To be published in Robotics and Autonomous Systems, 2018, 106, pp. 47-57,which should be cited to refer to this work.

DOI: 10.1016/j.robot.2018.03.009

Estimation of the Instantaneous Centre of Rotation with Nonholonomic Omnidirectional Mobile Robots

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

In order to move safely and accurately, mobile platforms using steerable wheels require adequate coordination of their actuators. One possibility to achieve actuator coordination is to control the motion of the chassis' instantaneous centre of rotation (ICR) and motion around it. Considering the chassis as a rigid body, the ICR is located at the intersection of each wheel's zero motion axis. In practice however, these axes may not concur, in particular when compliant actuators are used for wheel steering. They then no more define precisely an ICR and only an estimation of its position can be computed. Moreover, most parametrizations of the ICR position bring in singularities with no physical meaning, which hinder estimation. This paper introduces the H representation, a new parametrization of the motion state space free of any non-structural singularities, and presents an algorithm which estimates the ICR within the joint space. The proposed approach is compared in terms of reliability, efficiency, accuracy and robustness with three methods working within the operational space. Results suggest that the proposed estimation approach provides the best compromise for these performance indicators.

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

1. Introduction

Omnidirectional mobile platforms can move in any direction without manœuvring, making them suitable for operation in tight areas and crowded spaces. One mechanism to provide such capability are omnidirectional wheels, like wheels made with passive rollers attached along their circumference [1]. Steerable wheels are another alternative, allowing a platform to move in any direction by only changing the orientation of its wheels. Compared to omnidirectional wheels, the use of steerable wheels provides more precise odometry and lower mechanical complexity [2]. Many platforms use steerable wheels, like Meka B1 [3], the EXOMARS rover [4], Willow Garage PR2 [5], Rollin' Justin [6, 7], Care-O-bot [8, 9, 10] and AZIMUT [11, 12, 13]. Yet, omnidirectional platforms using steerable wheels are nonholonomic, since the wheels need to be reoriented to change the direction of motion. The wheels thus constrain the chassis motion and need to be carefully coordinated for the platform to move [14] without generating high internal forces and slippage caused by actuator antagonism [7, 12, 15]. To handle this coordination, motion of the individual actuators must be linked through kinematic models with the motion of the chassis, which may be described using two paradigms: the chassis' twist (instantaneous linear and angular velocities) or the chassis' motion around its instantaneous centre of rotation (ICR). In the case of a rigid body undergoing planar movement, the ICR is the point in the body's referential frame which has zero velocity at a given instant in time. Since steerable wheels can only move in the wheel plane, the zero motion axis (i.e., the axis perpendicular to the wheel plane and going through the wheel's centre, called hereafter propulsion axis) of each of the platform's wheels should then concur at the ICR location [16].

The twist paradigm is often used to control motion of robots using steerable wheels and is adapted to control quasi-holonomic platforms (e.g., using active caster wheels), since the twist components then provide independent control inputs [2]. When considering purely nonholonomic platforms, however, the direction of motion cannot change instantly and the twist components then become linked to each other, making it difficult to only use the twist paradigm [15, 17]. As an alternative, the ICR paradigm provides independent control inputs: the ICR position and the motion around the ICR. ICR-based control enforces the

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

nonholonomic constraints, since the propulsion axes must intersect at the ICR location for the platform to move safely and accurately. It thus provides an abstraction of the platform's actuators for control [18] that also enables easy recovery of the instantaneous physical state of the platform when needed. Without chassis motion, motion of the steering actuators can still be controlled by moving the desired ICR, for example to make very sharp turns that require the robot to stop and reconfigure its wheels' position, whereas a null twist gives no information to control steering and a separate control algorithm is needed to do the reconfiguration. In addition, the kinematic models of platforms using steerable wheels exhibit structural singularities when the ICR is on a steering axis [7, 14, 15]. Using the ICR paradigm makes it possible to define these singularities with a minimal number of conditions (i.e., the number of steerable wheels), whereas using the twist paradigm requires checking an infinite number of conditions, which complexifies the design and implementation of a motion controller. Hence, even though motion control of nonholonomic platforms is possible using the twist paradigm [19], switching to the ICR paradigm is beneficial to easily handle structural singularities [7].

However, using the ICR paradigm also brings challenges to overcome:

1. Most ICR parametrizations introduce singularities that are not related to singularities in the chassis motion, such as undefinedness of direction at infinity (for 2D Cartesian coordinates), undefinedness at the pole (for polar and spherical coordinates) or discontinuities in a parameter when crossing some meridian (for spherical coordinates) [17]. Those parametrization-induced singularities need to be dealt with and a singularity-free parametrization could provide more efficient and simpler ICR determination and motion control.
2. The ICR is a mathematical concept which enables abstraction of the platform geometry and actuators, but the only available source of information to determine its location – which is needed for ICR-based motion control – are the actuators' sensors. With infinite stiffness actuators, perfect sensors and careful coordination, the assumption of the complete robot (chassis and wheels) as a single rigid body would hold and the propulsion axes would concur at the ICR location. However, a real robot is always imperfect: it has geometric imperfections, actuators with finite stiffness, stiction and limitations, as well as sensors with noise and quantification. The robot should then be considered as a rigid body (the chassis) connected to rigid bodies (the wheels) that can move wrt the chassis, and the propulsion axes may not create a well-defined intersection when considering robots with three or more steerable wheels. This phenomenon is only made evident after long experimentation times with stiff actuators [7, 8]. But when compliant actuators are used for wheel steering to help minimize actuator antagonism or to make the robot more responsive and secure to physical contacts, like with AZIMUT-3 [12, 20], this is always perceptible. In that case, the ICR location still exists, but it cannot be determined *directly* by the actuators' sensors. In practice however, since motion of the platform is happening, the propulsion axes must have globally a minimal deviation from the ICR location, which has been confirmed by experimentation [21]. Compliant actuators bring the challenge that the propulsion axes do not have a well defined intersection during motion, but also help cope with that fact by ensuring that minimal deviations do not create actuator antagonism leading to slippage. Thus, as long as motion control ensures a careful coordination of the actuators and is fed with the most reliable ICR position, the ICR-based control model stays valid. Determining the most reliable ICR position may be done either in the operational space or in the joint space. Using a least squares estimation (LSE) in the operational space is standard practice [22], but it has been shown this does not give the most reliable ICR estimation when the propulsion axes are close to parallel [21]. As an alternative, an Extended Kalman filter (EKF) has been used in [4], but according to its authors, it does not give better results than the algorithm proposed in [21], which works in the joint space.

The aim and contribution of this paper are to present an approach for ICR estimation that addresses these challenges. It starts with the definition of a new parameterization of the ICR position having its roots in projective geometry and free of parametrization-induced singularities, and an associated representation in $\mathbb{R}^3$. We collectively refer to the parameterization and representation as the H representation. An iterative ICR estimation algorithm working in the joint space, based on [21], then leverages the H representation properties to obtain the best possible estimation when the ICR is ill-defined, i.e., the propulsion axes do not concur. Our algorithm is compared to three other algorithms working in the operational space: one doing no estimation, one using a LSE and one computing the motion constraints' null-space. Their relative performance is evaluated in terms of reliability, efficiency, accuracy and robustness, demonstrating the benefits of using our approach compared to the others evaluated. Demonstration of the applicability and usefulness of the proposed H representation and ICR estimation algorithm for ICR-based motion

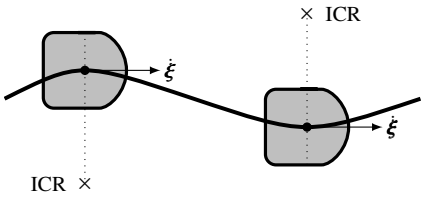

Figure 1: ICR position in the Cartesian plane during a transition between a right turn and a left turn: a slight change in the motion direction corresponds to a drastic relocation of the ICR position.

control of nonholonomic omnidirectional platforms is presented in [20, 23, 24].

