Instantaneous Centre of Rotation Estimation of an Omnidirectional Mobile Robot

Lionel Clavien
Génie électrique et génie informatique
Université de Sherbrooke
Sherbrooke QC – Canada
Email: lionel.clavien@usherbrooke.ca

Michel Lauria
University of Applied Sciences
Western Switzerland (HES-SO)
Geneva – Switzerland
Email: michel.lauria@hesge.ch

François Michaud
Génie électrique et génie informatique
Université de Sherbrooke
Sherbrooke QC – Canada
Email: francois.michaud@usherbrooke.ca

***Abstract*—Redundantly actuated mobile robots using conventional wheels need a precise coordination of their actuators in order to guarantee a safe and precise motion without generating high internal forces and slippage. Using the instantaneous centre of rotation (ICR) of the chassis to describe this motion is a well established method. But the ICR is a mathematical concept which is hardly achieved on a real robot. This paper addresses the problem of ICR estimation of a non-holonomic omnidirectional mobile robot using conventional wheels. Instead of estimating the ICR in the working space, our approach estimates it in the actuators' space. The algorithm is presented in its general form and then adapted for a particular robot. The use of the algorithm with other omnidirectional robots is also discussed. Results from extensive testing done in simulation as well as with a real robot are presented, demonstrating the effectiveness of the proposed method.**

I. Introduction

Omnidirectional wheeled mobile robots have become quite popular due to their high degree of manoeuvrability. They can be holonomic or non-holonomic [1], [2]. Non-holonomic omnidirectional robots have a reduced velocity state space compared to holonomic ones. Using the concepts introduced by Campion et al. [3], they are characterized by having a degree of steerability of two ($\delta_s = 2$) and a degree of mobility of one ($\delta_m = 1$). Using the rotation around the instantaneous centre of rotation (ICR) of their motion is then a very appropriate way of describing their velocity state. This ICR is defined as being the point in the robot frame that instantaneously does not move in relation to the robot. For a non-holonomic robot using conventional wheels, this corresponds to the point where the propulsion axis of each wheel intersect. As the motion in the space of instantaneously accessible velocities is constrained, a trajectory between two velocity states has to be calculated for the robot to move. On real robots, the intersection of the propulsion axes is not always well defined (see Fig. 1(b)) and therefore the best estimation of the current ICR must be found given the combination of all those axes. However, most non-holonomic robots that explicitly use the ICR to control their motion do not estimate it. They assume an always well defined ICR by relating it to each wheel independently. This allows them to simply use the inverse kinematics model to estimate their state [4]. With very stiff actuators and a short experimentation time, this assumption is valid. However, if compliant actuators are used to make the robot more responsive and secure to physical contacts, or if the robot is used for a very long period of time, this assumption is no longer valid.

The position of the ICR in the robot frame is defined (non redundantly) by two coordinates. However, most omnidirectional robots have more than two active wheels. As the number of controllable degrees of freedom (DOF) is higher than the DOF of the platform, the system of equations needed to calculate the ICR becomes over-determined. A simple way to solve this problem is to use a least squares estimation (LSE) [5]. But by design, using this method to estimate the ICR when it is close to infinity and ill-defined is difficult (see Sect. II).

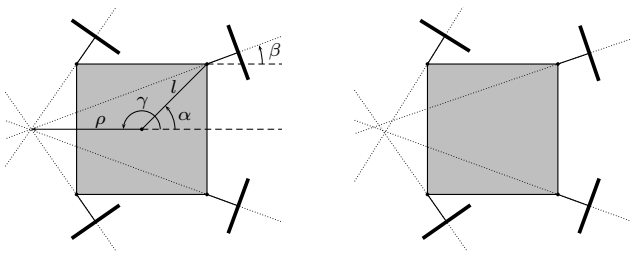
This paper presents an innovative solution for ICR estimation of non-holonomic omnidirectional robots using conventional wheels. The solution is based on a real-time iterative method which estimates the ICR as a projection in the actuators' space. The aim of the proposed solution is to obtain the best possible estimation when the ICR is close to infinity. A good estimation is being defined as the estimated ICR stays close to infinity when the propulsion axes are closely parallel.

The paper is organized as follows. The concept of ICR and its parameterization are first introduced. Then, the issues related to the standard way of estimating the ICR are presented, followed by the details of the new method. Adaptation of the method to the AZIMUT robot is then presented, followed by comparative results on that platform.

