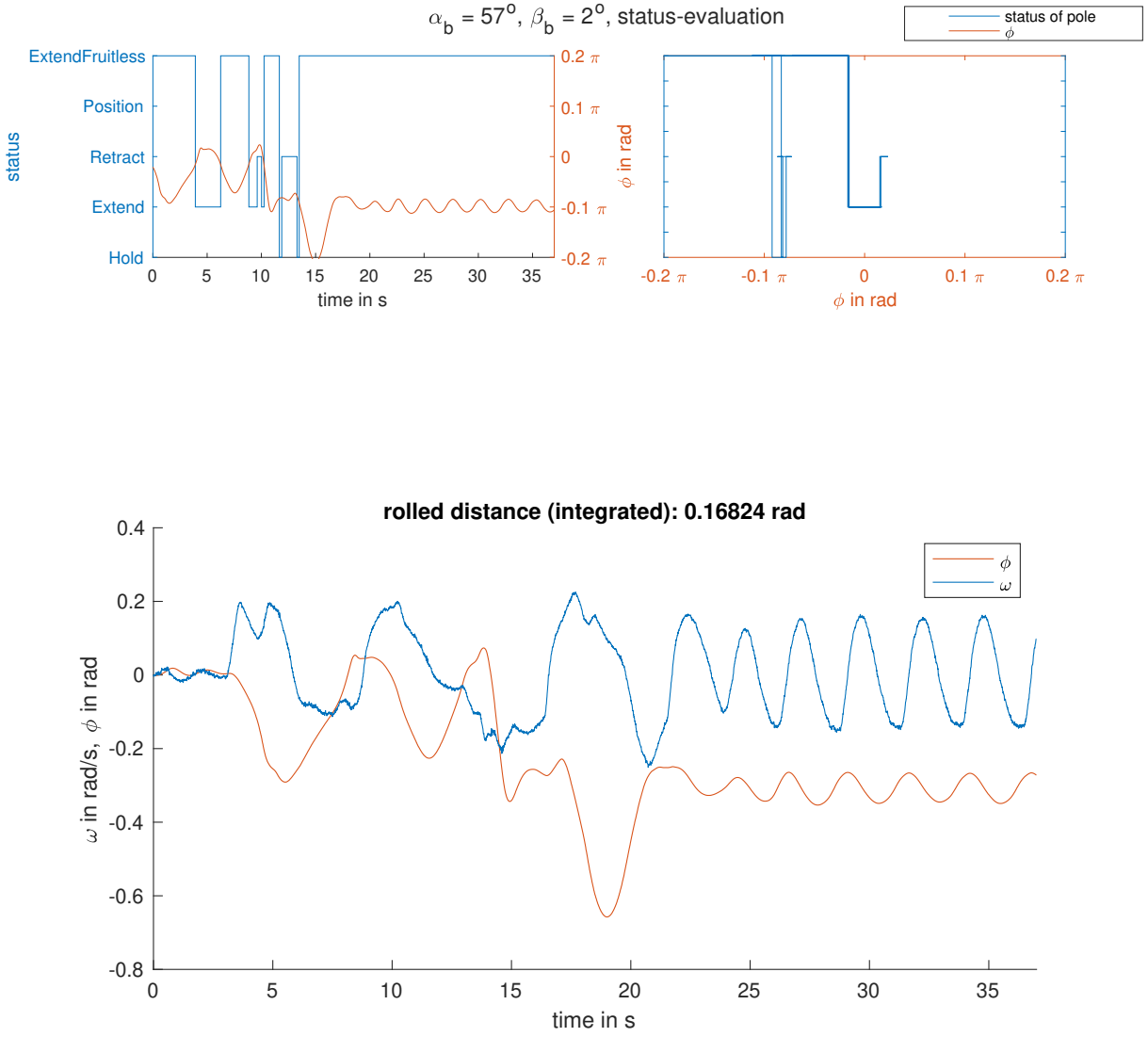



**Figure 6.24:** First combined locomotion and balancing of the full spherical prototype.

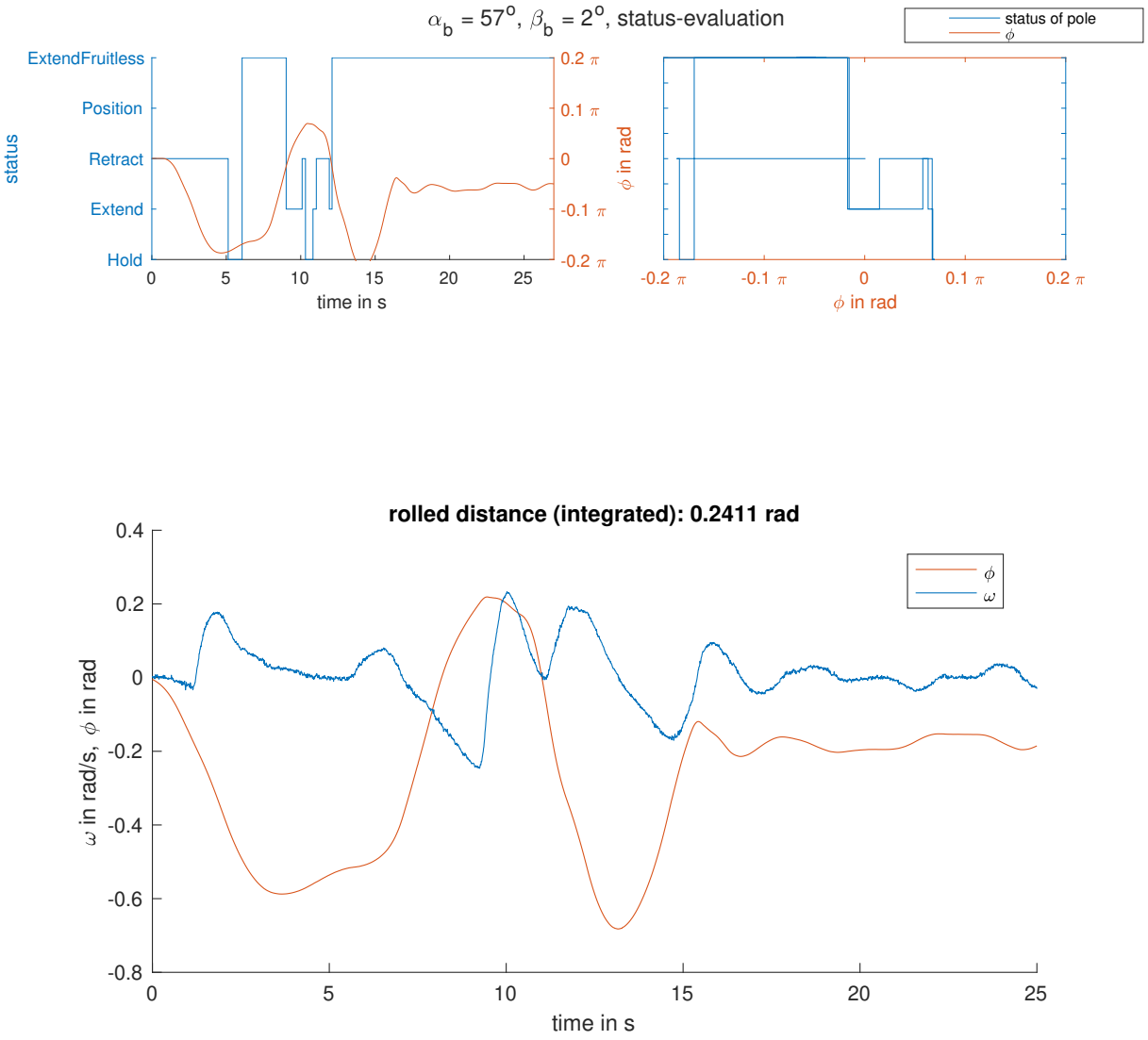



**Figure 6.25:** Second combined locomotion and balancing of the full spherical prototype.

**Figure 6.26:** Endpose of the sphere outside with the push and leverage approach and stabilizing. The force of the poles is not enough for further rotation.

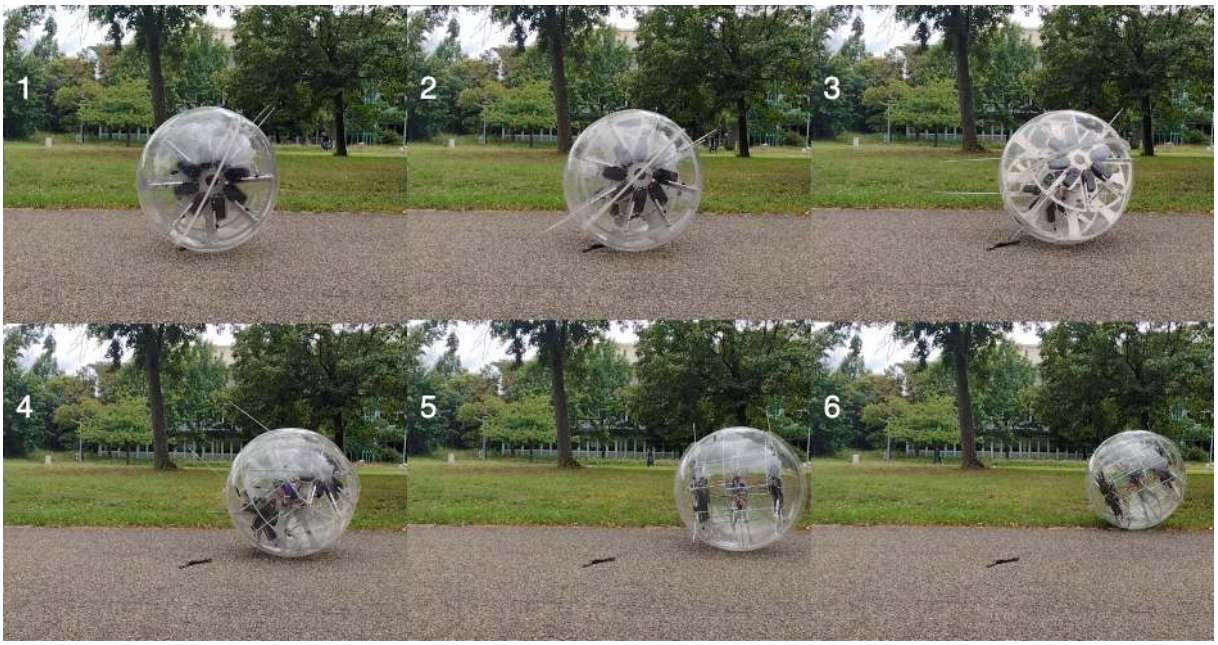



**Figure 6.27:** Push and Leverage locomotion approach for the full robot with shell on gravelly ground. The starting position of the sphere was optimized empirically by hand as the start position exploits the imbalance of the sphere and the not optimally flat ground to its advantage.

**Figure 6.28:** Deformed poles after pushing on gravelly ground. The deformation happened due to their own force.

# Chapter 7

# Conclusions

This thesis introduced the TLDR robot. Based on the DAEDALUS project, the general scope of application and the overall advantages of this robot were identified. The mathematical and physical models were derived, showing the capabilities and limitations of such a robot in terms of locomotion and balancing. For both aspects, the formulas provide help for dimensioning this and similar robots, as well as possible speeds, slopes, and obstacles. All formulas leave plenty of room for fine-tuning for a specific robot. The concept of VPIP was introduced as a control strategy for the TLDR robot with a virtual plane. We also gave an outlook on VPIM, which extends the virtual plane to a map. The prototype was designed, described, built, and tested for various scenarios. The prototype shows the feasibility of the concepts but has huge limitations due to the sub-optimally chosen actuators. This often leads to outcomes of experiments due to peculiarities of the actuators rather than conceptual influences.