This paper is organized as follows. Section 2 introduces the H representation for ICR representation and parametrization. Section 3 details the proposed ICR estimation algorithm and the three other algorithms used for comparison. Section 4 then presents their use on AZIMUT-3 [11], an omnidirectional platform using four compliant steerable wheels, with results using simulated and real data.

2. Motion Representation

As formulated by Campion *et al.* [16], a robot motion can be seen instantaneously as a rotation around the ICR. Using two-dimensional Cartesian ($\mathbb{R}^2$) or polar ($\mathbb{R}\times\mathbb{S}$) coordinates seems a natural choice for parametrizing its position. But those parametrizations have singularities with no physical meaning [17]. For instance, going from a slight right turn to a slight left one, as illustrated on Fig. 1, involves a continuous motion of the chassis but causes discontinuities in a two-dimensional Cartesian parametrization: the ICR position is near infinity on the right and goes directly to infinity on the left without going through the centre of the chassis. Even though they are not critical to handle, these singularities occur in a frequently used region of the state space and interfere with ICR estimation and motion control.

2.1. The Real Projective Plane

Since the different regions representing infinity in $\mathbb{R}^2$ are not connected ($x = -\infty$ is not identified with $x = \infty$, and the same for y), parametrizing the ICR motion in $\mathbb{R}^2$ without singularities is impossible. A new two-dimensional topological space is needed, where all points are multiplily-connected instead of simply-connected like in $\mathbb{R}^2$. Manifolds obtained by boundary identification of their definition domain are topological spaces that may provide such connectivity [25] and are thus well adapted to avoid parametrization-induced singularities. One interesting manifold to represent the ICR position is the real projective plane $\mathbb{P}_2(\mathbb{R})$ (also called the two-dimensional projective space and noted $\mathbb{RP}^2$), which is obtained by twisted boundary identification of the $[0, 1] \times [0, 1]$ domain.

Manifolds are abstract topological spaces and most are difficult to represent intuitively, but thanks to its projective nature, $\mathbb{P}_2(\mathbb{R})$ may be represented easily in $\mathbb{R}^3$. Indeed, since all points aligned with the origin are identified (i.e., all represent a line through that origin), the space is homeomorphic to $\mathbb{S}^2$ with antipodal points identified: $\mathbb{P}_2(\mathbb{R}) \cong \{\mathbb{S}^2 \setminus (x, y) \sim -(x, y)\}$. This means that $\mathbb{S}^2$, which is embeddable in $\mathbb{R}^3$ as the unit sphere, may be used to visualize the ICR position. $\mathbb{S}^2$ can be parametrized by using spherical coordinates, which still leads to parametrization-induced singularities [17]. This is because $\mathbb{S}^2$ is not homeomorphic to $\mathbb{R}^2$, so it is impossible to find a globally singularity-free parametrization of it using a single bi-dimensional map. To avoid such singularities, an atlas of overlapping, locally singularity-free charts that cover the whole sphere (with antipodal points identified) has to be constructed [25]. A related but more complicated approach using modified spherical coordinates has been proposed in [15] to locally hide singularities. Yet, employing these modified spherical coordinates inhibits usage of simple and efficient vector calculus, like the use of three-dimensional Cartesian coordinates enables.

Let Λ be the set of identified antipodal points on the sphere, and $\boldsymbol{\lambda}$ be one of its elements:

$$\boldsymbol{\lambda} \in \Lambda = \{(u, v, w) \in \mathbb{S}^2 \setminus (u, v, w) \sim (-u, -v, -w)\} \quad (1a)$$

$$\text{where} \quad \mathbb{S}^2 = \{(u, v, w) \in \mathbb{R}^3 | u^2 + v^2 + w^2 = 1\} \quad (1b)$$

A chart consists of a domain U and a map ϕ (i.e., a coordinate function). As a convention, let the North and South hemispheres with antipodal points identified be U_w, one of the domains:

$$U_w = \{(u, v, w) \in \mathbb{S}^2 | w \neq 0\} \quad (2)$$

and the associated map ϕ_w be:

$$\phi_w(u, v, w) = (u, v) \quad (3)$$

The corresponding parametrization is:

$$\phi_w^{-1}(u, v) = (u, v, \sqrt{1 - u^2 - v^2}) \quad (4)$$

The two other domains are defined using respectively the $u \neq 0$ and $v \neq 0$ identified hemispheres. The corresponding maps and parametrizations are obtained by circular permutation of the components of (3) and (4). Given a point on the sphere, the corresponding chart is selected so that the point lies closest to the centre of the chart's domain. Since coordinates changes between maps are continuous and the charts cover the whole manifold, by defining the preceding

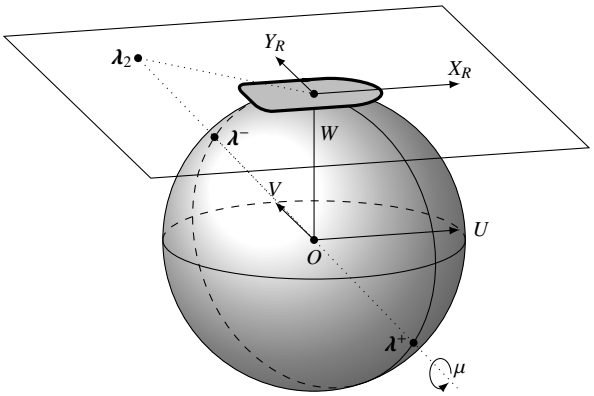

Figure 2: H representation with $\boldsymbol{\lambda}_2$ as mapping of $\boldsymbol{\lambda}$ in the tangent plane. $\boldsymbol{\lambda}$ is represented by $\boldsymbol{\lambda}^-$ and $\boldsymbol{\lambda}^+$.

atlas, $\mathbb{P}_2(\mathbb{R})$ is equipped with a smooth structure to obtain a smooth manifold. Differentiation operations (used by the iterative algorithm presented in Section 3) are then well defined.

Motion around the ICR is parametrized in $\mathbb{R}$ and noted μ. It may be visualized as the spin of the oriented line representing the ICR position. As illustrated by Fig. 2, the configuration space thus proposed to efficiently represent the motion of a nonholonomic mobile platform is $\mathrm{H} = \Lambda \times \mathbb{R}$, an element of which is noted $\boldsymbol{\eta} = (u\ v\ w\ \mu)^T$. This space and representation is what we refer to as the H representation. Its parametrization is singularity-free because any continuous chassis motion leads to a continuous change of coordinates. In particular, the situation illustrated by Fig. 1 now corresponds to an ICR moving along a spherical arc between two points near the sphere's equator.

2.2. Mapping between $\mathbb{R}^2$ and Λ

Mapping of positions between $\mathbb{R}^2$ and Λ may be required, for example to express the robot geometry (only known in $\mathbb{R}^2$) on the unit sphere or to map back an ICR on the plane (for visualization purposes). As illustrated on Fig. 2, this may be done by embedding $\mathbb{R}^2$ as the tangent plane to an unit sphere at its North pole, i.e., as the $w = 1$ plane. The conversions are done by considering the line linking the point $(x\ y)^T$ on the plane and the sphere's centre. As $\boldsymbol{\lambda}$ is a vector on the unit sphere, mapping from $\mathbb{R}^2$ to Λ (North hemisphere) is done by finding the director vector of that line:

$$\boldsymbol{\lambda} = \begin{pmatrix} u & v & w \end{pmatrix}^T = \frac{1}{\left\| \begin{pmatrix} x & y & 1 \end{pmatrix}^T \right\|} \begin{pmatrix} x & y & 1 \end{pmatrix}^T \qquad (5)$$

and mapping back from Λ to $\mathbb{R}^2$ is done by finding the point on that line where $w = 1$:

$$\boldsymbol{\lambda}_2 = \begin{pmatrix} x & y \end{pmatrix}^T = \frac{1}{w} \begin{pmatrix} u & v \end{pmatrix}^T \qquad (6)$$

This is the definition of the gnomonic projection [26], also called the central projection or rectilinear projection, which is at the roots of the projective geometry. It is an azimuthal perspective projection which maps all great circles (diameters, which are shortest paths on the sphere) to straight lines. Points on the sphere's equator, which are in the $w = 0$ plane, are thus mapped to points at infinity in $\mathbb{R}^2$.