II. ICR estimation

There are two standard ways to define the velocity state of a robot chassis. The first is by using its twist (linear and angular velocities), and the second is by using its rotation around the ICR of its motion [3].

The twist representation is well adapted for holonomic robots. But for non-holonomic ones, instantaneously accessible velocities are limited and the representation using the rotation around the ICR is more adapted. This is particularly true for redundantly actuated robots using conventional wheels, where all the wheels must be precisely coordinated to enable motion.



(a) Mathematics and notation (b) Reality

Fig. 1. ICR defined by the intersection of the propulsion axes and notation

The ICR position in the robot frame can be parameterized using two independant parameters. In the following, a parameterization using polar coordinates is used. Fig. 1(a) presents, for one axis, the main variables used throughout this paper. The polar coordinates of the ICR are (ρ, γ) and the propulsion axis' angle is named β.

When the ICR is close to the robot, it is usually well defined during motion and its estimation is not difficult to achieve. But typically, when the robot moves, the ICR is close to infinity (see Fig. 7(b) for an example) and this is where the estimation becomes challenging, because a small variation of β implies a big variation of the ICR position.

Robots using compliant actuators have drawn a lot of interest recently [6], [7]. The use of such actuators makes it possible to compensate for the fact that the propulsion axes may not converge to a single point because of wheel slippage, sensing errors, etc. Fig. 1(b) illustrates this situation for a four-wheel-steer/four-wheel-drive robot. However, compliant actuators provide lower stiffness, contributing in making the ICR ill-defined. But even if non-compliant actuators were used, at initialization the wheels could be in any position, making the ICR ill-defined. Therefore, a method to estimate the ICR is required.

One solution is based on least squares. The ICR is estimated as the point that minimizes its distance to each propulsion axis [5]. Using a LSE to estimate the ICR has the advantage of being a fast method with well known implementations [8]. However, LSE does not cope well with parallelism or near-parallelism. Indeed, when most or all the wheels are parallel, the system of equations describing the relationships between the wheels' axes and the ICR position degenerates up to the point where no solution exists (because there are more unknowns than available equations).

As an alternative solution, the idea consists in considering the ICR not as a free point in the working space, but as a constrained point in the actuators' space. Indeed, all ICR reachable by the propulsion axes create a hyper-surface in the actuators' space, referenced from now on by the constraining surface. Estimating the ICR is then done by finding the orthogonal projection of the input point (the values of the propulsion axes) onto the surface. For this purpose, our approach uses a first order geometric iterative algorithm which finds the projection by successive approximations, as described in [9].

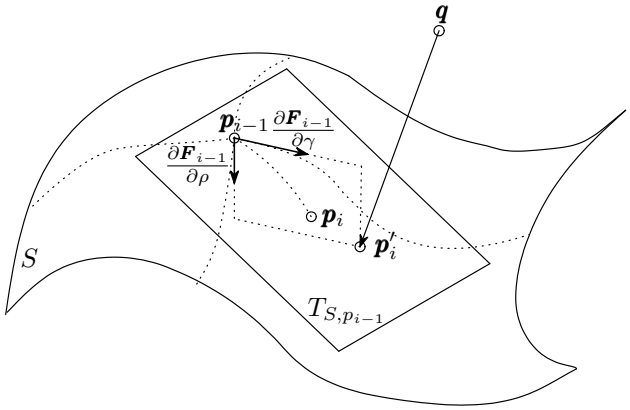


Fig. 2. Illustration of the ith iteration of the algorithm

Let S be the constraining surface (in parametric form) and n the number of conventional wheels.

$$S\colon (\beta_1, \cdots, \beta_n) = (F_1(\rho, \gamma), \cdots, F_n(\rho, \gamma)) \tag{1}$$

where

$$F_k(\rho, \gamma) = \arctan \frac{\rho \sin\gamma - l \sin\alpha_k}{\rho \cos\gamma - l \cos\alpha_k},\ k = 1, \ldots, n \tag{2}$$

Given an input point $\boldsymbol{q}$ and assuming an initial point $\boldsymbol{p}_0(\rho_0, \gamma_0)$ on the surface, the linear approximation of the surface at that point is the tangent plane T_{S,p_0} (see Fig. 2 with $i = 1$), which can be parameterized as:

$$T_{S,p_0}\colon \boldsymbol{x} - \boldsymbol{p}_0 = \frac{\partial \boldsymbol{F}_0}{\partial \rho}\Delta\rho + \frac{\partial \boldsymbol{F}_0}{\partial \gamma}\Delta\gamma \tag{3}$$