The movement of the TLDR robot has more potential with variable speed poles, as there are many constraints regarding the mono-speed problem. Then, a fruitful combination of the pushing and leverage approach is possible. The described and calculated braking techniques of the robot often stress the poles if mono-speed poles are used. This needs to be improved for those specific poles in order to enable efficient and material-friendly braking. The slope estimation gives limits of the possible inclination, which were later met by the evaluations. The obstacle evaluation shows new approaches to overcome obstacles and gives the TLDR robot, in theory, a wide range of locomotion possibilities. Strong enough actuators are necessary to verify this as the built prototype did not give any chance of obstacle evaluation. Balancing showed expected good behavior despite using basic control structures. For combining balancing and locomotion, we introduced the idea of a virtual plane controlling the robot, the virtual pose instruction plane (VPIP), and established a mathematical solution. Primary calculations indicate its suitability as a control method. A possible control law was described, but we assume it to highly depend on the specific robot. The concept introduces two controllable parameters (roll and pitch of the plane), whereas the other concepts for combining balancing and locomotion have no real parameters that can be used for controlling and are just simple algorithms. We then extended this idea to include not a plane, but a whole map of the near environment, leading to the virtual pose instruction map (VPIM). Open points were analyzed and described, as well as the overall idea of generating this map in situ using LiDAR sensors. This leads to the fruitful symbiosis of

LiDAR sensors as a scientific payload and sensors for the control mechanism. This opens huge possibilities for overall robot systems, as they do not have to plan sensors for control tasks and additional payload sensors. Both domains, the payload and robot itself, benefit as this double-usage opens multiple possibilities of using the gained space inside the robot: for redundancy of the sensor by adding further LiDAR-sensors, for optimizing the external dimensions of the robot, which may enable other routes, or for adding other mechanisms supporting the robot and therefore scientific data generation.

The prototype was described in terms of its design decision, structure, and components. For the actuator, we discussed different solutions and approaches. The prototype relies on low-cost linear actuators, which turned out to be a sub-optimal choice, as often peculiarities and characteristics of the actuators defined behaviors of the robot, and huge parts of the used code only deal with the peculiarities of the poles. The power was sufficient for the cylindrical prototype experiments, with the exception of the perpendicular obstacle experiments, as this was to be expected since it requires multiple times the power provided. For the full sphere, the single poles had too little power for locomotion. The main problems identified were the slow extension and retraction speed, as well as the most obstructive flexibility and low break-resistance of the poles. Also, the lack of feedback on the current pole length made controlling and evaluation difficult and imprecise.

Needless to say, a lot of work remains to be done. Further research needs to include a more robust version of the prototype, especially targeting the more stable and powerful actuators. With this, the evaluation of steady rotation speeds, which are not achievable with the given prototype due to the mono-speed actuators, becomes possible. Also, further research needs to investigate the combination with internal weight-shifting in more detail. The first step is to evaluate the already described calculated VPIP approach on a prototype and then target the open points described for VPIM. Also, the integration of a sensor platform, most likely LiDAR, into the prototype and the efficient symbiosis of both leaves much room for research. All in all, VPIM, if initial tests are successful, seems to be a promising approach for the TLDR to become a robust locomotion approach that makes good use of the given environment, which can then be used as a baseline especially for tough, unknown terrains.

# Bibliography


[1] National Aeronautics and Space Administration. Ammos: Call for ideas. https://ammos.nasa.gov/contributing/callforideas/. Accessed: 25-06-2021.

[2] National Aeronautics and Space Administration. Missions - category rovers - nasa solar system exploration. https://solarsystem.nasa.gov/missions/?order=launch_datedesc&per_page=50&page=0&search=&fs=&fc=151&ft=&dp=&category=151, Jun 2019. Accessed: 25-06-2021.

[3] European Space Agency. Esa: Current and future missions. https://www.esa.int/Enabling_Support/Operations/Current_and_future_missions. Accessed: 25-06-2021.

[4] European Space Agency. Sysnova. http://www.esa.int/Enabling_Support/Preparing_for_the_Future/Discovery_and_Preparation/SysNova2. Accessed: 25-05-2021.

[5] Adrian K Agogino, Vytas SunSpiral, and David Atkinson. Super ball bot-structures for planetary landing and exploration. *NASA Innovative Advanced Concepts (NIAC) Program*, 2018.

[6] H Vahid Alizadeh and Mohammad J Mahjoob. Quadratic damping model for a spherical mobile robot moving on the free surface of the water. In *2011 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pages 125–130. IEEE, 2011.

[7] J Alves and J Dias. Design and control of a spherical mobile robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 217(6):457–467, 2003.

[8] Rhodri Armour, Keith Paskins, Adrian Bowyer, Julian Vincent, and William Megill. Jumping robots: a biomimetic solution to locomotion across rough terrain. *Bioinspiration & biomimetics*, 2(3):S65, 2007.

[9] Rhodri H Armour and Julian FV Vincent. Rolling in nature and robotics: A review. *Journal of Bionic Engineering*, 3(4):195–208, 2006.

[10] Karl Johan Åström and Richard M Murray. *Feedback systems*. Princeton university press, 2010.

[11] Mohammad Awrangjeb. Effective generation and update of a building map database through automatic building change detection from lidar point cloud data. *Remote Sensing*, 7(10):14119–14150, 2015.

[12] Massimo Bandecchi, Bryan Melton, Bruno Gardini, and Franco Ongaro. The esa/estec concurrent design facility. *Proceedings of EUSEC*, 9:329–336, 2000.

[13] Christopher Batten and David Wentzlaff. Kickbot: A spherical autonomous robot. `http://www.princeton.edu/~wentzlaf/documents/Batten.2001.Class.Kickbot.pdf`, 2001.

[14] Shourov Bhattacharya and Sunil K Agrawal. Spherical rolling robot: A design and motion planning studies. *IEEE Transactions on Robotics and Automation*, 16(6):835–839, 2000.

[15] Stuart A Bowyer and Ferdinando Rodriguez y Baena. Dynamic frictional constraints in translation and rotation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2685–2692. IEEE, 2014.

[16] Jonathan Bruce, Ken Caluwaerts, Atil Iscen, Andrew P Sabelhaus, and Vytas SunSpiral. Design and evolution of a modular tensegrity robot platform. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3483–3489. IEEE, 2014.

[17] P-K Budig. The application of linear motors. In *Proceedings IPEMC 2000. Third International Power Electronics and Motion Control Conference (IEEE Cat. No. 00EX435)*, volume 3, pages 1336–1341. IEEE, 2000.

[18] JT Burwell and E Rabinowicz. The nature of the coefficient of friction. *Journal of Applied Physics*, 24(2):136–139, 1953.

[19] Richard Chase and Abhilash Pandya. A review of active mechanical driving principles of spherical robots. *Robotics*, 1(1):3–23, 2012.

[20] Brian Chemel, Edward Mutschler, and Hagen Schempf. Cyclops: Miniature robotic reconnaissance system. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 3, pages 2298–2302. IEEE, 1999.

[21] Lee-Huang Chen, Kyunam Kim, Ellande Tang, Kevin Li, Richard House, Edward Liu Zhu, Kimberley Fountain, Alice M Agogino, Adrian Agogino, Vytas Sunspiral, et al. Soft spherical tensegrity robot design using rod-centered actuation and control. *Journal of Mechanisms and Robotics*, 9(2), 2017.

[22] Wei-Hsi Chen, Ching-Pei Chen, Jia-Shiuan Tsai, Jackie Yang, and Pei-Chun Lin. Design and implementation of a ball-driven omnidirectional spherical robot. *Mechanism and Machine Theory*, 68:35–48, 2013.

[23] CL Clover and JE Bernard. Longitudinal tire dynamics. *Vehicle System Dynamics*, 29(4):231–260, 1998.

[24] Rotundus AB Corp. History of groundbot. `https://rotundus.se/history/`. Accessed: 06-06-2021.

[25] BR Fuller. Tensegrity. *Portfolio and Art News Annual Vol. 4*, 1961.

[26] James H Goldie, Kevin J Leary, and Michael J Gerver. High force electric linear motors. *SAE transactions*, pages 268–274, 1996.

[27] Shuxiang Guo, Shilian Mao, Liwei Shi, and Maoxun Li. Development of an amphibious mother spherical robot used as the carrier for underwater microrobots. In *2012 ICME International Conference on Complex Medical Engineering (CME)*, pages 758–762. IEEE, 2012.

[28] Aarne Halme, Torsten Schonberg, and Yan Wang. Motion control of a spherical mobile robot. In *Proceedings of 4th IEEE International Workshop on Advanced Motion Control-AMC'96-MIE*, volume 1, pages 259–264. IEEE, 1996.

[29] J Hermans. A symmetric sphere rolling on a surface. *Nonlinearity*, 8(4):493, 1995.

[30] J Hierrezuelo and C Carnero. Sliding and rolling: the physics of a rolling ball. *Physics Education*, 30(3):177, 1995.

[31] Peter C Hughes, Wayne G Sincarsin, and Kieran A Carroll. Trussarm - a variable-geometry-truss manipulator. *Journal of Intelligent Material Systems and Structures*, 2(2):148–160, 1991.

[32] Ian W Hunter, John M Hollerbach, and John Ballantyne. A comparative analysis of actuator technologies for robotics. *Robotics Review*, 2:299–342, 1991.

[33] TB Ivanova, AA Kilin, and EN Pivovarova. Controlled motion of a spherical robot of pendulum type on an inclined plane. In *Doklady Physics*, volume 63, pages 302–306. Springer, 2018.

[34] Yu L Karavaev and Aleksandr Aleksandrovich Kilin. Nonholonomic dynamics and control of a spherical robot with an internal omniwheel platform: Theory and experiments. *Proceedings of the Steklov Institute of Mathematics*, 295(1):158–167, 2016.

[35] Yury L Karavaev and Alexander A Kilin. The dynamics and control of a spherical robot with an internal omniwheel platform. *Regular and Chaotic Dynamics*, 20(2):134–152, 2015.

[36] Viktor Kaznov and Mattias Seeman. Outdoor navigation with a spherical amphibious robot. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5113–5118. IEEE, 2010.

[37] Festo AG & Co. KG. Video of bionicwheelbot. `https://www.festo.com/group/en/repo/assets/media/BionicWheelBot_HD_1280x720_sprachneutral.mp4`, 2018. Accessed: 22-07-2021.

[38] Seyedmeysam Khaleghian, Anahita Emami, and Saied Taheri. A technical survey on tire-road friction estimation. *Friction*, 5(2):123–146, 2017.

[39] Kyunam Kim. *On the locomotion of spherical tensegrity robots*. PhD thesis, UC Berkeley, 2016.

[40] Kyunam Kim, Lee-Huang Chen, Brian Cera, Mallory Daly, Edward Zhu, Julien Despois, Adrian K Agogino, Vytas SunSpiral, and Alice M Agogino. Hopping and rolling locomotion with spherical tensegrity robots. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4369–4376. IEEE, 2016.

[41] Won-Jong Kim, James H Goldie, Michael J Gerver, Jerome E Kiley, and John R Swenbeck. Extended-range linear magnetostrictive motor with double-sided three-phase stators. *IEEE Transactions on Industry Applications*, 38(3):651–659, 2002.

[42] Won-jong Kim and Ali Sadighi. A novel low-power linear magnetostrictive actuator with local three-phase excitation. *IEEE/ASME Transactions on mechatronics*, 15(2):299–307, 2009.

