To be published in Robotics and Autonomous Systems, 2018, 106, pp. 58-68,which should be cited to refer to this work.

DOI: 10.1016/j.robot.2018.03.014

# Instantaneous Centre of Rotation Based Motion Control for Omnidirectional Mobile Robots with Sidewards Off-centred Wheels

Lionel Clavien[a,*], Michel Lauria[b], François Michaud[a]

[a] *Université de Sherbrooke, Sherbrooke QC, Canada*
[b] *University of Applied Sciences Western Switzerland (HES-SO), Geneva, Switzerland*

**Abstract**

AZIMUT-3 is a nonholonomic omnidirectional platform design using sidewards off-centred compliant wheels. This design makes it possible to experiment with the use of the chassis' instantaneous centre of rotation (ICR) for motion control. Research on ICR-based motion controllers has focused on handling structural singularities and misses a more general consideration of the chassis' kinematic and physical constraints like steering, velocity and acceleration constraints. This paper presents the design of an ICR-based motion controller for AZIMUT-3. Leveraging a new parametrization of the motion state space and the associated representation in $\mathbb{R}^3$ (collectively referred to as the H representation) and adapting a time scaling principle initially developed for manipulator trajectories, the designed motion controller is able to handle actuators coordination and their physical limits, as well as structural singularities. Results of tests done with the platform are presented, demonstrating the applicability of the proposed motion controller in efficiently handling these issues.

*Keywords:* instantaneous centre of rotation (ICR), nonholonomic omnidirectional robots, motion control, kinematics, wheeled robots

## 1. Introduction

Nowadays, a variety of robotic platforms are using omnidirectional locomotion mechanisms. Many use omnidirectional wheels like the Mecanum wheel [1] used on ARMAR-III [2] or the Segway RMP [3], which provide true omnidirectional motion but bring limitations in terms of odometry precision, vibrations and obstacle crossing [4]. Another common solution is to use steerable wheels, which provides accurate odometry and lower mechanical complexity [5]. Different types of steerable wheels have been designed, each with specific kinematic properties. The centred wheel is the simplest type and is used by HERMES [6] and Rollin' Justin [7, 8]. The caster wheel is the most common one, as used by Corsero [9], Dynamaid [10], Meka B1 [11] and Willow Garage PR2 [12], and enables pseudo-holonomic motion. Both have in common a steering axis lying within the rolling plane of the wheel. A third type is the sidewards off-centred steerable wheel, used by platforms based on Neobotix MPO-700 [13] (like Care-O-bot 3 [14, 15], DESIRE [16] and [17]) and Johnny-0 [18, 19], called AZIMUT wheel [20]. As illustrated by

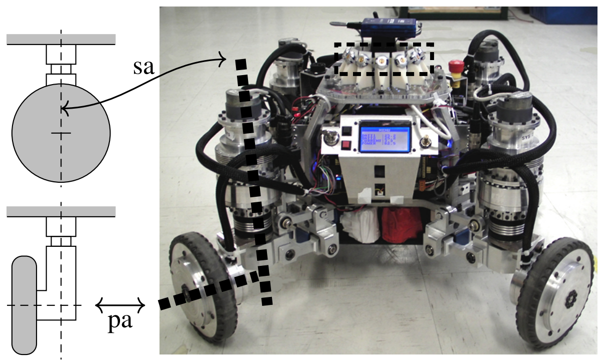


Figure 1: The AZIMUT-3 platform alongside an illustration of an AZIMUT wheel with its steering axis (sa) and propulsion axis (pa). Also visible are Optotrack wireless optical trackers (dashed rectangle) placed on top of the platform.

Fig. 1 and unlike the caster wheel, the steering axis of an AZIMUT wheel lies outside of its rolling plane. This creates a kinematic coupling between steering and propulsion, i.e., any steering motion needs an appropriate propulsion motion, even without chassis motion.

Employing steerable wheels leads however to nonholonomic platform designs, and the actuators then need to be carefully coordinated to guarantee safe and precise mo-

*Corresponding author
*Email address:* `l.clavien@ieee.org` (Lionel Clavien)

tion [5]. Using the chassis' instantaneous centre of rotation (ICR) and the motion around it is well suited to handle that actuator coordination for platforms equipped with centred and AZIMUT wheels [17, 21, 22, 23]. Amongst other advantages, it provides independent control inputs that can be directly mapped to the platform's degrees of freedom, which should provide simple and efficient kinematic models, easing motion control.

Also shown on Fig. 1, AZIMUT-3 is a nonholonomic omnidirectional platform equipped with four AZIMUT wheels. It is used as the mobile base of Johnny-0. Use of wheels with a sidewards offset was motivated by three reasons: 1) AZIMUT's first prototype used tracks instead of wheels, providing multi-modal locomotion capabilities [24]; 2) using compliant elastic actuators [25] for wheel steering allows the platform to sense wrenches applied on the platform [26] if such offset is present; and 3) it helps lowering the platform's centre of gravity, considering that we planned to use the platform with a humanoid torso [19]. This low centre of gravity design implies however that the wheels cannot rotate completely around their steering axis. AZIMUT-3 has therefore the limitation of having to stop for the wheels to be reoriented when they come too close to the chassis [27]: they then must do a 180° rotation to continue with the intended trajectory. This process is referred to as a wheel reconfiguration. In addition, steering and propulsion actuators cannot provide infinite velocity or infinite acceleration and thus set inherent limits on a wheel's motion.

Handling all these constraints is a critical part of motion coordination, usually handled at planning time [27, 28] or during trajectory tracking [29] using complex kinematic models. A more generic solution would be to take these constraints into consideration at the motion control level. For instance, Connette et al. [30] take actuators' physical limits into account at the motion control level using predictive potential fields, but only to ensure bounds on the steering velocities. Schwesinger et al. [22] and Sorour et al. [17] take both steering velocity and acceleration bounds into account by constraining the ICR velocity and acceleration, but do not take into account bounds on the propulsion.

In this paper, we present the design of an ICR-based realtime motion controller for AZIMUT-3. Using a trajectory time-scaling principle introduced by Hollerbach for industrial robotic manipulators [31], our motion controller outputs commands that satisfy both velocity and acceleration limits related to all actuators, taking into account the kinematic coupling of the wheels. It also handles the steering limitations and structural singularities of the platform.

The paper is organized as follows. Section 2 presents the state space representation used for the chassis' motion and the kinematic models of AZIMUT-3. Section 3 then details the proposed motion controller and methods to address handling of actuators' physical constraints and platform singularities. Section 4 presents results obtained using AZIMUT-3.

## 2. Motion Modelling

The design of a motion controller starts by defining a motion model of the platform to associate motion of the chassis with the individual actuators. Actuators are usually controlled in position or velocity and this model is thus based on the platform's kinematics. Since the model depends on how the motion is represented, the representation used is first described, followed by the kinematic models of AZIMUT-3.

### *2.1. ICR-based Motion Representation*

With omnidirectional robots, the ICR position may be located anywhere in the motion plane [32]. Two-dimensional Cartesian ($\mathbb{R}^2$) or polar ($\mathbb{R} \times \mathbb{S}$) coordinates are then natural choices for its parametrization, but this leads to parametrization-induced singularities [33]. Motion control being based on the kinematic models of the robot, which depend on the parametrization chosen for its motion, these parametrization-induced singularities needlessly complexify motion control.

This is why a different parameterization of the motion state space and a corresponding visualization, collectively referred to as the H representation and illustrated on Fig. 2, have been introduced in previous works [23, 27, 34]. The main difference with other ICR-based representations [33, 35] is that the configuration space for the ICR position is based on $\mathbb{P}_2(\mathbb{R})$ (the real projective plane) instead of $\mathbb{S}^2$ or $\mathbb{R}^2$. To ease visualization and computations, it is represented in $\mathbb{R}^3$ as the set $\Lambda$ of identified antipodal points on a unit sphere centred at the origin. The sphere is parametrized with three-dimensional Cartesian coordinates (noted $(u\ v\ w)$ in the $\{O, U, V, W\}$ frame), which leads to a singularity-free parametrization: any continuous motion of the chassis corresponds to a continuous trajectory in $\Lambda$. The motion around the ICR, parametrized by $\mathbb{R}$ and noted $\mu$, may be represented by the spin of the oriented line going through the antipodal points. The full configuration space for the chassis motion is thus $\mathrm{H} = \Lambda \times \mathbb{R}$ and one of its elements is noted $\boldsymbol{\eta} = (\boldsymbol{\lambda}\ \mu)^T$, with $\boldsymbol{\lambda} = (u, v, w)^T$. $\boldsymbol{\lambda}^+$ and $\boldsymbol{\lambda}^-$ are used to differentiate two antipodal points from each other when needed.

Even though other ICR-based motion controllers exist [33, 35], to our knowledge no motion planner or trajectory tracker directly generates ICR-based commands instead of twist-based commands. Hence, conversions between the two representations are needed. Converting a twist $\dot{\boldsymbol{\xi}} = (\dot{x}\ \dot{y}\ \dot{\theta})^T$ into an $\boldsymbol{\eta}$ can be done by considering the geometry of motion as pictured on Fig. 2. Since the chassis

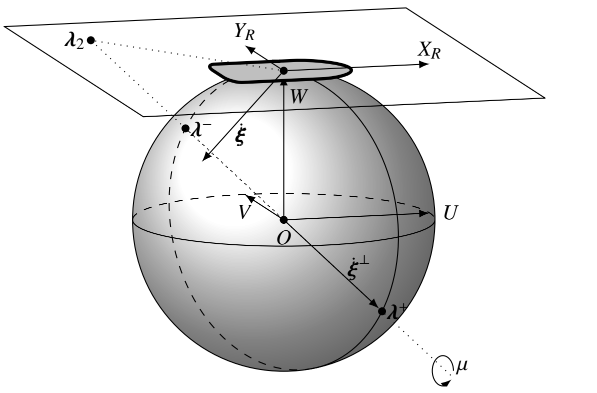


Figure 2: H representation with twist $\dot{\boldsymbol{\xi}}$ and ICR position on the motion plane $\boldsymbol{\lambda}_2$.

is a rigid body, its heading (given by the twist direction) is perpendicular to the line linking the ICR position $\boldsymbol{\lambda}_2$ to the chassis' centre. The vectors $\boldsymbol{\lambda}$ and $\dot{\boldsymbol{\xi}}^\perp = (-\dot{y}\ \dot{x}\ \dot{\theta})^T$ ($\dot{\boldsymbol{\xi}}$ rotated 90° around the $W$ axis) are thus collinear. In addition, motion around the ICR is directly proportional to the amplitude of the twist. Conversion from $\dot{\boldsymbol{\xi}}$ to $\boldsymbol{\eta}$ is thus given by:

$$\boldsymbol{\eta} = \frac{1}{||\dot{\boldsymbol{\xi}}||}\begin{pmatrix}-\dot{y} & \dot{x} & \dot{\theta} & ||\dot{\boldsymbol{\xi}}||^2\end{pmatrix}^T \tag{1}$$

The reverse conversion is given by:

$$\dot{\boldsymbol{\xi}} = \mu\begin{pmatrix}v & -u & w\end{pmatrix}^T \tag{2}$$

Equation (2) has no singularities and (1) only one structural singularity: if there is no motion, $\dot{\boldsymbol{\xi}}$ is null and no ICR can be computed out of it.

