Model-Based Fault Diagnosis and Reconfiguration of Robot Drives

Mathias Brandstötter Michael W. Hofbaur Gerald Steinbauer Franz Wotawa

Abstract— **Modern drives of mobile robots are complex machines. Because of this complexity, as well as of wear and aging of components, faults occurs in such systems quite frequently at runtime. In order to use such drives in truly autonomous robots it is desirable that the robot is able to automatically react to such faults. Therefore, the robot needs reasoning and reconfiguration capabilities in order to be able to detect, localize and repair such faults on-line. In this paper we propose a model-based diagnosis and reconfiguration framework which allows an autonomous robot to detect and compensate faults in its drive. Moreover, we present an implementation for a real robot platform. Finally, we report experimental results which shows that the proposed framework is able to correctly cope with injected faults in the drive hardware, like broken motors.**

I. INTRODUCTION

Modern drives of mobile robots are complex mechatronic devices and comprise a number of software and hardware components with different properties. The types of such drives range from simple differential drives to advanced mechanisms for outdoor and rescue applications. Because of the complexity and the close interaction between the components of a drive there is always the possibility that a fault occurs in the system at runtime. Such faults even occur if the components of the drive are well designed, implemented and tested. In [1] a quantitative and qualitative evaluation of these phenomena was presented for unmanned ground vehicles (UGV). The study clearly shows that even on commercial or military robot platforms such faults do occur during operation.

The key feature of an autonomous mobile robot is that the robot is able to perform a given task with limited or no human intervention. Therefore, the robot has to automatically react to any situation it is faced with in an intelligent and robust way. These may include situations that the developer or the robot never encountered before or the occurrence of a fault in the robot hardware and software. Such situations become even worse if one thinks about robots which work truly autonomously for a period of time with no or limited possibility for human intervention. Examples for such robots are planetary rovers on Mars. The developers of the NASA Mars rover, for example, report in [2] a situation where one motor of a rover started to fail. By a permanent and systematic examination of the parameters of the motor, e.g., drawn current, the operator was able to detect this circumstance and to reconfigure the rover's movement in order to extend the lifetime of the motor. This example clearly shows that the automatic detection, localization and repair of a fault is desirable, if not crucial, for truly autonomous robots.

In order to equip an autonomous mobile robot with appropriate reasoning and repair capabilities we propose to enrich the control framework of the robot by a fault diagnosis engine and a reconfiguration engine, as depicted in Fig. 1, that distinguishes three different fault scenarios. In the first scenario the fault or damage is not too serious. Once the framework has detected and localized the fault the framework adapts its behavior to the new situation and retains the full functionality of the robot. Such a reconfiguration of a drive can be transparently done without informing the higher level control system (path planner). Obviously, the robot has to have some level of redundancy to facilitate this capability. In the second situation the fault or damage cannot be fully compensated by the reconfiguration engine. In this situation the reconfiguration engine performs a controlled degradation of the functionality. For example, an omni-directional drive degrades to a differential drive. The reconfiguration provides a reduced but defined and known functionality. In this case the higher level control system is informed about the reduced functionality in order to adapt its behavior accordingly. Finally, a fault can be too serious so that the robot loses its capability to carry out a desired task. However, a fault detection can initiate appropriate actions to switch the robot to a save state or to inform a human operator.

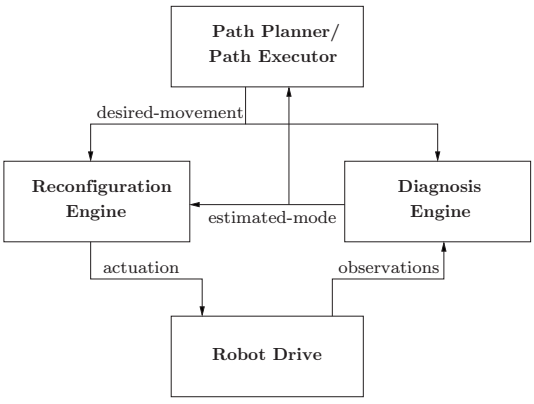

Fig. 1. The information flow in the proposed diagnosis and reconfiguration framework.