3. ICR Estimation

The motion representation presented in Section 2 gives a description of the robot chassis' motion, abstracting its geometry and actuators. However, the parameters $\boldsymbol{\lambda}$ and μ needed by such a representation are not directly measured from the platform sensors. What can be sensed are the steer angle $\beta \in \mathbb{S}$ and the propulsion angular speed $\dot{\varphi} \in \mathbb{R}$ of the n wheels. ICR estimation is thus concerned by establishing a relation $F\colon \mathbb{T}^n \mapsto \Lambda$, where $\mathbb{T}^n$ (the n-torus) is the Cartesian product of n times $\mathbb{S}$. We only consider robots with at least three steerable wheels ($n \geq 3$) and therefore the relation is over-determined: the initial space is at least of dimension three and the target space is of dimension two. With a perfect robot and appropriate control, the propulsion axes would always define an intersection and thus the ICR position, and any relevant algorithm could be used for estimation. However, on a real robot and in particular when using compliant actuators for steering, the propulsion axes will not always concur (see Section 3). What is then sought is an algorithm that gives the best possible *estimation* of the ICR given by the set of sensors' readings, as pictured on Fig. 3.

There are two paradigms that can be used to do the estimation: either work in the operational space (Λ or $\mathbb{R}^2$) by representing the steer angle values as great circles (Λ) or lines ($\mathbb{R}^2$), or work directly in the joint space ($\mathbb{T}^n$) by representing the ICR position as a point. Working in the joint space brings in a key advantage. Indeed, when working in the operational space, the ICR is a *free* point that can be located anywhere in the space. Any algorithm working in that space will only find a point that minimises its distance to the propulsion axes, without any constraints on the estimated ICR position. However, all ICR definable by the propulsion axes create an hyper-surface in $\mathbb{T}^n$ (one or more torus patches), referred to as the constraining surface, which creates a *constraint* on the estimated ICR position. An algorithm that would minimise the distance between a steer angle configuration (which represents the propulsion axes ori-

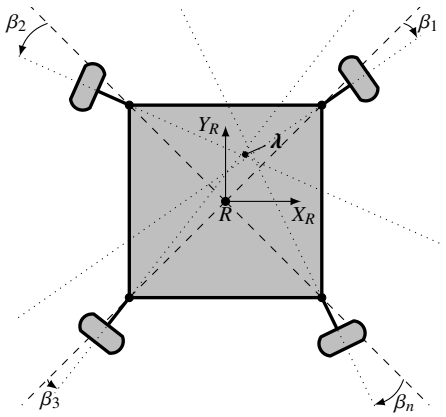

Figure 3: Problem statement: estimate $\boldsymbol{\lambda} \in \Lambda$, knowing $\boldsymbol{q} = \{\beta_1, ..., \beta_n\} \in \mathbb{T}^n$.

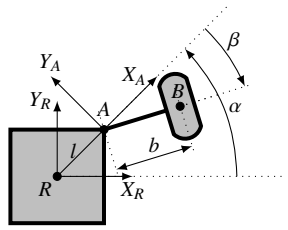

Figure 4: Geometry of the robot chassis and steering mechanism.

entations) and the constraining surface would thus find the best possible estimation of the current ICR position, in the sense it would be the one that minimises the energy needed to correct the deviations. Our research interest is oriented towards the AZIMUT wheel [12], illustrated on Fig. 4, since it is used on our mobile platform AZIMUT-3. However, both the AZIMUT wheel and the centred wheel are equivalent as regards ICR estimation, since the b offset does not appear in any of the following equations.

Let us define $\boldsymbol{a}$ and $\boldsymbol{a}^\perp$ as the orthogonal projection on the sphere's equatorial plane of the basis vectors of $\{A, X_A, Y_A\}$ (associated with the steering axis $\boldsymbol{s} = \boldsymbol{OA}$) and $\boldsymbol{l}$ as the wheel's distance to the chassis' centre ($l = \|\boldsymbol{RA}\|$) reported along the W axis (shown on Fig. 2):

$$\boldsymbol{a} = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \end{pmatrix}^T \tag{7a}$$

$$\boldsymbol{a}^\perp = \begin{pmatrix} -\sin\alpha & \cos\alpha & 0 \end{pmatrix}^T \tag{7b}$$

$$\boldsymbol{s} = \begin{pmatrix} l\cos\alpha & l\sin\alpha & 1 \end{pmatrix}^T \tag{7c}$$

$$\boldsymbol{l} = \begin{pmatrix} 0 & 0 & l \end{pmatrix}^T \tag{7d}$$

A great circle is defined as the intersection of the sphere with a plane going through its origin. An oriented great circle may thus be described completely by the normal vector of the plane that contains it. For example, the great circle going through two arbitrary points $\boldsymbol{P}'_1$ and $\boldsymbol{P}'_2$ is defined by the vector:

$$\boldsymbol{c} = \boldsymbol{P}'_1 \times \boldsymbol{P}'_2 \tag{8}$$

As points on the sphere have projections on the tangent plane via (6), the same great circle may be directly defined by two points in that tangent plane (or even by a point on the sphere and one in the tangent plane) without first being projected on the sphere. This eases computations for the four ICR estimation algorithms presented in the following subsections and would not be possible using non-Cartesian coordinates.

3.1. Algorithm Working in the Joint Space (IT)

When working in the joint space ($\mathbb{T}^n$), an algorithm should find the ICR-defining steerable configuration closest to the measured steering configuration (see beginning of the section). This can be done done by finding the orthogonal projection of the measured steering configuration (some point in $\mathbb{T}^n$) onto the constraining surface.

Some robots have steering limitations for their wheels. These limitations define the injectivity of the relationship between an ICR and steering configurations. Since β and $\beta + \pi$ represent the same axis, there are two steer angle values per wheel compatible with a given ICR. If there is no steering limitation, a given ICR corresponds to 2^n steering configurations. If the limitation is more than π, some ICR are defined by more than one steering configuration. If the limitation is less than π, some ICR cannot be defined by a steering configuration and the robot is no more omnidirectional. It is only when the limitation is of π that the relation is bijective: for each given ICR, there exists a unique steering configuration. Without loss of generality and to ease computations, only the case where the relation is bijective (i.e., when there is a symmetric limitation of $\pm\frac{\pi}{2}$ for each β) is considered in the following. For the other cases, simple linear transforms may be applied before and after the algorithm.

$\mathbb{T}^n$ is not homeomorphic to $\mathbb{R}^n$, but it may be embedded in $\mathbb{R}^{2n}$. Since $n \geq 3$, the constraining surface cannot be represented easily in the standard three-dimensional Cartesian space. However, for a robot having three wheels, it can be represented by plotting independently its steering coordinates, as shown on Fig. 5, and remembering that each opposite sides of the $]-\pi;\pi]^3$ cube are identified.