### 2.2. *Kinematic Models*

Under the assumption that a wheel may be modelled as a rigid body with no sliding at the contact point, a steerable wheel is subject to two kinematic constraints: pure rolling in the wheel's motion direction, and no sliding in the perpendicular direction to that motion. By using the twist representation, Campion *et al.* [32] have studied the kinematics of two specific variants of a steerable wheel, namely the centred and caster wheels. Using the H representation, we extend this study to the AZIMUT wheel, which leads to the definition of AZIMUT-3 inverse and direct kinematic models.

#### 2.2.1. *Kinematic Constraints*

Figure 3 illustrates the parameters of the AZIMUT wheel with radius $r$ and rotational velocity $\dot{\varphi}$. There are four important frames of reference related to the wheel: the inertial frame $\{I, X_I, Y_I\}$; the chassis' frame $\{R, X_R, Y_R\}$; the frame

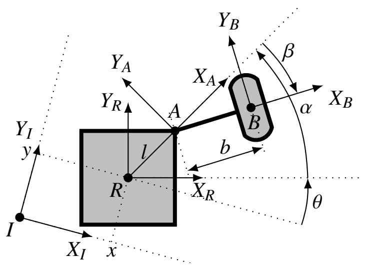


Figure 3: Geometry of the chassis and steering mechanism.

$\{A, X_A, Y_A\}$ associated with the steering axis and making an angle $\alpha$ with $\{R, X_R, Y_R\}$); and the frame $\{B, X_B, Y_B\}$ associated with the wheel centre $B$ and making an angle $\beta$ with $\{A, X_A, Y_A\}$.

Let us define some shorthand notation. An orthogonal matrix representing a rotation of angle $\delta$ is expressed in $\mathbb{R}^2$ as:

$$\boldsymbol{M}_2(\delta) = \begin{bmatrix} \cos\delta & -\sin\delta \\ \sin\delta & \cos\delta \end{bmatrix} \tag{3}$$

Let us also define the following vectors in $\mathbb{R}^3$:

$$\boldsymbol{a} = (\cos\alpha\ \ \sin\alpha\ 0)^T \tag{4a}$$
$$\boldsymbol{a}^\perp = (-\sin\alpha\ \ \cos\alpha\ 0)^T \tag{4b}$$
$$\boldsymbol{s} = (l\cos\alpha\ l\sin\alpha\ 1)^T \tag{4c}$$
$$\boldsymbol{b} = (0\ 0\ b)^T \tag{4d}$$
$$\boldsymbol{l} = (0\ 0\ l)^T \tag{4e}$$

$\boldsymbol{a}$ and $\boldsymbol{a}^\perp$ are the orthogonal projections on the sphere's equatorial plane of the basis vectors of $\{A, X_A, Y_A\}$, and $\boldsymbol{b}$ and $\boldsymbol{l}$ are the characteristic lengths of the wheel ($b = ||\boldsymbol{AB}||$, $l = ||\boldsymbol{RA}||$) reported along the $W$ axis (shown on Fig. 2). The vectors $(\boldsymbol{a} - \boldsymbol{l})$ and $\boldsymbol{a}^\perp$ define an orthogonal basis of the plane going through the origin of the sphere and perpendicular to the line through $A$, whose direction vector is $\boldsymbol{s}$, i.e.,

$$(\boldsymbol{a} - \boldsymbol{l}) \cdot \boldsymbol{a}^\perp = 0 \quad \text{and} \quad (\boldsymbol{a} - \boldsymbol{l}) \times \boldsymbol{a}^\perp = \boldsymbol{s} \tag{5}$$

Thus, any linear combination of them lies in that plane, in particular the two vectors:

$$\boldsymbol{s}_1^\perp(\beta) = \sin\beta(\boldsymbol{a} - \boldsymbol{l}) - \cos\beta\boldsymbol{a}^\perp \tag{6a}$$
$$\boldsymbol{s}_2^\perp(\beta) = \cos\beta(\boldsymbol{a} - \boldsymbol{l}) + \sin\beta\boldsymbol{a}^\perp \tag{6b}$$

As those vectors are orthogonal, the following holds:

$$\dot{\boldsymbol{s}}_1^\perp(\beta) = \dot{\beta}\boldsymbol{s}_2^\perp(\beta) \quad \text{and} \quad \dot{\boldsymbol{s}}_2^\perp(\beta) = -\dot{\beta}\boldsymbol{s}_1^\perp(\beta) \tag{7}$$

The two kinematic constraints may be expressed by setting the coordinates of the velocity vector $\boldsymbol{v_B}$ of point $B$

in $\{B, X_B, Y_B\}$ to respectively 0 (no lateral sliding) and $-r\dot{\varphi}$ (pure rolling):

$$^{B}\boldsymbol{v_B} = \boldsymbol{M}_2^{-1}(\alpha + \beta)^{R}\boldsymbol{v_B} = \begin{pmatrix} 0 \\ -r\dot{\varphi} \end{pmatrix} \tag{8}$$

Under the rigid body assumption, the same velocity in $\{R, X_R, Y_R\}$ may be computed as:

$$^{R}\boldsymbol{v_B} = \frac{d}{dt}{}^{R}\boldsymbol{IB} = \frac{d}{dt}{}^{R}\boldsymbol{IR} + \frac{d}{dt}{}^{R}\boldsymbol{RA} + \frac{d}{dt}{}^{R}\boldsymbol{AB} \tag{9}$$

and (8) becomes:

$$\boldsymbol{M}_2^{-1}(\alpha + \beta)\boldsymbol{M}_2^{-1}(\theta)\begin{pmatrix} {}^{I}\dot{x} \\ {}^{I}\dot{y} \end{pmatrix} + \dot{\theta}\boldsymbol{M}_2^{-1}(\beta)\begin{pmatrix} 0 \\ l \end{pmatrix} + (\dot{\theta} + \dot{\beta})\begin{pmatrix} 0 \\ b \end{pmatrix} = \begin{pmatrix} 0 \\ -r\dot{\varphi} \end{pmatrix} \tag{10}$$

which, by using (1) and (6), may be rewritten in the H representation as:

$$\begin{bmatrix} \boldsymbol{s}_1^{\perp}(\beta)^T \\ (\boldsymbol{s}_2^{\perp}(\beta) - \boldsymbol{b})^T \end{bmatrix} \boldsymbol{\lambda}\mu = \begin{bmatrix} 0 \\ b\dot{\beta} + r\dot{\varphi} \end{bmatrix} \tag{11}$$

The system (11) accepts two sets of solutions for its first equation: $\mu = 0$, meaning that if there is no motion, there cannot be any sliding; and

$$\boldsymbol{s}_1^{\perp}(\beta) \cdot \boldsymbol{\lambda} = 0 \tag{12}$$

expressing the fact that if there is motion, it must be done with the ICR direction perpendicular to $\boldsymbol{s}_1^{\perp}(\beta)$, otherwise lateral sliding will occur. The motion of the chassis is thus constrained by the wheel and leads to the construction of nonholonomic robots. Since the steering angle is expressed solely as a function of the ICR position and the robot's geometry, the H representation is well suited to express the motion state of nonholonomic wheeled robots like AZIMUT. Indeed, by definition of $\boldsymbol{s}_1^{\perp}(\beta)$ in (6), the steering angle $\beta$ may be expressed as:

$$\tan\beta = \frac{\boldsymbol{a}^{\perp} \cdot \boldsymbol{\lambda}}{(\boldsymbol{a} - \boldsymbol{l}) \cdot \boldsymbol{\lambda}} \tag{13}$$

and (6) may in turn be expressed solely as a function of the ICR position and robot geometry:

$$\boldsymbol{s}_1^{\perp}(\beta) = \boldsymbol{s}_1^{\perp}(\boldsymbol{\lambda}) \quad \text{and} \quad \boldsymbol{s}_2^{\perp}(\beta) = \boldsymbol{s}_2^{\perp}(\boldsymbol{\lambda}) \tag{14}$$

### 2.2.2. *Inverse Kinematic Model*

Deriving twice (11) and using (7), (12) and (14) leads to:

$$\dot{\beta} = -\frac{\boldsymbol{s}_1^{\perp}(\boldsymbol{\lambda}) \cdot \dot{\boldsymbol{\lambda}}}{\boldsymbol{s}_2^{\perp}(\boldsymbol{\lambda}) \cdot \boldsymbol{\lambda}} \tag{15a}$$

$$\ddot{\beta} = -\frac{2\dot{\beta}\boldsymbol{s}_2^{\perp}(\boldsymbol{\lambda}) \cdot \dot{\boldsymbol{\lambda}} + \boldsymbol{s}_1^{\perp}(\boldsymbol{\lambda}) \cdot \ddot{\boldsymbol{\lambda}}}{\boldsymbol{s}_2^{\perp}(\boldsymbol{\lambda}) \cdot \boldsymbol{\lambda}} \tag{15b}$$

$$\dot{\varphi} = \frac{(\boldsymbol{s}_2^{\perp}(\boldsymbol{\lambda}) - \boldsymbol{b}) \cdot \boldsymbol{\lambda}\mu - b\dot{\beta}}{r} \tag{15c}$$

$$\ddot{\varphi} = \frac{(\boldsymbol{s}_2^{\perp}(\boldsymbol{\lambda}) - \boldsymbol{b}) \cdot (\dot{\boldsymbol{\lambda}}\mu + \boldsymbol{\lambda}\dot{\mu}) - b\ddot{\beta}}{r} \tag{15d}$$

Equations (15a) and (15b) associate motion of the ICR with the corresponding steering velocity and acceleration, while (15c) and (15d) associate motion around the ICR with the corresponding propulsion velocity and acceleration. These equations define the inverse kinematic model of the AZIMUT wheel, where $\dot{\boldsymbol{\lambda}}$, $\ddot{\boldsymbol{\lambda}}$ and $\dot{\mu}$ are the inputs and $\boldsymbol{\lambda}$ and $\mu$ are the states.