[43] Young-Min Kim, Sung-Su Ahn, and Y Lee. Kisbot: new spherical robot with arms. In *10th WSEAS International Conference on Robotics, Control and Manufacturing Technology, Hangzhou, China*, pages 63–67, 2010.

[44] Ralf Simon King. *BiLBIQ: A biologically inspired robot with walking and rolling locomotion*. Springer Science & Business Media, 2012.

[45] Yuusuke Koizumi, Mizuho Shibata, and Shinichi Hirai. Rolling tensegrity driven by pneumatic soft actuators. In *2012 IEEE International Conference on Robotics and Automation*, pages 1988–1993. IEEE, 2012.

[46] Atsushi Koshiyama and Kazuo Yamafuji. Design and control of an all-direction steering type mobile robot. *The International journal of robotics research*, 12(5):411–419, 1993.

[47] H Kwun and KA Bartels. Magnetostrictive sensor technology and its applications. *Ultrasonics*, 36(1-5):171–178, 1998.

[48] Yaxin Li, Shuxiang Guo, and Chunfeng Yue. Preliminary concept and kinematics simulation of a novel spherical underwater robot. In *2014 IEEE International Conference on Mechatronics and Automation*, pages 1907–1912. IEEE, 2014.

[49] Jiarong Lin, Chunran Zheng, Wei Xu, and Fu Zhang. R2live: A robust, real-time, lidar-inertial-visual tightly-coupled state estimator and mapping. *arXiv preprint arXiv:2102.12400*, 2021.

[50] Daliang Liu, Hanxv Sun, Qingxuan Jia, and Liangqing Wang. Motion control of a spherical mobile robot by feedback linearization. In *2008 7th World Congress on Intelligent Control and Automation*, pages 965–970, 2008.

[51] James Lux. Alternative way of shifting mass to move a spherical robot. *NTRS: NASA Technical Reports*, 2005.

[52] Saber Mahboubi, Mir Masoud Seyyed Fakhrabadi, and Ahmad Ghanbari. Design and implementation of a novel spherical mobile robot. *Journal of Intelligent & Robotic Systems*, 71(1):43–64, 2013.

[53] James L Meriam, L Glenn Kraige, and Jeff N Bolton. *Engineering mechanics: dynamics*. John Wiley & Sons, 2020.

[54] Saeed Moazami, Srinivas Palanki, and Hassan Zargarzadeh. Kinematics of norma, a spherical robot, rolling over 3d terrains. In *2019 American Control Conference (ACC)*, pages 1330–1335. IEEE, 2019.

[55] Puyan Mojabi et al. Introducing august: a novel strategy for an omnidirectional spherical rolling robot. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 4, pages 3527–3533. IEEE, 2002.

[56] Ranjan Mukherjee. Spherical mobile robot, 2001. US Patent 6,289,263.

[57] Ranjan Mukherjee, Mark A Minor, and Jay T Pukrushpan. Simple motion planning strategies for spherobot: a spherical mobile robot. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, volume 3, pages 2132–2137. IEEE, 1999.

[58] Vijay Muralidharan and Arun D Mahindrakar. Geometric controllability and stabilization of spherical robot dynamics. *IEEE Transactions on Automatic Control*, 60(10):2762–2767, 2015.

[59] Balazs Nagy and Csaba Benedek. Real-time point cloud alignment for vehicle localization in a high resolution 3d map. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[60] David Newton, Ephrahim Garcia, and Garnett C Horner. A linear piezoelectric motor. *Smart Materials and Structures*, 7(3):295, 1998.

[61] Harry Duong Nguyen, Gim Soh, Shaohui Foong, and Kristin Wood. De-coupled dynamics control of a spherical rolling robot for waypoint navigation. In *2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 562–567, 11 2017.

[62] A Pinto and M Fiolhais. Rolling cylinder on a horizontal plane. *Physics Education*, 36(3):250, 2001.

[63] Deepak Pokhrel, Nutan Rajm Luitel, Sukanta Das, and Dip Narayan Ray. Design and development of a spherical robot (spherobot). In *Proceedings of the 1st International and 16th National Conference on Machines and Mechanisms (iNaCoMM2013), IIT Roorkee, India*, pages 735–741, 2013.

[64] Zhan Qiang, Liu Zengbo, and Cai Yao. A back-stepping based trajectory tracking controller for a non-chained nonholonomic spherical robot. *Chinese Journal of Aeronautics*, 21(5):472–480, 2008.

[65] Jia Qingxuan, Zheng Yili, Sun Hanxu, Cao Hongyu, and Li Hongyi. Motion control of a novel spherical robot equipped with a flywheel. In *2009 International Conference on Information and Automation*, pages 893–898. IEEE, 2009.

[66] Ingo Rechenberg, Ulrich Berg, Joachim Berg, Ivan Santibanez-Koref, Heinrich Frontzek, Elias Knubben, and Mart Moerdijk. Bionicwheelbot - walk and roll like a flic-flac spider. `https://www.festo.com/net/SupportPortal/Files/492805/Festo_BionicWheelBot_en.pdf`, 2018. Accessed: 22-07-2021.

[67] Giulio Reina, Mario Foglia, Annalisa Milella, and Angelo Gentile. Rough-terrain traversability for a cylindrical shaped mobile robot. In *Proceedings of the IEEE International Conference on Mechatronics, 2004. ICM'04.*, pages 148–153. IEEE, 2004.

[68] Angelo Pio Rossi, Francesco Maurelli, Vikram Unnithan, Hendrik Dreger, Kedus Mathewos, Nayan Pradhan, Dan-Andrei Corbeanu, Riccardo Pozzobon, Matteo Massironi, Sabrina Ferrari, Claudia Pernechele, Lorenzo Paoletti, Emanuele Simioni, Pajola Maurizio, Tommaso Santagata, Dorit Borrmann, Andreas Nüchter, Anton Bredenbeck, Jasper Zevering, Fabian Arzberger, and Camilo Andres Reyes Mantilla. Daedalus - descent and exploration in deep autonomy of lava underground structures. Technical Report 21, Institut für Informatik, 2021.

[69] Andrew P Sabelhaus, Jonathan Bruce, Ken Caluwaerts, Yangxin Chen, Dizhou Lu, Yuejia Liu, Adrian K Agogino, Vytas SunSpiral, and Alice M Agogino. Hardware design and testing of superball, a modular tensegrity robot. In *The 6th World Conference of the International Association for Structural Control and Monitoring (6WCSCM)*, 2014.

[70] Aravind Seeni, Bernd Schafer, Bernhard Rebele, and Nikolai Tolyarenko. Robot mobility concepts for extraterrestrial surface exploration. In *2008 IEEE Aerospace Conference*, pages 1–14. IEEE, 2008.

[71] Mizuho Shibata, Fumio Saijyo, and Shinichi Hirai. Crawling by body deformation of tensegrity structure robots. In *2009 IEEE international conference on robotics and automation*, pages 4375–4380. IEEE, 2009.

[72] Andrea Spaggiari, Igor Spinella, and Eugenio Dragoni. Design of a telescopic linear actuator based on hollow shape memory springs. *Journal of materials engineering and performance*, 20(4):489–496, 2011.

[73] K Spanner. Survey of the various operating principles of ultrasonic piezomotors. In *Proceedings of the 10th International Conference on New Actuators*, 2006.

[74] Shigeki Sugano, Qiang Huang, and Ichiro Kato. Stability criteria in controlling mobile robotic systems. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, volume 2, pages 832–838. IEEE, 1993.

[75] Yuuta Sugiyama and Shinichi Hirai. Crawling and jumping of deformable soft robot. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 4, pages 3276–3281. IEEE, 2004.

[76] Yoichi Tatara. Mechanochemical actuators. *Advanced Robotics*, 2(1):69–85, 1987.

[77] J.L. Tate. Toy, 1893. US Patent 508,558.

[78] Filip Tomik, Shahin Nudehi, Louis L Flynn, and Ranjan Mukherjee. Design, fabrication and control of spherobot: a spherical mobile robot. *Journal of Intelligent & Robotic Systems*, 67(2):117–131, 2012.

[79] Josep M Mirats Tur and Sergi Hernàndez Juan. Tensegrity frameworks: Dynamic analysis review and open problems. *Mechanism and Machine Theory*, 44(1):1–18, 2009.

[80] Nathan S Usevitch, Zachary M Hammond, and Mac Schwager. Locomotion of linear actuator robots through kinematic planning and nonlinear optimization. *IEEE Transactions on Robotics*, 36(5):1404–1421, 2020.

[81] Keith W Wait, Philip J Jackson, and Lanny S Smoot. Self locomotion of a spherical rolling robot using a novel deformable pneumatic method. In *2010 IEEE International Conference on Robotics and Automation*, pages 3757–3762. IEEE, 2010.

[82] Wanjun Wang and Ilene Busch-Vishniac. A high precision micropositioner based on magnetostriction principle. *Review of scientific instruments*, 63(1):249–254, 1992.

[83] Thomas Wiemann, Andres Nüchter, Kai Lingemann, Stefan Stiene, and Joachim Hertzberg. Automatic construction of polygonal maps from point cloud data. In *2010 IEEE Safety Security and Rescue Robotics*, pages 1–6. IEEE, 2010.

[84] Maotao Yang, Tianping Tao, Jianwen Huo, Konstantin A Neusypin, Zikun Zhang, Hua Zhang, and Mingming Guo. Design and analysis of a spherical robot with two degrees of freedom swing. In *2020 Chinese Control And Decision Conference (CCDC)*, pages 4913–4918. IEEE, 2020.

[85] Tomi Ylikorpi and Jussi Suomela. Ball-shaped robots. In *Climbing & Walking Robots, Towards New Applications*, pages 235–256. INTECH Open Access Publisher, 2007.

[86] Joong-Cheol Yoon, Sung-Su Ahn, and Yun-Jung Lee. Spherical robot with new type of two-pendulum driving mechanism. In *2011 15th IEEE International Conference on Intelligent Engineering Systems*, pages 275–279. IEEE, 2011.

[87] Jasper Zevering, Anton Bredenbeck, Fabian Arzberger, Dorit Borrmann, and Andreas Nüchter. Luna-a laser-mapping unidirectional navigation actuator. In *Experimental Robotics: The 17th International Symposium*, page 85. Springer Nature, 2020.

[88] Jasper Zevering, Anton Bredenbeck, Fabian Arzberger, Dorit Borrmann, and Andreas Nüchter. Imu-based pose-estimation for spherical robots with limited resources. In *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2021. [Manuscript accepted for publication].

[89] Qiang Zhan, Yao Cai, and Caixia Yan. Design, analysis and experiments of an omni-directional spherical robot. In *2011 IEEE International Conference on Robotics and Automation*, pages 4921–4926. IEEE, 2011.

[90] Wei Zhao, Han Xu Sun, Qing Xuan Jia, Yan Heng Zhang, and Tao Yu. Mechanical analysis of the jumping motion of a spherical robot. In *Advanced Materials Research*, volume 591, pages 1457–1460. Trans Tech Publ, 2012.

