ER-Force 2026
Extended Team Description Paper

Theodor Böhm, Undine Hahn and Lukas Wegmann

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Faculty of Engineering,
Department of Computer Science, Distributed Systems and Operating Systems
Robotics Erlangen e.V., Martensstr. 1, 91058 Erlangen, Germany
Homepage: `https://www.robotics-erlangen.de/`
Contact Email: `info@robotics-erlangen.de`

Abstract. This paper presents the proceedings of ER-Force, the RoboCup Small Size League team from Erlangen located at Friedrich-Alexander-University Erlangen-Nürnberg, Germany.
It describes our recent improvements to the omni wheels and provides insight into the manufacturing of our dribbler suspension. Moreover, it provides a design guide for developing motor drivers, sharing the considerations and constraints of our most recent design iteration, together with the lessons we learned. Additionally, we describe and critique a common robot control strategy and propose an alternative structure that we have used ourselves.

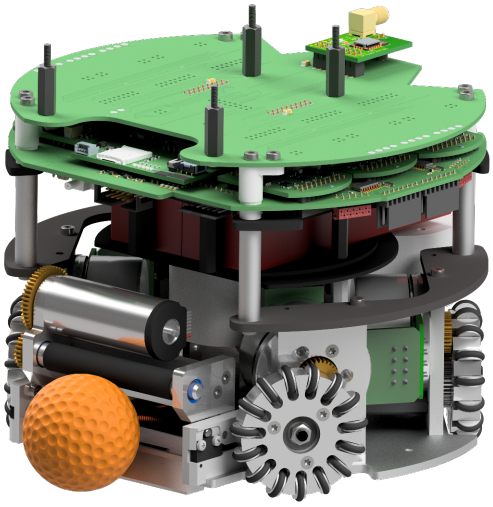


Fig. 1: ER-Force robot design from 2023

1 Introduction

This ETDP presents some of our latest improvements to the mechanics, a guide for designing motor drivers and a comparison of two control structures. Section 2 details improvements to our wheels and provides insight into the manufacturing of our dribbler suspension. Subsequently, section 3 is a design guide for developing motor drivers. It shares our considerations and constraints for the design process and lessons we learned in the most recent iteration. Finally, section 4 highlights a flaw of a common motion control strategy and describes an alternative approach, which we have used in the past.

2 Mechanics

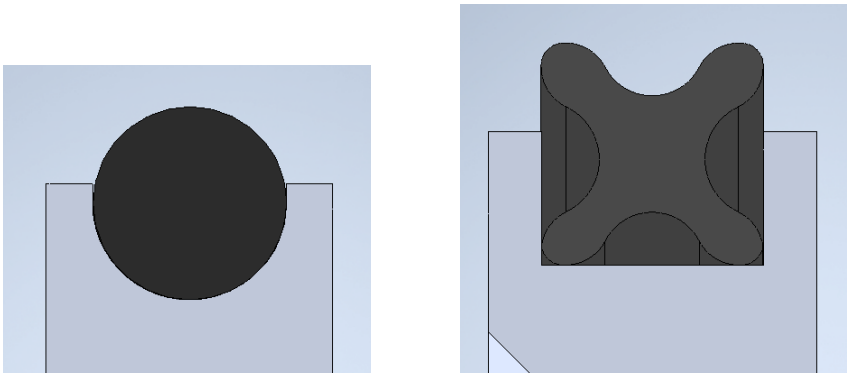
The following sections will deal with the mechanical development of three components. Regarding driving behavior, the initial focus is on the subwheels. Subsequently, the design of the omniwheel will be revised. Finally, the material evolution of the dribbler suspensions will be discussed.

2.1 Subwheels

Until now, the robots have been using O-rings on the subwheels. These wear out very slowly and are made of a relatively hard material. As a result, they hardly deform and have a relatively small contact area with the carpet. This can cause the wheels to slip. To counteract this, X-rings made of NBR with Shore 70 are used. With these, each subwheel has two contact surfaces with the carpet instead of just one. Therefore, the individual contact surfaces are smaller, and the material above them is correspondingly thinner compared to the O-ring. The thinner and softer material makes the X-Rings more flexible. We expect this behavior to provide more grip on the ground. When manually rolling the omniwheels, a noticeable difference emerges. The wheel with the X-rings rolls significantly more smoothly and quietly. The higher friction and smoother driving behavior have already been observed by TIGERs, Delft, and PARSIAN [1,2,3]. However, TIGERs also mention higher wear.

The previous design of the subwheel bodies was adapted to the O-rings and featured a rounded groove into which the O-ring was inserted, as shown in figure 2a. Due to the different cross-section of the X-rings, this shape of the subwheel body is unsuitable. The X-ring has an X-shaped cross-section, which means the contact surfaces on the subwheel are located further outwards. Thus, the X-ring would rest on a larger diameter than the O-ring due to the round groove in the subwheel body. This causes the X-ring to stretch more, resulting in a larger outer diameter for the assembled subwheel. Because of that, the X-ring tends to twist. This in turn leads to high wear of the X-rings and a slightly larger outer diameter of the entire omniwheel. To counteract this, the inner groove of the subwheel bodies is modified as shown in figure 2b. This reduces the amount of stretching in the X-ring and prevents it from twisting. Furthermore, the X-ring

is more laterally enclosed by the subwheel body, which further stabilizes it. This results in an outer diameter of 11 mm for the subwheel.