The coupling induced by the $b$ offset of the AZIMUT wheel is highlighted by (15c) and (15d). Indeed, (15c) may be rewritten as:

$$\dot{\varphi} = \dot{\varphi}_p + \dot{\varphi}_s \tag{16}$$

with

$$\dot{\varphi}_p = \frac{(\boldsymbol{s}_2^{\perp}(\boldsymbol{\lambda}) - \boldsymbol{b}) \cdot \boldsymbol{\lambda}\mu}{r} \quad \text{and} \quad \dot{\varphi}_s = -\frac{b}{r}\dot{\beta} \tag{17}$$

$\dot{\varphi}_p$ gives the propulsion axis velocity needed to enable chassis motion and $\dot{\varphi}_s$ gives the contribution needed to accommodate a non-null steering velocity $\dot{\beta}$ without sliding. (15d) may be rewritten in the same way.

The system (15) becomes singular when:

$$\boldsymbol{s}_2^{\perp}(\boldsymbol{\lambda}) \cdot \boldsymbol{\lambda} = 0 \tag{18}$$

Since (12) must also hold, (18) is satisfied only when $\boldsymbol{\lambda}$ is simultaneously perpendicular to $\boldsymbol{s}_1^{\perp}(\boldsymbol{\lambda})$ and $\boldsymbol{s}_2^{\perp}(\boldsymbol{\lambda})$ which, by (5), means that $\boldsymbol{\lambda}$ is collinear with $\boldsymbol{s}$. The inverse kinematic model may thus be used as long as the ICR is not on the wheel's steering axis. Equation (13) has the same singularity: the steering angle cannot be uniquely determined when the ICR is on the steering axis. In the remaining of this section, the ICR is supposed not to be on a steering axis. This structural singularity is dealt with by the motion controller presented in Section 3.

The inverse kinematic model of a wheeled robot is made of the set of its wheels' inverse kinematic models. By considering only the actuators velocity and using the fact that the wheels have the same geometry, the resulting inverse kinematic model for AZIMUT-3 may be written as:

$$\begin{bmatrix} \dot{\boldsymbol{\beta}} \\ \dot{\boldsymbol{\varphi}} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{C} & \boldsymbol{0}_{4\times1} \\ \frac{b}{r}\boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\lambda}} \\ \mu \end{bmatrix} = \boldsymbol{J}(\boldsymbol{\lambda}) \begin{bmatrix} \dot{\boldsymbol{\lambda}} \\ \mu \end{bmatrix} \tag{19}$$

where

$$\boldsymbol{C} = \begin{bmatrix} \frac{1}{\boldsymbol{s}^{\perp}_{2,1}(\boldsymbol{\lambda})\cdot\boldsymbol{\lambda}} \boldsymbol{s}^{\perp}_{1,1}(\boldsymbol{\lambda})^T \\ \vdots \\ \frac{1}{\boldsymbol{s}^{\perp}_{2,4}(\boldsymbol{\lambda})\cdot\boldsymbol{\lambda}} \boldsymbol{s}^{\perp}_{1,4}(\boldsymbol{\lambda})^T \end{bmatrix} \quad \text{and} \quad \boldsymbol{D} = \begin{bmatrix} \frac{(\boldsymbol{s}^{\perp}_{2,1}(\boldsymbol{\lambda})-\boldsymbol{b})\cdot\boldsymbol{\lambda}}{r} \\ \vdots \\ \frac{(\boldsymbol{s}^{\perp}_{2,4}(\boldsymbol{\lambda})-\boldsymbol{b})\cdot\boldsymbol{\lambda}}{r} \end{bmatrix} \tag{20}$$

and $\dot{\boldsymbol{\beta}}$ and $\dot{\boldsymbol{\varphi}}$ represent the set of $\dot{\beta}$ and $\dot{\varphi}$ for all wheels. $\boldsymbol{s}^{\perp}_{1,k}(\boldsymbol{\lambda})$ and $\boldsymbol{s}^{\perp}_{2,k}(\boldsymbol{\lambda})$ are the vectors $\boldsymbol{s}^{\perp}_{1}(\boldsymbol{\lambda})$ and $\boldsymbol{s}^{\perp}_{2}(\boldsymbol{\lambda})$ for wheel $k$.

$\boldsymbol{J}$ is expressed solely as a function of the ICR position and the robot's geometry; the model is thus implementable and usable to control the robot.

### 2.2.3. *Direct Kinematic Model*

Since $\boldsymbol{J}$ is full rank, it may be pseudoinverted to find $\dot{\boldsymbol{\lambda}}$ and $\mu$ from the sensors' readings. It requires having direct access to the measure of the steering velocities $\dot{\boldsymbol{\beta}}$. This is however not the case with AZIMUT-3, where only the steering angles $\boldsymbol{\beta}$ are available. Yet, observation of the structure of (19) shows that the coupling induced by the AZIMUT wheel introduces redundancy in the model. Using the properties of the H representation, i.e., that motion of a point on a sphere is constrained to be tangential to its current position, this redundancy may be replaced by a single constraint:

$$\boldsymbol{\lambda} \cdot \dot{\boldsymbol{\lambda}} = 0 \tag{21}$$

(19) thus reduces to:

$$\begin{bmatrix} 0 \\ \dot{\boldsymbol{\varphi}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\lambda}^T & 0 \\ \frac{b}{r}\boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\lambda}} \\ \mu \end{bmatrix} = \boldsymbol{K}(\boldsymbol{\lambda}) \begin{bmatrix} \dot{\boldsymbol{\lambda}} \\ \mu \end{bmatrix} \tag{22}$$

Since AZIMUT-3 has four wheels, the system is overdetermined in the considered case. It may then be solved for $\dot{\boldsymbol{\lambda}}$ and $\mu$ by using a standard least squares method to get AZIMUT-3's direct kinematic model. As $\boldsymbol{K}$ depends on $\boldsymbol{\lambda}$, the model takes for granted that the ICR is already known from the sensors' readings. ICR estimation for platforms using centred and AZIMUT wheeels can be done reliably in real-time [23, 36].

## 3. Motion Control for AZIMUT-3

To emphasize the role of a motion controller, Fig. 4 sketches the global control architecture used on AZIMUT-3. The motion controller receives commands for a desired motion state (given as a motion around an ICR $\boldsymbol{\eta}_d$ or a twist $\dot{\boldsymbol{\xi}}_d$) from a high-level controller described in [18] and sends motion commands ($\boldsymbol{\beta}_c$ and $\dot{\boldsymbol{\varphi}}_c$) to the independent steering and propulsion low-level controllers for each wheel. It also gets from the low-level controllers the measured steering and propulsion state ($\boldsymbol{\beta}_m$ and $\dot{\boldsymbol{\varphi}}_m$) and uses it to estimate the current platform state $\boldsymbol{\eta}_e$ and to compute the odometry $\boldsymbol{\xi}_e$, which are sent back to the high-level controller.

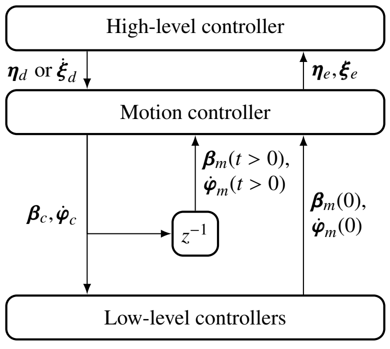

High-level controller
$\boldsymbol{\eta}_d$ or $\dot{\boldsymbol{\xi}}_d$
$\boldsymbol{\eta}_e, \boldsymbol{\xi}_e$
Motion controller
$\boldsymbol{\beta}_m(t>0)$,
$\dot{\boldsymbol{\varphi}}_m(t>0)$
$\boldsymbol{\beta}_c, \dot{\boldsymbol{\varphi}}_c$
$z^{-1}$
$\boldsymbol{\beta}_m(0)$,
$\dot{\boldsymbol{\varphi}}_m(0)$
Low-level controllers

Figure 4: AZIMUT-3 global control architecture, with $z^{-1}$ representing the feedforward model used.

To avoid input filtering that would lower command responsiveness, the current implementation of the motion controller uses a feed-forward configuration, with the platform's actions modelled as an one-step delay:

$$\boldsymbol{\beta}_m(t) = \boldsymbol{\beta}_c(t - \Delta t) \tag{23a}$$

$$\dot{\boldsymbol{\varphi}}_m(t) = \dot{\boldsymbol{\varphi}}_c(t - \Delta t) \tag{23b}$$

where $\Delta t$ is the control step duration. The measured steering and propulsion configuration ($\boldsymbol{\beta}_m(0)$ and $\dot{\boldsymbol{\varphi}}_m(0)$) is read from the low-level controllers only at (re)initialization. The motion controller design, which ensures the commands $\boldsymbol{\beta}_c$ and $\dot{\boldsymbol{\varphi}}_c$ are guaranteed to be executable, and the performant low-level controllers available onboard AZIMUT-3 motivated this choice.

Three important questions arose when designing a motion controller for a platform like AZIMUT-3: Q1) how to handle structural singularities such as an ICR on a steering axis (18); Q2) how to handle differential constraints such as actuator velocity limits; and Q3) how to handle algebraic constraints such as steering limitations. Algorithm 1 gives an overview of one step of the motion control algorithm devised for AZIMUT-3, which addresses all these questions.

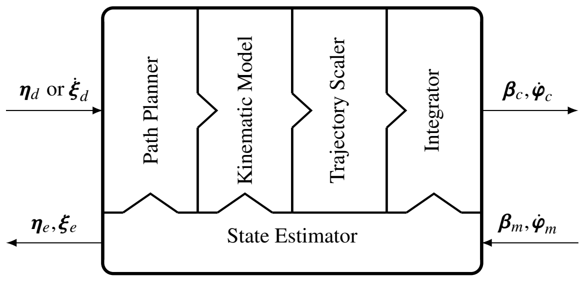
The motion controller has been split into five logical modules which are organized as pictured on Fig. 5 and are described in the following subsections.