$\Delta\rho$ and $\Delta\gamma$ are the free parameters and $\boldsymbol{x}$ the variable. A point of particular interest on that plane is the orthogonal projection $\boldsymbol{p}'_1$ of $\boldsymbol{q}$, as it is a first order approximation of the projection of $\boldsymbol{q}$ onto S. For that particular point, we have:

$$\boldsymbol{p}'_1 - \boldsymbol{p}_0 = \frac{\partial \boldsymbol{F}_0}{\partial \rho}\Delta\rho_1 + \frac{\partial \boldsymbol{F}_0}{\partial \gamma}\Delta\gamma_1 \tag{4}$$

Multiplying (4) ($\langle , \rangle$ denotes the scalar product) with respectively $\frac{\partial \boldsymbol{F}_0}{\partial \rho}$ and $\frac{\partial \boldsymbol{F}_0}{\partial \gamma}$ and using the fact that $\langle \boldsymbol{p}'_1 - \boldsymbol{p}_0, \frac{\partial \boldsymbol{F}_0}{\partial \rho}\rangle = \langle \boldsymbol{q} - \boldsymbol{p}_0, \frac{\partial \boldsymbol{F}_0}{\partial \rho}\rangle$ and $\langle \boldsymbol{p}'_1 - \boldsymbol{p}_0, \frac{\partial \boldsymbol{F}_0}{\partial \gamma}\rangle = \langle \boldsymbol{q} - \boldsymbol{p}_0, \frac{\partial \boldsymbol{F}_0}{\partial \gamma}\rangle$ gives (5) and (6).

$$\langle \boldsymbol{q} - \boldsymbol{p}_0, \frac{\partial \boldsymbol{F}_0}{\partial \rho}\rangle = \langle \frac{\partial \boldsymbol{F}_0}{\partial \rho}, \frac{\partial \boldsymbol{F}_0}{\partial \rho}\rangle \Delta\rho_1 + \langle \frac{\partial \boldsymbol{F}_0}{\partial \gamma}, \frac{\partial \boldsymbol{F}_0}{\partial \rho}\rangle \Delta\gamma_1 \tag{5}$$

$$\langle \boldsymbol{q} - \boldsymbol{p}_0, \frac{\partial \boldsymbol{F}_0}{\partial \gamma}\rangle = \langle \frac{\partial \boldsymbol{F}_0}{\partial \rho}, \frac{\partial \boldsymbol{F}_0}{\partial \gamma}\rangle \Delta\rho_1 + \langle \frac{\partial \boldsymbol{F}_0}{\partial \gamma}, \frac{\partial \boldsymbol{F}_0}{\partial \gamma}\rangle \Delta\gamma_1 \tag{6}$$

$\Delta\rho_1$ and $\Delta\gamma_1$ can then be computed as the solution of the regular system of linear equations (5) and (6). Finally, the new starting point $\boldsymbol{p}_1$ belonging to S that will be used for the next iteration is computed by substituing the improved parameters $(\rho_0 + \Delta\rho_1, \gamma_0 + \Delta\gamma_1)$ into (2). The algorithm stops when $\|\boldsymbol{p}_i - \boldsymbol{p}_{i-1}\|^2$ is lower than a preset threshold. The value of that threshold represents a compromise between convergence speed and precision of the estimation.