In [3] we formally described such a general framework which allows an autonomous mobile robot to intelligently cope with faults in its drive. This paper demonstrates this

Authors are listed in alphabetic order. This research has been funded in part by the Austrian Science Fund (FWF) under grant P17963-N04.

M. Brandstötter and M. Hofbaur are with the Institute for Automation and Control, Graz University of Technology, Kopernikusgasse 24/2, A-8010 Graz, Austria. michael.hofbaur@TUGraz.at

G. Steinbauer and Franz Wotawa are with the Institute for Software Technology, Graz University of Technology, Inffeldgasse 16b, A-8010 Graz, Austria. {steinbauer,wotawa}@ist.tugraz.at

framework for a specific omni-directional robot drive that we use in our RoboCup robots and shows experimental results for its operation.

The remainder of this paper is organized as follows. In the next section we describe the robot platform that we use as an example throughout the rest of the paper. In Section III we introduce the model-based diagnosis methods that we use for the fault detection and localization. The following section details the reconfiguration strategy of our model-based motion controller. Section V reports experimental results for the proposed framework. Finally, in the last section we will draw some conclusions and present future work.

II. EXPAMPLE ROBOT PLATFORM

Fig. 2 shows the robot platform and the omni-directional drive of our RoboCup soccer team [4]. We use the robot and the drive as example throughout the paper. The robot comprises an omni-directional drive, a pneumatic kicker, a central PC running Linux and an omni-directional camera system. All major components are connected by a CAN-bus.

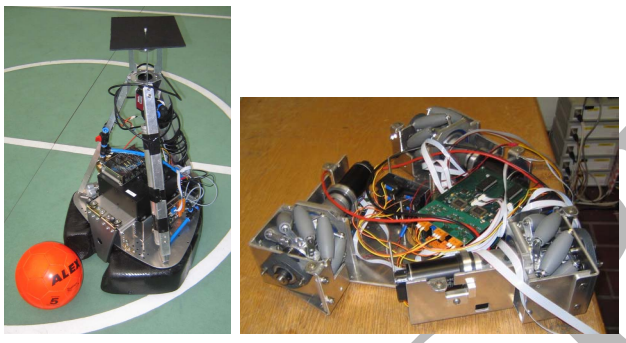

Fig. 2. An autonomous robot of our RoboCup soccer team on the left. On the right the omni-directional drive of the robot is shown.

The omni-directional drive of the robot comprises three Mecanum wheels (Swedish 45-degree wheels). Fig. 3 depicts the drive with its three Mecanum wheels, the associated brush-less DC-motors (M_1, M_2, M_3) and its geometric parameters.

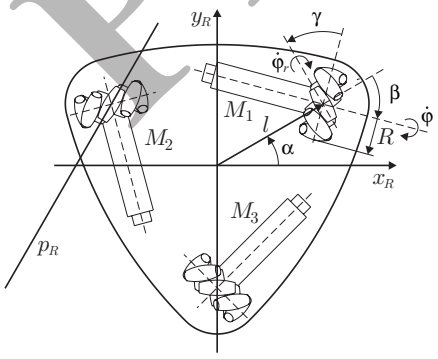

Fig. 3. The arrangement of the omni-directional drive.

The wheel arrangement of the robot defines its kinematics. Each wheel imposes a so-called rolling and sliding constraint. The constraints of the wheels can be combined to describe the kinematics of the robot. A motion command from the path planner / path executer is given in the form $\dot{\xi}_R = [\dot{x}_R \;\; \dot{y}_R \;\; \dot{\Theta}_R]^T$, where $\dot{x}_R$ and $\dot{y}_R$ denote the velocity of the robot along its axes. $\dot{\Theta}_R$ denotes the angular velocity around the vertical axis of the robot. According to [5] the rolling constraint of one Swedish wheel can be written as

$$[\sin(\alpha+\beta+\gamma) \;\; -\cos(\alpha+\beta+\gamma) \;\; -l\cdot\cos(\beta+\gamma)]\cdot\dot{\xi}_R - R\dot{\varphi}\cdot\cos\gamma = 0\,. \quad (1)$$