### 3.1. *State Estimator*

The State Estimator module first determines the current ICR $\boldsymbol{\lambda}_e$ from $\boldsymbol{\beta}_m$ (using an adapted version of the algorithm

**Algorithm 1** Overview of a control step.

```
1: procedure DoControlStep($\boldsymbol{\beta}_m, \dot{\boldsymbol{\varphi}}_m, \boldsymbol{\lambda}_d, \mu_d, \Delta t$)
2:     $\boldsymbol{\lambda}_e \leftarrow$ EstimateLambda($\boldsymbol{\beta}_m$)
3:     $\mu_e \leftarrow$ EstimateMu($\dot{\boldsymbol{\varphi}}_m, \boldsymbol{\lambda}_e$)
4:     $\boldsymbol{\xi}_e \leftarrow$ ComputeOdometry($\boldsymbol{\lambda}_e, \mu_e, \Delta t$)
5:     $k_b \leftarrow 1, backtrack \leftarrow 1$
6:     while $backtrack$ do
7:         $(\boldsymbol{\lambda}', \boldsymbol{\lambda}'', \mu') \leftarrow$ ComputeChassisMotion($\boldsymbol{\lambda}_d, \boldsymbol{\lambda}_e, \mu_d, \mu_e, k_b$)
8:         $(\boldsymbol{\beta}', \boldsymbol{\beta}'', \dot{\boldsymbol{\varphi}}_p, \dot{\boldsymbol{\varphi}}'_p) \leftarrow$ ComputeActuatorsMotion($\boldsymbol{\lambda}', \boldsymbol{\lambda}'', \mu'$)
9:         $(\dot{s}_l, \dot{s}_u, \ddot{s}_l, \ddot{s}_u) \leftarrow$ ComputeScalingBounds($\boldsymbol{\beta}', \boldsymbol{\beta}'', \dot{\boldsymbol{\varphi}}_p, \dot{\boldsymbol{\varphi}}'_p$)
10:        if $\dot{s}_l \leq \dot{s}_u \wedge \ddot{s}_l \leq \ddot{s}_u$ then
11:            $backtrack \leftarrow 0$
12:        else
13:            $k_b \leftarrow$ UpdateBacktrackingParameter($k_b$)
14:        end if
15:    end while
16:    $(\dot{s}, \ddot{s}) \leftarrow$ ComputeScalingParameters($\dot{s}_l, \dot{s}_u, \ddot{s}_l, \ddot{s}_u$)
17:    $(\dot{\boldsymbol{\beta}}, \ddot{\boldsymbol{\beta}}, \ddot{\boldsymbol{\varphi}}_p) \leftarrow$ ScaleMotion($\boldsymbol{\beta}', \boldsymbol{\beta}'', \dot{\boldsymbol{\varphi}}'_p, \dot{s}, \ddot{s}$)
18:    $(\boldsymbol{\beta}_c, \dot{\boldsymbol{\varphi}}_c) \leftarrow$ IntegrateMotion($\dot{\boldsymbol{\beta}}, \ddot{\boldsymbol{\beta}}, \dot{\boldsymbol{\varphi}}_p, \ddot{\boldsymbol{\varphi}}_p, \Delta t$)
19: end procedure
```



$\boldsymbol{\eta}_d$ or $\dot{\boldsymbol{\xi}}_d$
Path Planner
Kinematic Model
Trajectory Scaler
Integrator
$\boldsymbol{\beta}_c, \dot{\boldsymbol{\varphi}}_c$
$\boldsymbol{\eta}_e, \boldsymbol{\xi}_e$
State Estimator
$\boldsymbol{\beta}_m, \dot{\boldsymbol{\varphi}}_m$

Figure 5: Logical organisation of the motion controller designed for AZIMUT-3. The arrows indicate the data flow.

described in [36], referred to as the Iterative Algorithm for ICR estimation in [23]) and computes the corresponding steer angles $\boldsymbol{\beta}_e$ using (13). Using $\dot{\boldsymbol{\varphi}}_m$, it then solves (22) to estimate $\mu_e$ and finally computes the odometry $\boldsymbol{\xi}_e = (x\ y\ \theta)^T$ using (2):

$$\boldsymbol{\xi}_e(t) = \boldsymbol{\xi}_e(t - \Delta t) + \boldsymbol{M}_3(\theta)\dot{\boldsymbol{\xi}}_e \Delta t \tag{24}$$

where $\boldsymbol{M}_3(\theta)$ means the three dimensional orthogonal matrix representing a rotation of angle $\theta$ in the $W$ plane. If $\boldsymbol{\lambda}_e$ is on steering axis $k$ (Q1), the estimation algorithm sets $\beta_{e,k}$ to $\beta_{m,k}$.

### *3.2. Path Planner*

The Path Planner module is the control entry point and implements the control laws.

When a new desired state is received, which may be set either by inputting a motion around an ICR $\boldsymbol{\eta}_d$ or a twist $\dot{\boldsymbol{\xi}}_d$ (internally converted to $\boldsymbol{\eta}_d$ using (1)), it is checked and adapted if needed:

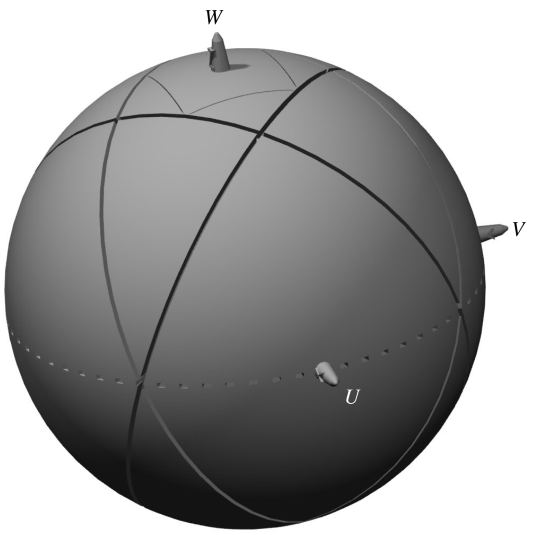


Figure 6: Spherical patches defined by the steering limitations of AZIMUT-3. The small spherical square at the top represents the chassis projection onto the sphere, each wheel being located at one of the corners.

- If $\boldsymbol{\lambda}_d$ is on a steering axis (Q1), $\boldsymbol{\eta}_d$ is discarded, ensuring that a chassis' rest configuration is never on a structural singularity.
- A null twist (Q1) is handled by setting $\boldsymbol{\lambda}_d$ to $\boldsymbol{\lambda}_e$ and $\mu_d$ to 0, ensuring smooth immobilization of the platform.
- Since the ICR is lying on a sphere with antipodal points identified, there are always two distinct paths between $\boldsymbol{\lambda}_e$ and $\boldsymbol{\lambda}_d$: $\boldsymbol{\lambda}_e$ to $\boldsymbol{\lambda}_d^+$ and $\boldsymbol{\lambda}_e$ to $\boldsymbol{\lambda}_d^-$. When a new command is received, the module needs to choose which path to follow. To be efficient, the shortest path should be chosen. However, each steering limitation (Q3) creates a great circle on the sphere and these great circles then define the boundaries of spherical patches, some of which are shown for AZIMUT-3 on Fig. 6. A continuous ICR motion between the patches leads to a discontinuous motion of at least one steering axis, which implies a platform stop and a wheel reconfiguration. Hence, this wheel reconfiguration cost is taken into account when choosing the path: if the shortest path leads to a wheel reconfiguration but not the second one, the second path is chosen; if both paths lead to a wheel reconfiguration, the shortest path is kept.
- $\mu$ is linearly proportional with $\dot{\varphi}_p$, like a rewrite of (17) shows:

$$\mu = \frac{r}{(\boldsymbol{s}_2^{\perp}(\boldsymbol{\lambda}) - \boldsymbol{b}) \cdot \boldsymbol{\lambda}} \dot{\varphi}_p = f(\boldsymbol{\lambda}) \dot{\varphi}_p \tag{25}$$

Since the ICR position lies on a closed surface (i.e., $f(\boldsymbol{\lambda})$ is bounded) and $\dot{\varphi}_p$ is bounded, then $\mu$ is also bounded:

$$\mu_{min}(\dot{\varphi}_{min}, \boldsymbol{\lambda}) \le \mu \le \mu_{max}(\dot{\varphi}_{max}, \boldsymbol{\lambda}) \tag{26}$$

Hence, a first step in handling propulsion velocity limits (Q3) is done when receiving a new command: if $\mu_d$ is not in the range (26), it is clamped to the nearest allowed limit.

Once a valid path between $\boldsymbol{\eta}_e$ and a new $\boldsymbol{\eta}_d$ has been determined or during normal path execution, the Path Planner module also computes the required commands $\boldsymbol{\lambda}'$, $\boldsymbol{\lambda}''$ and $\mu'$ (state variables derivatives) to reach $\boldsymbol{\eta}_d$ from $\boldsymbol{\eta}_e$. For $\mu'$, a simple proportional control law is used:

$$\mu' = k_b k_\mu (\mu_d - \mu_e) \tag{27}$$

where $k_\mu \ge 1$ is the proportional gain and $k_b$ a modulating factor described later and initialized to 1. Since the ICR moves on the surface of a sphere, its derivatives must comply to the following constraints:

$$\boldsymbol{\lambda}_e \cdot \boldsymbol{\lambda}' = 0 \quad \text{and} \quad \boldsymbol{\lambda}_e \cdot \boldsymbol{\lambda}'' = -||\boldsymbol{\lambda}'||^2 \tag{28}$$

Multiplying the derivatives by a common factor $k_\lambda \ge 1$ does not influence the constraints, and proportional control laws may thus also be used for $\boldsymbol{\lambda}'$ and $\boldsymbol{\lambda}''$. However, they should be designed to ensure convergence of $\boldsymbol{\lambda}_e$ to $\boldsymbol{\lambda}_d$, which needs care when working on a sphere. A possible solution is:

$$\boldsymbol{\lambda}' = k_b k_\lambda \left(\boldsymbol{\lambda}_d - (\boldsymbol{\lambda}_e \cdot \boldsymbol{\lambda}_d)\boldsymbol{\lambda}_e\right) \tag{29a}$$

$$\boldsymbol{\lambda}'' = k_b^2 k_\lambda^2 \left((\boldsymbol{\lambda}_e \cdot \boldsymbol{\lambda}_d)\boldsymbol{\lambda}_d - \boldsymbol{\lambda}_e\right) \tag{29b}$$

where the control error is given by the projection of $\boldsymbol{\lambda}_e$ on $\boldsymbol{\lambda}_d$, and the ICR moves along the great circle going through $\boldsymbol{\lambda}_e$ and $\boldsymbol{\lambda}_d$. The proposed control laws satisfy (28) and ensure convergence of $\boldsymbol{\eta}_e$ to $\boldsymbol{\eta}_d$. They also only involve the state variables and not their derivatives, which enables the use of time scaling in the Time Scaler module. Using only scalar products, they are also efficient and simple to implement.