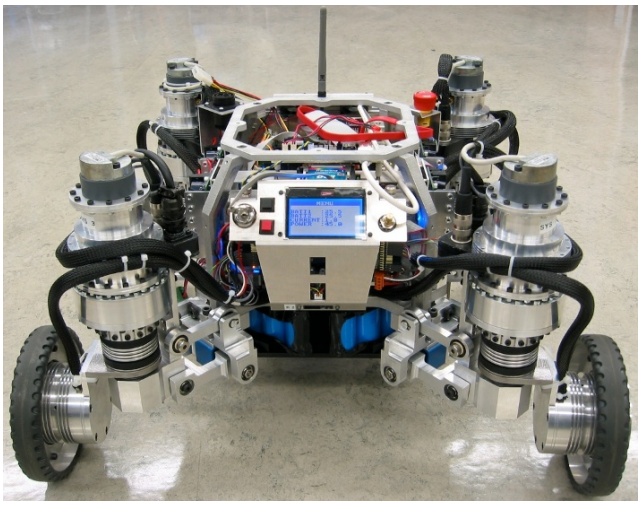


Fig. 3. The AZIMUT-3 platform

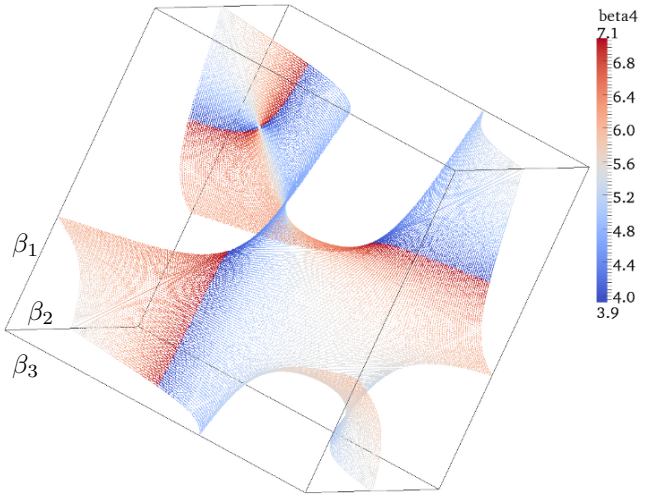


Fig. 4. 3D representation of the grid points, the color map representing the fourth axis

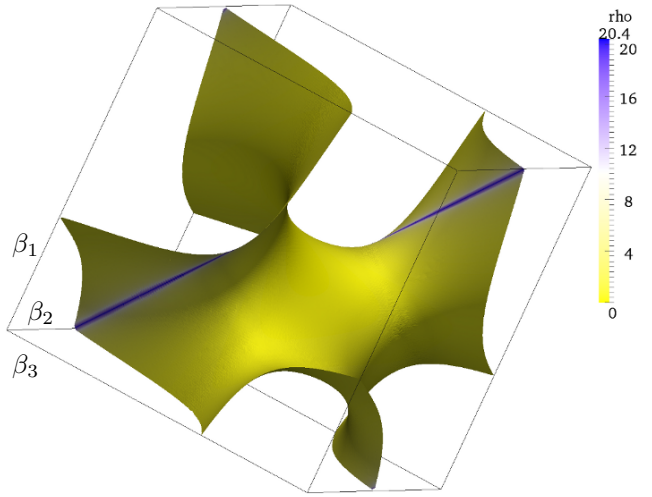


Fig. 5. 3D representation of the constraining surface, the color map representing the distance of the ICR in relation to the robot

III. ICR ESTIMATION ON THE AZIMUT ROBOT

Like most iterative Newton-based first order algorithms, the algorithm depicted in Sect. II is quite sensitive to the choice of the starting point and to the local curvature of the surface. Moreover, it works well with continuous surfaces, but the constraining surface can become quite irregular with real robots. For the algorithm to work, it is therefore important to find a good starting point.

One simple solution is to have the surface discretized and use an algorithm to find the best candidate on the generated grid. The constraining surface and the corresponding grid being specific to each robot, we illustrate our approach using the AZIMUT robot [7], [10]. It is a non-holonomic omnidirectional four-wheel-steer/four-wheel-drive symmetric platform, whose third prototype is pictured on Fig. 3. Here are the issues that need to be addressed when the algorithm is used on that particular robot:

1) When the ICR is at infinity, $\frac{\partial \boldsymbol{F}_{i-1}}{\partial \rho}$ becomes null and the system of equations (5), (6) degenerates.
2) When the ICR is located on direction axis k, $\frac{\partial F_{i-1,k}}{\partial \rho}$ and $\frac{\partial F_{i-1,k}}{\partial \gamma}$ are undefined, because the axis' angle may have any value [11].
3) Each wheel can rotate with β_k defined in the range $]-\frac{\pi}{4}+(k-1)\frac{\pi}{2}, \frac{3\pi}{4}+(k-1)\frac{\pi}{2}]$, where $k \in \{1,2,3,4\}$ represents each wheel starting from the front right corner of the chassis and going counter-clockwise. This splits the constraining surface in unconnected patches.