# Appendix A

# Code

**Listing A.1:** Matlab simulation for rod extension

```
clear all;
close all;

alpha = 57;%deg
beta = 2;%deg
maximumLength = 3;%m
radius = 2;%m
speed=20; %m/s (maximum speed of extension and retraction)
minspeed=0; %m/s (minimum speed of extension and retraction)
%desired omega, for this simultaion omega will be constant if achivable by
%pushing
omega =1;%rad/s
tEnd=8; %s (simulation duration)

%%%%%%%%%%%%%%%%%%%%%%%%%%
hz = 100;
zeta = 0;%current angle
if(minspeed>speed)
    minspeed=speed;
end
speeddt = speed/hz;
minspeeddt = minspeed/hz;
gamma= 0;
epsilon=0;
l=0;%currentlength
alpha = alpha/360*2*pi;
beta = beta/360*2*pi;

```

```
29
30
31
32 for(t=1:tEnd*hz)
33
34     gamma= getNextGamma(speed, radius, omega);
35     epsilon = getNextEpsilon(gamma, omega, radius, maximumLength, speed);
36
37     l=getNextLength(l,zeta,speeddt,minspeeddt,speed, maximumLength, ...
           alpha, beta,epsilon, gamma, radius, omega);
38
39     zeta= getNextZeta(zeta, omega, hz);
40
41     lengthArray(t)=l;
42     zetaArray(t)=zeta;
43
44    %plot maximum possible extension. Negative(to the sky) are suppresed
45     realArray(t)=radius/cos(2*pi-zeta)-radius;
46     if(realArray(t)<0||realArray(t)>maximumLength)
47         realArray(t)=nan;
48     end
49
50
51
52 end
53
54
55 %just plotting
56 time= linspace(0,tEnd,tEnd*hz);
57
58 t=tiledlayout(1,2);
59 t.TileSpacing = 'compact';
60 nexttile
61 hold on;
62 yyaxis left
63 plot(time,lengthArray);
64 plot(time,realArray);
65 ylabel('Extension i m')
66 yyaxis right
67 plot(time,zetaArray);
68 ylim([0 2*pi]);
69 ylabel('\zeta in rad')
70 yticks([0 (0.5*pi) pi (1.5*pi) (2*pi)])
71 yticklabels({'0','0.5 \pi','\pi','1.5 \pi','2 \pi'})
```

```
72 xlabel('time in s')
73
74
75 legend({'extension of pole','possible ...
       extension','\zeta'},'Position',[0.825 0.74 0.17 0.2])
76
77 nexttile;
78
79 polarplot(zetaArray,lengthArray,'Color',[ 0 0.4470 0.7410]);
80 hold all;
81 polarplot(zetaArray,realArray, 'Color',[0 0.4470 0.7410],'LineStyle', '--');
82
83 ax = gca;
84 hold(ax)
85 d = ax.ThetaDir;
86 ax.ThetaDir = 'clockwise';
87 ax.ThetaZeroLocation = 'bottom'
88 thetaticks([0 90 180 270])
89 thetaticklabels({'0','0.5 \pi','\pi','1.5 \pi'})
90
91 rticks([0 0.5*maximumLength maximumLength])
92 rticklabels({'', 'half extension', 'full extension'})
93 rtickangle(45)
94 ax.RAxisLocation = 135;
95 set(gca,'thetacolor',[0.8500 0.3250 0.0980]);
96 set(gcf, 'Position', [100, 100, 700, 250]);
97
98 title(t,("\alpha = "+(alpha/2/pi*360)+"^o, "+"\beta = ...
       "+(beta/2/pi*360)+"^o, "+"\omega = "+omega+"rad/s, "+"extension ...
       speed = "+speed +"m/s, "+"radius = "+ radius+"m"+", l_{max} = "+ ...
       maximumLength+"m"))
99 saveas(gcf,"exports/alpha="+(alpha/2/pi*360)+"beta="+(beta/2/pi*360)...
100    +"omega="+omega+"speed="+speed+"maximumLength"+maximumLength+".fig");
101 saveas(gcf,"exports/alpha="+(alpha/2/pi*360)+"beta="+(beta/2/pi*360)...
102    +"omega="+omega+"speed="+speed+"maximumLength"+maximumLength,'epsc');
103
104
105
106
107
108
109 function l = getNextLength(length, zeta,speeddt,minspeeddt,speed, ...
       maximumLength, alpha, beta ,epsilon, gamma, radius, omega)
110   %pushing sector
```

```
111 if(zeta≥beta && zeta<alpha)
112
113     nextlength = radius / cos(zeta) - radius;
114     %is this possible with max speed?
115     if(abs(nextlength-length)>speeddt)
116         %at this moment the pole can not hold up with the given omega
117         % now you can just extend at full speed
118         nextlength= length + speeddt;
119         disp('This omega is not possible with the given parameters')
120         % or you can retract, as the pole does not help for the rotation
121         nextlength= retractAFAP(length,speeddt);
122
123     end
124     %is this possible with maximum length?
125     if(nextlength>maximumLength)
126         nextlength=maximumLength;
127     end
128
129     l=nextlength;
130     return;
131 end
132
133
134 %leverage sector
135 %first check if we can just skip everything else
136
137 touchpoint = 2 * pi - acos(radius / (radius + maximumLength ));
138
139 if(speed≥abs(radius*omega*tan(touchpoint)*sec(touchpoint)))
140
141     %if here, no need for gamma and epsilon check
142     %just check if extension or slow retraction
143
144
145     if(zeta>pi && zeta<touchpoint)
146     l= extendAFAP(length,speeddt, maximumLength);
147     return;
148     end
149
150     if(zeta<pi && zeta<touchpoint)
151     l =retractAFAP(length,speeddt);
152     return;
153     end
154
```

```
155
156
157      nextlength = radius / cos(2*pi-zeta) - radius;
158     %is this possible with max speed?
159     if(abs(nextlength-length)<minspeeddt)
160         nextlength= length - minspeeddt;
161     end
162     %is this possible with maximum length?
163     if(nextlength<0)
164         nextlength=0;
165     end
166     if(nextlength>maximumLength)
167         nextlength=maximumLength;
168     end
169
170     l=nextlength;
171     return;
172
173
174
175
176
177 elseif (zeta≥gamma)
178     %if we are above gamma we are fast enough for retraction not AFAP
179
180      nextlength = radius / cos(2*pi-zeta) - radius;
181     %is this possible with max speed?
182     if(abs(nextlength-length)<minspeeddt)
183         nextlength= length - minspeeddt;
184     end
185      if(abs(nextlength-length)>speeddt)
186         nextlength= length - speeddt;
187     end
188     %is this possible with maximum length?
189     if(nextlength<0)
190         nextlength=0;
191     end
192     if(nextlength>maximumLength)
193         nextlength=maximumLength;
194     end
195
196     l=nextlength;
197     return;
198
```

```
199 elseif (zeta>epsilon)
200     %retraxct as fast as possible to be at gamma at time
201     l =retractAFAP(length, speeddt);
202     return;
203 elseif(zeta > epsilon-omega*maximumLength/speed && zeta >alpha)
204     %extend for elevarage maybe before pi
205     %the condition zeta>alpha checks that pushing preferenced before
206     %levarage
207     l= extendAFAP(length,speeddt, maximumLength);
208     return;
209 end
210
211 %start extending for levaraging, if we are over pi or if we need to extend
212 %on the right side to meet gamma
213 if(zeta>pi)
214     l= extendAFAP(length,speeddt, maximumLength);
215     return;
216 end
217
218
219  % if non of the conditions is met, just retract.
220 l =retractAFAP(length,speeddt);
221
222
223 end
224
225 function lret= retractAFAP(length,speeddt)
226
227 lret= length-speeddt;
228 if(lret<0)
229     lret=0;
230 end
231
232 end
233
234 function lret= extendAFAP(length, speeddt, maximumLength)
235
236 lret= length+speeddt;
237 if(lret>maximumLength)
238     lret=maximumLength;
239 end
240
241 end
242
```

```
243
244
245 function gR= getNextGamma(speed, radius, omega)
246
247 a= -speed/(radius*omega);
248 gR =pi + 2*atan(1/2*sqrt(1/a^2 + 4) + sqrt(1/a^2 - 4/(a*sqrt(1/a^2 + 4)) ...
        - 1/(a^3*sqrt(1/a^2 + 4)))/sqrt(2) - 1/(2*a));
249
250 gR= abs(gR);
251 end
252
253 function z= getNextZeta(zeta, omega, hz)
254
255 z=zeta+omega/hz;
256 if(zeta≥2*pi)
257     z=0;
258 end
259 end
260
261 function eps= getNextEpsilon(gamma, omega, radius, maximumLength, speed)
262
263 if(maximumLength-(radius/(cos(2*pi-gamma))-radius)≤0)
264     %gamma is capable of retracting. For saftey, should not occur due to
265     %separate condition check
266     eps=gamma
267     return;
268 end
269
270 eps = gamma - omega ...
        *(maximumLength-(radius/(cos(2*pi-gamma))-radius))/speed;
271
272 eps=abs(eps);
273
274 end
```

**Listing A.2:** Omega calculataion for rod extension simulation

```
function o = getOmega(zeta, alpha, beta, speed, radius, omega)

%calculate which pole is the relevant for locomotion in that moment
relevantZeta = mod(zeta-beta, pi/4) + beta

o= speed/(radius*tan(relevantZeta)*sec(relevantZeta));

%maximum acceleration of omega
maxAcc=1; % rad/s^2
if(o-omega>maxAcc/100)
    o= omega+maxAcc/100
end


end
```

**Listing A.3:** Arduino code for serial communication with pi and controlling relais

```
1
2
3
4  void setup() {
5     //initialize desired pins as output pins
6     for(int i =22; i≤53; i++){
7          pinMode(i, OUTPUT);
8          digitalWrite(i, HIGH);
9       }
10    //initalize serial-communication wih a baud rate of 9600
11   Serial.begin(9600);
12 }
13
14 void loop() {
15
16
17
18 //check if serial is received
19  if (Serial.available() > 0) {
20    //Read serial
21    String data = Serial.readStringUntil('\n');
22
23    //check if data is valid
24
25    if (validateMessage(data)){
26
27      //Read the command
28      char c = data.charAt(0);
29      // Read the side
30      int side = data.substring(2,3).toInt();
31      //Read actuator number
32      int number = data.substring(4,5).toInt();
33
34      // execute function corresponing to command
35      switch(c){
36
37        case 'E': extend(side, number);
38                  Serial.print("ACC extending \""+data+"\"\n");
39                  break;
40        case 'R': retract(side, number);
41                    Serial.print("ACC retracting \""+data+"\"\n");
42                  break;
```

```
43          case 'H': hold(side, number);
44                    Serial.print("ACC holding \""+data+"\"\n");
45                    break;
46          case 'X': resetAll();
47                    Serial.print("ACC resetting \""+data+"\"\n");
48                    break;
49          default: Serial.print("ERR unknown command \""+data+"\"\n");
50          }
51
52      }
53
54
55    }
56  }
57
58
59
60  //checks if the format is C_S_N with C=Command letter, S= side number, ...
        N= number
61  bool validateMessage(String s){
62
63    //check for length 5
64    if (s.length()!=5 ) {
65      Serial.print("ERR not valid, too short \""+s+"\"\n");
66      return false;}
67
68    //cheks for the underscores
69    String command = s.substring (0,1);
70    if (!s.substring(1,2).equals("_") || !s.substring(3,4).equals("_")) {
71          Serial.print("ERR not valid, underscores missing/wrong place ...
              \""+s+"\"\n");
72      return false;}
73
74
75    //check if substringing are in range (if 0 , then atoi failed, meaning ...
          there is no number)
76    if(!(s.substring(2,3).toInt()≥1 && s.substring(2,3).toInt()≤2 && ...
          s.substring(4,5).toInt()≥1 && s.substring(4,5).toInt()≤8)){
77       Serial.print("ERR not valid, out of range \""+s+"\"\n");
78
79      return false;
80      }
81
82    return true;
```

```
83   }
84
85   //extend actuator
86   void extend(int side, int number){
87
88     //get power and signal pin
89     int * i = numberToPin(side,number);
90
91     //energy: on, signal on:
92     digitalWrite(*i, LOW);
93     digitalWrite(*(i+1), LOW);
94
95
96
97   }
98
99  void retract(int side, int number){
100
101    //get power and signal pin
102    int * i = numberToPin(side,number);
103
104    //energy: on, signal off:
105  digitalWrite(*i, LOW);
106    digitalWrite(*(i+1), HIGH);
107
108  }
109
110  //retract all poles for 5 seconds
111  void resetAll(){
112
113      for(int side=1; side≤2; side++){
114        for(int number=1; number≤8; number++){
115          retract(side, number);
116        }
117      }
118      delay(5000);
119
120
121
122  }
123
124  void hold(int side, int number){
125
126       //get power and signal pin
```

```
127       int * i = numberToPin(side,number);
128
129       //energy: off, signal off:
130         digitalWrite(*i, HIGH);
131     digitalWrite(*(i+1), HIGH);
132
133    }
134
135
136     //returns for a given side and the number of the pole the ...
            correspondending pins of the power and the signal relai
137   int * numberToPin(int side, int number){
138      static int r[2];
139
140      // side 1 are all even, side 2 are all uneven (this is just )
141      // signal relai is 8 above the power relai
142      // example: side 1/pole 1: 22 is the power relai and 30 the signal ...
             relai
143      // example: side 2/ pole 3: 27 is the power relai and 35 the signal
144      r[0] = 20 + 2*number + (side-1);
145      r[1] = r[0]+8;
146
147      // return array with both values
148      return r;
149
150
151    }
```