(a) Old subwheel with O-ring (b) New subwheel with X-ring

Fig. 2: The different subwheel crosssections. The old form of the subwheel body with the round groove and an O-ring (a) and the new form of the subwheel body with the square groove and an X-ring (b).

The new subwheel bodies are made of aluminum instead of plastic. These were sometimes a bit stiff on the steel axles of the subwheels. To prevent the aluminum subwheels from rubbing against the steel axles, plain bearings are installed. These allow for significantly easier rotation of the subwheels and thus a smoother driving behavior. KIKS and TIGERs both use plain bearings in their subwheels [4,1]. KIKS use plain bearings made of POM, and TIGERs use manufactured plain bearings from igus[1]. The latter are made from specially developed polymers, which makes those used by TIGERs shock-absorbing and gives them good dry-run capabilities. For this reason, we opted for the GSM-0203-03 plain bearings from igus.

2.2 Omniwheels

So far, the robots have had 15 subwheels per omniwheel. This means they drive on a pentadecagon, which results in rather bumpy driving behavior. By using X-rings, each of these corners is replaced by two very tightly spaced corners. However, the main problem with the pentadecagon still remains. For smoother driving behavior, a polygon with more corners would be required; therefore, omniwheels with more subwheels are needed. In order to maintain the robot's height, the outer diameter of the omniwheel must remain the same. To ensure this, the new subwheel size must be taken into account. Since the new shape of

[1] igus SE & Co. KG, Cologne, Germany

the subwheel body results in dimensions similar to the old one, the previously used diameter of the wheel body of 45 mm will be retained. The shape of the recess for the subwheels is adapted to the shape of the assembled subwheel. Care must be taken to ensure that the subwheel does not have too much tolerance to the side, but still leaves enough space so that it does not rub. The depth should also be minimized, in figure 3 to the left of the subwheel. Attention should be paid to leave enough space so that the wheel does not become blocked if lint gets in.

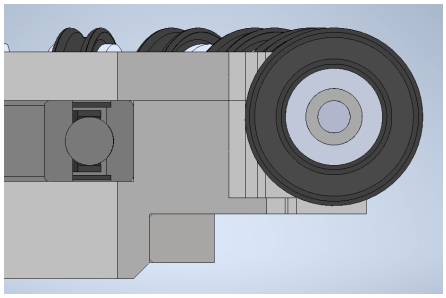


Fig. 3: A cross-section of the assembled Omniwheel. Visible is the space for the subwheel and the reinforcement of the subwheel suspension struts.

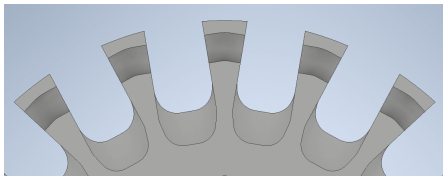


Fig. 4: Shape of the wheel body from the front with the recesses, the subwheel suspension struts and their reinforcement.

Based on the size of this recess, it can then be determined how many subwheels can be positioned around the wheel body. Caution is advised regarding the thickness of the suspension struts of the subwheels (see fig. 4). If they become too thin, they can deform and break. In our case, 19 subwheels fit around one omniwheel. This means we have fewer subwheels than TIGERs with 20, Delft with 24, and TurtleRabbit with 30 [1,2,5]. However, these teams all have a larger wheel diameter, with TurtleRabbit even using two rows of subwheels. PARSIAN, on the other hand, has fewer subwheels than us with 16, but apparently also uses smaller wheel bodies and possibly larger subwheels [3]. With the 19 subwheels, the struts of the wheel body are 1.124 mm thick at their thinnest point. To stabilize it, the recess at the back is closed as much as possible. Thus, the wheels resemble a nonadecagon and enable a significantly smoother ride. A robot equipped with four prototypes of this Omniwheel has already competed at RoboCup 2025. Since this worked very well, the remaining robots will now also be equipped with these wheels in order to compete in the RoboCup 2026.

2.3 Dribbler Suspension

Until the beginning of 2025, the dribbler suspensions of the robots were printed from PA12 using the SLS printing process. This was a very cost-effective production, as the Department of Additive Manufacturing from the Institute of Polymer

Technology at our university printed it for us free of charge. These components are attached to the base plate of our robots using screws. The problem is that the metal screw wears down the plastic thread relatively quickly. Because Loctite does not adhere to this material, the screws loosen. This results in high wear and tear on these components. In order to have sufficient replacements for the GermanOpen and the SchubertCup, the two tournaments in Germany, more had to be manufactured. However, the institute's SLS printer is currently undergoing maintenance and is therefore unavailable, which is why an epoxy resin printer is used. During the two tournaments, it became clearly apparent that epoxy resin is too brittle for this application. The chip plate is attached to this component. Furthermore, it protrudes slightly from the robot, which means it is involved in many collisions. This leads to the components regularly shattering and their remains having to be collected from the field, resulting in fouls. In addition, the chip plate is no longer sufficiently attached and is ejected from the robot during chipping. The dribbler suspension also serves as a mount for the light barrier. Since the ones we use are expensive, care should be taken to protect them. If their mounting breaks, they can be damaged.

