On-line Kinematics Reasoning for Reconfigurable Robot Drives

Michael Hofbaur   Mathias Brandstötter   Christoph Schörghuber   Gerald Steinbauer

Abstract— **The control system for a mobile robot typically assumes fixed kinematics according to the drive's geometry and functionality. Faults in the system, for example a blocked steering actuator, will then lead to an undesired behaviour, unless one takes care of specific single and/or multiple faults explicitly. We present a novel model-programmed procedure for on-line kinematics reasoning that allows a robot to deduce the (inverse)-kinematics of the drive and also its kinematic abilities for the specific modes of operation and some falt modes during operation. As a consequence, we can reconfigure a robot drive to compensate for some faults and also inform a higher level control system about changed mobility capabilities of a robot. Being fault tolerant is, however, only one advantage of our approach that derives the kinematics control strategy from a geometric and functional model of the drive. We can easily adapt the controller for various robot drives, handle drives that change their geometry and functionality during run-time and also provide the basis for a flexible control scheme for self-configuring multi-robot systems.**

I. INTRODUCTION

Robot kinematics are typically considered at the design stage of a robot. The resulting drive-geometry and functionality reflects the desired mobility capabilities for a robot according to its application. Kinematic analysis [2] provides the basis for the control law that converts a drive command from higher level control (e.g. path planning) into the corresponding actuation (steering angles and rotational speeds) of the individual wheels. This approach implies several limits in robot design and application:

1) Operational faults in the drive, for example a blocked steering, can result in significantly different kinematics so that a fault tolerant control scheme has to consider these cases as well and switch to the appropriate controller upon fault detection. However, given the complex mechatronic designs of typical robot drives, it is easy to see that one might only consider single faults since the number of multiple fault scenarios becomes too large quickly.
2) The path planner that operates a robot typically assumes a specific mobility capability for the robot drive. This assumption is maintained implicitly so that a change in the robot's kinematics (e.g. due to a fault) would require a non-trivial interaction with the path planning procedure.
3) Some applications might require robot drives that change in functionality and geometry. For example, consider a robot that can extend extra wheels for improved traction [5] or a robot that can widen its base to improve its stability (e.g. [7]). An implicit kinematics solution for the drive control only captures the very specific drive design and is unlikely to generalise to a wider range of applications.

Contrarily to this conventional approach for low-level robot control, we want to propose a *model-programmed* control scheme that easily adapts to a diverse set of possible robot drive geometries and functionalities. Even more, we will present a method that allows us to analyse the drive's kinematics on-line during operation. This enables us to handle drives that change their functionality due to faults or even due to a desired operation. It also provides an explicit awareness about a the drive's mobility capabilities. Low-level control can use this information to validate a drive-command from higher level control and, for example, to re-configure a drive-command to compensate for faults in the drive. High-level path planning, on the other hand, can utilise the information to actively plan a path that does not demand movements that are beyond the drive's capabilities. Our control architecture shown in Fig. 1 consists of several model-based control units that can deal with changes in the drive during run-time. The controller actuates each steered and/or actuated wheel of the robot drive through the *drive-unit handler*. The unit represents the interface to the wheel's low-level servo controller and additionally provides the geometric and functional specification of the wheel. This specification represents the single-component building-block

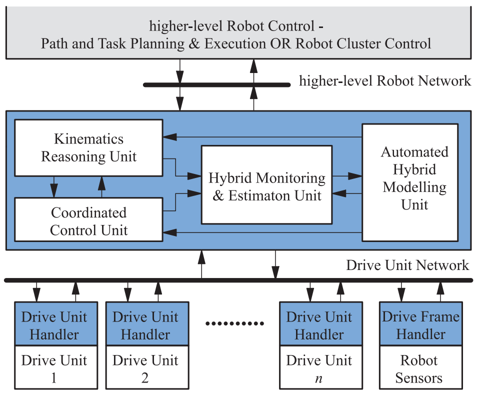
higher-level Robot Control -
Path and Task Planning & Execution OR Robot Cluster Control
higher-level Robot Network
Kinematics Reasoning Unit
Hybrid Monitoring & Estimaton Unit
Automated Hybrid Modelling Unit
Coordinated Control Unit
Drive Unit Network
Drive Unit Handler
Drive Unit Handler
Drive Unit Handler
Drive Frame Handler
Drive Unit 1
Drive Unit 2
Drive Unit n
Robot Sensors

Fig. 1. Single-robot model-programmed control architecture

This research has been funded by the Austrian Science Fund (FWF) under grant P20041-N15.