**Listing A.4:** Code running on the prototype of the TLDRrobot

```
#include "TLDRrobot.h"


// Signal-safe flag for whether shutdown is requested
sig_atomic_t volatile g_request_shutdown = 0;


bool quiet=false, pushOnly=false, leverageOnly=false, noSerial=false, ...
    poleUpTest=false,printRoll=false, ...
    printPitch=false,calibrateRoll=false, calibratePitch=false, ...
    useAlgortihm=false, backwards=false, printLeverage=false, ...
    useStabilization=false;

float maximumAllowedLength=POLE_LENGTH*0.5;

// Replacement SIGINT handler
void mySigIntHandler(int sig) {
    g_request_shutdown = 1;
}

void imuCallback(const sensor_msgs::Imu::ConstPtr& msg) {


 qW=msg->orientation.w;
 qX=msg->orientation.x;
 qY=msg->orientation.y;
 qZ=msg->orientation.z;

wX=msg->angular_velocity.x;
wY=msg->angular_velocity.y;
wZ=msg->angular_velocity.z;

aX=msg->linear_acceleration.x;
aY=msg->linear_acceleration.y;
aZ=msg->linear_acceleration.z;

roll = atan2f(qW*qX + qY*qZ, 0.5f - qX*qX - qY*qY);
if(printRoll){ROS_INFO("Initial Roll is %f", roll);}
pitch = asinf(-2.0f * (qX*qZ - qW*qY));
yaw = atan2f(qX*qY + qW*qZ, 0.5f - qY*qY - qZ*qZ);

pitch +=PI12;
```

```
39 if(roll<0){
40 pitch= PI21-pitch;
41 }
42 pitch += compensatePitch; // compensate for 1;
43 if(pitch>PI21){
44
45 pitch-=PI21;
46 }
47 if(pitch <0){
48 pitch+=PI21;
49 }
50 pitch= PI21-pitch;
51
52
53 roll = fabs(roll);
54
55 roll= fmod(roll+compensateRoll,PI);
56
57 roll-=PI12;
58
59 if(printPitch){ROS_INFO("Pitch is %f",pitch);}
60 if(printRoll){ROS_INFO("Roll is %f", roll);}
61
62 //ROS_INFO("Omega is %f",wZ);
63 }
64 int nPoleManual =0; //number which pole is currently controlled
65 void joyCallback(const sensor_msgs::Joy::ConstPtr& msg){
66
67 if(msg->buttons[9]==1){
68 ROS_INFO("Starting algortihm");
69 useAlgortihm=true;
70 }
71 if(msg->buttons[8]==1){
72 ROS_INFO("Ending Algortihm");
73 useAlgortihm=false;
74 retractAll();
75 }
76
77 if(msg->buttons[5]==1){
78 if(useStabilization){
79 useStabilization=false;
80 retractAll();
81 ROS_INFO("Disabiling Stabilization");
82
```

```
83 }else{
84 useStabilization=true;
85 ROS_INFO("Activating Stabilization!");
86 
87 }
88 }
89 
90 if(msg->buttons[4]==1){
91 cw= msg->axes[1]*0.4;
92 if(cw<-0.01){backwards=true;}
93 else{backwards=false;}
94 }
95 
96 if(msg->axes[4]>0.9){
97 
98     if(!leverageOnly && !pushOnly){
99     ROS_INFO("Switching to: Push Only Mode");
100    pushOnly=true;
101    leverageOnly=false;
102    }else if(pushOnly){
103    ROS_INFO("Switching to: Leverage Only Mode");
104    pushOnly=false;
105    leverageOnly=true;
106    }else{
107    ROS_INFO("Switching to: Push and Leverage Mode");
108    leverageOnly=false;
109    pushOnly=false;
110    }
111 
112 
113 }
114 
115 if(msg->axes[4]<-0.9){
116 
117         if(!leverageOnly && !pushOnly){
118         ROS_INFO("Switching to: Leverage Only Mode");
119         leverageOnly=true;
120    pushOnly=false;
121         }else if(leverageOnly){
122         ROS_INFO("Switching to: Push Only Mode");
123         pushOnly=true;
124         leverageOnly=false;
125         }else{
126         ROS_INFO("Switching to: Push and Leverage Mode");
```

```
127         leverageOnly=false;
128         pushOnly=false;
129         }
130
131
132 }
133
134 if(msg->axes[5]>0.9){
135     nPoleManual= ++nPoleManual % 9;
136     ROS_INFO("Controlling pole %d",nPoleManual);
137     }
138
139 if(msg->axes[5]<-0.9){
140     nPoleManual=(nPoleManual-1+9)%9;
141     ROS_INFO("Controlling pole %d",nPoleManual);
142     }
143 if(nPoleManual!=0){
144
145     int start=0;
146     int end=1;
147
148     if(msg->buttons[6]==1){end=0;}
149     if(msg->buttons[7]==1){start=1;}
150
151     for(int i = start ;i≤end;i++){
152
153     if(msg->buttons[0]==1){
154         pole_arr[i][nPoleManual-1].state=Hold;
155         pole_arr[i][nPoleManual-1].manual=true;
156          pole_arr[i][nPoleManual-1].change=true;
157     }
158  if(msg->buttons[1]==1){
159                 pole_arr[i][nPoleManual-1].state=Extend;
160                 pole_arr[i][nPoleManual-1].manual=true;
161          pole_arr[i][nPoleManual-1].change=true;
162         }
163
164  if(msg->buttons[2]==1){
165                 pole_arr[i][nPoleManual-1].state=Retract;
166                 pole_arr[i][nPoleManual-1].manual=true;
167         pole_arr[i][nPoleManual-1].change=true;
168
169         }
170
```

```
171  if(msg->buttons[3]==1){
172                 pole_arr[i][nPoleManual-1].manual=false;
173
174         }
175     }
176
177 }
178 }
179
180
181 int init(){
182
183
184
185     if (wiringPiSetup () == -1) {
186         ROS_ERROR("Unable to start Serial Communication");
187     return -1 ;
188         }
189      if((fd=serialOpen(SERIAL_DEVICE,BAUD_RATE))<0){
190                 ROS_ERROR("Unable to open serial device: ...
                        %s\n",SERIAL_DEVICE);
191                 return -1;
192         }
193
194     ROS_INFO("Opening Serial Port sucessful");
195
196
197     //initiate every Pole
198     for(int i =0;i≤1;i++){
199         for(int j =0; j≤7;j++){
200
201             pole_arr[i][j]={0,i+1,j+1,Retract,false,false,0};
202         }
203     }
204
205     if(!noSerial){
206     serialPrintf(fd,"X_1_1\n");
207     delay(1000);
208      while (serialDataAvail (fd)) {
209             printf ("%c", serialGetchar(fd));
210          }
211     }
212
213     qW=2;
```

```
214     ros::Duration(0.5).sleep();
215     ros::spinOnce();
216     while(qW==2){
217
218     ROS_INFO("Waiting for Pose Data (External)");
219     ros::Duration(2).sleep();
220     ros::spinOnce();
221     }
222     ROS_INFO("Recieving Pose Data");
223
224
225
226     if(calibratePitch){
227
228     ROS_INFO("Zeroing, make sure Pole 1 points up");
229     ros::Duration(2).sleep();
230         ros::spinOnce();
231
232     pitch = asinf(-2.0f * (qX*qZ - qW*qY));
233     roll = atan2f(qW*qX + qY*qZ, 0.5f - qX*qX - qY*qY);
234
235     pitch +=PI12;
236     if(roll<0){
237     pitch= PI21-pitch;
238     }
239     compensatePitch = PI-pitch;
240     ROS_INFO("Zeroing done, compensation for pitch is %f", compensatePitch);
241     }
242     else{
243     ROS_INFO("Using 0,033 as pitch compensation, for calibration start ...
            program with -calibratePitch flag");
244     compensatePitch=0.033;
245
246     }
247     if(calibrateRoll){
248
249     ROS_INFO("Zeroing, make sure Robot is ad desired roll 0");
250         ros::Duration(2).sleep();
251         ros::spinOnce();
252
253         roll = atan2f(qW*qX + qY*qZ, 0.5f - qX*qX - qY*qY);
254
255
256     roll = fabs(roll);
```

```
257     roll -= PI12;
258     compensateRoll=-roll;
259
260
261     ROS_INFO("Zeroing done, compensation for Roll is %f", compensateRoll);
262     }else{
263     compensateRoll=0.0;
264     ROS_INFO("Using 0.00 PI as roll compensation, for calibration start ...
            program with -calibrateRoll flag");
265     }
266
267     //initial calculation of the touchangle
268     touchpoint = PI21 - acos(RADIUS / (RADIUS + POLE_LENGTH ));
269
270     return 1;
271 }
272
273
274
275
276 int main (int argc, char **argv)
277 {
278         ros::init(argc, argv, "TLDRrobot_node", ...
                ros::init_options::NoSigintHandler);
279      ros::NodeHandle nh;
280
281     ros::Subscriber subImu = nh.subscribe("orientation", 2, imuCallback);
282     ros::Subscriber subJoy = nh.subscribe("joy",10,joyCallback);
283
284
285     ros::Publisher pub = ...
            nh.advertise<TLDRrobot::TLDR_msg>("TLDRstatusPub", 1000);
286
287     ros::Rate rate(RATE);
288
289     signal(SIGINT, mySigIntHandler);
290
291     argumentHandler(argc, argv);
292
293
294
295     init();
296
297     serialPrintf(fd,"X_1_1\n");
```

```
298
299      while (ros::ok() && !g_request_shutdown)
300         {
301
302         calculatVariables();
303         updatePoleStates();
304
305
306         updatePoleLength();
307
308
309         if(!noSerial){
310         updateSerialCommunication();}
311
312         pub.publish(getStatusMsg());
313         ros::spinOnce();
314         rate.sleep();
315     }
316     ROS_INFO("Closing, Goodbye!");
317     serialPrintf(fd,"X_1_1\n");
318        ros::shutdown();
319 }
320 //send for each pole the message in form C_S_N (Comman Side Number) if ...
       the change indicator is true (prohibiting multiple sends)
321 void updateSerialCommunication(){
322
323 for(int i =0;i≤1;i++){
324     for(int j =0; j≤7;j++){
325
326
327     if(pole_arr[i][j].change||((n%100)==pole_arr[i][j].number*10)){
328
329     if(pole_arr[i][j].state==Hold) ...
           serialPrintf(fd,("H_"+std::to_string(pole_arr[i][j].side)
330 +"_"+std::to_string((pole_arr[i][j].number-i+7)%8+1)+"\n").c_str());
331     if(pole_arr[i][j].state==Extend
332         ||pole_arr[i][j].state==ExtendFruitless) ...
              serialPrintf(fd,("E_"+std::to_string(pole_arr[i][j].side)
333 +"_"+std::to_string((pole_arr[i][j].number-i+7)%8+1)+"\n").c_str());
334     if(pole_arr[i][j].state==Retract) ...
           serialPrintf(fd,("R_"+std::to_string(pole_arr[i][j].side)
335 +"_"+std::to_string((pole_arr[i][j].number-i+7)%8+1)+"\n").c_str());
336             pole_arr[i][j].change=false;
337             }
```

```
338
339
340                 }
341         }
342
343
344
345
346 }
347 //update the theoratical pole length. There is no real feedback to ...
       validate the length
348 void updatePoleLength(){
349
350
351 for(int i =0;i≤1;i++){
352                 for(int j =0; j≤7;j++){
353
354
355                         if(pole_arr[i][j].state==Extend) ...
                               pole_arr[i][j].length += POLE_SPEED * dt;
356                         if(pole_arr[i][j].state==Retract) ...
                               pole_arr[i][j].length -= POLE_SPEED * dt;
357             if(pole_arr[i][j].length>POLE_LENGTH) ...
                   pole_arr[i][j].length=POLE_LENGTH;
358              if(pole_arr[i][j].length<0) pole_arr[i][j].length=0;
359
360
361                         }
362                 }
363
364
365
366 }
367
368 void avoidCloseRangeProblem(){
369
370
371
372     for(int i =0;i≤1;i++){
373                 for(int j =0; j≤7;j++){
374
375         if(pole_arr[i][j].state==Retract && pole_arr[i][j].change && ...
               pole_arr[i][j].length<CLOSE_RANGE && ...
               pole_arr[i][j].length>0.000001){
```

```
376         pole_arr[i][j].state=Extend;
377         }
378 
379         if(pole_arr[i][j].length≥maximumAllowedLength && ...
                pole_arr[i][j].state!=Retract){
380         //pole_arr[i][j].state=Hold;
381         }
382 
383     }
384 
385     }
386 
387 }
388 
389 
390 void retractAll(){
391 
392 
393 
394         for(int i =0;i≤1;i++){
395                 for(int j =0; j≤7;j++){
396 
397                 pole_arr[i][j].state=Retract;
398 
399                 }
400 
401         }
402     serialPrintf(fd,"\n");
403          serialPrintf(fd,"X_1_1\n");
404 
405 }
406 
407 //stabilize algortihm. Retun -1 = error, Return 0: success but may be ...
       overwritten, Return 1, sucess but is not allowed to be overwritten
408 int stabilize(Pole & p, float angleRoll, float anglePitch, bool locomotion){
409 
410 
411 
412 if(fabs(angleRoll)>38.9*DEG2RAD){
413 ROS_WARN("We are falling over and there is nothing i can do");
414 //return -1;
415 }
416  float zeta = anglePitch + (p.number-1)*PI14;
417 if(zeta>PI21){
```

```
418 zeta-=PI21;
419 }
420
421 float l_balance = RADIUS_M /((float)cosf(zeta))-RADIUS;
422
423  if(backwards){
424        anglePitch = PI21-anglePitch;
425        zeta= PI21-zeta;
426        }
427
428
429 float threshold=0.05;
430 float err = -angleRoll;
431 if(p.side==2){
432 err= angleRoll;
433 }
434
435
436 if(zeta>beta && zeta < alpha){
437     if(err>threshold)
438         {
439         p.state=ExtendFruitless;
440         return 1;
441         }
442     if(err<-threshold){
443     p.state=Position;
444     p.cLength= RADIUS_M-RADIUS;
445     return 1;
446     }
447     if(p.length<l_balance){
448     p.state=Extend;
449     return 1;
450     }
451     p.state=Hold;
452     return 0;
453 }
454
455 if(zeta< PI21-beta && zeta > PI21-alpha && !locomotion){
456
457
458       if(err>threshold)
459                 {
460                 p.state=ExtendFruitless;
461                 return 1;
```

```
462                 }
463         if(err<-threshold){
464         p.state=Position;
465         p.cLength= RADIUS_M-RADIUS;
466         return 1;
467         }
468
469         if(p.length<l_balance){
470         p.state=Extend;
471         return 1;
472         }
473         p.state=Hold;
474         return 1;
475
476
477 }
478 p.state=Retract;
479 return 0;
480
481
482
483
484 }
485
486
487 void changePointUp(Pole & p, float angle){
488
489  float zeta = angle + (p.number-1)*PI14;
490 if(zeta>PI21){
491 zeta-=PI21;
492 }
493
494
495 if(zeta > PI-PI18 && zeta<PI+PI18){
496 p.state= Extend;
497 ROS_INFO("Pole %d is looking upwards!", p.number);
498 }else{
499 p.state = Retract;
500 }
501
502 }
503
504
505
```

```
506
507
508 void updatePoleStates(){
509
510
511
512
513
514     for(int i =0;i≤1;i++){
515                 for(int j =0; j≤7;j++){
516             if(!pole_arr[i][j].manual){
517             State temp = pole_arr[i][j].state;
518
519             int overrideStabi=0;
520             if(useStabilization){
521              overrideStabi=stabilize(pole_arr[i][j],roll,pitch,useAlgortihm);
522             }
523
524             if(useAlgortihm && overrideStabi == 0){
525                 //For TEsting:
526                 if(poleUpTest){changePointUp(pole_arr[i][j],pitch);}
527                 //actual algorithm
528                 else{ changePoleState(pole_arr[i][j],pitch );}
529             }
530             if(pole_arr[i][j].state==Position){
531                 positionStateProcessing(pole_arr[i][j]);
532             }
533             avoidCloseRangeProblem();
534             if(temp != pole_arr[i][j].state){pole_arr[i][j].change = true;}
535             }
536                 }
537         }
538
539
540
541
542
543
544 }
545
546 void positionStateProcessing(Pole & p){
547
548
549 if(p.state!=Position)return;
```

```
550
551 if (p.cLength<CLOSE_RANGE)p.cLength=0;
552 if(p.cLength>POLE_LENGTH)p.cLength=POLE_LENGTH;
553 if(p.cLength< p.length-0.01){p.state=Retract;}
554 else if(p.cLength>p.length+0.01){p.state=Extend;
555 }
556 else{p.state=Hold;}
557
558 }
559
560
561 void changePoleState(Pole & p, float angle){
562
563     float zeta = std::fmod(angle + (p.number-1)*PI14, PI21);
564     float zetaNext = std::fmod(zeta + wZ*dt, PI21);
565
566
567     if(backwards){
568     angle = PI21-angle;
569     zeta= PI21-zeta;
570     zetaNext=PI21-zetaNext;
571     }
572
573
574     if(fabs(cw)<0.001){
575         p.state=Retract;
576
577         return;
578     }
579
580     if(zeta ≤alpha && !leverageOnly){
581         if(zeta≥ beta){p.state= Extend;}
582         else{p.state=Hold;}
583     return;
584     }
585
586     //just pushing behaviour=> return now.
587     if(pushOnly){
588     p.state= Retract;
589     return;}
590
591     if(printLeverage)ROS_INFO("What am i going to do with zeta %f of ...
          pole number %d", zeta , p.number);
592     //minimum problem
```

```
593     if(zeta> PI21-acos(RADIUS/(RADIUS+CLOSE_RANGE))){
594     if(printLeverage)ROS_INFO("I am retracting, to avoid minium problem");
595     p.state=Retract;
596     return;
597     }
598
599     //climbinign, saftey, as otherwise pole damage
600     if(!pushOnly && !leverageOnly){
601     if(zeta>PI&&zeta < (PI+PI14)){p.state=Extend; return;}
602     else{p.state=Retract; return;}}
603
604     if(max_retractionspeed_needed < POLE_SPEED){
605 if(printLeverage)ROS_INFO("Leverage: retraction is fast enough, ...
        therefore extension after PI and retraction from moment we woudl ...
        touch ground");
606
607         float rattle=0; //if zeta is eaxactly on one value this would ...
              lead to rattle. Thefore this rattle factor leads to a hysteresis
608         if(p.state==Retract){rattle=0.05;}
609         if(zeta +rattle > PI && zetaNext+rattle<touchpoint){
610 //(RADIUS/cos(PI21-zetaNext-rattle)-RADIUS > p.length+L_MARGIN
611 // || zetaNext+rattle<touchpoint)){
612             p.state=Extend;
613             if(printLeverage)ROS_INFO("Extending, becauese i am fast ...
                  enough and not in danger ");
614         }
615         else{
616         if(printLeverage)ROS_INFO("Retracting, i would otherwise touch ...
              the ground");
617         p.state= Retract;}
618
619     return;
620     }
621     if(zeta > gammaAngle){
622
623         if( RADIUS/cos(PI21-zetaNext)-RADIUS < p.length-L_MARGIN){
624         p.state = Retract;
625         if(printLeverage)ROS_INFO("Retracting as i am over gamma, but ...
              would touch the ground");
626         }else{
627          if(printLeverage)ROS_INFO("Holding, as i am over gamma but will ...
               not touch the ground");
628         p.state= Hold;
629         if(p.length<CLOSE_RANGE){
```

```
630         p.state=Retract; // minimum distance problem
631             }
632         }
633         return;
634     }
635     if(zeta≥epsilon){
636     p.state=Retract;
637     if(printLeverage){ROS_INFO("Reteacting, as fast as i can, because i ...
            have not reached gamma bit am over epsilon.");}
638     return;}
639     if((zeta > epsilon_s || zeta > PI) && zeta > alpha){
640     p.state = Extend;
641     if(printLeverage)ROS_INFO("Extending, as i am over pi or epsioln s ...
            (and of course over alpha)");
642     return;
643     }
644     p.state=Retract;
645
646
647 }
648 void calculatVariables(){
649
650 float angVel= fabs(wZ); //choose abs(cw) for save operation. If fast ...
        enough poles choose abs(wZ)
651
652 //max_speed_needed- calculation; if the speed is slower then we cna ...
        provide, no need for further calculations.
653 max_retractionspeed_needed = ...
        fabs(RADIUS*angVel*tan(touchpoint)/cos(touchpoint));
654
655 if(max_retractionspeed_needed < POLE_SPEED){
656 return;}
657
658
659 //Gamma calculation
660
661 float a= -POLE_SPEED/(RADIUS*angVel);
662 float aa = a*a;
663 gammaAngle = PI + 2.0*atan(0.5*sqrt(1.0/aa + 4.0) + sqrt(1.0/aa - ...
        4.0/(a*sqrt(1.0/aa + 4)) - 1.0/(aa*a*sqrt(1.0/aa + 4.0)))/sqrt(2.0) ...
        - 1.0/(2.0*a));
664
665
666
```

```
667
668
669 //epsilon calculation
670 if(POLE_LENGTH -(RADIUS/(cos(PI21-gammaAngle))-RADIUS)<=0){
671     //gamma is capable of retracting. For saftey, should not occur due ...
        to separate condition check
672     epsilon =gammaAngle;
673     }
674 else{
675 epsilon = gammaAngle - angVel ...
    *(POLE_LENGTH-(RADIUS/(cos(PI21-gammaAngle))-RADIUS))/POLE_SPEED;
676
677 epsilon=fabs(epsilon);
678
679 }
680
681 epsilon_s= epsilon-angVel*POLE_LENGTH/POLE_SPEED;
682 if(printLeverage){
683 ROS_INFO("Variables: epsilon %f deg, epsilon_s %f deg, gamma %f deg, ...
    touchpoint %f deg with cw of %f",
684 (epsilon*RAD2DEG),(epsilon_s*RAD2DEG),(gammaAngle*RAD2DEG),
685 (touchpoint*RAD2DEG),angVel);
686 }
687
688 }
689
690
691 TLDRrobot::TLDR_msg getStatusMsg(){
692
693
694     TLDRrobot::TLDR_msg msg;
695
696     msg.header.stamp = ros::Time::now();
697     msg.header.seq= n++;
698
699     msg.pitch=pitch;
700     msg.roll=roll;
701     msg.cw=cw;
702     msg.w=wZ;
703
704     msg.ext_length= POLE_LENGTH;
705
706     float factor = 1.0/POLE_LENGTH;
707
```

```
708     for(int i =0;i≤1;i++){
709                 for(int j =0; j≤7;j++){
710
711
712             msg.pole_stat.push_back(pole_arr[i][j].state);
713             msg.pole_ext.push_back(pole_arr[i][j].length*factor);
714
715                         }
716                 }
717     return msg;
718
719 }
720 int argumentHandler(int argc, char *argv[]) {
721
722     for (int i = 1; i < argc; ++i) {
723         std::string argi = argv[i];
724         if (argi.compare("-q") == 0) {
725             quiet = true;
726             ROS_INFO(
727                     "Quiet Mode activated! No receiing data or repetetive ...
                        Messages will be shown. Start-up information, ...
                        Errors and warning etc will be shown!!\n");
728         }
729      if (argi.compare("-noSerial") == 0) {
730             noSerial= true;
731             ROS_INFO("No Serial COmmunication during algortihm. There ...
                    will be no actual movement. Initalization will still be ...
                    done.");
732         }
733
734
735     if (argi.compare("-activateSerial") == 0) {
736             noSerial= false;
737             ROS_INFO("Startign with serial communication");
738         }
739
740             if (argi.compare("-useAlgortihm") == 0) {
741             useAlgortihm= true;
742             ROS_INFO("Starting using the algortihm");
743         }
744
745         if (argi.compare("-pushOnly") == 0) {
746
747                 pushOnly = true;
```

```
748                 ROS_INFO("Push only mode");
749         if(leverageOnly){
750                 ROS_WARN("Leverage only Only was activated, will be ...
                        deactivated");
751                 leverageOnly=false;
752                 }
753         }
754       if (argi.compare("-leverageOnly") == 0) {
755 
756                 leverageOnly = true;
757                 ROS_INFO("leverage only mode activated");
758                 if(pushOnly){
759         ROS_WARN("Push Only was activated, will be deactivated");
760         pushOnly=false;
761         }
762         }
763       if (argi.compare("-poleUpTest") == 0) {
764 
765                 poleUpTest = true;
766 
767         }
768 
769   if (argi.compare("-useStabilization") == 0) {
770 
771                 useStabilization = true;
772 
773         ROS_INFO("Stabilization of roll activated.");
774 
775         }
776 
777       if (argi.compare("-printPitch") == 0) {
778 
779                 printPitch = true;
780 
781         }
782       if (argi.compare("-printRoll") == 0) {
783 
784                 printRoll = true;
785 
786         }
787      if (argi.compare("-calibratePitch") == 0) {
788 
789                 calibratePitch = true;
790 
```

```
791         }
792      if (argi.compare("-calibrateRoll") == 0) {
793
794                 calibrateRoll = true;
795
796         }
797      if (argi.compare("-printLeverage") == 0) {
798
799                 printLeverage = true;
800
801         }
802         if (argi.compare("-alpha") == 0) {
803             if (i + 1 < argc) {
804                 i++;
805                 alpha = atof(argv[i]);
806                 ROS_INFO("alpha set to %f deg\n", alpha);
807         alpha = alpha*DEG2RAD;
808
809             }
810         }
811
812     if (argi.compare("-beta") == 0) {
813             if (i + 1 < argc) {
814                 i++;
815                 beta = atof(argv[i]);
816                 ROS_INFO("beta set to %f deg \n", beta);
817         beta = beta*DEG2RAD;
818
819             }
820         }
821         if (argi.compare("-cw") == 0) {
822             if (i + 1 < argc) {
823                 i++;
824                 cw = atof(argv[i]);
825                 ROS_INFO("command omega cw set to %f rad/s\n", cw);
826
827             }
828         }
829 }
830 return 1;
831 }
```