The motion perpendicular to this direction is not constrained because of the free rotation $\dot{\varphi}_r$ of the wheel's rollers. Therefore, the sliding constraint is

$$[\cos(\alpha+\beta+\gamma) \;\; \sin(\alpha+\beta+\gamma) \;\; l\cdot\sin(\beta+\gamma)]\cdot\dot{\xi}_R - R\dot{\varphi}\cdot\sin\gamma - r\,\dot{\varphi}_r = 0\,. \quad (2)$$

The combination of the constraints for all wheels of the robot provides the kinematic constraints in vector-form

$$\mathbf{J}_1\dot{\xi}_R - \mathbf{J}_2\dot{\varphi} = 0 \quad (3)$$

$$\mathbf{C}_1\dot{\xi}_R - \mathbf{K}(\dot{\varphi},\dot{\varphi}_r) = 0 \quad (4)$$

The arrangement of the wheels and the motors form the basis for the model-based fault diagnosis. The rolling and sliding constrains are later used for the model-based reconfiguration.

III. MODEL-BASED DIAGNOSIS

The diagnosis engine for our robot drive tracks its operation in terms of estimating continuous variables such as the rotational speed of the wheels and its mode of operation/failure. The basis for this process is a *hybrid model* of the robot drive that captures the drive's continuous behavior that is interwoven with discrete mode changes, such as abrupt faults or commanded mode changes. Monitoring and diagnosis is then framed as *hybrid estimation* that builds upon a hybrid automaton model for the system. We apply our component based probabilistic framework [6], [7] that models the system in terms of a concurrent composition of *probabilistic hybrid automata (PHA)* component models and provides a sub-optimal, but computationally efficient algorithm for hybrid estimation that combines numerically accurate filtering methods [8] with techniques from the toolkit of model-based reasoning/diagnosis [9].

The basic building block of our modeling framework is a probabilistic hybrid automaton (PHA) that is used to model the individual components of the robot, for example, electric motors, gears, current amplifier, or entire subsystems such as a complete drive unit that comprises the wheel, the actuating motor and its gear-box, the associated power electronics, optical wheel encoders, and the low-level rotational speed control unit. The PHA is expressed as a tuple

$$\mathcal{A} := \langle \mathbf{x}, \mathbf{w}, \mathbf{y}, F, T, N \rangle \quad (5)$$

where $\mathbf{x}$ denotes the *state* of the automaton that is composed of the discrete state or *mode* $\mathbf{x}_d$ and the *continuous state* $\mathbf{x}_c$.

$\mathbf{w}$ captures the discrete ($\mathbf{w}_d$) and continuous ($\mathbf{w}_c$) *input-/output-variables*. These I/O variables are used to interconnect the automaton with other automata and to the outside world. The *continuously valued evolution* of the automaton is defined through F which specifies *sets of differential or difference equations* and/or *algebraic equations/constraints* that hold for a particular mode of operation or failure[1]. The automaton entity T defines the *discretely valued evolution* of the automaton through specifying *probabilistic mode-transitions* that capture commanded or autonomous abrupt changes of the modelled system. Since we model real world systems, we also include *disturbances* or *noise* that act on the continuous state and the measurements. The automaton entity N specifies these disturbances for every mode of the automaton.

The overall model is then built through the *concurrent composition* of its component automata $\mathcal{A} = \mathcal{A}_1 \parallel \mathcal{A}_2 \parallel \cdots \parallel \mathcal{A}_\zeta$ together with the specification of the interconnection to the outside world. This leads to what we call the *concurrent probabilistic hybrid automaton (cPHA - $\mathcal{CA}$)*. Fig. 4 shows a cPHA composition for the model of our example robot drive. The drive of our RoboCup soccer robot consists of 3 actuated Mecanum wheels arranged in a 120 degree configuration (Fig. 2). As a consequence, our model consists of 3 PHA components for the drive units ($\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$) that specify the hybrid dynamics of each drive unit (Mecanum wheel, DC-motor and sensors and the low-level rotational-speed control unit) and a PHA component ($\mathcal{A}_4$) for the robot frame that specifies the kinematics of the drive as algebraic (rolling/sliding) constraints. Modeling the frame as independent component facilitates the possibility to model *reconfigurable robot drives*, where one can adapt the operational mode of the wheels as well as the *wheel arrangement*, e.g. through retraction of redundant wheels or the variation of the wheel geometry.