## 3.3. Kinematic Model

Using (15), the Kinematic Model module transforms the chassis motion commands $\boldsymbol{\lambda}'$, $\boldsymbol{\lambda}''$ and $\mu'$ output by the Path Planner module into the corresponding motion for each actuator $\boldsymbol{\beta}'$, $\boldsymbol{\beta}''$, $\dot{\boldsymbol{\varphi}}_p$ and $\dot{\boldsymbol{\varphi}}'_p$. There are however situations where the model (15) cannot be used directly:

- When the ICR is on steering axis $k$ (Q1), part of the inverse kinematic model for wheel $k$ is undefined. In that case, both $\beta'_k$ and $\beta''_k$ are set to 0.
- When a wheel reconfiguration occurs, the ICR becomes undefined and so is the global inverse kinematic model. Handling this requires a two-step process:
  1. The platform must first be stopped by setting $\mu_d = 0$, as motion of the chassis is possible only with a defined ICR;
  2. Once the platform has stopped, a proportional control law is applied to each steering axis so as to reach the wheel configuration $\boldsymbol{\beta}_d$ corresponding to $\boldsymbol{\lambda}_d$:

  $$\boldsymbol{\beta}' = k_\beta(\boldsymbol{\beta}_d - \boldsymbol{\beta}_e) \tag{30}$$

  where $k_\beta$ is the proportional gain. $\boldsymbol{\beta}''$, $\dot{\boldsymbol{\varphi}}_p$ and $\dot{\boldsymbol{\varphi}}'_p$ are all set to $\mathbf{0}$.
- At controller initialization, the ICR could be undefined. This is handled similarly to a wheel reconfiguration: a proportional control law is applied to each steering axis so as to reach the wheel configuration $\boldsymbol{\beta}_e$:

  $$\boldsymbol{\beta}' = k_\beta(\boldsymbol{\beta}_e - \boldsymbol{\beta}_m) \tag{31}$$

To handle these different tasks and allow easy coordination with the other modules, the Kinematic Model module is built around a finite state machine. The other modules communicate events (new command received, wheel reconfiguration needed, etc.), which generate transitions between the different states.

## 3.4. Time Scaler

Due to the simplicity of the control laws used in the Path Planner module, the amplitude of the values generated by the Kinematic Model module might be arbitrarily high and are not guaranteed to be executable by the actuators. The Time Scaler module is thus used to scale $\boldsymbol{\beta}'$, $\boldsymbol{\beta}''$ and $\dot{\boldsymbol{\varphi}}'_p$ into $\dot{\boldsymbol{\beta}}$, $\ddot{\boldsymbol{\beta}}$ and $\ddot{\boldsymbol{\varphi}}_p$ such that differential constraints (Q2) on each actuator's motion are respected. To that end, a modified version of the trajectory time scaling method proposed in [31] is used: instead of scaling statically the whole motion at planning time, scaling is done dynamically at each control step. Time scaling is based on a fundamental property of differentiation:

$$\dot{x} = \frac{d}{dt}x(t) = \frac{d}{ds}x(s)\frac{d}{dt}s(t) = x'\dot{s} \tag{32}$$

i.e., a given trajectory $x(t)$ may be re-parametrized using $s$ to follow the same path as $x(t)$ but with a different timing function $\dot{s}$.

The state variables derivatives may thus be expressed as:

$$\dot{\boldsymbol{\lambda}} = \boldsymbol{\lambda}'\dot{s} \tag{33a}$$

$$\ddot{\boldsymbol{\lambda}} = \boldsymbol{\lambda}''\dot{s}^2 + \boldsymbol{\lambda}'\ddot{s} \tag{33b}$$

$$\dot{\mu} = \mu'\dot{s} \tag{33c}$$

As (33) suggests, only motion equations involving directly the state variable derivatives may be considered for time scaling. The time scalable variables from (15) are then:

$$\dot{\beta} = \beta' \dot{s} \tag{34a}$$
$$\ddot{\beta} = \beta'' \dot{s}^2 + \beta' \ddot{s} \tag{34b}$$
$$\ddot{\varphi}_p = \dot{\varphi}'_p \dot{s} \tag{34c}$$

and the limits on the actuators motion considered for scaling are:

$$\dot{\beta}_{min} \leq \dot{\beta} \leq \dot{\beta}_{max} \tag{35a}$$
$$\ddot{\beta}_{min} \leq \ddot{\beta} \leq \ddot{\beta}_{max} \tag{35b}$$
$$\ddot{\varphi}_{min} \leq \ddot{\varphi}_p \leq \ddot{\varphi}_{max} \tag{35c}$$

This leads to the following scaling bounds (with $\beta' > 0$ and $\dot{\varphi}'_p > 0$):

$$\frac{\dot{\beta}_{min}}{\beta'} \leq \dot{s} \leq \frac{\dot{\beta}_{max}}{\beta'} \tag{36a}$$
$$\frac{\ddot{\beta}_{min} - \beta'' \dot{s}^2}{\beta'} \leq \ddot{s} \leq \frac{\ddot{\beta}_{max} - \beta'' \dot{s}^2}{\beta'} \tag{36b}$$
$$\frac{\ddot{\varphi}_{min}}{\dot{\varphi}'_p} \leq \dot{s} \leq \frac{\ddot{\varphi}_{max}}{\dot{\varphi}'_p} \tag{36c}$$

The inequalities are reversed for $\beta' < 0$ or $\dot{\varphi}'_p < 0$. When $|\beta'| \cong 0$ or $|\dot{\varphi}'_p| \cong 0$, the corresponding constraint is not considered for time scaling.

Nonholonomic systems cannot be asymptotically stabilized to a rest configuration by means of smooth state feedback laws like (29a), (29b) and (27); they need smooth time-varying feedback laws to do so. Trajectory tracking may be done with dynamic feedback laws, but it cannot enclose rest configurations. Those matters are discussed in detail by Thuilot et al. [35], who designed a hybrid control law to switch between dynamic tracking and time-varying stabilization. Control of the state variables in the motion controller faces the same challenges: a trajectory between $\boldsymbol{\eta}_e$ and $\boldsymbol{\eta}_d$ must be tracked, and the rest configuration $\boldsymbol{\eta}_d$ must then be stabilized. Time scaling provides a simple way to handle these challenges. Indeed, (33) shows that the state feedback laws are dynamically converted into time-varying laws. If $\dot{s} > 1$, then the motion imposed by the control laws is amplified; it is unaffected when $\dot{s} = 1$; it is slowed down when $\dot{s} < 1$. Ensuring:

$$0 < \dot{s} \leq 1 \tag{37}$$

then allows for a smooth transition between tracking and stabilization: while the limits (35) are exceeded, the control laws are scaled down; but as soon as they are within limits, $\dot{s} = 1$ is enforced and the control laws are unmodified, thus stabilizing $\boldsymbol{\eta}_e$ to $\boldsymbol{\eta}_d$. Another condition for that rest configuration to be possible is:

$$\dot{\beta}_{min} \leq 0 \leq \dot{\beta}_{max} \tag{38a}$$
$$\ddot{\beta}_{min} \leq 0 \leq \ddot{\beta}_{max} \tag{38b}$$
$$\ddot{\varphi}_{min} \leq 0 \leq \ddot{\varphi}_{max} \tag{38c}$$

i.e., the limits need to be of opposite sign; but they do not need to be symmetric nor identical for all wheels. When $\boldsymbol{\beta}' \cong \mathbf{0}$, there are no active constraints to define $\ddot{s}$ and it is set to 1, so as not to influence the control laws.

The set of scaling constraints (36) and (37) create, for each wheel, a range of possible values for $\dot{s}$ and $\ddot{s}$. The intersection of all those ranges gives the respective intervals in which $\dot{s}$ and $\ddot{s}$ should be chosen to ensure that the desired path for each actuator is followed without violating its motion limits. To make the motion controller efficient, the upper bound of the interval allowed for $\dot{s}$ and $\ddot{s}$ is always chosen. When the interval is empty (which may happen for $\ddot{s}$), the trajectory is not executable while satisfying all differential constraints (Q2). In that case, $k_b$ (which $\boldsymbol{\beta}', \boldsymbol{\beta}''$ and $\dot{\boldsymbol{\varphi}}'_p$ depend on through $\boldsymbol{\lambda}', \boldsymbol{\lambda}'', \mu'$) is scaled down by halving its value until the interval is no more empty.

Once $\beta''$ and $\dot{\varphi}'_p$ have been scaled and are thus bounded, the coupling effect of the AZIMUT wheel on $\ddot{\varphi}$ may be handled. This is done by first finding a scaling factor $f_{dd}$ such that:

$$\ddot{\boldsymbol{\varphi}}_{min} \leq f_{dd}(\ddot{\boldsymbol{\varphi}}_p - \frac{b}{r}\ddot{\boldsymbol{\beta}}) \leq \ddot{\boldsymbol{\varphi}}_{max} \tag{39}$$

holds, and then use it to scale $\ddot{\boldsymbol{\beta}}$ and $\ddot{\boldsymbol{\varphi}}_p$.

### 3.5. Integrator

The Integrator module generates the commands $\boldsymbol{\beta}_c$ and $\dot{\boldsymbol{\varphi}}_c$ for the individual actuators by integrating $\dot{\boldsymbol{\beta}}, \ddot{\boldsymbol{\beta}}$ and $\ddot{\boldsymbol{\varphi}}_p$ output by the Time Scaler module:

$$\boldsymbol{\beta}_c = \boldsymbol{\beta}_e + \dot{\boldsymbol{\beta}}\Delta t + \frac{1}{2}\ddot{\boldsymbol{\beta}}(\Delta t)^2 \tag{40a}$$
$$\dot{\boldsymbol{\varphi}}_c = (\dot{\boldsymbol{\varphi}}_p - \frac{b}{r}\dot{\boldsymbol{\beta}}) + (\ddot{\boldsymbol{\varphi}}_p - \frac{b}{r}\ddot{\boldsymbol{\beta}})\Delta t \tag{40b}$$

The module also handles the two algebraic constraints (Q3), which are the steer angle limits and the maximum propulsion velocity:

$$\boldsymbol{\beta}_{min} \leq \boldsymbol{\beta}_c \leq \boldsymbol{\beta}_{max} \tag{41a}$$
$$\dot{\boldsymbol{\varphi}}_{min} \leq \dot{\boldsymbol{\varphi}}_c \leq \dot{\boldsymbol{\varphi}}_{max} \tag{41b}$$