To prevent this and to reduce wear on the components, the dribbler suspension is manufactured of aluminum. The components manufactured in this way were used on the RoboCup 2025. Since then, wear has been significantly reduced, and the screws can be secured with Loctite, so they no longer come loose. This also means fewer pieces are lost on the field, thus reducing the likelihood of fouls for this reason. In severe collisions, dents can still occur. These can sometimes stiffen or completely lock the chip plate in place, preventing the robot from chipping the ball. However, this rarely occurs and can be fixed quickly enough. Therefore, it still seems to be the best solution.

3 Design guide for developing motor drivers

Our previous motor controller design was done during the global semiconductor shortage, and this resulted in some hard design choices. The microcontroller, for example, was not chosen based on technical criteria but on availability. We bought 100 chips in a BGA package before starting the design to avoid creating a board, which components would no longer be available once the design is finished. The BGA package made the design much more expensive because of buried and blind vias and much more complex because of the high pin density. The controllers worked but had some reliability issues, which were not apparent before ordering the assembly of the series. When an unexpected brownout or power loss happened, the motor driver would not be able to carefully slow down the wheel; instead, the stored energy of the wheel would irreparably damage the half bridges of the driver. For our current design, we decided on a longer design process as described in the following subsections. These findings can be considered an extension on the process described in [6].

3.1 Choosing and evaluating components

We began by collecting the design requirements from the firmware and motor control team to choose the microcontroller (MCU) and motor driver.

Important considerations for the microcontroller were DMA channels to keep the system real-time capable, timers for software timing, status LEDs and PWM generation, analog digital converters to measure currents and voltages, SPI to communicate with the driver, position encoder, and the rest of the robot, and UART for "printf"-debugging.

The driver IC was chosen based on the current and voltage requirements for our motors, as well as important control and safety features. Since our new controller is field-oriented instead of trapezoidal, it is a requirement that there is some form of current sense system available. Protection systems for events like cross-conduction, overcurrent, and overtemperature protect the driver itself and the motor from permanent damage.

Based on these considerations we chose the STM32F401RET6 by STMicroelectronics for the MCU and the DRV8316C by Texas Instruments as the motor driver. Many options were available, but secondary constraints led us to these components because we needed a chip that is compatible with our real-time embedded operating system version and wanted to save on space by choosing a fully integrated motor driver instead of a gate driver with discrete MOSFETs.

We verified the capabilities of both devices using the evaluation kits designed by the manufacturers.

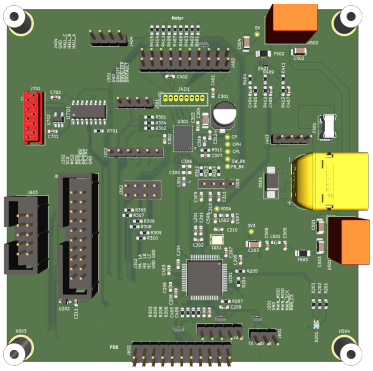
3.2 Design of the prototype

The first prototype, seen in Figure 5a, was created based on the reference designs described in the datasheets of the chips. The schematic contains the new motor controller itself, as we intended it for the final design, as well as the required voltage generation and programming interface, which will be provided by the robot power distribution board for the final board. This first version was much larger than the final version was going to be. This was done in order to add test points to all relevant parts of the board and have enough space to hack in additional components, if required. Pull-up/down resistors were added to most signals to force a defined state of the circuit in case the microcontroller or driver IC are not driving the connection. Using this strategy, one can avoid undefined (floating) signals and prevent the components from misbehaving during a reset, fault, or other unexpected events. The test board and the final version are available as an open-source publication[2].

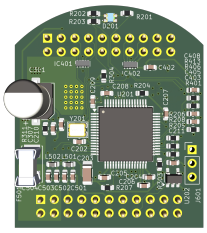
3.3 Evaluating prototypes

The evaluation prototype was thoroughly tested by the electronics, firmware, and control teams. The first tests were done by the electronics team in order

[2] Robotics Erlangen e.V., Open-Source Hardware, GitHub Repository, `https://github.com/robotics-erlangen/hardware`



(a) This is the large evaluation board with test points for every relevant connection (100mm x 100mm).



(b) This is the final board (42.5mm x 35mm).