M. Hofbaur and M. Brandstötter, Institute of Automation and Control Engineering, Private University UMIT, Eduard Wallnöver Zentrum 1, A-6060 Hall in Tyrol, Austria, michael.hofbaur@umit.at; Ch. Schörghuber, Institute of Automation and Control, Graz University of Technology, Kopernikusgasse 24/2, A-8010 Graz, Austria; G. Steinbauer, Institute of Software Technology, Graz University of Technology, Inffeldgasse 16b/2, A-8010 Graz, Austria

that is used by the *automated hybrid modelling unit* to deduce the kinematic and dynamic model for overall robot drive during run-time. The dynamically generated model is mostly used within the *hybrid monitoring and estimation unit* [3] which infers the current mode of operation / failure for the system. This mode, together with the kinematic model from the modelling unit is then used in the *kinematics reasoning unit* to analyse the kinematic capabilities of the drive, to perform necessary reconfigurations and to deduce the currently active inverse kinematics for the drive. This inverse kinematic is then used to compute appropriate set-points for the individual drive units through the *coordinated drive control unit* in an orchestrated way.

Additionally to this novel control scheme, we developed a *modular robotic platform* that allows us to quickly configure wheeled robots with diverse geometry and functionality. The modular system utilises a *6-edge honey-comb shaped prism* as its basic building block. Fig. 2 shows configuration examples for robot drives that can be built with this versatile geometric form.

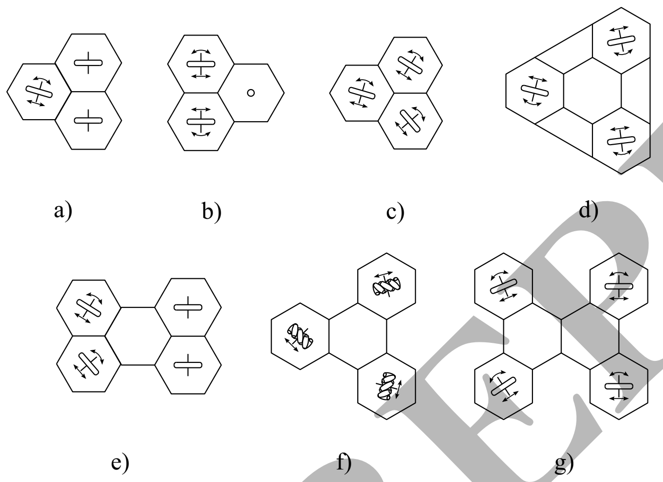

Fig. 2. COMB modular robot drive configurations: (c) shows an omni-directional drive with three steered and actuated standard wheels whereas (b) realizes an omni-drive with two standard wheels and a ball-wheel and (f) denotes an omni-drive with three swedish wheels. Robot drives with both, steered and unsteered wheels for a tricycle- (a) and a car/ackermann configuration (e).

II. Kinematics Reasoning

The *kinematics reasoning unit* represents the key component for our control architecture. Its functionality builds upon the *qualitative kinematic constraints* that we introduced in [1] for Mecanum wheels. These constraints provide the basis to deduce (on-line) the space of *admissible and controllable* motions B for a robot drive with specific geometric configuration and mode of operation/failure. The explicit knowledge about this space provides valuable information since we can always check, whether a motion set-point from higher-level control

$$\dot{\xi} = [\dot{x} \;\; \dot{y} \;\; \dot{\Theta}]^T \tag{1}$$

that describes the desired longitudinal velocities ($\dot{x}$ and $\dot{y}$) and the angular velocity ($\dot{\Theta}$) of the robot (within its local frame of reference) is feasible for the robot at its current configuration. Whenever the condition

$$\dot{\xi} \in B \tag{2}$$

holds, we can use the kinematic model to automatically deduce the *inverse kinematics* of the robot that provides the procedure to compute the set-points for the wheel's steering angles β_i and angular velocities $\omega_i = \dot{\varphi}_i$. We will use the explicit knowledge of the space of admissible and controllable motions B in two ways. Firstly, we provide this information to higher level control such as the path planner for the robot. Secondly, we utilise it to check, whether a drive command from the path planner is admissible for the drive in its current mode of operation/failure. A violation of the condition (2) can either stop the robot and inform the path planner about the failure to execute a desired drive command, or trigger a re-configuration to compute an alternative set-point $\dot{\xi}^*$ within B as we will show below.