Since the steering angles are independent of each other but are linked by the ICR, only a parametric representation of the constraining surface is known. There is however no generic algebraic method known to compute the orthogonal projection of a point onto a parametric surface. Thus,

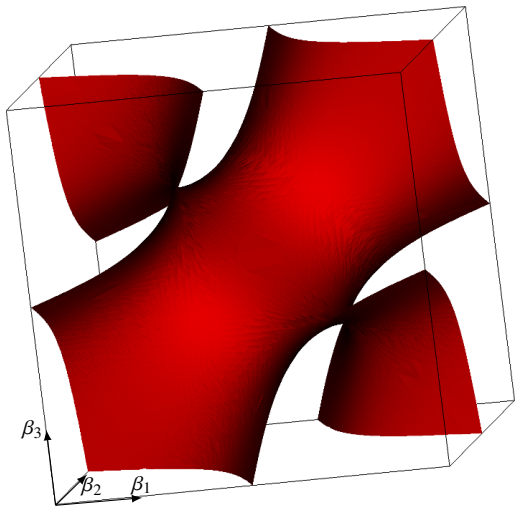

Figure 5: Cartesian representation of the constraining surface for a three-wheeled symmetric robot having symmetric steering limitations of π. Centre of the surface is located at the centre of the cube.

estimating the ICR is done by using a first-order geometric iterative algorithm. The base of the algorithm consists of a simplified version of the Newton-Raphson method to minimize the distance between a point and a parametric surface [27]. A preliminary version of this algorithm has been described in [21], with the ICR position parametrized using polar coordinates. It has further been optimized, adapted to use the H representation for more simplicity and improved efficiency, and is now described for the $\{U_w, \phi_w\}$ chart (see Section 2). Adaptation to the other charts is obtained by circular permutation in the choice of the independent parameters. Algorithm 1 gives a high level overview of the four principal steps now described.

3.1.1. Computing Parameters

The core of the algorithm, with one iteration illustrated by Fig. 6, lies in Lines 9–10. Let $S(\boldsymbol{\lambda})$ be the constraining surface in $\mathbb{T}^n$, defined by the parametric equations:

$$S(\boldsymbol{\lambda}) : (\beta_1, \cdots, \beta_n) = (F_1(\boldsymbol{\lambda}), \cdots, F_n(\boldsymbol{\lambda})) \tag{9}$$

where $F_k(\boldsymbol{\lambda})$ expresses the relation between the ICR and the kth steering angle β. Given an input point $\boldsymbol{q}$, assuming a known initial point $\boldsymbol{p}_0 = S(\boldsymbol{\lambda}_0)$ on the surface and knowing the derivatives at that point ($\boldsymbol{S}_{u,0} = \boldsymbol{S}_u(\boldsymbol{\lambda}_0)$ and $\boldsymbol{S}_{v,0} = \boldsymbol{S}_v(\boldsymbol{\lambda}_0)$), the linear approximation of the surface at the point is the tangent plane $T_{p_0}(S)$, which can be parametrized as:

$$T_{p_0}(S) : \boldsymbol{x} - \boldsymbol{p}_0 = \boldsymbol{S}_{u,0}\Delta u + \boldsymbol{S}_{v,0}\Delta v \tag{10}$$

Δu and Δv are the free parameters and $\boldsymbol{x}$ is the variable: any point $\boldsymbol{x}$ of this plane can be computed given the numerical

Algorithm 1 Iterative ICR estimation algorithm

```
1:  function EstimateLambda(q)
2:      for all λ_0 ← FindStartingPoints(q) do
3:          λ ← λ_0
4:          if ||q − S(λ)|| ≤ ε_δ then
5:              found ← 1
6:          else
7:              count ← 0
8:              repeat
9:                  (S_u, S_v) ← ComputeDerivatives(λ)
10:                 (Δ_u, Δ_v) ← Solve(S_u, S_v, q, λ)
11:                 λ_t ← UpdateParameters(λ, Δ_u, Δ_v)
12:                 λ_t ← HandleSingularities(λ_t)
13:                 if ||q − S(λ_t)|| > ||q − S(λ_0)|| then
14:                     found ← 0
15:                     break
16:                 else
17:                     found ← ||λ − λ_t|| ≤ ε_λ
18:                 end if
19:                 λ ← λ_t, count ← count + 1
20:             until found ∨ count ≥ maxIter
21:         end if
22:         if found then
23:             return λ
24:         end if
25:     end for
26:     return closest λ
27: end function
```

values of the free parameters and $\boldsymbol{p}_0$. A point of particular interest on that plane is the orthogonal projection $\boldsymbol{p}'_1$ of $\boldsymbol{q}$, since it is a first-order approximation of the projection of $\boldsymbol{q}$ onto S. For that particular point:

$$\boldsymbol{p}'_1 - \boldsymbol{p}_0 = \boldsymbol{S}_{u,0}\Delta u_1 + \boldsymbol{S}_{v,0}\Delta v_1 \tag{11}$$

Multiplying (11) with respectively $\boldsymbol{S}_{u,0}$ and $\boldsymbol{S}_{v,0}$, and using the fact that every point on the axis through $\boldsymbol{q}$ and $\boldsymbol{p}'_1$ has the same orthogonal projection, i.e.:

$$(\boldsymbol{p}'_1 - \boldsymbol{p}_0) \cdot \boldsymbol{S}_{u,0} = (\boldsymbol{q} - \boldsymbol{p}_0) \cdot \boldsymbol{S}_{u,0} \tag{12a}$$
$$(\boldsymbol{p}'_1 - \boldsymbol{p}_0) \cdot \boldsymbol{S}_{v,0} = (\boldsymbol{q} - \boldsymbol{p}_0) \cdot \boldsymbol{S}_{v,0} \tag{12b}$$

yields (13a) and (13b).

$$(\boldsymbol{q} - \boldsymbol{p}_0) \cdot \boldsymbol{S}_{u,0} = \boldsymbol{S}_{u,0} \cdot \boldsymbol{S}_{u,0}\Delta u_1 + \boldsymbol{S}_{v,0} \cdot \boldsymbol{S}_{u,0}\Delta v_1 \tag{13a}$$
$$(\boldsymbol{q} - \boldsymbol{p}_0) \cdot \boldsymbol{S}_{v,0} = \boldsymbol{S}_{u,0} \cdot \boldsymbol{S}_{v,0}\Delta u_1 + \boldsymbol{S}_{v,0} \cdot \boldsymbol{S}_{v,0}\Delta v_1 \tag{13b}$$

Δu_1 and Δv_1 may then be computed as the solution of the regular system of linear equations (13).

Looking at Fig. 2 and Fig. 4, the relation F between an ICR $\boldsymbol{\lambda}$ and the corresponding steering angle β can be derived by expressing the fact that $\boldsymbol{\lambda}$, $\boldsymbol{s}$ and $\boldsymbol{B}'$ are on the same great circle, where $\boldsymbol{B}' = (\cos(\alpha+\beta)\ \ \sin(\alpha+\beta)\ \ 0)^T$ is the point on the sphere's equator which makes an angle $\alpha+\beta$ with the U axis (shown on Fig. 2). Using (8), this yields:

$$\boldsymbol{s} \times (\cos(\alpha+\beta)\ \ \sin(\alpha+\beta)\ \ 0)^T \cdot \boldsymbol{\lambda} = 0 \tag{14}$$

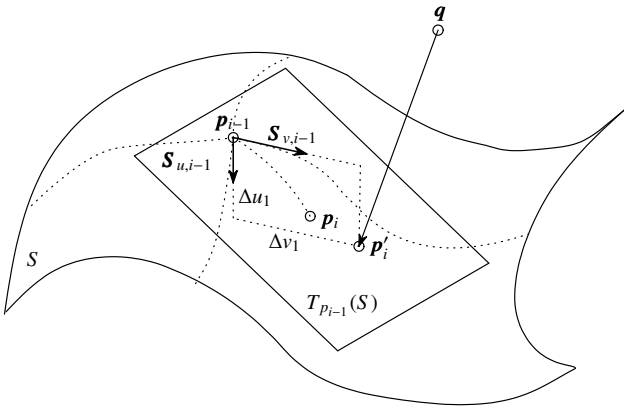

Figure 6: Illustration of one iteration of the ICR estimation algorithm.

which, by using (7), may be rewritten as:

$$\left(\sin\beta(\boldsymbol{a}-\boldsymbol{l}) - \cos\beta\boldsymbol{a}^{\perp}\right)\cdot\boldsymbol{\lambda} = 0 \tag{15}$$

Making explicit β in (15) gives:

$$\tan\beta = \frac{\boldsymbol{a}^{\perp}\cdot\boldsymbol{\lambda}}{(\boldsymbol{a}-\boldsymbol{l})\cdot\boldsymbol{\lambda}} \tag{16}$$