Since with AZIMUT wheels propulsion velocity depends on steering velocity, the order in which the limitations are handled is important: propulsion velocity limitations are handled first, followed by steering limitations. The Path Planner and Time Scaler modules ensure that each component of (40b) is bounded. Dynamic handling of propulsion limitations then resumes to finding a scaling factor $f_d$ such that:

$$\dot{\boldsymbol{\varphi}}_{min} \leq f_d \dot{\boldsymbol{\varphi}}_c \leq \dot{\boldsymbol{\varphi}}_{max} \tag{42}$$

holds and use it to scale $\dot{\boldsymbol{\varphi}}_c$, $\dot{\boldsymbol{\beta}}$ and $\ddot{\boldsymbol{\beta}}$. Once $\dot{\boldsymbol{\beta}}$ and $\ddot{\boldsymbol{\beta}}$ have been scaled, (40a) is computed and $\boldsymbol{\beta}_c$ is checked: if (41a) does not hold, a steering limitation crossing is detected and the Kinematic Model module needs to handle a wheel reconfiguration. To avoid stopping the platform with an invalid wheel configuration, $\boldsymbol{\beta}_c$ is set to its preceding value:

$$\boldsymbol{\beta}_c(t) = \boldsymbol{\beta}_c(t - \Delta t) \tag{43}$$

## 4. Experimental Results

The motion controller for AZIMUT-3 has been implemented within the ROS software framework [37]. The platform has four identical wheels with an offset $b = 0.09$ m and a radius $r = 0.079$ m. The wheels are located at the corners of a square centred at the chassis' centre, thus $\alpha_k = -\frac{\pi}{4} + (k-1)\frac{\pi}{2}$ rad where $k$ is the wheel number starting at 1 for the front-right wheel and counting up in the trigonometric direction. Each wheel has a distance of $l = 0.257$ m to the chassis' centre and may be steered with $-\frac{\pi}{2} < \beta \leq \frac{\pi}{2}$ rad. Motion limits for the actuators are:

$$-1.75 \leq \dot{\beta} \leq 1.75 \text{ rad/s} \tag{44a}$$
$$-15 \leq \ddot{\beta} \leq 15 \text{ rad/s}^2 \tag{44b}$$
$$-13 \leq \dot{\varphi} \leq 13 \text{ rad/s} \tag{44c}$$
$$-20 \leq \ddot{\varphi} \leq 20 \text{ rad/s}^2 \tag{44d}$$

The control step duration is $\Delta t = 10$ ms and the control gains are $k_\lambda = k_\mu = k_\beta = 40$.

After having validated the feedforward model used, the motion controller has been tested to evaluate odometry precision, control efficiency and handling of singularities, in order to globally validate the handling of all control constraints.

A video recording of the conducted experiments is available at `https://youtu.be/Bucyf4iBQLo`.

### 4.1. Feedforward Model Validation

To evaluate the validity of using a one-step delay as feedforward model, the following arbitrary trajectory has been executed: initialization to $\boldsymbol{\eta}_1 = (0\ 1\ 0\ 0)^T$, step on $\mu$ at $t = 0$ s to $\boldsymbol{\eta}_2 = (0\ 1\ 0\ 0.5)^T$, followed by a step on $\boldsymbol{\lambda}$ at $t = 1.7$ s to $\boldsymbol{\eta}_3 = (0\ 0.6\ 0\ 0.5)^T$. Figure 7 compares the commands sent by the motion controller with the state read from the low-level controllers. To ease comparison, the feedforward model has been taken into account in the results: the

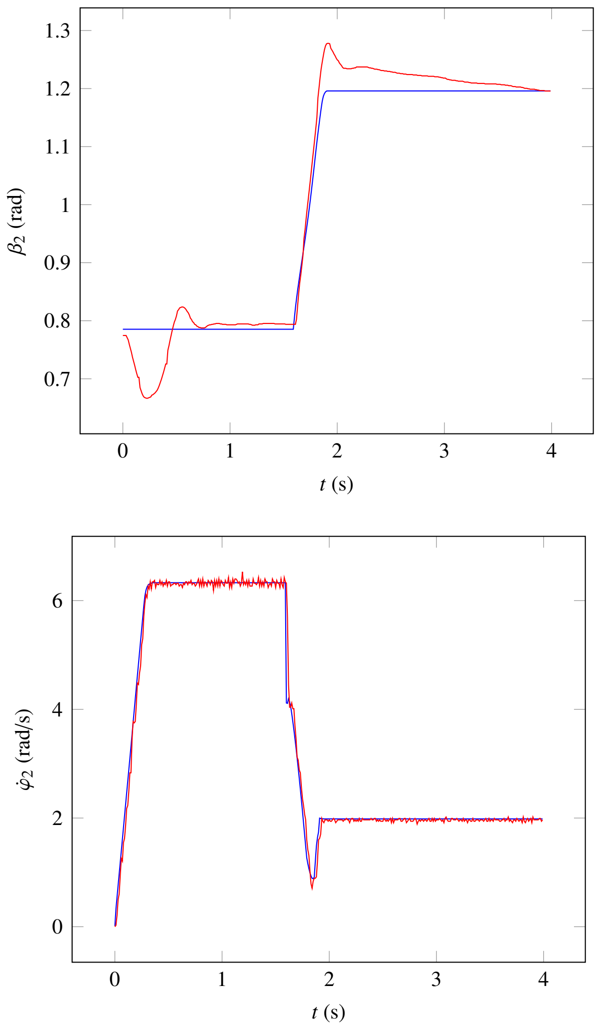


Figure 7: Feedforward model validation, with command (blue) and delayed state (red) for $\beta_2$ (top) and $\dot{\varphi}_2$ (bottom).

state plotted for $t$ is the state that was read at $t + \Delta t$. The top part shows the steering angle $\beta_2$: some overshoot and static error are caused by the wheel's low-level controller dynamics and the compliant steering, but tracking performance is globally acceptable. The bottom part shows the propulsion velocity $\dot{\varphi}_2$: despite small oscillations when commanding a constant velocity, which are again inherent to the low-level controllers, the tracking performance looks good. Most importantly, there is no static lag between the command and the delayed state. Table 1 reports the detailed relative error measures for each wheel. Due to stiction, the propulsion actuators need at least two control steps to start motion; these steps have not been taken into account in the statistics presented. The 3.75% combined average error for $\beta$ and 2.25% for $\dot{\varphi}$ suggest that the hypothesis of using a one-step delay

Table 1: Feedforward model error (in %).

| | | Wheel 1 | Wheel 2 | Wheel 3 | Wheel 4 |
|---|---|---|---|---|---|
| $\beta_k$ | Mean | 4.08 | 2.82 | 2.68 | 5.37 |
| | Std dev. | 3.03 | 3.20 | 3.76 | 3.48 |
| | Max. | 16.09 | 15.14 | 15.33 | 17.68 |
| $\dot{\varphi}_k$ | Mean | 1.77 | 3.16 | 2.16 | 1.92 |
| | Std dev. | 5.08 | 6.98 | 4.50 | 6.18 |
| | Max. | 51.86 | 61.49 | 32.61 | 75.93 |

as feedforward model is valid.

### 4.2. *Odometry Precision*

To evaluate the global precision of actuator coordination, odometry computed by the motion controller has been compared to measurements of the platform's position using a NDI Optotrak system (consisting of four cameras and twelve markers distributed symetrically around the chassis, visible on Fig. 1), with a root mean square error of around 0.4 mm.Three different trajectories were used: T1) a circle; T2) a spiral; and T3) an infinity sign made of lines and arcs. Each trajectory started at $\boldsymbol{\xi}_e = \mathbf{0}$ with the platform at rest ($\mu_e = 0$), ended also at rest (when $\mu_e = \mu_d = 0$) and was executed with the chassis heading tangent to the trajectory.

Executing movements at high velocity involves more acceleration from the actuators and could lead to time scaling issues. To evaluate the impact of motion velocity on the quality of the odometry, each trajectory was executed ten times with two different values for $\mu_d(0)$: $\mu_1 = 0.5$ was chosen to evaluate the performance at medium velocity that should not require special care from the motion controller, and $\mu_2 = 1$ was chosen as a relatively high velocity which, in the case of discontinuous ICR motion (T3), needs full scaling of the propulsion acceleration and dynamic handling of propulsion velocity limits (as done in the Trajectory Scaler and Integrator modules).

Figure 8 illustrates a typical trial for the trajectories executed with $\mu_2$. The computed odometry closely matches the position measurements, even for the high velocity trajectories considered. Error is mostly accumulated on orientation, in particular when there are sharp transitions between ICR positions, i.e., between the lines and arcs of T3. Apart from the well-known errors due to geometric uncertainties, sensors' noise and non systematic errors, error is caused by the compliant steering actuators, which generate a little oscillation of the chassis when they stop. Part of the error may also been explained by the static error between the real steering angles and $\boldsymbol{\beta}_c$. Table 2 presents the relative error on position and orientation at the end of each trajectory and confirms that the error stays small, with a maximum of about 2%. These results have been computed by dividing the positional

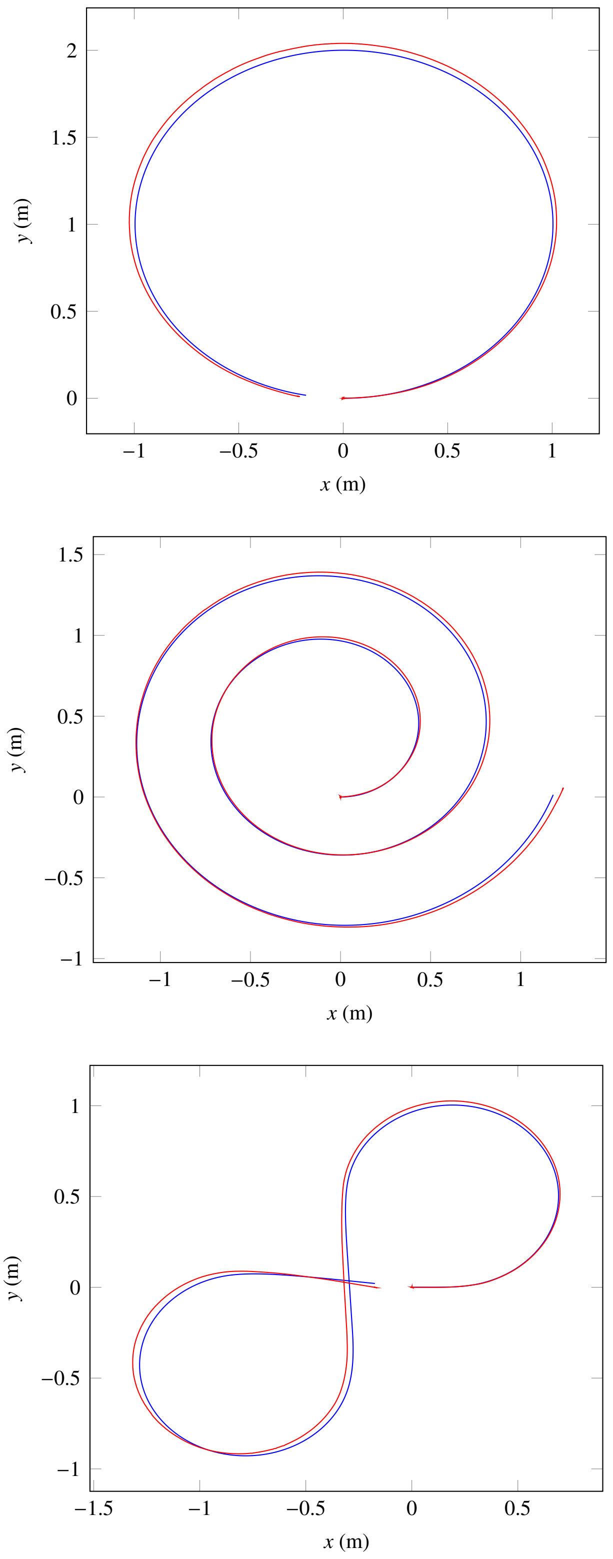


Figure 8: Robot position as computed by odometry (blue) and as measured by the external tracking system (red) for T1 (top), T2 (middle), and T3 (bottom), all executed with $\mu_2$.