**Listing A.5:** Header for the code running on the prototype of the TLDRrobot

```
#ifndef TLDR_H
#define TLDR_H

#include <ros/ros.h>
#include <wiringSerial.h>
#include <wiringPi.h>
#include <sensor_msgs/Imu.h>
#include <sensor_msgs/Joy.h>
#include <iostream>
#include <string>
#include <unistd.h>
#include <signal.h>
#include "TLDRrobot/TLDR_msg.h"
#include <cmath>

#define SERIAL_DEVICE "/dev/ttyUSB0"
#define BAUD_RATE 9600 //per s
#define POLE_SPEED 0.14 // in m per s
#define POLE_LENGTH 0.95 // in m
#define RATE 125 //calculation rate
#define DEFAULT_ALPHA 57 //in degree
#define DEFAULT_BETA 2 //in degree
#define RADIUS 0.4 //in m
#define RADIUS_M 0.50
#define CLOSE_RANGE 0.15 // in m, tareting a region of the actuator ...
    where retracting can not be started.

//constants
#define DEG2RAD 0.017453293
#define RAD2DEG 57.2957795
#define PI 3.1415926536
#define PI12 1.5707963268
#define PI14 0.7853981634
#define PI18 0.3926990817
#define PI21 6.2831853072
#define PI32 4.7123889804

#define L_MARGIN 0.05 // in m. This will compensates fot the uncertainty ...
    of the pole length

//Hold, extend and retract pole. With state position the pole is ...
    extended or rectected dependend on the commanded length cLength. ...
```

```
    With ExtendFruitless, Extending is initiated but no length is integrated
enum State {Hold, Extend, Retract, Position,ExtendFruitless};

struct Pole {
    float length;
    int side;
    int number;
    State state;
    bool change;
    bool manual;
    float cLength;
};

struct Pole pole_arr[2][8];


// orientation and velocity data by imu
float qW;
float qX;
float qY;
float qZ;

float aX;
float aY;
float aZ;

float wX;
float wY;
float wZ;

float roll;
float pitch;
float yaw;


float compensatePitch;
float compensateRoll;
//commaned omega
float cw;


//serial communication id
int fd;
//variables from mathematical model
```

```
83 float alpha= DEFAULT_ALPHA*DEG2RAD;
84 float beta= DEFAULT_BETA*DEG2RAD;
85 float gammaAngle;//gamma already defined by standard liberys
86 float epsilon;
87 float epsilon_s;
88 float max_retractionspeed_needed;
89
90
91 //angle at whoch fully extend pole will touch groudn. Calculatetd in init.
92 float touchpoint;
93
94
95 //current dt
96 float dt = 1.0/RATE;
97
98
99 int n=0;
100
101 void updatePoleStates();
102 void updateSerialCommunication();
103 void updatePoleLength();
104 void changePoleState(Pole & p, float angle);
105 void calculatVariables();
106 int argumentHandler(int argc, char *argv[]);
107 void avoidCloseRangeProblem();
108 void retractAll();
109 int stabilize(Pole & p, float angleRoll, float anglePitch, bool locomotion);
110 void positionStateProcessing(Pole & p);
111 TLDRrobot::TLDR_msg getStatusMsg();
112 #endif
```