Fig. 5: These are the 3d renders of the motor controllers (not to scale).

to test the hardware features of the board. The capabilities of the MCU were tested one by one by deploying a minimal firmware that only makes use of one feature at a time and gives instant feedback over UART. A similar approach was employed to test the driver IC, but some extra tools were required. A laboratory power supply can be used to simulate the currents of a single motor phase by forcing a known current through the driver and checking the resulting values in the MCU to gauge the current sense amplifiers and ADCs. The overtemperature protection was tested using a heat gun to artificially overheat the chips. The cross-conduction and overcurrent protection was intentionally triggered by an explicitly faulty firmware. The additional resistors were successively removed, and then undefined states were simulated by manually causing faults in the board to observe the behavior and judge whether they are needed or not.

After these initial hardware tests, the firmware team was able to start developing the firmware for the evaluation and the final PCB. When problems arose in the firmware development, the extra test points were very useful to use logic analyzers and oscilloscopes to get direct information about the nature of the issue.

Once the first draft of the firmware was running, the control team was able to start developing current and velocity controllers for a single wheel.

3.4 Problems and lessons

After the successful testing of the larger design, all test points and additional support components were removed for the final design as seen in Figure 5b. The finished motor controller PCBs could be inserted into the robot right after receiving the boards from the board house. The firmware and control algorithms developed on the evaluation hardware worked out of the box.

A problem of the new motor controller is the heat produced by the driver IC when handling a spike in required motor power. This issue could be observed on

the larger test board but was not as severe and was judged to be not that critical. On the smaller version used in the robot, this became a persistent issue. We were able to massively reduce the amount and frequency of overtemperature events by adding heat sinks and fans to the driver IC. Overall, the design process can be considered a success, but in the next design we will pay much more attention to the thermal behavior of our drivers. Another lesson learned is that removing all test points was a bit too aggressive, and some commonly used ones, like the SPI to rest the of the robot or the analog signals of the motor currents, should have stayed on the board for better in-robot debugging capabilities.

4 A case against the most common robot control structure

There are a multitude of architectures for the control of omnidirectional mobile robots. However, the RoboCup SSL has in large part converged on a common structure that is now widely employed in both Division A and B. It consists of sending (local) body-velocity commands to the robot, which will be converted to required wheel velocities. Each wheel is then separately controlled to follow these velocity commands.

This approach and variations thereof are widely described in both conventional literature (e.g. [7,8]) and TDPs (e.g. [9,10]). It is both easy to understand and straightforward to implement, which contributes to its popularity. However, we believe this approach to be flawed in most implementations.

In this section, we will model the robot, briefly outline the control strategy, and then highlight its issues. Then, we will suggest an easily implementable extension that does not require the presence of an IMU (inertial measurement unit) or any significant redesign. This is to enable teams to see quick improvements without requiring huge investments of time or money. Without loss of generality, we will demonstrate our findings on a three-wheeled robot to maintain symmetry and keep the dimensions smaller. The results generalize to four-wheeled robots without modification.

4.1 Modeling

The robot type under consideration is shown in figure 6. We differentiate between a fixed coordinate system s (space-frame) and a moving coordinate system b (body-frame) that lies in the geometric center of the robot, which is assumed to also be the center of mass.

Kinematics: The kinematics (i.e. the geometric relationship between velocity and wheel angular velocity) have been thoroughly derived in [11]. For convenience we quickly summarize the important equations here. Assuming no slip, the inverse kinematics of the robot in the moving body-frame are described as

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \boldsymbol{W} \underbrace{\begin{bmatrix}{}^b v_\mathrm{x}\\ {}^b v_\mathrm{y}\\ \dot{\phi}\end{bmatrix}}_{{}^b\boldsymbol{v}}, \quad \boldsymbol{W} = \frac{1}{r_\mathrm{w}}\begin{bmatrix}0 & 1 & L\\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} & L\\ \frac{\sqrt{3}}{2} & -\frac{1}{2} & L\end{bmatrix} \tag{1}$$

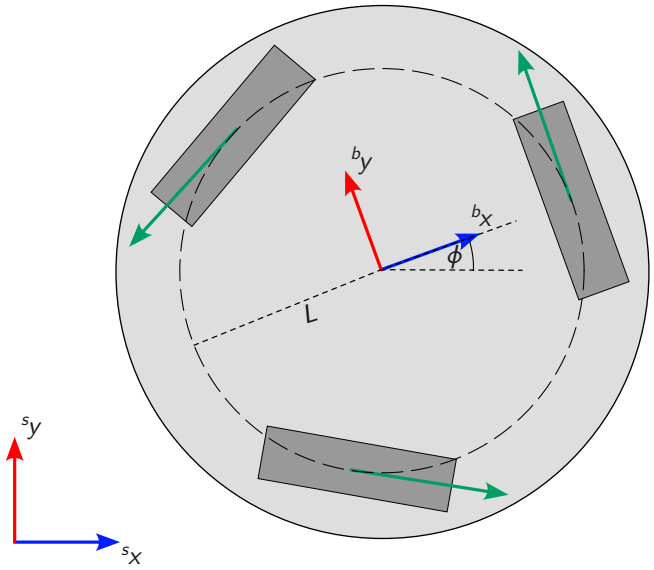


Fig. 6: Three-wheeled omnidirectional mobile robot, with illustration of the moving b-frame and stationary s-frame.

where $\boldsymbol{W}$ denotes the 3×3 wheel Jacobian mapping robot velocity to wheel speeds. The matrix $\boldsymbol{W}$ depends only on the wheel geometry and is constant in the body frame. The distance of the wheels to the center of the robot is denoted as L and the wheel radius is r_w. The left side superscript b signifies that the quantities are expressed in the b-frame. With a desired body-speed $\boldsymbol{v}^*$ of the robot, the inverse kinematics (1) yield the desired wheel velocities ω_i^* with $i = 1, 2, 3$.