A. Qualitative Kinematics Constraints

The key algorithm for this single robot kinematics reasoning is the deduction of B. The kinematics of robot drives is typically described in terms of the *rolling* and *sliding constraints* that one obtains from the geometric alignment of the wheels and their type (steered and un-steered standard wheels, mecanum wheels, castor wheels, etc.) in the drive [8], [2]. The usual rolling (3) and sliding (4) constraints for standard wheels relate $\dot{\xi}$ and the angular wheel velocities $\dot{\phi} = [\dot{\varphi}_1, \ldots, \dot{\varphi}_n]$ through the drive's geometry ($\mathbf{J}_1, \mathbf{C}_1$) and the radius R_i of the wheels $\mathbf{J}_2 = \text{diag}[R_1, \ldots, R_n]$

$$\mathbf{J}_1(\beta_s)\dot{\xi} = \mathbf{J}_2\dot{\phi} \tag{3}$$

$$\mathbf{C}_1(\beta_s)\dot{\xi} = \mathbf{0}. \tag{4}$$

However, the matrices $\mathbf{J}_1$ and $\mathbf{C}_1$ depend upon the (time-varying) steering angles $\beta_s = [\beta_1, \ldots, \beta_n]^T$. To remove this dependancy for analysis purposes, we formulate *qualitative rolling and sliding constraints* with matrices $\mathbf{J}_q$ and $\mathbf{C}_q$ that capture the matrices for *all* possible steering angles in compact form.

For example, consider the standard wheel at geometric position $\langle l, \alpha \rangle$ as shown in Fig. 3. The wheel defines the

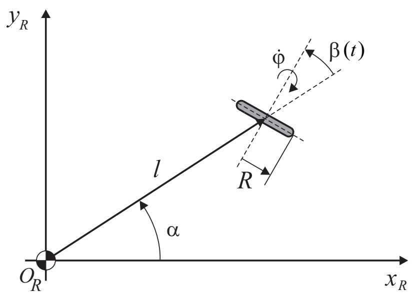

Fig. 3. Geometric parameters for a standard steered wheel

following row

$$\mathbf{j}_1(\beta) = \left[\; \sin(\alpha+\beta) \quad -\cos(\alpha+\beta) \quad -l\,\cos(\beta) \;\right] \tag{5}$$

in the matrix $\mathbf{J}_1(\beta_s)$ of the rolling constraint. In terms of drive-space analysis we are interested in the sub-space of $\mathbb{R}^3$ that the row-vector $\mathbf{j}_1$ of (5) defines for the parametrised steering angle with range $-\pi < \beta \leq \pi$. This leads to what we call the *qualitative rolling constraint* $\mathbf{j}_q$ that encodes the subspace in terms of *two linear independent rows* for $\mathbf{J}_q$. The two perpendicular angles $\beta = 0$ and $\beta = \frac{\pi}{2}$ ensures this property, so that we define:

$$\mathbf{j}_q := \left[\begin{array}{ccc} \sin(\alpha) & -\cos(\alpha) & -l\ \cos(0) \\ \sin(\alpha + \frac{\pi}{2}) & -\cos(\alpha + \frac{\pi}{2}) & -l\ \cos(\frac{\pi}{2}) \end{array} \right]. \quad (6)$$

Both rows together define the sub-space of controllable velocities for the actuated wheel. A wheel with blocked steering simply acts as un-steered wheel so that $\mathbf{j}_q$ simply becomes $\mathbf{j}_q = \mathbf{j}_1(\beta)$. The situation for a drive with impaired rotational actuation (e.g. a freely spinning wheel or a wheel with blocked rotation) requires a definition for $\mathbf{j}_q$ that expresses the failure to control the velocity through the wheel's actuation. In terms of our qualitative constraints we express this fact through the null-vector

$$\mathbf{j}_q = \left[\begin{array}{ccc} 0 & 0 & 0 \end{array} \right]. \quad (7)$$

We apply an analogue principle for the sliding constraint $\mathbf{c}_q$ that a single wheel contributes for $\mathbf{C}_q$.[1] A wheel with blocked steering contributes the standard sliding constraint

$$\mathbf{c}_q = \mathbf{c}_1(\beta) = \left[\begin{array}{ccc} \cos(\alpha + \beta) & \sin(\alpha + \beta) & l\ \sin(\beta) \end{array} \right] \quad (8)$$

for a wheel at the fixed steering angle β. In operational/fault modes with operating steering we can always direct the wheel so that it does not impose a constraint on the movement. As a consequence we use the null-vector

$$\mathbf{c}_q = \left[\begin{array}{ccc} 0 & 0 & 0 \end{array} \right] \quad (9)$$

for the qualitative sliding constraint. A fully blocked wheel (steering and rotational actuation) does not allow any longitudinal movement. We can express this fact through the two sliding constraints for $\mathbf{C}_q$ with perpendicular angles $\beta = 0$ and $\beta = \pi/2$

$$\mathbf{c}_q = \left[\begin{array}{ccc} \cos(\alpha) & \sin(\alpha) & l\ \sin(0) \\ \cos(\alpha + \frac{\pi}{2}) & \sin(\alpha + \frac{\pi}{2}) & l\ \sin(\frac{\pi}{2}) \end{array} \right]. \quad (10)$$