Table 2: Relative odometry error at trajectory end (in %).

| | | | T1 Circle | T2 Spiral | T3 Inf. sign |
|---|---|---|---|---|---|
| $\mu_1$ | Position | Mean | 0.69 | 2.09 | 0.67 |
| | | Max. | 0.75 | 2.14 | 2.31 |
| | Orientation | Mean | 0.69 | 1.17 | 1.12 |
| | | Max. | 0.91 | 1.21 | 1.25 |
| $\mu_2$ | Position | Mean | 0.74 | 0.55 | 0.31 |
| | | Max. | 0.83 | 0.61 | 0.38 |
| | Orientation | Mean | 0.92 | 0.10 | 0.86 |
| | | Max. | 1.14 | 0.20 | 0.94 |

error by the total traveled distance and the absolute angular error by the total angle traveled. The table also shows that high velocity trajectories do not imply larger odometry error than low velocity trajectories, which is expected since only commands executable by the low-level controllers are output by the motion controller. As a result, no loss of coordination due to actuator saturation happens, which would lead to large odometry errors.

### 4.3. *Control Efficiency and Differential Constraints*

To ensure a timely execution, a motion controller should execute a desired command as fast as possible, while satisfying all differential constraints on the actuators. For the start of T3 (i.e., acceleration on a straight line) executed with $\mu_1$, the top part of Fig. 9 shows the evolution of the absolute error $\delta_\mu$ between $\mu_d$ and $\mu_e$ and the evolution of the time scaling factor $\dot{s}$, and the bottom part shows the corresponding evolution of $\ddot{\varphi}_1$ and $\ddot{\varphi}_2$. Except for the last 0.07 s of the trajectory for $\boldsymbol{\eta}$ (time scaling stops at $t = 0.29$ s when $\dot{s} = 1$ and $\delta_\mu$ is null at $t = 0.36$ s), there is always at least one actuator whose velocity or acceleration is at its extremal value. Motion is thus executed at the fastest allowable velocity and acceleration, except for a small stabilization phase at the end.

### 4.4. *Singularities Handling*

Contrary to most motion controllers that need to avoid having the ICR move close to a singularity so that no arbitrarily high steering velocity command is generated for the concerned wheel ([22, 38, 39]), the use of time scaling enables our motion controller to handle an ICR moving very close to a singularity or even going through it. Figure 11 illustrates the evolution of the steering velocities for the ICR trajectory depicted on Fig. 10, i.e., the ICR trajectory is parallel to the steering limitation of wheel 1 and has a minimum distance to the singularity of around 1 cm. As expected, all steering velocities remain bounded by the extremal values allowed.

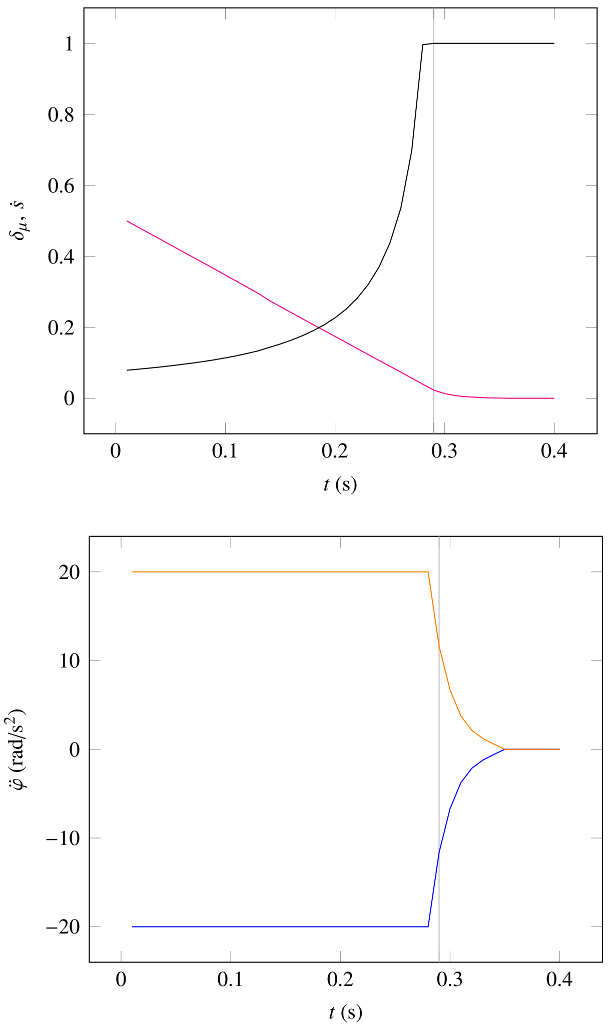


Figure 9: Evolution of $\delta_\mu$ (top, magenta), $\dot{s}$ (top, black), $\ddot{\varphi}_1$ (bottom, blue) and $\ddot{\varphi}_2$ (bottom, orange) for the start of T2 executed with $\mu_1$. The vertical line at $t = 0.29$ s indicates the first time when $\dot{s}$ becomes equal to 1, i.e., when time scaling stops.

For platforms having wheels with steering limitations like AZIMUT-3, the singularities of the kinematic models have an interesting property: if the ICR trajectory goes through one steering axis, no wheel reconfiguration should occur since the corresponding wheel may be freely controlled to have a compatible direction with the ICR motion. Figure 13 illustrates the evolution of the steering angles for the ICR trajectory depicted on Fig. 12, i.e., the ICR goes over the steering axis of wheel 2. As the graph shows, no discontinuities occur in any steering angle trajectory, which indicates that no wheel reconfiguration was generated.

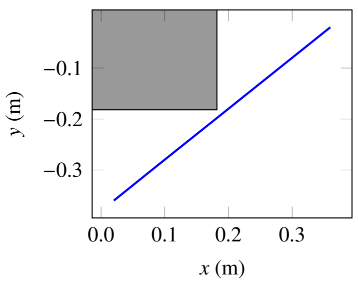


Figure 10: ICR trajectory (blue) moving close to a singularity. The grey box represents the relevant part of the chassis, which is centred at origin.

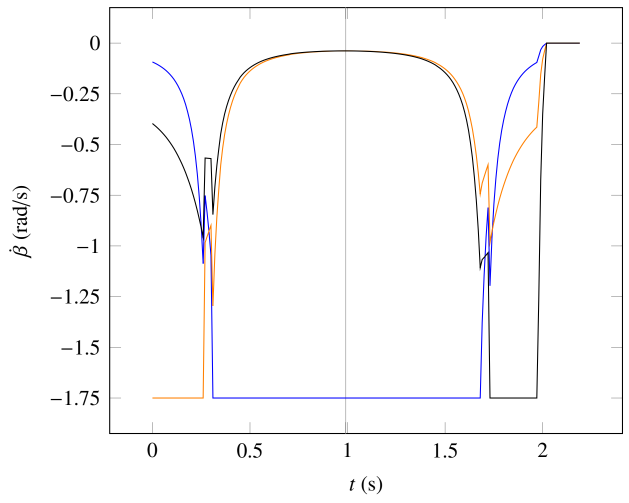


Figure 11: Evolution of $\dot{\beta}_1$ (blue), $\dot{\beta}_2$ (orange) and $\dot{\beta}_4$ (black) when the ICR moves close to a singularity. $\dot{\beta}_3$ is not shown, since its corresponding constraint is never activated during this particular trajectory. The vertical line at $t = 0.99$ s indicates the time when the ICR is closest to the singularity.

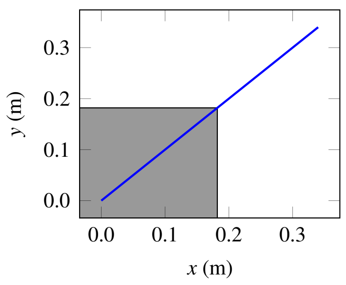


Figure 12: ICR trajectory (blue) going through a singularity. The grey box represents the relevant part of the chassis, which is centred at origin.

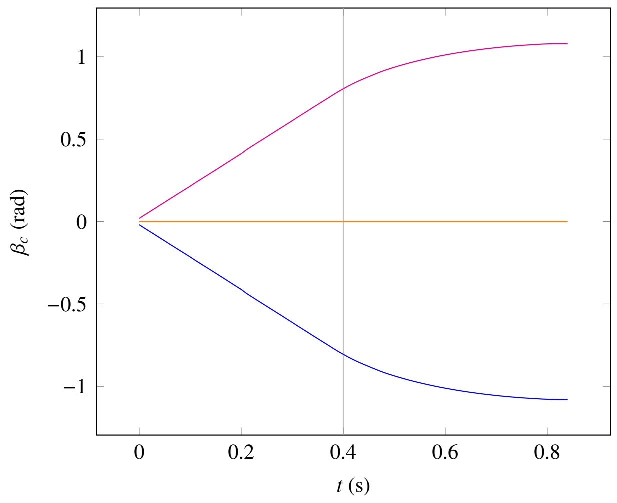


Figure 13: Evolution of $\beta_{c,1}$ (blue), $\beta_{c,2}$ (orange) and $\beta_{c,3}$ (magenta) when the ICR goes through a singularity. $\beta_{c,4}$ is not shown, as it follows exactly the same trajectory as $\beta_{c,2}$. The vertical line at $t = 0.40$ s indicates the time when the ICR goes over the singularity.

## 5. Conclusion

Designing a motion controller for a nonholonomic robot like AZIMUT-3 requires addressing issues such as actuator coordination within their physical limits, platform singularities and command execution efficiency. To do so, this paper presents the design of an ICR-based motion controller that takes advantage of the H representation, a singularity-free parametrization of the ICR position and motion around it based on the real projective plane, leading to simple and efficient kinematic models and control laws which ensure proper actuator coordination and easy handling of structural singularities. In addition, the use of trajectory time scaling enables to maintain command execution efficiency, while ensuring in a unified way that actuator physical limits are never reached. Results acquired on AZIMUT-3 demonstrate these capabilities, with accurate odometry and fast command execution.