Dynamics: The robot dynamics (i.e. the relationship between forces and accelerations) in the body frame have been derived in [12] using Lagrangian mechanics and are described by

$$\boldsymbol{M}^{*\,b}\dot{\boldsymbol{v}} + \dot{\phi}\boldsymbol{Q}\,{}^b\boldsymbol{v} = \boldsymbol{W}^\mathrm{T}\boldsymbol{\tau} \tag{2}$$

where $\boldsymbol{\tau}$ are the applied wheel torques and $\boldsymbol{W}$ is once again the coupling matrix. The inertia matrix

$$\boldsymbol{M}^* = \underbrace{\begin{bmatrix} m_\mathrm{r} & 0 & 0 \\ 0 & m_\mathrm{r} & 0 \\ 0 & 0 & J_\mathrm{r} \end{bmatrix}}_{\boldsymbol{M}} + \boldsymbol{W}^\mathrm{T} \underbrace{\begin{bmatrix} J_\mathrm{w} & 0 & 0 \\ 0 & J_\mathrm{w} & 0 \\ 0 & 0 & J_\mathrm{w} \end{bmatrix}}_{\boldsymbol{J}} \boldsymbol{W} \tag{3}$$

includes the mass of the robot m_r, the *robot* rotational inertia J_r and the *wheel* inertia J_w. Therefore, to accelerate the robot ($\dot{\boldsymbol{v}}$) both the inertia of the entire robot (the matrix $\boldsymbol{M}$) as well as the rotational inertia of the wheels (the matrix $\boldsymbol{J}$) has to be overcome. The term

$$\dot{\phi}\boldsymbol{Q}\,{}^b\boldsymbol{v} \tag{4}$$

considers the effects of the potentially rotating frame of reference with the skew symmetric matrix

$$\boldsymbol{Q} = \begin{bmatrix} 0 & -m_\mathrm{r} & 0 \\ m_\mathrm{r} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5}$$

and the robot angular velocity $\dot{\phi}$.

Wheel dynamics: Now we shift the perspective to that of the PID controllers that only observe the dynamics of the respective wheel. A common but naïve approach would yield the SISO-system

$$\dot{\omega} = -\frac{\mu_\mathrm{r}}{J_\mathrm{w}}\omega + \frac{k_\mathrm{t}}{J_\mathrm{w}}u \tag{6}$$

with the wheel inertia J_w (not to be confused with the robot inertia J_r), the torque constant k_t to relate motor current with produced torque and the viscous friction coefficient μ_r. On the surface this model seems perfectly reasonable and is physically plausible. However, that is not the case and the deficiencies of this system description are closely related to the deficiencies of the decentralized control approach.

The wheel dynamic description of (6) is incomplete. The issue lies in the effective wheel inertia J_w. This would only be an accurate system description if the wheel is unloaded, as the rotational inertia of the wheel is usually small relative to the mass of the robot that is driven. This can be clearly seen in the body-dynamics (3), where the applicable inertia consists both of the wheel inertias $\boldsymbol{J}$ (that are considered in the wheel model) and the robot inertia $\boldsymbol{M}$ (which is not considered).

The natural question arises whether that can be taken into account. For a rigid wheel geometry, the wheel Jacobian $\boldsymbol{W}$ depends only on the robot geometry and is therefore constant in the body frame, such that ${}^b\dot{\boldsymbol{v}} = \boldsymbol{W}^{-1}\dot{\boldsymbol{\omega}}$. Substituting the kinematics (1) into the dynamics (2) yields

$$\boldsymbol{M}^*\boldsymbol{W}^{-1}\dot{\boldsymbol{\omega}} + \dot{\phi}\boldsymbol{Q}\boldsymbol{W}^{-1}\boldsymbol{\omega} = \boldsymbol{W}^\mathrm{T}\boldsymbol{\tau}\,. \tag{7}$$

which can be rearranged to

$$\dot{\boldsymbol{\omega}} = \boldsymbol{W}\boldsymbol{M}^{*-1}\boldsymbol{W}^\mathrm{T}\boldsymbol{\tau} - \dot{\phi}\boldsymbol{W}\boldsymbol{M}^{*-1}\boldsymbol{Q}\boldsymbol{W}^{-1}\boldsymbol{\omega} \tag{8}$$

and results, after considering the friction of (6), in

$$\dot{\boldsymbol{\omega}} = \boldsymbol{W}\boldsymbol{M}^{*-1}\boldsymbol{W}^\mathrm{T}(\boldsymbol{\tau} - \mu_\mathrm{r}\boldsymbol{\omega}) - \dot{\phi}\boldsymbol{W}\boldsymbol{M}^{*-1}\boldsymbol{Q}\boldsymbol{W}^{-1}\boldsymbol{\omega}\,. \tag{9}$$

Note the difference between the scalar angular velocity ω of a single wheel as opposed to the vector-valued velocity of *all* wheels $\boldsymbol{\omega}$ (denoted by bold lettering).