To help the algorithm cope with those matters, the grid is generated in two phases. First, a two dimensional mesh is generated using β_1 and β_2. For each point of the mesh, (2) is used to calculate an ICR and the corresponding β_3 and β_4. The method is then repeated with all possible combinations of β_k. This generates many duplicates, but ensures a grid with a mostly uniform density. In the second phase, the three issues are addressed. To help handle infinity and surface discontinuities, a particular curve (circle for the infinity, line for the borders of the patches) is discretized in the working space, and then each point is translated into the actuators' space using (2). For an ICR on one direction axis, the three other β_k are determined with (2), and equidistant points in its range are generated for the considered axis. Once all the points have been generated, duplicates or points too close to each other are removed.

Fig. 4 shows a 3D representation of the grid points. The three axes correspond to β_1, β_2, β_3, and the color map illustrates β_4. Even if the surface seems continuous in the 3D space, looking at the color map clearly indicates the discontinuities in the 4D space, located at direct transitions between dark blue and dark red.

Fig. 5 is the result of a 3D Delaunay triangulation of the grid points. The color map illustrates the distance to the robot of the corresponding ICR. This clearly shows how infinity and its vicinity represent a very small part of the constraining surface, albeit representing a very big part of the working space. Estimating the ICR in this area is therefore challenging.

Having the grid points, the next step is to select the best candidates for initializing the iterative algorithm. For this purpose, we use a nearest neighbor finding algorithm. As our algorithm is sensitive to the starting point and to the curvature of the surface, we search for four nearest neighbors. The iterative algorithm is initialized with each point until a projection is found. If no projection is found after evaluating each point, the solution with the least error is kept. Note that to avoid being stuck in local minima, the algorithm is stopped after a maximum of twelve iterations.

To handle the three issues with AZIMUT, the algorithm described in Sect. II needs some adaptations:

1) Dealing with an ICR at infinity is done using a simple trick. As we use a finite threshold (ρ_∞) to describe infinity, the system of equations does not degenerates, but its solution gives very high values for $\Delta\rho$. We simply test for $\rho_i > \rho_\infty$ and stop if the condition is true, assuming a found solution.
2) To handle the fact that the constraining surface is made of unconnected patches, each patch is given an id (we call it a mode) and if the modes of $\boldsymbol{p}_{i-1}$ and $\boldsymbol{p}_i$ are different, the algorithm is stopped.
3) To handle an ICR on one direction axis, each singularity of this kind gets its own mode assigned. If an iteration of the algorithm steps on it, it will then be stopped. But if the starting point is already in such a mode, a modified algorithm adapted to iterate on a line is used to confirm that the input point is on a singularity.

Overall, the approach can easily be adapted for other non-holonomic robots with three or more standard wheels. The main difficulty is to adapt the second item above to the wheel configuration of a particular robot.

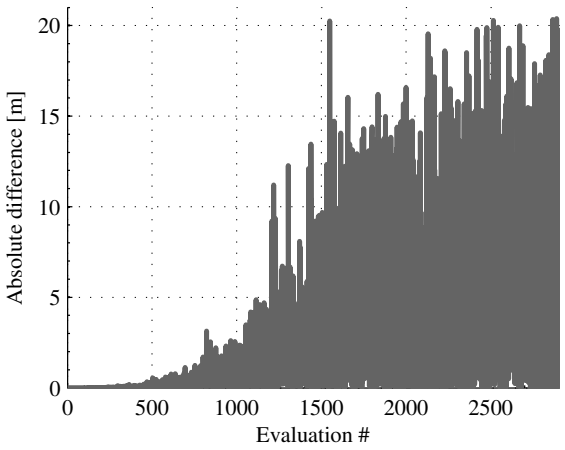
IV. Results

To demonstrate our approach, we compare ICR estimation using LSE and our iterative algorithm, first with simulated data (to make extensive testing) and then with data recorded on the AZIMUT-3 robot. More specifically, we focus on the estimation of the ICR distance relative to the robot (ρ coordinate), because all the tests indicate that no significant differences (<10^{-2} rad) are observed for the ICR angle estimation (γ coordinate) when both algorithms output a similar distance estimation. Note that we assume that for $\rho \geq 20.44$ m, the ICR is at infinity. This value is related to the precision of the robot sensors.