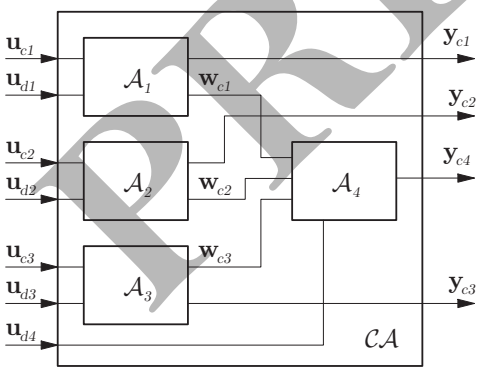

Fig. 4. Concurrent hybrid automata model of the drive.

The PHA model for each drive unit captures 1+3 modes. The nominal operational mode (m_1), a failure mode m_2 where the wheel becomes stuck ($\omega_i = 0$), a failure mode m_3 where the wheel rotates freely and a generic unknown mode m_0 that covers all other non-modelled situations. Fig. 5 shows the probabilistic transitions among the modes of a drive unit. Note that our model includes ambiguity in the transition outcome. This allows us to model the fact that the drive unit remains with high likelihood in the nominal mode m_1 but there is always a possibility for failure so that the transition spreads among the other three failure modes m_2, m_3, m_0. Fig. 5 details the transition from mode m_1 (τ_1) with likelihoods $P_0, \ldots, P_4$, respectively.

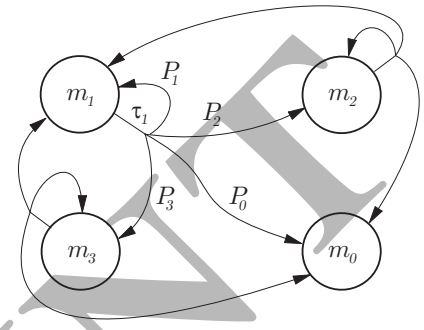

Fig. 5. PHA transitions for the drive automaton.

In terms of the continuous behavior, we model a drive unit through the actual rotational speed ω_i as the continuous state $\mathbf{x}_{c\,i,k} = \omega_i(k)$. Furthermore, the state acts as I/O variable $\mathbf{w}_{c\,i,k} = \mathbf{x}_{c\,i,k}$ to the PHA that models the robot frame ($\mathcal{A}_4$). Note that we do not predefine the causal direction of $\mathbf{w}_{c\,i}$ since the variable can act as *output* in modes m_1 and m_2 where the drive unit's actor imposes a desired rotational speed or blocks the wheel. However, in mode m_3, where the wheel can rotate freely, $\mathbf{w}_{c\,i,k}$ acts as an *input* to the drive-unit's PHA since the movement of robot imposes a rotation of the wheel according to the drive's kinematics. The cPHA specification for the overall drive then specifies the interconnection of the concurrent PHA composition to the outside world.

The continuously-valued inputs that act upon the system $\mathbf{u}_{c\,i,k}$ with $i = 1, \ldots, 3$ specify the desired rotational speed of the wheel of the drive unit i at time-step k. The discretely-valued input $\mathbf{u}_{d\,i,k}$ with $i = 1, \ldots, 4$ can be used to enforce mode-changes or to signal mode changes in the control system, e.g. a re-configuration of the drive. The outputs of the cPHA $\mathbf{y}_{c\,i,k}$ specify the observed system variables that can be used for estimation and control. In our case, we directly measure the rotational speed of the wheels

$$\mathbf{y}_{c\,i,k} = k\,\mathbf{x}_{c\,i,k} + \mathbf{v}_{c\,i,k}, i = 1, \ldots, 3 \tag{6}$$

through encoders and model these measurements with additional *measurement noise* $\mathbf{v}_{c\,i,k}$. The scaling factor k summarizes the gear transmission and the encoder resolution.