Looking at the now more accurate wheel dynamic description leads to several critical conclusions that should be appreciated:

- The wheel loading depends on the direction of the commanded acceleration. That is a consequence of $\boldsymbol{W}\boldsymbol{M}^{*-1}\boldsymbol{W}^{\mathrm{T}}$ not being isotropic in wheel space.
- The wheel dynamics depend on the robot angular velocity.
- The dynamics of a single wheel are influenced by the torques of *all* wheels, i.e. all wheels are coupled.

There are various ways in which the wheel dynamics change during normal robot operation. In fairness it should be said that in general, the qualitative behavior of the system is still mostly linear. But the effects of acceleration direction and rotation can be significant and substantially change the behavior of the wheel dynamics from the perspective of the wheel speed controller.

4.2 The issues of decentralized control

While most teams have implemented inverse kinematics to determine desired wheel speeds, the robot dynamics are only rarely considered. The assumption is that sufficient performance of the underlying wheel speed controllers will result in the desired motion of the whole robot. The wheel speed controllers are most often in the form of heuristically tuned PI(D) controllers.

In general, an error feedback controller such as PID is reactive in nature and requires the presence of deviations before any control output is generated. It is therefore expected that with a constantly changing wheel speed reference $\omega^*(t)$ there will be some amount of delay until the actual wheel speed ω follows the desired command. This is the case even in the complete absence of any external disturbance, and the delay is strongly dependent on the tuning of the controller gains. Depending on the robot, the tuning process can be challenging, as the goal of trajectory tracking by pure feedback can require high gains, which in turn can result in issues with actuator saturation, windup, and instability. Thankfully, SSL teams employing this approach are mostly aware of these issues.

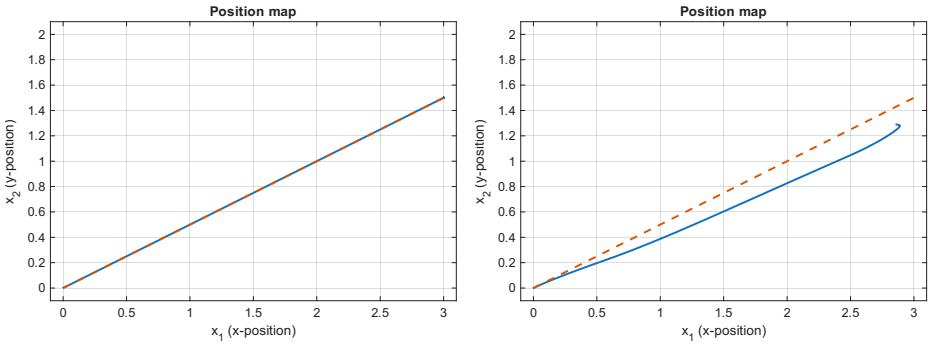
What is often critically underestimated, however, are the effects of the varying wheel dynamics that we have shown in the previous subsection. We do not see much discussion about that, and personal conversations during the last few RoboCups have shown that teams are generally unaware of this issue. Since the controller gains are fixed, the shifting dynamics depending on orientation, rotation, and acceleration direction inevitably lead to inconsistent system behavior of the individual wheels, especially after a change in setpoint. Without external information, these additional terms are unknowable from the perspective of a wheel controller.

As the magnitude of the wheel speeds relative to one another begins to change, the kinematics of the robot result in the speed of the robot differing not only in magnitude but also in direction. Furthermore, the imbalance of the wheel speeds then also results in a rotation of the robot, as equilibrium between the wheels is no longer maintained. Until that drift is corrected, the robot will then accelerate in the wrong direction. In the absence of vision data and IMU onboard sensor fusion, this can only be compensated after the whole vision-to-robot delay (consisting of vision processing delay, tracking filter time constants, AI processing,

and radio delay), resulting in a visible deviation during games. Once the team's AI issues a corrective command, the same problem applies again, resulting in a fishtailing zig-zagging motion of the robot that we observe in many teams.

This effect can even be seen in simulation in the absence of disturbance. To see how we investigate two elementary bang-bang trajectories that accelerate the robot in a straight line. For the first trajectory, the robot orientation is fixed, while during the second trajectory, the robot is also supposed to rotate. For the simulation, the wheel speed setpoints are determined by the kinematic model (1). A heuristically tuned PI controller determines the applied wheel torques, given the deviation between the desired and current wheel speed. The robot motion is then determined from the robot dynamics (2). We assume that there is no slip and no external disturbance.