Table I summarise the qualitative constraints $\mathbf{j}_q$ and $\mathbf{c}_q$ for standard wheels in various mode of operation/failure.

B. Drive-Space Computation

We now use the matrices $\mathbf{J}_q$ and $\mathbf{C}_q$ to compute the space of admissible and controllable motions B for a robot drive as follows:

In the first step we compute the *space of the admissible motions* Z for a specific mode in the usual way through the *null-space* or *kernel*:

$$\text{kernel}(\mathbf{C}_1(\beta_s)) \subseteq \mathbb{R}^3 \,. \quad (11)$$

[1]The matrix $\mathbf{C}_q$ generalises the matrix $\mathbf{C}_{1,f}$ that specifies sliding constraints for fixed wheels of the robot only.

To eliminate the dependency on the steering angles, we use our qualitative form of $\mathbf{C}_1$ instead and obtain

$$Z = \text{kernel}(\mathbf{C}_q) \subseteq \mathbb{R}^3 \,. \quad (12)$$

However, one has to ensure, that an admissible velocity $\dot{\xi} \in Z$ can be actuated (controlled) through the actuated robot wheels. Re-writing (3) we obtain

$$\mathbf{J}_2^{-1}\mathbf{J}_1(\beta_s)\dot{\xi} = \dot{\phi} \quad (13)$$

It becomes evident that we can compute the *non-controllable* velocities $\bar{S}$ by means of

$$\text{kernel}(\mathbf{J}_1(\beta_s)) \subseteq \mathbb{R}^3 \,. \quad (14)$$

Our qualitative form of $\mathbf{J}_1$, i.e. the matrix $\mathbf{J}_q$ encodes the (again, steering angle independent) rolling constraint information of the robot wheels with rotational actuation so that we obtain the *non-controllable* velocities $\bar{S}$ through

$$\bar{S} = \text{kernel}(\mathbf{J}_q) \subseteq \mathbb{R}^3 \,. \quad (15)$$

Whenever the two spaces do intersect, i.e. $Z \cap \bar{S} \neq \emptyset$, we have to refine the admissible velocities Z to exclude those movements that cannot be actuated through the robot's wheels. By computing the complement of $\bar{S}$

$$S = \text{kernel}(\bar{\mathbf{S}}^T) \subseteq \mathbb{R}^3 \,, \quad (16)$$

where $\bar{\mathbf{S}}$ denotes the *matrix of basis vectors* for $\bar{S}$, we obtain the *controllable* velocities so that, finally, the intersection

$$Z \cap S =: B \quad (17)$$

defines the space of *admissible and controllable* velocities for a given mode of operation or failure of the robot.

Of course, the line of argument more or less summarises text-book knowledge from robot kinematics (e.g. [8]). However, the introduction of our qualitative sliding constraints and, in particular, rolling constraints enables us to formulate the line of argument in an efficient algorithmic form that allows on-line drive-space and control/motorization analysis. The computations rely on several null-space computations (e.g. via singular-value decomposition) and the vector-space intersection (17) (e.g. with the Zassenhaus algorithm [10]). All operations together require less than 1 ms on our robot's National Instruments cRIO real-time CPU. As a consequence, we can utilise the reasoning concept directly within the drive's control loop and thus reactively adapt the drive's control mechanism to the kinematics for the onset of operational mode and faults in the drive.