The odometry measurements serve as the main source of information for the diagnosis results reported in this paper. However, we are currently fitting our robots with additional gyroscopes and merge these measurements with data obtained from our robot localization so that we will also provide the monitoring and diagnosis unit with $\mathbf{y}_{c\,4,k}$

[1] Since a hybrid automaton allows us to define multiple modes of operation we can formalize not only one nominal mode, but several ones and also particular failure modes.

that captures the longitudinal $(\dot{x}_R, \dot{y}_R)$ and rotational speed $(\dot{\Theta}_R)$ of the robot. Furthermore, we evaluate whether the diagnosis results can be improved even further through using DC motor current and voltage measurements.

A drive unit automaton $\mathcal{A}_i$ with $i = 1, \ldots, 3$ models the DC motor with gear-box, power electronics and the associated low-level speed control unit. As a consequence, we can describe the continuous dynamics for the drive unit i in the nominal mode m_1 for diagnosis purposes approximately through a 1^{st} order system with the difference equation

$$\mathbf{x}_{c\,i,k+1} = a_i \mathbf{x}_{c\,i,k} + b_i \mathbf{u}_{c\,i,k}. \tag{7}$$

The blocking mode m_2, for example, enforces no angular velocity regardless of the control input $\mathbf{u}_{c\,i,k}$. Thus, the difference equation at mode m_2 becomes

$$\mathbf{x}_{c\,i,k+1} = 0. \tag{8}$$

The mode m_3 where the wheel rotates freely imposes no difference equation but an algebraic constraint only. The constraint relates the measured rotational speed that is obtained through odometry (the sensor in $\mathcal{A}_i$) with the deduced speed that $\mathcal{A}_4$ provides on the basis of its kinematic constraints and the gyroscopes (the sensors in $\mathcal{A}_4$), more specifically

$$\mathbf{y}_{c\,i,k} + \mathbf{v}_{c\,i,k} = \mathbf{w}_{c\,i,k}\,. \tag{9}$$

The unknown mode m_0 that covers unmodeled situations, such as a fault in the drive's odometry, simply specifies *no equations* at all for the particular drive unit.

IV. RECONFIGURATION AND REPAIR

The kinematics of a wheeled robot drive can be described by rolling and sliding constraints [5]. Based on these constrains several properties regarding the mobility of the robot drive can be derived. In order to give the model-based controller the capability to decide if a transparent reconfiguration, a degradation or a safety stop is necessary we use a relaxed version of the constraints that we call *qualitative constraints* [10]. In this way we remove controllable (but still undetermined) parameters such as angles β_i for steered wheels or uncontrollable parameters such as the free rotation $\dot{\varphi}_r$ of the wheels' rollers from the sliding and rolling constraints. Table I depicts the qualitative constrains of an omni-directional wheel. In order to express that the sliding motion is not constrained in the nominal mode (the wheel's small rollers can rotate freely) we set the vector c_{iq} to zero. This means there are no (sliding) constraints for wheel i. In the mode *stuck* an omni-directional wheel acts through its rollers as a non-driven standard wheel but blocks the rotation of the wheel ($\mathbf{j}_{iq} = \mathbf{0}$).

Now the rolling and sliding constraints can be combined to the matrices $\mathbf{J}_{1q}$ and $\mathbf{C}_{1q}$ by stacking the constraint-vectors $\mathbf{j}_{iq}$ and $\mathbf{c}_{iq}$ for each wheel according to its estimated mode of operation/failure. In order to determine the kinematic capabilities of a drive we have to calculate the space of *admissible and controllable* velocities. In our previous work [10], [3] we introduced an algorithmic approach to solve this task on-line. The underlying theory can be summarized as follows:

Mode	$\mathbf{j}_{iq}$	$\mathbf{c}_{iq}$
nominal	$\begin{bmatrix} \sin(\alpha+\beta+\gamma) \\ -\cos(\alpha+\beta+\gamma) \\ -l\cdot\cos(\beta+\gamma) \end{bmatrix}^T$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^T$
stuck	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^T$	$\begin{bmatrix} \cos(\alpha+\beta+\gamma+\frac{\pi}{2}) \\ \sin(\alpha+\beta+\gamma+\frac{\pi}{2}) \\ l\cdot\sin(\beta+\gamma+\frac{\pi}{2}) \end{bmatrix}^T$