(16) is independent of b and so is valid for centred wheels as well. Using trigonometric identities and (16) gives:

$$\sin\beta = \frac{\mathrm{s}(\boldsymbol{\lambda})(\boldsymbol{a}^{\perp}\cdot\boldsymbol{\lambda})}{\sqrt{((\boldsymbol{a}-\boldsymbol{l})\cdot\boldsymbol{\lambda})^2 + (\boldsymbol{a}^{\perp}\cdot\boldsymbol{\lambda})^2}} \tag{17a}$$

$$\cos\beta = \frac{\mathrm{s}(\boldsymbol{\lambda})(\boldsymbol{a}-\boldsymbol{l})\cdot\boldsymbol{\lambda}}{\sqrt{((\boldsymbol{a}-\boldsymbol{l})\cdot\boldsymbol{\lambda})^2 + (\boldsymbol{a}^{\perp}\cdot\boldsymbol{\lambda})^2}} \tag{17b}$$

where $\mathrm{s}(\boldsymbol{\lambda}) = \mathrm{sgn}((\boldsymbol{a}-\boldsymbol{l})\cdot\boldsymbol{\lambda})$. Applying the $\frac{\partial}{\partial u}$ operator to (15) and using (17), the derivative along u (one component of $\boldsymbol{S}_{u,0}$) may be expressed as:

$$\beta_u = \frac{\partial\beta}{\partial u} = -\frac{[(\boldsymbol{a}^{\perp}\cdot\boldsymbol{\lambda})(\boldsymbol{a}-\boldsymbol{l}) + ((\boldsymbol{a}-\boldsymbol{l})\cdot\boldsymbol{\lambda})\boldsymbol{a}^{\perp}]\cdot\frac{\partial\boldsymbol{\lambda}}{\partial u}}{[((\boldsymbol{a}-\boldsymbol{l})\cdot\boldsymbol{\lambda})(\boldsymbol{a}-\boldsymbol{l}) - (\boldsymbol{a}^{\perp}\cdot\boldsymbol{\lambda})\boldsymbol{a}^{\perp}]\cdot\boldsymbol{\lambda}} \tag{18}$$

and similarly for the other derivatives β_v and β_w. This formulation makes it very easy to handle the dependent parameters, as the ICR derivatives are grouped in a single term. For the $\{U_w, \phi_w\}$ chart and the u coordinate, they are:

$$\frac{\partial\boldsymbol{\lambda}}{\partial u} = \begin{pmatrix}1 & 0 & -\frac{u}{w}\end{pmatrix}^T \tag{19}$$

3.1.2. Handling Invalid Parameters

The Newton-Raphson algorithm is known to generate out-of-bound parameters when dealing with bounded surfaces [28], so care must be taken when updating the parameters:

$$u_i = u_{i-1} + \Delta u_i \tag{20a}$$

$$v_i = v_{i-1} + \Delta v_i \tag{20b}$$

Two different situations have to be dealt with when the parameters are dependent:

- u_i and v_i are invalid, because they do not verify the condition $u_i^2 + v_i^2 \leq 1$ and the resulting point is thus no more on the sphere. As a solution, they are recursively multiplied by the norm $\sqrt{u_{i-1}^2 + v_{i-1}^2}$ until they fulfil the condition.
- u_i and v_i are valid, but determine a steering configuration which is far away from the input. This indicates divergence of the algorithm.

To avoid divergence of the algorithm, the following solutions may be considered:

1. Discard the new iteration and accept the last one as a possible solution. This can prove acceptable if multiple starting points close to the solution are evaluated and the best solution is kept. It has been used successfully in [21], but it needs a resource-intensive starting points selection mechanism to work and this should be avoided.
2. Individually saturate the value of the parameter which goes out-of-bound for an open surface or wrap the value if it is closed [27]. This is a simple solution, but does not preserve the search direction and works only for independent parameters.
3. Find the intersection of the line linking the current and new parameters with the domain boundary [28]. This is a more complex approach, but it has the advantage of preserving the search direction. However, for surfaces parametrized with dependent parameters, a discontinuity in the surface does not necessarily correspond to a bound in the parameters' domain. Numerical errors then make the solution impractical.
4. Backtrack, by reducing the step size $(\Delta u_i, \Delta v_i)$ made until an acceptable solution is found. This preserves the search direction. Press et al. [29] propose a sophisticated algorithm which guarantees to find a solution if there is no local minima, but it requires a lot of computational resources. We use the simpler approach of recursively halving the step size until an acceptable solution is found, ensuring a minimum value for this step to avoid infinite recursion. Since there is however no more guarantee of finding a solution, if that minimum value is reached (called backtracking failure in the remaining), the last successful iteration is kept as the current best solution and a new starting point is tried.

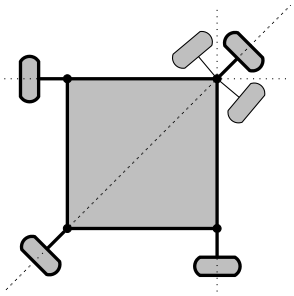

Figure 7: ICR on a wheel, represented with three possible steering configurations for the same ICR.

3.1.3. Handling Singularities

Since the proposed parametrization for the ICR position is singularity-free, only structural singularities (i.e., those that arise when linking the chassis' motion to the actuators' motion) have to be dealt with. These singularities happen when (16) is undefined for a particular wheel, which also corresponds to the singularity of (18). Since $\mathbf{0} \notin \mathbb{S}^2$, this happens only when $\boldsymbol{\lambda}$ is simultaneously perpendicular to $\boldsymbol{a}^\perp$ and $\boldsymbol{a} - \boldsymbol{l}$:

$$\boldsymbol{\lambda} = \frac{\boldsymbol{a}^\perp \times (\boldsymbol{a} - \boldsymbol{l})}{\| - \boldsymbol{a}^\perp \times (\boldsymbol{a} - \boldsymbol{l})\|} = \frac{\boldsymbol{s}}{\|\boldsymbol{s}\|} \tag{21}$$

i.e., when the ICR is on a steering axis. This singularity being structural, it cannot be lifted by a change of parametrization, but it is easy to handle. Indeed, since the robot has at least three unaligned wheels, the ICR is globally always defined: if it is on a steering axis, the other wheels unambiguously define it, as illustrated by Fig. 7. Since the ICR may only be on a particular steering axis k at any given time, only the k-coordinate of $\boldsymbol{S}_u$ and $\boldsymbol{S}_v$ is then undefined, and it is simply set to zero. This discards the contribution of wheel k when solving (13). If the test point $\boldsymbol{\lambda}_t$ corresponding to the next iteration is still on the steering axis, the singularity is kept as the solution and β_k is copied from the input vector.

3.1.4. Starting Point Selection

Having a good initial starting point is key to finding a solution when using Newton-based algorithms. However, no generic method exists to find one for parametric surfaces. The computer-aided design community has some interesting propositions [30, 31], but they all require the surface to be represented as a non-uniform rational B-spline (NURBS).

Since any starting point could lead to divergence of the algorithm or to a backtracking failure, more than one starting point should be evaluated. The ICR being defined by the intersection of all propulsion axes, a sensible choice of starting points are the different intersections defined by each wheel pairs. Indeed, if the ICR is well defined, the first well-defined intersection (i.e., when both propulsion axes are not collinear) gives the true value and no iteration is necessary (Line 5). If the ICR is not very well defined, one intersection gives a point close to the solution, and the algorithm converges in few iterations. If the ICR is completely undefined, the different intersections give distant starting points, which helps avoid being stuck in local minima.

The intersections are simply found by taking the cross-product of the normal vectors to the great circles defined by each propulsion axis, using (8). If the propulsion axes are collinear, that "intersection" is not considered. To maximize the algorithm efficiency, the starting points are ordered by their distance to the input point and the closest point is tried first. Only the first three starting points are kept, since tests have shown that evaluating more than three points does not enhance significantly the results compared to the computing resources used.