**Listing A.6:** Message structure of the TLDR msg used for ROS

```
Header header
string child_frame_id

float64[] pole_ext
uint8[] pole_stat

float64 ext_length

float64 w
float64 cw

float64 pitch
float64 roll
```

# Appendix B

# Calculations

$$s(x) = \sin(x) \quad as(x) = \arcsin(x) \quad c(x) = \cos(x) \quad ac(x) = \arccos(x) \quad t(x) = \tan(x) \quad at(x) = \arctan(x)$$

$$l = \frac{(-t(\phi_{\text{VPIP}}) \cdot s_f \cdot c(\phi_r) \cdot d_m^s - r_m + s_f \cdot s(\phi_r) \cdot d_m^s) \cdot ||d_{mt}|| \cdot r_s + l_{\max}}{t(\phi_{\text{VPIP}}) \cdot d_{mtx} + t(\theta_{\text{VPIP}}) \cdot d_{mty} + d_{mtz}} - r_s$$

$$l = \frac{\begin{array}{l}(-t(\phi_{\text{VPIP}}) \cdot s_f \cdot c(\phi_r) \cdot s\left(ac\left(\frac{r_s}{r_m}\right)\right) \cdot r_m - r_m + s_f \cdot s(\phi_r) \cdot s\left(ac\left(\frac{r_s}{r_m}\right)\right) \cdot r_m) \\ \cdot \sqrt{\begin{array}{r}(s_f \cdot c(\phi_r) \cdot s\left(ac\left(\frac{r_s}{r_m}\right)\right) \cdot r_m - c(\zeta) \cdot r_s \cdot s(\phi_r))^2 + (s(\zeta) \cdot r_s)^2 \\ + (r_m - s_f \cdot s(\phi_r) \cdot s\left(ac\left(\frac{r_s}{r_m}\right)\right) \cdot r_m - c(\zeta) \cdot r_s \cdot c(\phi_r))^2\end{array}} \cdot r_s + l_{\max}\end{array}}{\begin{array}{r}t(\phi_{\text{VPIP}}) \cdot (s_f \cdot c(\phi_r) \cdot s\left(ac\left(\frac{r_s}{r_m}\right)\right) \cdot r_m - c(\zeta) \cdot r_s \cdot s(\phi_r)) - t(\theta_{\text{VPIP}}) \cdot s(\zeta) \cdot r_s + r_m \\ - s_f \cdot s(\phi_r) \cdot s\left(ac\left(\frac{r_s}{r_m}\right)\right) \cdot r_m - c(\zeta) \cdot r_s \cdot c(\phi_r)\end{array}} - r_s \tag{B.1}$$