Although the motion controller and kinematic models presented in this paper are specifically geared towards AZIMUT-3, the motion controller architecture may easily be adapted to other platforms for which the State Estimator, Path Planner and Kinematic Model modules can be adequately defined. The proposed inverse kinematic model is directly applicable to a robot equipped with any number of centred or sidewards off-centred wheels. The proposed direct kinematic model is also directly applicable to a robot with three or more sidewards off-centred wheels. For centred wheels, $\mu$ may be directly estimated by solving (25) for any number of those wheels, and estimation of $\dot{\boldsymbol{\lambda}}$ is not necessary as the proposed control laws do not depend on it. As indicated in [23, 33, 36], ICR estimation is possible for three or more centred or sidewards off-centred wheels. After implementing these additions and validating the resulting motion controller, the proposed approach could be used with any platform using three or more centred or sidewards off-centred wheels.

## Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada, the Fonds de recherche du Québec – Nature et technologies, the Canada Foundation for Innovation and the Canada Research Chair program. The authors would like to thank Dominic Létourneau, François Ferland and all the members of IntRoLab for making this work possible.

[1] S. L. Dickerson, B. D. Lapin, Control of an omni-directional robotic vehicle with Mecanum wheels, in: Proceedings of the National Telesystems Conference, IEEE, 1991, pp. 323–328 vol. 1.

[2] T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, R. Dillmann, ARMAR-III: An integrated humanoid platform for sensory-motor control, in: Proceedings of the International Conference on Humanoid Robots, IEEE-RAS, 2006, pp. 169–175.

[3] Segway Inc., 2017, Segway Robotic Mobility Platforms (RMPs) Segway RMP400 Omni, URL: `http://rmp.segway.com/about-segway-robotics/segway-rmp400-omni/`.

[4] B. Woods, Omni-Directional Wheelchair, Ph.D. thesis, The University of Western Australia, School of Mechanical, Materials and Mechatronics Engineering, 2006.

[5] G. Campion, W. Chung, Wheeled robots, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer, Berlin, 2008, pp. 391–410.

[6] R. Bischoff, V. Graefe, HERMES – An intelligent humanoid robot designed and tested for dependability, in: B. Siciliano, P. Dario (Eds.), Experimental Robotics VIII, volume 5 of *Springer Tracts in Advanced Robotics*, Springer, Berlin, Heidelberg, 2003, pp. 64–74. doi:`10.1007/3-540-36268-1_4`.

[7] M. Fuchs, C. Borst, P. R. Giordano, A. Baumann, E. Krämer, J. Langwald, R. Gruber, N. Seitz, G. Plank, K. Kunze, R. Burger, F. Schmidt, T. Wimböck, G. Hirzinger, Rollin' Justin – Design considerations and realization of a mobile platform for a humanoid upper body, in: Proceedings of the International Conference on Robotics and Automation, IEEE, 2009, pp. 4131–4137.

[8] A. Dietrich, T. Wimböck, A. Albu-Schäffer, G. Hirzinger, Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom, IEEE Robotics and Automation Magazine 19 (2012) 20–33.

[9] J. Stückler, M. Schwarz, S. Behnke, Mobile manipulation, tool use, and intuitive interaction for cognitive service robot Cosero, Frontiers in Robotics and AI 3 (2016) 1–20.

[10] J. Stückler, S. Behnke, Dynamaid, an anthropomorphic robot for research on domestic service applications, Automatika – Journal for Control, Measurement, Electronics, Computing and Communications 52 (2011) 233–243.

[11] Meka Robotics, 2013, B1 Omni Base | Meka Robotics, URL: `http://mekabot.com/products/omni-base/`.

[12] Willow Garage, 2017, Hardware Specs | Willow Garage, URL: `http://www.willowgarage.com/pages/pr2/specs`.

[13] Neobotix GmbH, 2017, MPO-700 - Neobotix, URL: `http://www.neobotix-robots.com/omnidirectional-robot-mpo-700.html`.

[14] U. Reiser, C. P. Connette, J. Fischer, J. Kubacki, A. Bubeck, F. Weisshardt, T. Jacobs, C. Parlitz, M. Hägele, A. Verl, Care-O-bot 3 – Creating a product vision for service robot applications by integrating design and technology, in: Proceedings of the International Conference on Intelligent Robots and Systems, IEEE / RSJ, 2009, pp. 1992–1998.

[15] C. P. Connette, C. Parlitz, B. Graf, M. Hägele, A. Verl, The mobility concept of Care-O-bot 3, in: Proceedings of the International Symposium on Robotics, VDE Verlag, 2008, pp. 746–750.

[16] E. Prassler, R. Bischoff, W. Burgard, R. Haschke, M. Hägele, G. Lawitzky, B. N. amd Paul Plöger, U. Reiser, M. Zöllner (Eds.), Towards Service Robots for Everyday Environments, Springer Tracts in Advanced Robotics, Springer, Berlin Heidelberg, 2012. doi:`10.1007/978-3-642-25116-0`.

[17] M. Sorour, A. Cherubini, P. Fraisse, R. Passama, Motion discontinuity-robust controller for steerable mobile robots, IEEE Robotics and Automation Letters 2 (2017) 452–459.

[18] F. Michaud, F. Ferland, D. Létourneau, M.-A. Legault, M. Lauria, Toward autonomous, compliant, omnidirectional humanoid robots for natural interaction in real-life settings, Paladyn. Journal of Behavioral Robotics 1 (2010) 57–65.

[19] F. Ferland, A. Aumont, D. Létourneau, M.-A. Legault, F. Michaud, Johnny-0, a compliant, force-controlled and interactive humanoid autonomous robot, in: Proceedings of the Annual International Conference on Human-Robot Interaction, ACM / IEEE, 2012, pp. 417–418. doi:`10.1145/2157689.2157826`.

[20] M. Lauria, I. Nadeau, P. Lepage, Y. Morin, P. G. F. Gagnon, D. Létourneau, F. Michaud, Design and control of a four steered wheeled mobile robot, in: Proceedings of the Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2006, pp. 4020–4025.

[21] C. P. Connette, M. Hägele, A. Verl, Singularity-free state-space representation for non-holonomic, omnidirectional undercarriages by means of coordinate switching, in: Proceedings of the International Conference on Intelligent Robots and Systems, IEEE / RSJ, 2012, pp. 4959–4965.

[22] U. Schwesinger, C. Pradalier, R. Siegwart, A novel approach for steering wheel synchronization with velocity/acceleration limits and mechanical constraints, in: Proceedings of the International Conference on Intelligent Robots and Systems, IEEE / RSJ, 2012, pp. 5360–5366. doi:`10.1109/IROS.2012.6385644`.

[23] L. Clavien, M. Lauria, F. Michaud, Estimation of the instantaneous centre of rotation with nonholonomic omnidirectional mobile robots, Robotics and Autonomous Systems (2018).

[24] F. Michaud, D. Létourneau, M. Arsenault, Y. Bergeron, R. Cardin, F. Gagnon, M.-A. Legault, M. Millette, J.-F. Paré, M.-C. Tremblay, P. Lepage, Y. Morin, J. Bisson, S. Caron, Multi-modal locomotion robotic platform using leg-track-wheel articulations, Autonomous Robots 18 (2005) 137–156.

[25] M. Lauria, M.-A. Legault, M.-A. Lavoie, F. Michaud, Differential elastic actuator for robotic interaction tasks, in: Proceedings of the International Conference on Robotics and Automation, IEEE, 2008, pp. 3606–3611.

[26] J. Frémy, F. Ferland, M. Lauria, F. Michaud, Force-guidance of a compliant omnidirectional non-holonomic platform, Robotics and Autonomous Systems 62 (2014) 579–590.

[27] S. Chamberland, E. Beaudry, L. Clavien, F. Kabanza, F. Michaud, M. Lauria, Motion planning for an omnidirectional robot with steering constraints, in: Proceedings of the International Conference on Intelligent Robots and Systems, IEEE / RSJ, 2010, pp. 4305–4310.

[28] S. M. LaValle, Planning Algorithms, Cambridge University Press, Cambridge, 2006.

[29] J. B. Coulaud, G. Campion, Optimal trajectory tracking for differentially flat systems with singularities, in: Proceedings of the International Conference on Robotics and Automation, IEEE, 2007, pp. 1960–1965.

[30] C. P. Connette, A. Pott, M. Hägele, A. Verl, Addressing input saturation and kinematic constraints of overactuated undercarriages by predictive potential fields, in: Proceedings of the International Conference on Intelligent Robots and Systems, IEEE / RSJ, 2010, pp. 4775–4781.

[31] J. M. Hollerbach, Dynamic scaling of manipulator trajectories, in: Proceedings of the American Control Conference, IEEE, 1983, pp. 752–756.

[32] G. Campion, G. Bastin, B. d'Andréa-Novel, Structural properties and

classification of kinematic and dynamic models of wheeled mobile robots, IEEE Transactions on Robotics and Automation 12 (1996) 47–62.

[33] C. P. Connette, A. Pott, M. Hägele, A. Verl, Control of an pseudo-omnidirectional, non-holonomic, mobile robot based on an ICM representation in spherical coordinates, in: Proceedings of the Conference on Decision and Control, IEEE, 2008, pp. 4976–4983.

[34] F. Ferland, L. Clavien, J. Frémy, D. Létourneau, F. Michaud, M. Lauria, Teleoperation of AZIMUT-3, an omnidirectional non-holonomic platform with steerable wheels, in: Proceedings of the International Conference on Intelligent Robots and Systems, IEEE / RSJ, 2010, pp. 2515–2516.

[35] B. Thuilot, B. d'Andréa-Novel, A. Micaelli, Modeling and feedback control of mobile robots equipped with several steering wheels, IEEE Transactions on Robotics and Automation 12 (1996) 375–390.

[36] L. Clavien, M. Lauria, F. Michaud, Instantaneous centre of rotation estimation of an omnidirectional mobile robot, in: Proceedings of the International Conference on Robotics and Automation, IEEE, 2010, pp. 5435–5440.

[37] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng, ROS: an open-source robot operating system, in: Proceedings of the Open-source Software Workshop of the IEEE International Conference on Robotics and Automation, 2009.

[38] C. P. Connette, C. Parlitz, M. Hägele, A. Verl, Singularity avoidance for over-actuated, pseudo-omnidirectional, wheeled mobile robots, in: Proceedings of the International Conference on Robotics and Automation, IEEE, 2009, pp. 4124–4130.

[39] A. Dietrich, T. Wimböck, A. Albu-Schäffer, G. Hirzinger, Singularity avoidance for nonholonomic, omnidirectional wheeled mobile platforms with variable footprint, in: Proceedings of the International Conference on Robotics and Automation, IEEE, 2011, pp. 6136–6142.