3.2. Algorithms Working in the Operational Space

Iterative algorithms are often associated with lack of efficiency and precision, mostly because they are not fixed-time and use thresholds. Thus, to challenge IT, three non-iterative algorithms have been designed: the first does no estimation at all, the second uses a standard least squares approximation and the third uses a null-space computation. They all work in the operational space.

3.2.1. No Estimation (NE)

This algorithm is similar to the starting-point selection mechanism used for IT. It corresponds to the case where no estimation is done and ICR definiteness is taken for granted.

The intersection of the great circles corresponding to the first and second wheel's propulsion axis is computed and considered as the estimated ICR. When this computation gives a null vector (i.e., the great circles are coplanar), the second wheel is not considered and the third wheel is considered instead. Since the robot is guaranteed to have its steering axes not all aligned, this second computation is guaranteed to give a non-null vector when the first one is null. If the robot has more than three wheels, those wheels are not considered for the computation.

3.2.2. Least Squares Estimation (LS)

Each propulsion axis defines a line in $\mathbb{R}^2$ which goes through the steering axis A_k (see Fig. 4) and has a direction vector $\boldsymbol{d}_k = (\cos\gamma_k \;\; \sin\gamma_k)^T$, where $\gamma_k = \alpha_k + \beta_k$ is the angle the propulsion axis makes with the X_R axis. Its parametric equations are then:

$$x = l_k \cos\alpha_k + p_k \cos\gamma_k \tag{22a}$$

$$y = l_k \sin\alpha_k + p_k \sin\gamma_k \tag{22b}$$

where p_k is the free parameter.

For motion to be possible, the ICR must lie on that line. Thus, for each wheel, there exists a p_k that gives the ICR position, and conversely, each p_k imposes a constraint on the ICR position. When considering all wheels, those constraints may be expressed in the following matrix form:

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{B} \tag{23}$$

where $\boldsymbol{A}$ is a sparse $2n \times (2+n)$ matrix, $\boldsymbol{B}$ is a $2n \times 1$ matrix and $\boldsymbol{X}$ is a $(2+n) \times 1$ matrix. For a two-wheeled robot, they would be:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & -\cos\gamma_1 & 0 \\ 0 & 1 & -\sin\gamma_1 & 0 \\ 1 & 0 & 0 & -\cos\gamma_2 \\ 0 & 1 & 0 & -\sin\gamma_2 \end{pmatrix} \tag{24}$$

$$\boldsymbol{B} = \begin{pmatrix} l_1\cos\alpha_1 & l_1\sin\alpha_1 & l_2\cos\alpha_2 & l_2\sin\alpha_2 \end{pmatrix}^T \tag{25}$$

$$\boldsymbol{X} = \begin{pmatrix} x & y & p_1 & p_2 \end{pmatrix}^T \tag{26}$$

Finding the coordinates x and y of the ICR position as the intersection of all lines may then be found by solving (23). Since $n \geq 3$, the system is overdetermined and is solved using a least squares approximation. Once x and y have been estimated, (5) is used to derive $\boldsymbol{\lambda}$.

Note that when $\gamma_i = 0$ or $\gamma_i = \frac{\pi}{2}$ for all i (i.e., all lines are horizontal or vertical), $\boldsymbol{A}$ is degenerate and cannot be inverted. Both cases need to be handled separately, by setting the corresponding ICR to respectively $\boldsymbol{\lambda} = (0\ 1\ 0)^T$ and $\boldsymbol{\lambda} = (1\ 0\ 0)^T$.

3.2.3. Null-space Computation (NS)

This algorithm is an adaptation of LS to the H representation. The ICR existence condition (15) gives a constraint on each wheel in the form:

$$\left(\sin\beta_k(\boldsymbol{a}_k - \boldsymbol{l}_k) - \cos\beta_k\boldsymbol{a}_k^{\perp}\right) \cdot \boldsymbol{\lambda} = \boldsymbol{s}_{1,k}^{\perp}(\boldsymbol{\lambda}) \cdot \boldsymbol{\lambda} = 0 \tag{27}$$

When considering all wheels, the ICR must then lie in the null space of the $n \times 3$ matrix $\boldsymbol{S}$ whose lines are the transpose of the $\boldsymbol{s}_{1,k}^{\perp}(\boldsymbol{\lambda})$ vectors. Contrary to $\boldsymbol{A}$ of LS, $\boldsymbol{S}$ is never degenerate and so there are no special cases to handle. When $\boldsymbol{\beta}$ does not correspond to a well-defined ICR, the SVD used to compute the null space of $\boldsymbol{S}$ will give an approximation of $\boldsymbol{\lambda}$.

4. Experimental Results

The algorithms presented in Section 3 are generic, in the sense that they are valid for any mobile robot having three or more centred or AZIMUT wheels. To evaluate their use in a motion controller on-board a real robot, data was acquired from the AZIMUT-3 robot [12] pictured on Fig. 8. It is a symmetric, nonholonomic omnidirectional platform having four AZIMUT wheels. For this prototype, steering uses differential elastic (compliant) actuators [32] and propulsion uses standard stiff actuators. The platform has steering limitations: each wheel is able to steer with $\beta \in]-\frac{\pi}{2};\frac{\pi}{2}]$ rad.

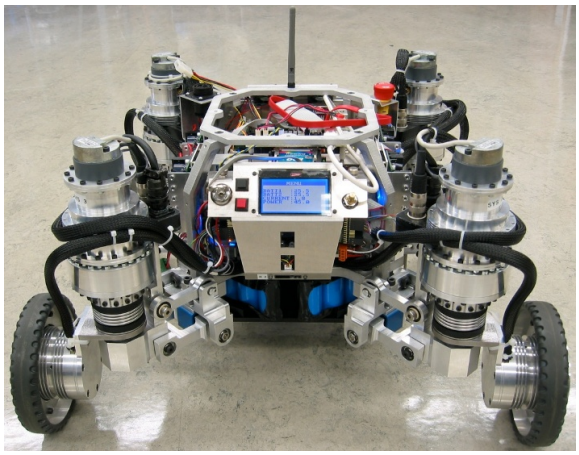

Figure 8: AZIMUT-3, a nonholonomic omnidirectional mobile platform.

Figure 9 illustrates a trajectory followed by AZIMUT-3 in $\mathbb{R}^2$ and Fig. 10 illustrates that trajectory in the steer angles' space using the same representation depicted in Fig. 5 (a closer view of the top left corner), with the constraining surface as reference. The first part of the trajectory (upper right portion of the figure) is a spiral, i.e., a continuously moving ICR with concurrent propulsion axes. It is followed by a circle, i.e., a constant ICR with concurrent propulsion axes, and terminated by a square with rounded corners, i.e., switches between parallel propulsion axes (lower part of the figure) and concurrent ones (upper part of the figure), with sharp transitions in-between.

The observed trajectory confirms that the propulsion axes seldom concur (i.e., the steer angles' trajectory does not stay on the constraining surface) during real motion, which means they do not define an ICR and it needs to be estimated. It is however never completely undefined, since the trajectory remains in a bounded region around the constraining surface.

4.1. Evaluation of ICR Estimation Algorithms

An ICR estimation algorithm should be both reliable and efficient on real robots, i.e., when the steer angle' trajectory stays in a bounded region around the constraining surface. It should also be accurate and robust, i.e., be usable in the whole domain of definition ($\mathbb{T}^n$), as the robot could start in any steering configuration or collide with an object that would change the orientation of its wheels. To help evaluate

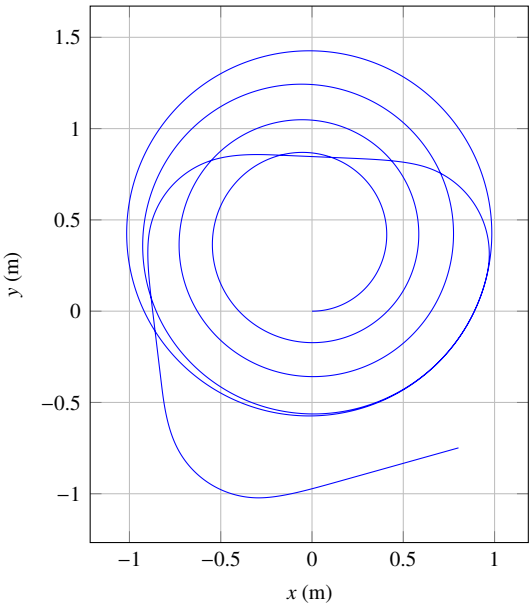

Figure 9: A trajectory followed by AZIMUT-3 in $\mathbb{R}^2$.