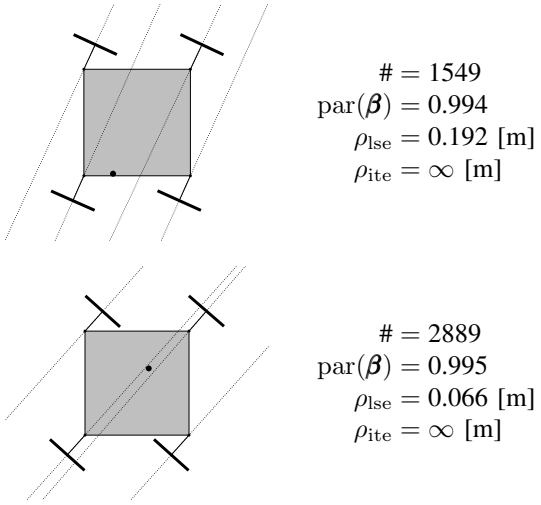
To help evaluate the results, it is useful to define a metric quantifying the parallelism of the propulsion axes, which will make more evident what an "ICR close to infinity" means. To construct this metric, we simply use the relative difference of the β_k. We define

$$\mathrm{par}(\boldsymbol{\beta}) = 1 - \frac{\sigma_\beta}{\sigma_{\beta,max}} \tag{7}$$

where σ_β is the standard deviation of the β_k and $\sigma_{\beta,max} = \frac{\sqrt{11}}{8}\pi$ is the maximum possible value on AZIMUT-3.



(a) Difference between the two algorithms distance estimation



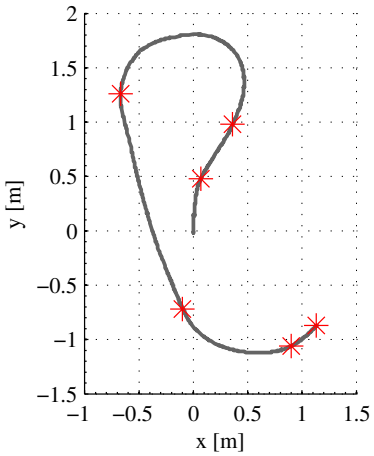
(b) Details of two evaluations

Fig. 6. Results with simulated noise

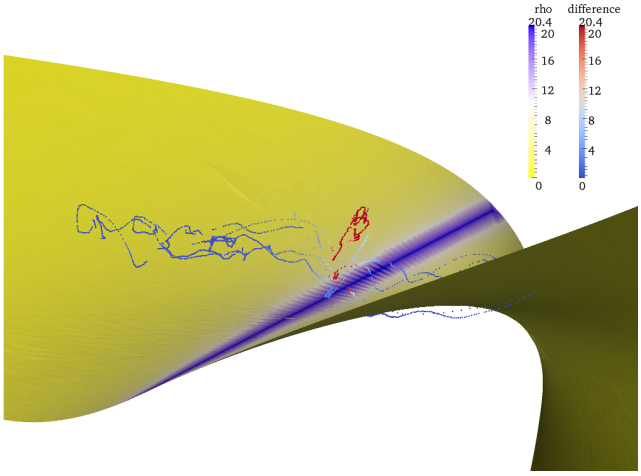
We now have a percentage indicating how parallel to each other the propulsion axes are.

We first conducted tests with simulated data using well-defined ICR to confirm that both LSE and our iterative algorithm provide the same results. To be as exhaustive as possible, a tight Archimedean spiral ($\rho = 0.05\gamma$) was computed to cover the whole working space, from the origin to infinity, and the resulting points in the actuators' space were input to both algorithms. As expected, there were no significant differences (<10^{-6} m) between both algorithms.

We then conducted trials with ill-defined ICR. Using the same points generated precedently, we added a bounded (± 0.02 rad) white noise to each β_k before inputting the point to both algorithms. Fig. 6(a) shows the absolute difference between the estimation from both algorithms. We see that close to the robot, when the ICR is still well defined despite the added noise, both algorithms give similar results. But as the ICR moves towards infinity, the algorithms give quite diverging results. Fig. 6(b) shows the details of two evaluations and demonstrates how the proposed algorithm

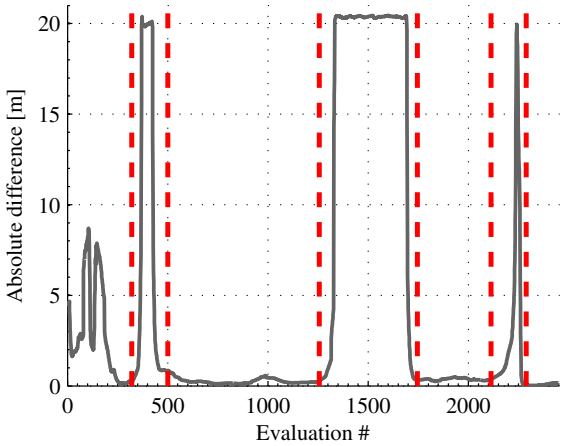


(a) Trajectory in the Cartesian space, starting at (0, 0)