The respective trajectories on the playing field are shown in figure 7. Despite the required robot movement being effectively identical in both cases, different only by rotation, the resulting robot motion is quite different from before. The designed controller tracks the translational command nearly perfectly, but the same tuning fails while rotating. This is despite the fact that for these kinds of robots, the wheel torques required for robot rotation are generally small compared to the wheel torques required for translation (as many teams demonstrate each year with wildly spinning robots).



(a) Pure translation (b) Translation with rotation (180°)

Fig. 7: Comparison of simulated robot trajectories under identical translational commands. Reference trajectory is shown in dashed orange.

Of course, in regular implementations the trajectories would be quickly overwritten after some delay, depending on the respective team's update rate. These plots just show the inability of the structure to follow through on a fixed trajectory.

This is the main point of critique of that control structure and the major takeaway: **The decentralized wheel control approach makes the robot trajectory tracking performance unreliable, as the speed controller's**

performance is wildly inconsistent across operating points. While this can be somewhat mitigated by good controller tuning, it is a fundamental issue of this design that is worth addressing.

4.3 Better approaches

After understanding the core issue, we can now explore solution approaches. While a sheer infinite amount of possibilities exist, we try to limit ourselves to easily implementable approaches so that teams are encouraged to actually try them for themselves. For this reason, we assume that no IMU is available [3] and the structure of decentralized velocity control is maintained.

While we believe a centralized position or velocity control approach to be superior, the decentralized approach can be extended to address most of the mentioned adverse effects. An elegant way to mitigate them is the implementation of feedforward control. Using a model of the system, a control sequence $u^*(t)$ can be determined that would follow the desired trajectory of $\omega^*(t)$ exactly. The PID controller then only compensates for actual disturbances and model inaccuracies. This separates the concerns of setpoint tracking and disturbance rejection and also leads to an overall easier tuning procedure. In particular, the effect of the robot dynamics can be mostly compensated for such that the wheel dynamics are then predominantly described by the free-wheeling wheel dynamics (6) with additional disturbance. We have used this approach successfully up until and including RoboCup Eindhoven 2024, after which we switched to a position control architecture including on-board sensor fusion of vision and IMU data.

A simple and straightforward model to determine feedforward currents can be obtained by applying Newton's laws in the s-frame

$$ {}^s\dot{\boldsymbol{v}} = \underbrace{\begin{bmatrix} \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{J_\mathrm{r}} \end{bmatrix}}_{\boldsymbol{M}^{-1}} \underbrace{\begin{bmatrix} {}^sf_\mathrm{x} \\ {}^sf_\mathrm{y} \\ M \end{bmatrix}}_{{}^s\boldsymbol{F}} \tag{10} $$

Note that this simple expression only works in the s-frame, as the body frame is not inertial. The wrench-vector ${}^s\boldsymbol{F}$ results from the wheel torques and can be determined by

$$ {}^s\boldsymbol{F} = \underbrace{\begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{R}(\phi)} \underbrace{\boldsymbol{W}^\mathrm{T} \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}}_{{}^b\boldsymbol{F}} \tag{11} $$

where $\boldsymbol{R}(\phi)$ maps body-frame wrenches into the inertial frame. For a given continuously differentiable wheel speed trajectory (i.e. the desired acceleration ${}^s\dot{\boldsymbol{v}}^*$ is available or can be determined) the robot dynamics (10) can be algebraically rearranged to yield the required feedforward wheel torques

$$ \boldsymbol{\tau} = \boldsymbol{W}^{\mathrm{T}\,-1} \boldsymbol{R}(\phi)^\mathrm{T} \boldsymbol{M}\, {}^s\dot{\boldsymbol{v}}^* \,. \tag{12} $$

[3] If you *do* have one, see our remarks at the end of this section.

Intuitively, the required force is determined according to the desired acceleration and inertia of the robot ($\boldsymbol{M}\,{}^{s}\dot{\boldsymbol{v}}^{*}$), which is then transformed in the robot's frame of reference ($\boldsymbol{R}(\phi)$) and then distributed to the wheels according to the geometric configuration of the robot ($\boldsymbol{W}^{\mathrm{T}\,-1}$).

It is important to note that this has to be determined centrally and assumes the availability of the robot orientation. If the orientation is not available, the approach can still work, as long as there is little overlapping acceleration and rotation. We have used this until RoboCup 2023, and it can serve as a reasonable approximation in a pinch. To do it properly, however, requires the availability of orientation or consideration of the Coriolis effect. With an available gyrometer, the angular rate and orientation can be estimated quite well, making this approach even easier.

The wheel controllers can also benefit from feedforward control. Using the simplified model from (6) yields the required feedforward current

$$u^{*} = \frac{J_{\mathrm{w}}\dot{\omega}^{*} + \mu_{\mathrm{r}}\omega^{*}}{k_{\mathrm{t}}}\,. \tag{13}$$

If during implementation you struggle matching the feedforward currents to your actual robot model, try recording the torques that your wheel controllers produce. These give a sanity check for the required scale of feedforward torques for a given speed and acceleration and can help to verify your model as well as give a good approximation of friction parameters.