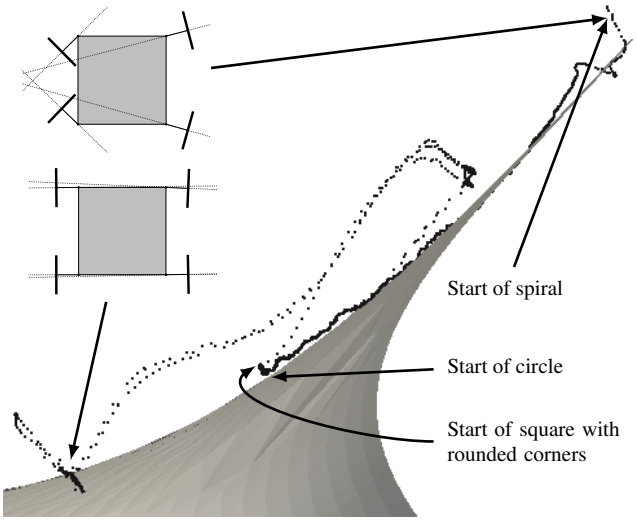

Figure 10: Representation in the steer angles' space of the trajectory (dots) shown on Fig. 9, with the constraining surface as reference and two example resulting wheel configurations.

those four performance indicators, three datasets have been designed:

- $D1$: A random sampling of Λ containing 15000 points.
- $D2$: The real-world trajectory depicted on Fig. 10 containing 2450 points (after removing duplicates).
- $D3$: A random sampling of $\mathbb{T}^4$ containing 15000 points.

To evaluate a good ICR estimation, a metric of quality is needed. That metric may only be relative to the input point, since ground truth is only known with $D1$. The orthogonal projection having the shortest distance between a point and its projection, the distance between the input steering configuration $\boldsymbol{q}$ and the projection $\boldsymbol{\beta}$ found on the constraining surface is used as the base for the metric. The maximum possible distance for each β_k is π, since β_k and $\beta_k + \pi$ represent the same ICR, and therefore it may be used for normalization. The base metric is then defined as:

$$m(\boldsymbol{\beta}_e, \boldsymbol{q}) = \frac{(\boldsymbol{q} - \boldsymbol{\beta}_e) \cdot (\boldsymbol{q} - \boldsymbol{\beta}_e)}{n\pi^2} \tag{28}$$

Since the maximum value for m is much larger than the majority of the values obtained during normal trajectories (i.e., those present in $D2$), it is useful to change the linearity of the metric to better discriminate between values close to zero. It is also more intuitive to have a high quality value represented as being close to one. For these reasons, the metric used to evaluate the quality $\mathrm{qua}(\boldsymbol{\beta}_e, \boldsymbol{q})$ of the estimation $\boldsymbol{\beta}_e$ of $\boldsymbol{q}$ is:

$$\mathrm{qua}(\boldsymbol{\beta}_e, \boldsymbol{q}) = 1 - \frac{\ln(fm + 1)}{\ln(f + 1)} \tag{29}$$

where $f \geq 1$ is a scaling factor. The value $f = 500$, used for the results presented hereafter, has been chosen because it provides a good discrimination for those results. The resulting value is a percentage indicating how close to the input point the estimated ICR is. It may also be seen as a way to quantify how well the propulsion axes define the ICR.

Most of the algorithms described in Section 3 use linear algebra computations. For IT, solving the regular linear system (13) is done using the Cramer method. For LS, the overdetermined linear system (23) is solved using LAPACK [33], which is also used to compute the singular value decomposition needed to find the null-space of $\boldsymbol{S}$ when using NS.

4.1.1. Reliability and Efficiency

Reliability is defined as the ability to find the closest ICR to a steering configuration encountered on a real robot. It has been evaluated by computing $\mathrm{qua}(\boldsymbol{\beta}_e, \boldsymbol{q})$ when $\boldsymbol{q}$ is in $D2$.

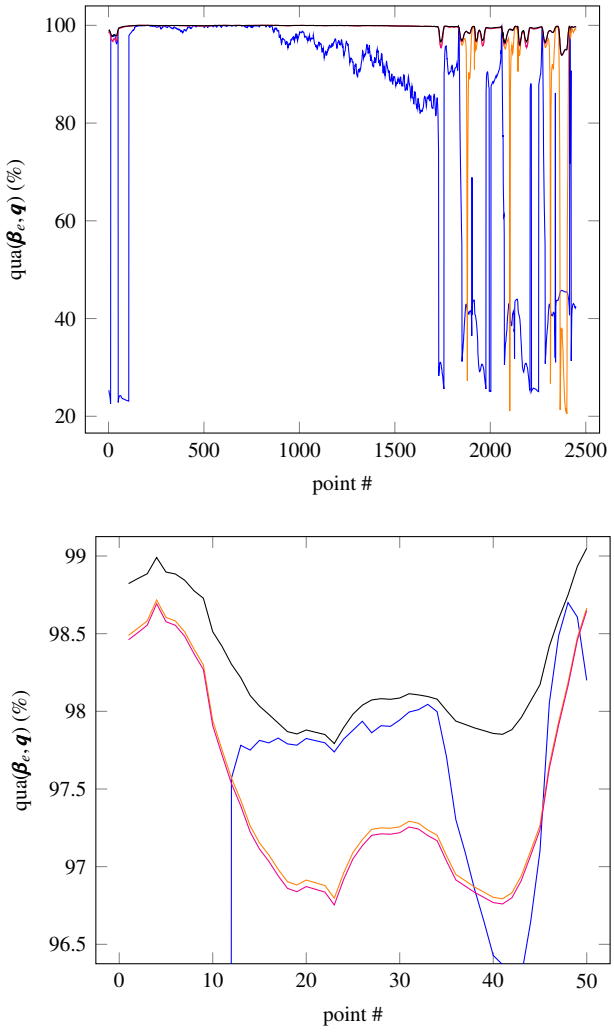

Figure 11: Reliability evaluation by computing qua($\boldsymbol{\beta}_e, \boldsymbol{q}$) when using $D2$, with IT (black), NE (blue), LS (orange) and NS (magenta): whole trajectory (top) and enlarged view for the first 50 points (bottom).

Figure 11 illustrates that both IT and NS perform well, with IT giving always the same or a better quality (as detailed in the bottom part of the figure). NE does not perform well except for very well defined ICR, and LS performs well only when the propulsion axes are not closely parallel.

Efficiency is defined by the ability to run inside a motion controller without affecting its performance too much. It has been evaluated by measuring the computing resources needed to run an algorithm. To make the measure not too dependent on the hardware, processor cycles were counted. Figure 12 presents a comparison of the algorithms against all datasets. The reported value is the total number of instructions used to process one dataset divided by the number of points it contains. Even if IT has not been tuned and optimized like the library used for LS and NS, it is already faster than those algorithms for most steering configurations encountered on a real robot (i.e., those in $D2$). As regards $D3$, most of the steering configurations it contains (like those shown on Fig. 13) would only be encountered on a real robot if its wheels would collide with objects, which should occur very rarely. This suggests that IT is fast enough to be used inside a real-time controller on-board a robot, which has been confirmed by all the tests done on AZIMUT-3 to date.