C. Drive command reconfiguration

As noted above, we use the information about the admissible and controllable drive space B to check the compatibility of a drive command $\dot{\xi}$ with a drive's mobility capabilities. Whenever the Kinematics Reasoning Unit identifies an incompatible drive command through $\dot{\xi} \notin B$ it performs two operations. Firstly, it communicates the incompatibility to higher order control / path planning to initiate re-planning. Secondly, it either stops the drive until it receives a new

TABLE I

QUALITATIVE ROLLING AND SLIDING CONSTRAINTS FOR A STANDARD WHEEL

Mode of Operation	$\mathbf{j_{iq}}$	$\mathbf{c_{iq}}$
OK: actuated rotation and operational steering	$\begin{bmatrix} \sin(\alpha) & \sin(\alpha+\frac{\pi}{2}) \\ -\cos(\alpha) & -\cos(\alpha+\frac{\pi}{2}) \\ -l\cdot\cos 0 & -l\cdot\cos\frac{\pi}{2} \end{bmatrix}^T$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^T$
Fault 1: actuated rotation and blocked steering	$\begin{bmatrix} \sin(\alpha+\beta) \\ -\cos(\alpha+\beta) \\ -l\cdot\cos\beta \end{bmatrix}^T$	$\begin{bmatrix} \cos(\alpha+\beta) \\ \sin(\alpha+\beta) \\ l\cdot\sin\beta \end{bmatrix}^T$
Fault 2: freely spinning wheel and blocked steering	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^T$	$\begin{bmatrix} \cos(\alpha+\beta) \\ \sin(\alpha+\beta) \\ l\cdot\sin\beta \end{bmatrix}^T$
Fault 3: blocked rotation and operational steering	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^T$	$\begin{bmatrix} \cos(\alpha) & \cos(\alpha+\frac{\pi}{2}) \\ \sin(\alpha) & \sin(\alpha+\frac{\pi}{2}) \\ l\cdot\sin 0 & l\cdot\sin\frac{\pi}{2} \end{bmatrix}^T$
Fault 4: freely spinning wheel and operational steering	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^T$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^T$

compatible drive command from the path planner or adapts the drive-command to a compatible set-point $\dot{\xi}^* \in B$. For example, one might want to make sure that the robot remains on track (i.e. maintain the longitudinal velocities $\dot{x}, \dot{y}$ of $\dot{\xi}$) and determine the robot's angular velocity according to the mobility constraint ($\dot{\Theta} \rightarrow \dot{\Theta}^*$).

Let us illustrate the drive-space computations and reconfiguration procedure with an omni-directional robot in a geometric configuration with three actuated and steered wheels as in Fig. 4 (and Fig. 2d). The regular 6-edge comb geometry of our modular robot leads to a drive with wheel parameters $\alpha_1 = 0, \alpha_2 = \frac{4\pi}{6}, \alpha_3 = \frac{8\pi}{6}$ and $l_1 = l_2 = l_3 = 0.26$. Fig. 4 shows two fault scenarios for such a robot that we now use to demonstrate our automated kinematic reasoning and the reconfiguration procedure. In the following we represent the vector spaces $B, Z, \bar{S}$ and S through 3×3 matrices $\mathbf{B}, \mathbf{Z}, \bar{\mathbf{S}}$ and $\mathbf{S}$, respectively, that contain the basis-vectors of the individual spaces in their columns.

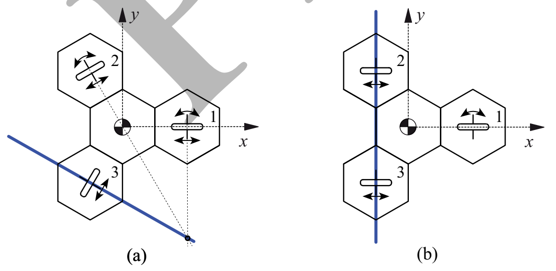

Fig. 4. 3-wheel COMB robot with (a) a single steering-fault in wheel 3 and (b) a triple fault scenario with steering-faults in wheel 2 and 3 and an actuation fault in wheel 1 (freely spinning wheel) leading to a tricycle configuration of the robot

Fig. 4a shows a scenario with two operational wheels (wheel 1 & 2) and a wheel with blocked steering (wheel 3) at $\beta_3 = \frac{-3\pi}{6}$. The angular speed actuation is operational for all three wheels. This mode implies qualitative constraint matrices[2]

$$\mathbf{J}_q = \left[\begin{array}{ccc} 0 & -1 & 0.26 \\ 1 & 0 & 0 \\ \hline 0.866 & 0.5 & -0.26 \\ -0.5 & 0.866 & 0 \\ \hline 0.5 & 0.866 & 0 \end{array}\right],$$

$$\mathbf{C}_q = \left[\begin{array}{ccc} 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline -0.86 & 0.5 & 0.26 \end{array}\right].$$