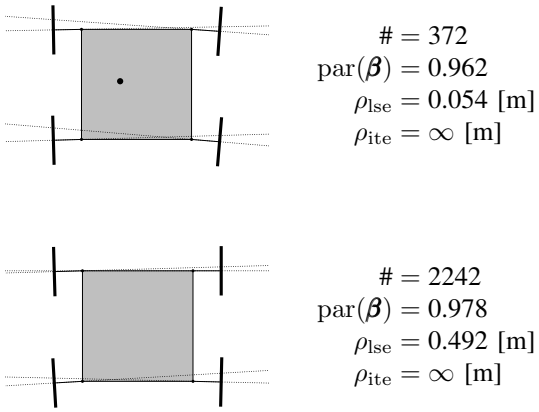


(b) Trajectory in the actuators' space

Fig. 7. Results with data recorded on AZIMUT-3



(a) Difference between the two algorithms distance estimation



(b) Details of two evaluations

Fig. 8. Results with data recorded on AZIMUT-3 (cont.)

improves the estimation compared to LSE. The big dot indicates the position of the ICR as estimated by LSE, ρ_{lse} is the corresponding distance and ρ_{ite} is the same distance estimated by our algorithm.

We finally conducted trials with data taken on the real robot. The trajectory illustrated in Fig. 7(a) shows one of the trial. The orientation of the robot is not represented, but it is used in a car-like fashion, with its velocity vector tangent to its trajectory. Fig. 7(b) shows the same trajectory in the actuators' space. It illustrates how important it is to have a good ICR estimation for ill-defined situations, because with real robot motion, the trajectory in the actuators' space can be quite far from the constraining surface. The color map for the surface indicates the vicinity of the infinity (dark blue band). As with Fig. 6(a), we see that when the ICR is close to the robot, it is quite well-defined and the trajectory in the actuators' space is close to the surface. But when the wheels are closely parallel, that trajectory goes far away from the surface, which means the ICR becomes really ill-defined. Fig. 8(a) shows the same information as the colormap for the trajectory on Fig. 7(b). The vertical dashed lines correspond to the markers on Fig. 7(a). There is a clear correspondence between moves with an ICR close to infinity and the spikes of Fig. 8(a). Two specific evaluations corresponding to those spikes, one at the beginning and one at the end of the trajectory, are detailed on Fig. 8(b). Both show a situation where the propulsion axes are close to parallel. We see that even for a simple trajectory involving no omnidirectional motion, LSE has much trouble estimating real life ICR, where the proposed algorithm gives a much better estimation.

Considering that our approach is iterative, it is important to examine the computational resources it requires, so as to evaluate its use on-board a robot. Analysis of the results from the trials (with simulated and real data) has shown that for most inputs, only the first starting point is evaluated and the algorithm converges in average after 3 iterations. Moreover, only the starting point selection and the projection finding is done on-line: the grid is created off-line and the tree needed for the nearest neighbor search is populated and balanced at initialization only.

Table I shows a comparison of the computing time for the different trials reported earlier. All tests were done on

TABLE I
COMPARISON OF COMPUTING RESOURCES NEEDED BY EACH ALGORITHM

	Algorithm	Simulation (without noise)	Simulation (with noise)	Real robot
Minimum time per evaluation [s]	LSE	3.66 E-6	3.76 E-6	3.69 E-6
	Iterative	9.28 E-7	3.28 E-6	4.63 E-6
Maximum time per evaluation [s]	LSE	1.88 E-5	2.88 E-5	1.87 E-5
	Iterative	3.28 E-5	9.84 E-5	4.71 E-5
Average time per evaluation [s]	LSE	4.97 E-6	4.99 E-6	5.00 E-6
	Iterative	7.61 E-6	1.30 E-5	1.43 E-5
	(Ratio)	1.53	2.61	2.86

the same computer with the same test program, changing only the input file. Both algorithms were run sequentially with each combination of β_k and timed independently. The time reported is the average of 20 calls to each algorithm. Even though our algorithm needs on average 3 times more computing time than LSE (with real data), the time required is still reasonable, considering that AZIMUT's control loop runs at 100 Hz. On the other hand, the algorithm is not as deterministic as the LSE, because not all starting points are always evaluated and not all projections take the same time to converge. This could be problematic in hard real-time situations, but again, for a robot like AZIMUT, this is negligible.