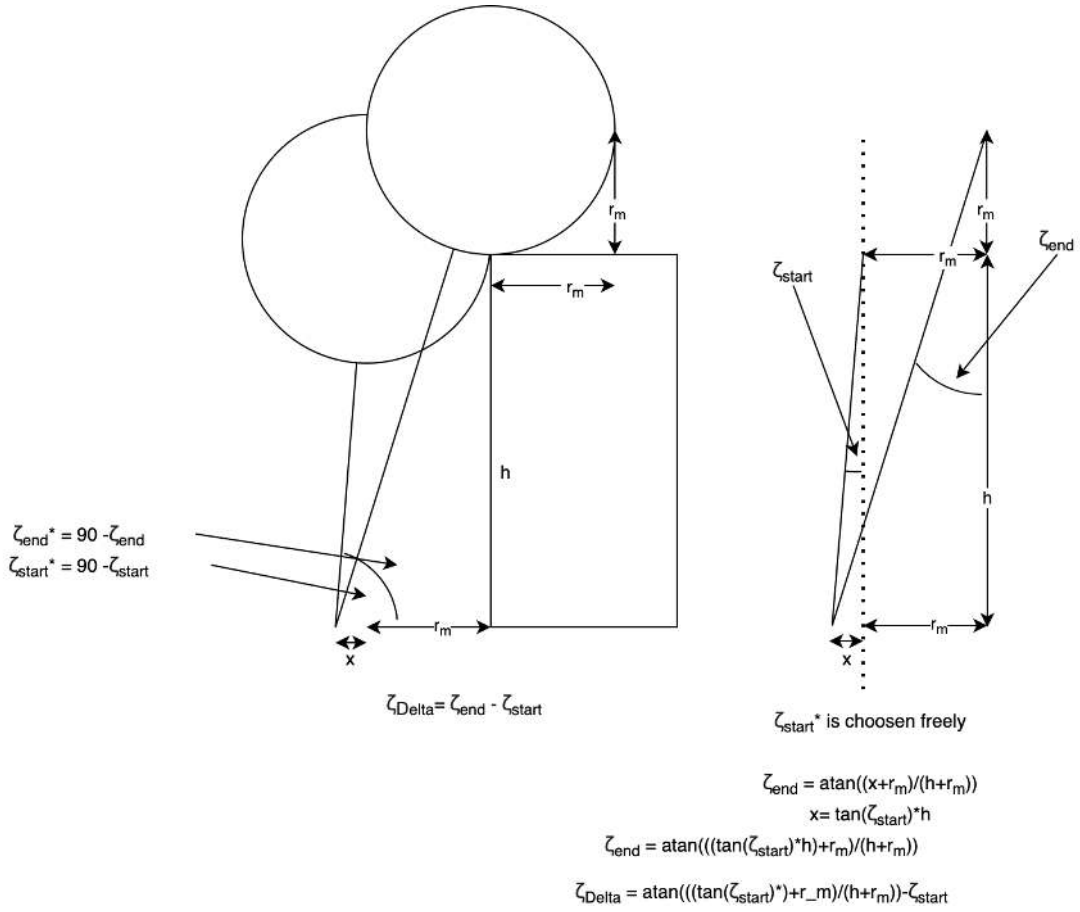



**Figure B.1:** Calculation of $\zeta_\Delta$ when overcoming an perpendicular obstacle and leverage mode is avaliable.

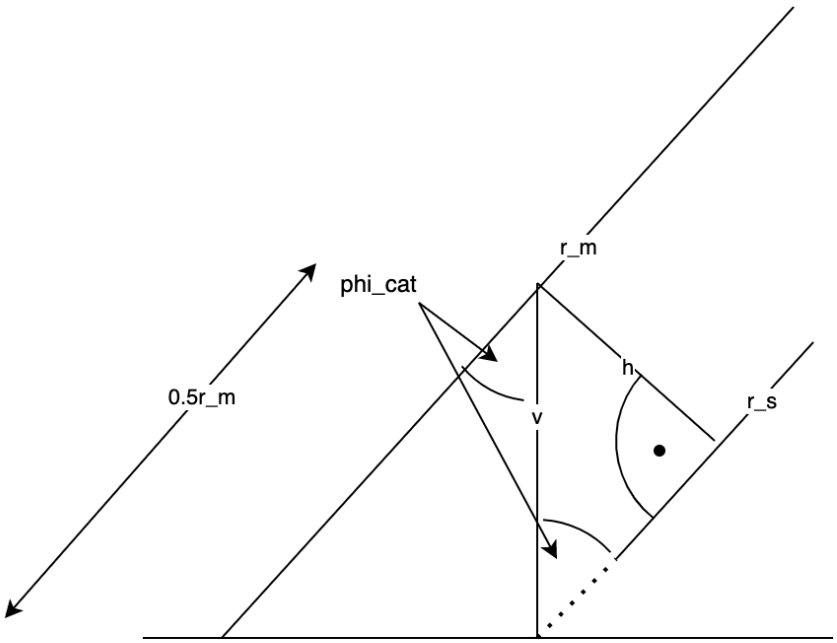



**Figure B.2:** Calculation of the $\phi_{\text{catastrophic}}$ with variable h (referred to as $d_m^s$)

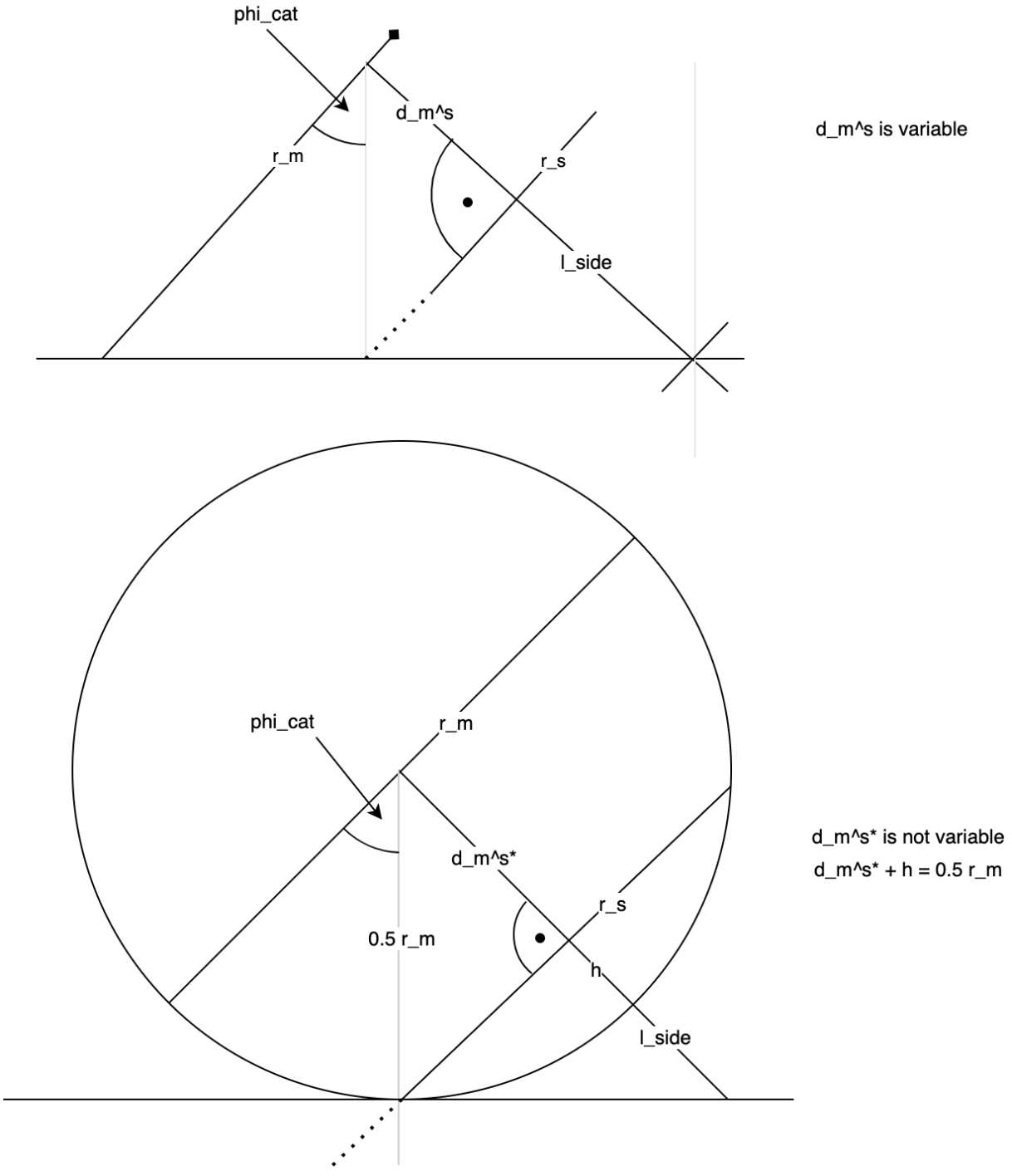



**Figure B.3:** Variable illustration for $l_{\text{side}}$ calculation for general disc robot (above) and spherical robot(below).

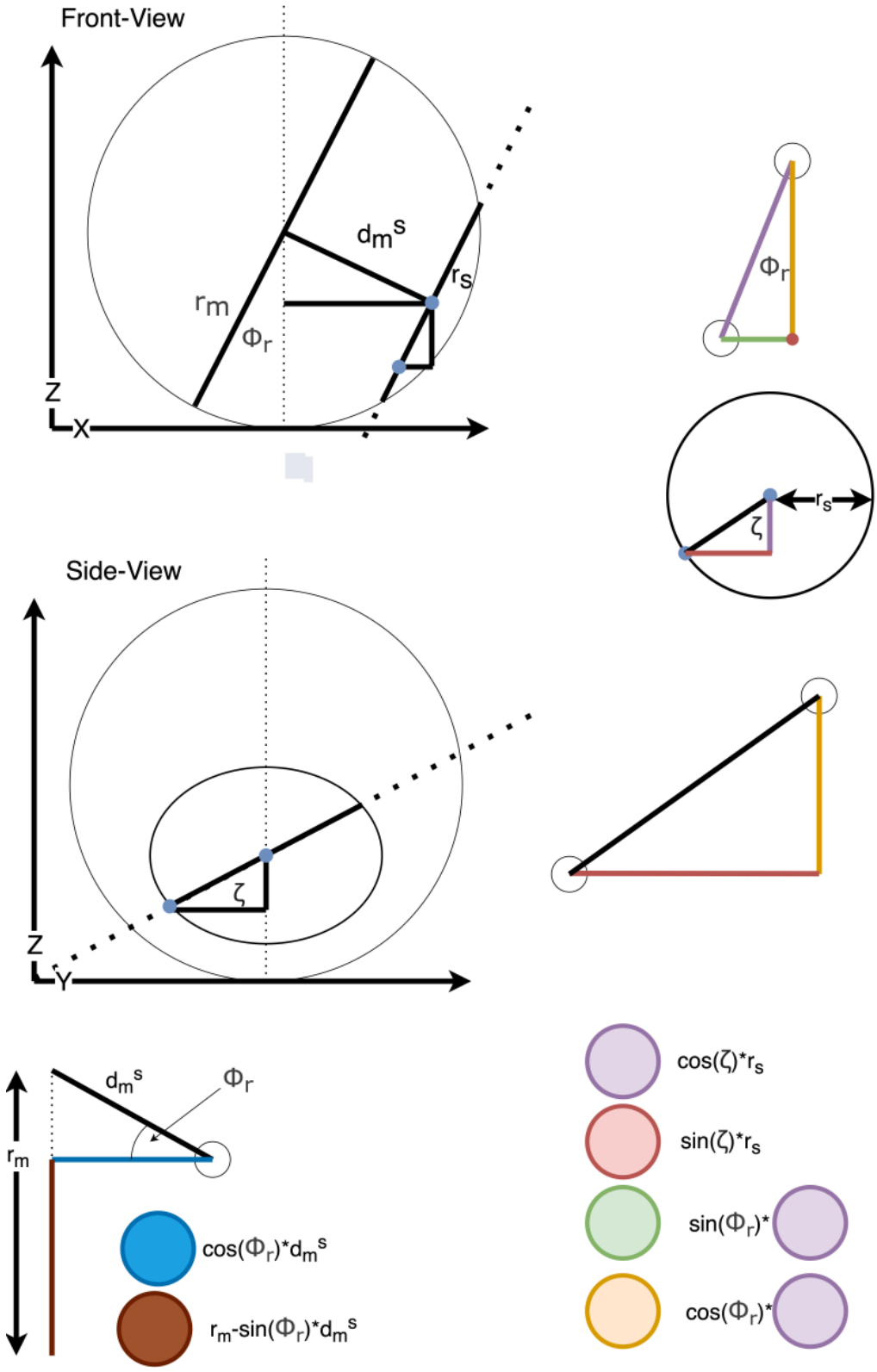



**Figure B.4:** Calculation visualization of line-estimation of single pole for the VPIP.

# Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Würzburg, September 2021