Because all wheels have operational rotational actuation we obtain a matrix $\mathbf{J}_q$ with full rank. This implies $\bar{\mathbf{S}} = \mathbf{0}$ and $\mathbf{S} = \mathbf{I}$ which indicates full control (motorization) in terms of the drive's actuators. The admissible and controllable space B or its matrix of basis vectors $\mathbf{B}$ is thus simply defined through the kernel of $\mathbf{C}_q$ as

$$\mathbf{B} = \mathbf{Z} = \begin{bmatrix} 0 & -0.545 & 0 \\ 0.461 & -0.744 & 0 \\ 0.887 & 0.387 & 0 \end{bmatrix}.$$

This defines a 2-dimensional subspace for $\dot{\xi}$ that specifies velocities with ICRs on the axis of the impaired wheel. A feasible drive command adaption $\dot{\Theta} \rightarrow \dot{\Theta}^*$ that maintains longitudinal velocities obtains the modified angular velocity $\dot{\Theta}^*$ through projecting $\dot{\xi}$ onto the plane as shown in Fig. 5.

Fig. 4b shows a triple-fault scenario with impaired steering in wheel 2 and 3 and no actuation in the steered wheel 1 (freely spinning wheel). This configuration defines the

[2]The horizontal lines separate the sections for the individual wheels.

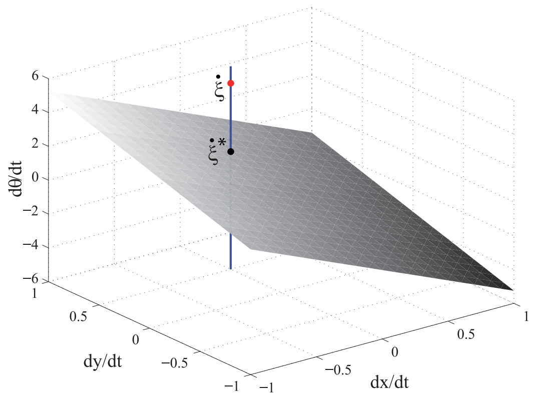

Fig. 5. Drive command adaption $\dot{\xi} \rightarrow \dot{\xi}^*$ to maintain the longitudinal velocities $\dot{x}, \dot{y}$

qualitative matrices and drive space matrices

$$\mathbf{J}_q = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -0.225 \\ -1 & 0 & -0.225 \end{bmatrix}, \mathbf{C}_q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -0.130 \\ 0 & -1 & 0.130 \end{bmatrix},$$

$$\bar{\mathbf{S}} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ 0.129 & 0 & 0 \\ 0.992 & 0 & 0 \end{bmatrix}.$$

It is easy to see that the space Z does not intersect with $\bar{S}$ $(\text{rank}([\bar{\mathbf{S}}, \mathbf{Z}]) = \text{rank}(\bar{\mathbf{S}}) + \text{rank}(\mathbf{Z}))$ so that we can conclude $B = Z$.

An additional actuation fault of wheel 2 (freely spinning wheel) leads to a different scenario with

$$\mathbf{J}_q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & -0.225 \end{bmatrix}, \mathbf{C}_q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -0.130 \\ 0 & -1 & 0.130 \end{bmatrix},$$

$$\bar{\mathbf{S}} = \begin{bmatrix} 0 & -0.220 & 0 \\ 1 & 0 & 0 \\ 0 & 0.976 & 0 \end{bmatrix}, \mathbf{S} = \begin{bmatrix} 0.976 & 0 & 0 \\ 0 & 0 & 0 \\ -0.220 & 0 & 0 \end{bmatrix},$$

$$\mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ 0.129 & 0 & 0 \\ 0.992 & 0 & 0 \end{bmatrix}.$$

Since Z intersects with $\bar{S}$ we have to compute B through $B = Z \cap S$ and obtain $B = \emptyset$! This result indicates that we cannot actuate the drive specifically enough so that it can perform movements within Z. Of course, one can argue that the robot is still capable of driving since the steerable front wheel provides enough (sideways) friction forces that ensures a specific movement. However, recall that $\mathbf{C}_q$ does not contain constraints for wheels with operational steering. As a consequence, our mechanism provides a conservative, but on the other hand safe[3] result for the drive. However, one could also easily extend the reasoning concept for the specific case $B = \emptyset, Z \neq \emptyset, S \neq \emptyset$ as follows: Apply Z to compute $\dot{\xi}^*$ and determine the appropriate steering angles for the steered wheels $\beta_s = [\beta_1, \ldots, \beta_n]^T$. These angles are then used to determine $\mathbf{C}_1(\beta_s)$. Whenever this matrix satisfies $\text{rank}(\mathbf{C}_1(\beta_s)) = 1$ we can conclude, that the drive sufficiently constrains the movement to enforce the desired movement through the sliding constraints. We can then actuate the desired movement whenever $\dot{\xi}^* \notin \bar{S}$.