The average time needed by our algorithm is higher with real data than with simulated one. This is because on the real robot, the input point is often quite far from the surface, and more iterations are then needed to obtain a precise estimation. On the other hand, the global quality of the estimation has been shown to be better. Interestingly, the minimum time required for the trials with simulated data is quite lower for our algorithm. This is because the added noise is not very important, so the input point is close or already on the constraining surface. If the input point is very close to a grid point (which can happen as there are many grid points), that point is chosen as the estimation and no iteration is performed, which is quite fast.

V. CONCLUSION AND FUTURE WORK

This work addressed the problem of ICR estimation of non-holonomic omnidirectional robots using three or more conventional wheels. A novel approach has been proposed, which does the estimation in the actuators' space instead of in the working space of the robot. It overcomes the shortcomings of the standard LSE approach, which fails in estimating ill-defined ICR in the vicinity of infinity. The adaptation of the approach to the AZIMUT robot demonstrated how it can be applied to a real platform. Simulation results showed the ability of the proposed approach to estimate ill-defined ICR. Results on a real platform confirmed that the ICR is ill-defined most of the time, even during simple trajectories.

Despite the fact that the approach is iterative, the computing requirements are small and the approach is suitable for soft real-time control.

We are currently working on solutions to cope with the non-deterministic nature of the approach and make it suitable for hard real-time control. In addition, to come up with a single representation of finite and infinite values, we are currently investigating a new parametrization with less singularities, similar to [4], [11], but using a more formal approach.

ACKNOWLEDGMENTS

This work is funded by the Natural Sciences and Engineering Research Council of Canada, the Canada Foundation for Innovation and the Canada Research Chairs. F. Michaud holds the Canada Research Chair in Mobile Robotics and Autonomous Intelligent Systems.

The authors gratefully acknowledge the contribution of their colleagues, François Ferland for his help with the timings results and Julien Frémy for providing raw results from the real robot.

REFERENCES

[1] J. A. Batlle and A. Barjau, "Holonomy in mobile robots," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 433–440, 2009.

[2] I. Kolmanovsky and N. H. McClamroch, "Developments in nonholonomic control problems," *IEEE Control Systems Magazine*, vol. 15, no. 6, pp. 20–36, 1995.

[3] G. Campion, G. Bastin, and B. d'Andréa-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 47–62, 1996.

[4] C. P. Connette, A. Pott, M. Hagele, and A. Verl, "Control of an pseudo-omnidirectional, non-holonomic, mobile robot based on an ICM representation in spherical coordinates," in *Proceedings of the 47th IEEE Conference on Decision and Control*, 9-11 Dec 2008, pp. 4976–4983.

[5] T. L. Lam, H. Qian, Y. Xu, and G. Xu, "Omni-directional steer-by-wire interface for four wheel independent steering vehicle," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, 12-17 May 2009, pp. 1383–1388.

[6] P. F. Santana, C. Candido, V. Santos, and J. Barata, "A motion controller for compliant four-wheel-steering robots," in *Proceedings of the 2006 IEEE International Conference on Robotics and Biomimetics*, 17-20 Dec 2006, pp. 532–537.

[7] M. Lauria, I. Nadeau, P. Lepage, Y. Morin, P. Giguère, F. Gagnon, D. Létourneau, and F. Michaud, "Design and control of a four steered wheeled mobile robot," in *Proceedings of the 32nd Annual Conference of the IEEE Industrial Electronics*, 7-10 Nov 2006, pp. 4020–4025.

[8] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[9] E. Hartmann, "Numerical implicitization for intersection and Gn-continuous blending of surfaces," *Computer Aided Geometric Design*, vol. 15, no. 4, pp. 377–397, 1998.

[10] F. Michaud, D. Létourneau, M. Arsenault, Y. Bergeron, R. Cadrin, F. Gagnon, M.-A. Legault, M. Millette, J.-F. Paré, M.-C. Tremblay, P. Lepage, Y. Morin, J. Bisson, and S. Caron, "Multi-modal locomotion robotic platform using leg-track-wheel articulations," *Autonomous Robots*, vol. 18, no. 2, pp. 137–156, 2005.

[11] B. Thuilot, B. d'Andréa-Novel, and A. Micaelli, "Modeling and feedback control of mobile robots equipped with several steering wheels," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 3, pp. 375–390, 1996.