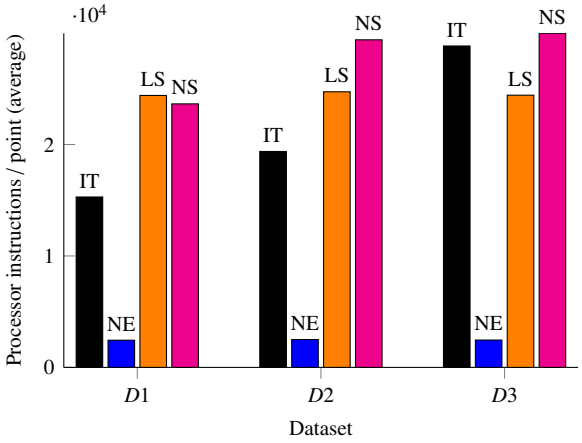

Figure 12: Efficiency evaluation by comparing the computing resources needed by the algorithms against the different datasets.

4.1.2. Accuracy and Robustness

Accuracy is defined as the ability to perfectly estimate a well-defined ICR. It has been evaluated by computing qua($\boldsymbol{\beta}_e, \boldsymbol{q}$) when $\boldsymbol{q}$ is in $D1$. qua($\boldsymbol{\beta}_e, \boldsymbol{q}$) = 1 was observed for all input points and each algorithm, which means they are all accurate. Robustness is defined as the ability to estimate an ICR that is always valid (i.e., a point in Λ) and that corresponds to the closest possible ICR-defining steering configuration. It has been evaluated by computing qua($\boldsymbol{\beta}_e, \boldsymbol{q}$) when $\boldsymbol{q}$ is in $D3$. All estimated ICR were first checked for validity, which was the case for all points and all algorithms. qua($\boldsymbol{\beta}_e, \boldsymbol{q}$) was then computed, as summarized in Table 1. It is observed that IT stands above NE, LS and NS. It is interesting to see that NE is on average a little better than LS. This explains the very good performance of IT. Indeed, it starts with the intersection of two wheels, as does NE. However, it evaluates if needed all possible intersections instead of only one, and locally improves these. Figure 13 illustrates the difference in robustness between IT and LS or NS: the latter give a solution in the "good" direction, but which tends to be too close to the chassis' centre.

Table 1: qua($\boldsymbol{\beta}_e, \boldsymbol{q}$) when using dataset $D3$.

	IT	NE	LS	NS
Min. %	32.47	15.56	12.04	12.21
Mean %	67.18	49.94	47.24	48.02

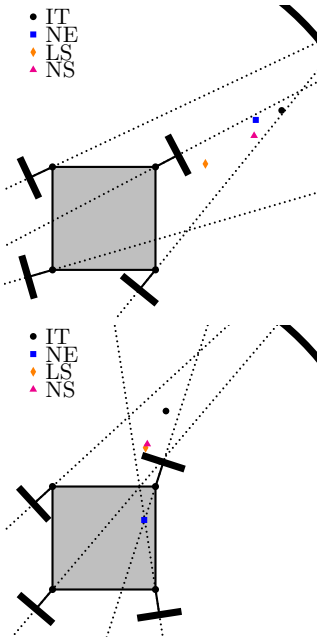

Figure 13: Robustness evaluation: steering configurations in dataset $D3$ and the corresponding orthogonal projection on the motion plane of the ICR as estimated by the algorithms. The bold arc (upper right) represents the unit sphere's equator.

4.2. Discussion

All algorithms evaluated in our study reveal to be accurate when the propulsion axes define an intersection, and they all estimate a valid ICR even with completely random steering configurations. They are thus safe to be used for ICR estimation. When considering a real robot and assuming its control is ICR-based, the direct kinematic model – for which ICR estimation is needed – might be used for trajectory planning and motion control. When doing motion planning, ICR existence is guaranteed and efficiency is paramount. In that situation, NE is a practical choice, because it is the fastest. When doing motion control, however, the ICR is seldom well defined by the propulsion axes and could also be badly defined after colliding with objects. Reliability and robustness are then paramount, but efficiency should also be considered as real-time control must be achievable. Our results suggest that IT delivers the best compromise in that situation.

Transforming IT into a time-capped algorithm is difficult, as it would require more accurate time-keeping capabilities than what is currently available on mobile processors. However, the estimation time worst case scenario depends only on the number of starting points evaluated, the maximum number of iterations allowed per starting point and the maximum number of backtracking steps allowed per iteration. Thus, if a hard timing constraint is necessary, those parameters may be adjusted consequently. Preliminary tests have shown that reducing the number of iterations and the number of backtracking steps does not influence significantly the robustness of the algorithm, but reducing the number of starting points evaluated does.

5. Conclusion

The ICR is a mathematical concept which can be used to control the motion of nonholonomic mobile platforms using steerable wheels. As it was observed on the mobile platform AZIMUT-3, this ICR is however most of the time not well defined during real trajectories and needs to be estimated. Our approach for ICR estimation leverages the newly defined H representation, which provides a globally singularity-free parametrization of the ICR position and motion around it based on the real projective plane. To overcome the limitations of standard approaches which do ICR estimation in the operational space and fail when the wheels are closely parallel, our approach does ICR estimation in the joint space. Despite being iterative, this approach reveals to be reliable, efficient, accurate and robust with real data.

As presented in [24], the next step in our work is to demonstrate the applicability and usefulness of the proposed H representation and ICR estimation algorithm for ICR-based motion control of nonholonomic omnidirectional platforms like AZIMUT-3.

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

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

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] R. Kittmann, T. Fröhlich, J. Schäfer, U. Reiser, F. Weißhardt, A. Haug, Let me introduce myself: I am Care-O-bot 4, a gentleman robot, in: S. Diefenbach, N. Henze, M. Pielot (Eds.), Mensch und Computer 2015 Tagungsband, Walter de Gruyter, Berlin, Boston, 2015, pp. 223–232.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] T. Jacobs, C. Connette, M. Haegele, A. Verl, Design of wheel modules for non-holonomic, omnidirectional mobile robots in context of the emerging control problems, in: Proceedings of the German Conference on Robotics (ROBOTIK), VDE Verlag, Berlin, Offenbach, 2012, pp. 135–138.

[19] P. R. Giordano, M. Fuchs, A. Albu-Schäffer, G. Hirzinger, On the kinematic modelling and control of a mobile platform equipped with steering wheels and movable legs, in: Proceedings of the International Conference on Robotics and Automation, IEEE, 2009, pp. 4080–4087.

[20] 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.

[21] 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.

[22] T. L. Lam, H. Qian, Y. Xu, G. Xu, Omni-directional steer-by-wire interface for four wheel independent steering vehicle, in: Proceedings of the International Conference on Robotics and Automation, IEEE, 2009, pp. 1383–1388.

[23] 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.

[24] L. Clavien, M. Lauria, F. Michaud, Instantaneous centre of rotation based motion control for omnidirectional mobile robots with sidewards off-centred wheels, Robotics and Autonomous Systems (2018).

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

[26] J. P. Snyder, Map Projections – A Working Manual, United States Government Printing Office, Washington, 1987.

[27] L. A. Piegl, W. Tiller, The NURBS Book, Monographs in Visual Communication, 2 ed., Springer, New-York, 1997.

[28] X.-M. Liu, L. Yang, J.-H. Yong, H.-J. Gu, J.-G. Sun, A torus patch approximation approach for point projection on surfaces, Computer Aided Geometric Design 26 (2009) 593–598.

[29] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes: The Art of Scientific Computing, 3 ed., Cambridge University Press, Cambridge, 2007.

[30] L. A. Piegl, W. Tiller, Parametrization for surface fitting in reverse engineering, Computer-Aided Design 33 (2001) 593–603.

[31] Y. L. Ma, W. T. Hewitt, Point inversion and projection for NURBS curve and surface: Control polygon approach, Computer Aided Geometric Design 20 (2003) 79–99.

[32] 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.

[33] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. D. J. Dongarra, J. D. Croz, A. Greenbaum, S. H. A. McKenney, D. Sorensen, LAPACK Users' Guide, 3rd ed., Society for Industrial and Applied Mathematics, Philadelphia, 1999.