[3]The configuration restricts movements to ones with an ICR on the axis of the wheels 2 and 3, except an ICR at the location of the only remaining wheel with actuator (wheel 3). An ICR that is slightly off the location of wheel 3, however, would impose high torque requirements onto the actuator that can easily overstress the drive.

D. Origin Shift and Multi-Robot Systems

Our procedure not only provides a sense of self awareness about the active kinematics for the robot drive but also explicit information for the path planner that should be used actively within the higher levels of control.

1) Origin Shift: Some drive geometries and control tasks have preferred positions for the robot's center. For example, for a robot with differential drive it is advantageous to locate the robot's center onto the common axis of the un-steered wheels ($O^{(2)}$ in Fig. 6a). Another example is shown in Fig. 6b. The robot control task of pushing a ball can be simplified through matching the robot's and the ball's center.

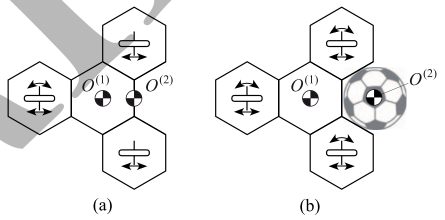

Fig. 6. Origin shift examples for a single robot

Of course, we do have full control over the geometric model of the robot drive. We could easily perform an origin shift through an adaption of the model. The on-line kinematics reasoning process will then take care of the adapted situation automatically. Another strategy is to perform a virtual origin shift through transformation. So the drive acts as a robot with (adaptive) center $O^{(2)}$ for higher-level control, whereas it uses a fixed center $O^{(1)}$ for kinematics reasoning and control. This functionality will also be enormously helpful for multi-robot control as we shall see later.

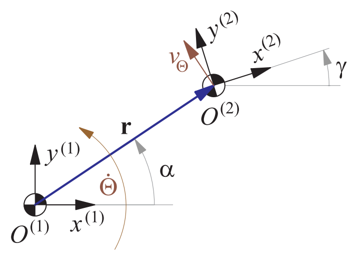

Fig. 7. Origin shift geometry

Mathematically, we have to perform the problem of velocity transformation [9] $\dot{\xi}^{(1)} \leftrightarrow \dot{\xi}^{(2)}$, where $\dot{\xi}^{(i)} = [\dot{x}_i, \dot{y}_i, \dot{\Theta}^{(i)}]^T$ represents a drive command with respect to a coordinate system with origin $O^{(i)}$. For example, $O^{(1)}$ represents the original robot origin and $O^{(2)}$ denotes the shifted (virtual) origin. We describe the transformation through a translation $\mathbf{r} = [x_r^{(1)}, y_r^{(1)}]^T$ and a rotation γ between the two coordinate systems for both origins as shown in Fig. 7 and obtain

$$\dot{\xi}^{(2)} = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[\dot{\xi}^{(1)} + \dot{\Theta}^{(1)} \begin{bmatrix} -y_r^{(1)} \\ x_r^{(1)} \\ 0 \end{bmatrix} \right]$$

Our kinematics analysis provides the mobility spaces with respect to the origin $O^{(1)}$, i.e. we write $B^{(1)}, Z^{(1)}$ and $\bar{S}^{(1)}$. The matrix $\mathbf{B}^{(1)}$ specifies the space $B^{(1)}$ through up to three basis vectors that define admissible and controllable velocities in $B^{(1)}$. As a consequence, we can easily compute the space $B^{(2)}$ for the shifted origin $O^{(2)}$ through applying the transformation column-wise and obtain $\mathbf{B}^{(2)}$. The transformations $Z^{(1)} \rightarrow Z^{(2)}$ and $\bar{S}^{(1)} \rightarrow \bar{S}^{(2)}$ are handled analogously so that we can provide a path planner with the mobility information with respect to the virtual origin $O^{(2)}$. Velocity set-points $\dot{\xi}^{(2)}$ from higher level control/path planner are then transformed to the drive's origin $O^{(1)}$ through the inverse transformation.

2) Multi-Robot Systems: We are particularly interested in controlling multi-robot systems where robots join to (temporarily) form *meta-robots* with diverse drive geometries. Fig. 8 shows such a robot that combines a 4-wheel Ackerman steering type robot with a 3-wheel omnidirectional robot.