With these two feedforward terms, large parts of the model are already accounted for, and, depending on the accuracy of the model parameters, the setpoints are already being followed decently even in the absence of any control action. That relegates the controllers to compensate for the remaining modeling inaccuracies and disturbances, which is a much simpler task. By implementing this approach, struggling teams should be able to observe vastly improved robot driving behavior and may even find that higher accelerations and speeds can now be (safely) achieved. In general, the full dynamics (9) are also solvable for $\boldsymbol{\tau}$, but we have not done so on our real robots and can't comment on effectiveness. The Newtonian model proved good enough, and we have since abandoned this decentralized architecture altogether in favor of centralized position control.

A quick note on IMUs : While our approach does not require them, teams that have them available have come up with various solution attempts that we wish to quickly comment on. Some teams have realized their zig-zagging issue and have implemented a separate, central control loop that immediately generates corrective speed commands if a deviation, such as an unwanted robot angular rate, is detected. This approach *can* improve the issue, but it is bad practice. There are no additional dynamics between the robot velocity and the wheel velocities, only kinematic constraints assuming no slip. Therefore, the wheel speed and central controllers are regulating the same process variable and may end up "fighting" each other. If you wish to use an IMU this way, consider a centralized control structure concerning solely the body velocity of the robot as

a process variable of interest and using the wheel torques as control input. The individual wheel speeds would *not* be controlled directly in this case. This leads nicely into the second currently employed approach, onboard position control (as opposed to speed control). This can compensate for all the mentioned deviations nicely but is much harder to implement, and we would not recommend it for new teams with limited manpower.

5 Conclusion

This paper describes improvements to our wheels and provides insight into the manufacturing of our dribbler suspension. The redesign of our wheels has resulted in a smoother ride. The number of fouls caused by parts on the playing field could also be reduced through a new manufacturing method for our dribbler suspensions. Moreover, this paper contains a design guide for developing motor drivers, detailing considerations, constraints, and lessons learned during the most recent design iteration of our motor drivers. This should help teams during their own motor driver designs and provide helpful ideas for streamlining their development process. Lastly, a common but flawed control structure was introduced and thoroughly investigated, after which an improved approach was presented. With the detailed implementation instructions, teams can easily adopt the feedforward control and see improvements in their driving behavior.

References

1. Andre Ryll, S.J.: TIGERs Mannheim (Team Interacting and Game Evolving Robots) Extended Team Description for RoboCup 2020. (2020)
2. Mercurians, T.D.: Delft Mercurians Team Description Paper RoboCup 2024. (2024)
3. Kolani, M.R., Behzadi, H.S., Onidin, M., Gharib, Y.A., MahdiMalverdi, M., Khalafi, K., Behzad, K., Khosravi, M.A.: PARSIAN 2020 Extended Team Description Paper. (2020)
4. Ohno, S., Naito, Y., Mimura, T., Ohno, K., Tsuruta, Y., Mitsuoka, R., Sako, R., Watanabe, M., Sugiura, T.: KIKS Extended Team Description for RoboCup 2019. (2019)
5. Trinh, L., Anzuman, A., Batkhuu, E., Chan, D., Graf, L., Gurung, D., Jamal, T., Namgyal, J., Ng, J., Tsang, W.L., Wang, X.R., Yilmaz, E., Obst, O.: TurtleRabbit 2024 SSL Team Description Paper. (2024)
6. Engelhardt, T., Heineken, T., Kuehn, T., Lindner, J., Schmidt, M., Schofer, F., Seifert, C., Stadler, M., Wegmann, L., Wendler, A.: ER-Force 2011 Extended Team Description. (2011)
7. Cáceres, C., Rosário, J., Amaya, D.: Design, simulation, and control of an omnidirectional mobile robot. International Review of Mechanical Engineering (2018)
8. Palacín, J., Rubies, E., Clotet, E., Martínez, D.: Evaluation of the path-tracking accuracy of a three-wheeled omnidirectional mobile robot designed as a personal assistant. Sensors (2021)
9. Neto, A.A.F., Freiberger, A., de Souza, A.C., da Silva, A.D.F., Yamamoto, B.D., dos Santos, J.P.S., Lima, J.M.A., dos Santos, J.V., Neves, L.G., de Almeida Santos, M.R., et al.: Itandroids small size league team description paper for robocup 2025. (2025)

10. Araujo, E., Paixao, M., Andrade, M., Xavier, D., Barros, E.: Robôcin small size league extended team description paper for robocup 2024. (2024)
11. Agulló, J., Cardona, S., Vivancos, J.: Kinematics of vehicles with directional sliding wheels. Mechanism and Machine Theory (1987)
12. Agullo, J., Cardona, S., Vivancos, J.: Dynamics of vehicles with directionally sliding wheels. Mechanism and Machine Theory (1989)