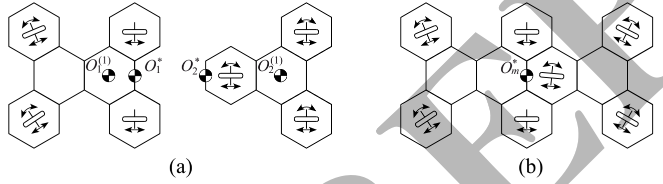

Fig. 8. Origin shift for a multi-robot control

Our control architecture (Fig. 1) allows us to dynamically add/remove wheels so that we could simply use the kinematics analysis outlined above for centralised control of the meta-robot. Another strategy is to merge the drive-space information of the individual robots into the drive-space information of the meta robot and use distributed/hierarchical control for the meta robot.

In a first step, we shift the robots origins $O_1^{(1)}$ and $O_2^{(1)}$ onto the desired contact points O_1^* and O_2^* (see Fig. 8a) and compute

$$Z_i^{(1)} \rightarrow Z_i^*, \quad \bar{S}_i^{(1)} \rightarrow \bar{S}_i^*, \quad i = 1, 2 \tag{18}$$

and merge the robots through control with respect to the shifted origins. An interconnection of the robot implies that the shifted origins O_1^* and O_2^* match and define the origin O_m^* of the meta-robot. The drive-space information (Z and $\bar{S}$) of the meta robot is then computed through intersection

$$Z_m^* = Z_1^* \cap Z_2^*, \quad \bar{S}_m^* = \bar{S}_1^* \cap \bar{S}_2^*. \tag{19}$$

Analogously as in the single robot case we have to refine the admissible space Z_m^* whenever it contains non-controllable velocities, i.e. $Z_m^* \in \bar{S}_m^* \neq \emptyset$, through $\bar{S}_m^* \rightarrow S_m^*$ and $B^* = Z_m^* \cap S_m^*$. We can apply this procedure recursively to handle multi-robot configurations with more than two robots.

III. Conclusion

We presented a robot control architecture that builds upon an on-line kinematics-reasoning capability in order to cope with faults in the system and virtual and physically changing robot-drive geometries. Many detailed analyses for robot kinematics can be found in literature and textbooks, for example [2], [6], [4], [8]. These papers provide profound and systematic techniques and methodologies for the off-line kinematics analysis of robot drives. Our formulation that builds upon generalised sliding and rolling constraints, however, allows us to perform kinematics-reasoning during run-time of the robot. This capability enables us to formulate a fault-tolerant, robust and adaptable control scheme for the robot drive. Furthermore, it opens interesting perspectives for intelligent drive-aware path planning and re-configurable robot drives and, in particular, coordinated control of heterogeneous multi-robot systems. Being able to analyse/drive robots with almost arbitrary wheel geometry generalises to multi-robot systems where the formation of heterogeneous robots forms a particular meta-robot with specific drive geometry. Our reasoning technique can directly analyse the kinematics of this meta robot in a centralised control setting but also supports hierarchical and distributed control schemes through the multi-robot kinematics reasoning scheme outlined above.

References

[1] M. Brandstötter, M. Hofbaur, G. Steinbauer, and F. Wotawa. Model-based fault diagnosis and reconfiguration of robot drives. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS07)*, pages 1203–1209, 2007.

[2] 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*, 12:47–62, 1996.

[3] M. W. Hofbaur and B. C. Williams. Hybrid estimation of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 34(5):2178–2191, October 2004.

[4] W. Kim, S. Lee, and B. Yi. Mobility analysis of planar mobile robots. In *Proceedings of the IEEE Internat. Conf. on Robotics and Automation (ICRA02)*, pages 2861–2867, 2002.

[5] E. Nüchter, E. Bahr, and Lohmüller H. *Omnidirektionales Fahrzeug, Fahrmodul und mobiler Industrieroboter*, German Patent Office, Offenlegungsschrift DE 10 2007 016 662 A1, 2008.

[6] R. Rajagopalan. A generic kinematic formulation for wheeled mobile robots. *Journal of Robotic Systems*, 14:77–91, 1997.

[7] R. Robuffo Giordano, M. Fuchs, A. Albu-Schäffer, and G. Hirzinger. On the kinematic modeling and control of a mobile platform equipped with steering wheels and movable legs. In *Proceedings of the IEEE Internat. Conf. on Robotics and Automation (ICRA09)*, pages 4080–4087, 2009.

[8] R. Siegwart and I. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.

[9] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2006.

[10] H. Zassenhaus. Über einen Algorithmus zur Bestimmung der Raumgruppen. *Commentarii Mathematici Helvetici*, 21:117–141, 1948.