TABLE I
QUALITATIVE CONSTRAINS FOR AN OMNI-DIRECTIONAL WHEEL

One can compute the space of the *admissible* motions $\mathcal{Z}$ for the *null-space* or *kernel*:

$$\mathcal{Z} = kernel(\mathbf{C}_{1q})\,. \tag{10}$$

Furthermore, the space of the *non-controllable* velocities $\bar{\mathcal{S}}$ can be computed by means of

$$\bar{\mathcal{S}} = kernel(\mathbf{J}_{1q})\,. \tag{11}$$

Whenever the two spaces do intersect, i.e. $\mathcal{Z} \cap \bar{\mathcal{S}} \neq 0$, we have to refine the admissible velocities $\mathcal{Z}$ to exclude those movements that cannot be performed by the robot's wheels. By computing the complement of $\bar{\mathcal{S}}$

$$\mathcal{S} = kernel(\bar{\mathbf{S}}^T)\,, \tag{12}$$

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

$$\dot{\Xi} := \mathcal{Z} \cap \mathcal{S} \tag{13}$$

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

The $rank(\dot{\Xi})$ determines to the *degree of freedom* (DOF) of maneuverability of the robot motion in the particular operation mode. If the rank is 1, only a rotation around a single point is possible. The corresponding *Instantaneous Center of Rotation* (ICR) for this rotation is defined through the non-trivial column of $\dot{\Xi}$. In this case it is quite unlikely that the robot is able to fulfill its task and the information about this is provided to the higher level control unit.

Whenever DOF is equal to 2 the robot is only able to perform motions with an ICR on a particular line p_R that extends the axis of the impaired wheel's rollers as shown in Fig. 3 for a blocked motor M_2. We call this line the *confining polhode line*. The polhode line can be deduced as follows: The fault leads to two non-trivial columns of the matrix $\dot{\Xi}$. Each column specifies a corresponding ICR.

$$\dot{\Xi} = \begin{bmatrix} \dot{x}_{R1} & \dot{x}_{R2} & 0 \\ \dot{y}_{R1} & \dot{y}_{R2} & 0 \\ \dot{\Theta}_{R1} & \dot{\Theta}_{R2} & 0 \end{bmatrix} \rightarrow I_1 = \begin{bmatrix} \dot{\Theta}_{R1} \\ -\dot{y}_{R1} \\ \dot{x}_{R1} \end{bmatrix}, I_2 = \begin{bmatrix} \dot{\Theta}_{R2} \\ -\dot{y}_{R2} \\ \dot{x}_{R2} \end{bmatrix}$$

Both points (I_1, I_2) can be interpreted as *homogeneous coordinates* and one can compute the confining polhode line

through the cross product

$$p_R = I_1 \times I_2 \ . \tag{14}$$

Compared to the fault-free omni-directional drive ($rank(\dot{\Xi}) = 3$) we loose one DOF in the described fault szenario. Therefore, one has to decide which DOF is given up by making it *dependent* on the remaining two DOF's. In general, it seems to be useful to give up the freedom of rotation, so that every desired path can be still traversed. The algorithm to compute the now dependent angular velocity $\dot{\Theta}_R$ is based on the following line of reasoning:

The desired trajectory with its translatory velocities $\mathbf{v} = [\dot{x}_R \ \dot{y}_R]^T$ determines the position of the center of the robot O_R over time. We are able to calculate a perpendicular line n_I to any point on the trajectory in homogeneous coordinates (Fig. 6). The intersection of this line with the confining polhode line p_R results in the momentary ICR.

$$ICR = p_R \times n_I \ . \tag{15}$$

This ICR guarantees that (a) the robot moves on the desired trajectory and (b) that the impaired wheel does not rotate. This determines a dependent $\dot{\Theta}^*_R$ as

$$\dot{\Theta}^*_R = \frac{v}{r} \ , \tag{16}$$

where r denotes the distance of the robot's center (O_R) to the computed ICR and $v = |\mathbf{v}|$ denotes the robots translational velocity. We combine the calculated value for $\dot{\Theta}^*_R$ with the desired longitudinal velocities and obtain a valid motion

$$\dot{\xi}^*_R = [\dot{x}_R \ \ \dot{y}_R \ \ \dot{\Theta}^*_R]^T \tag{17}$$

on the desired path that automatically compensates the fault.

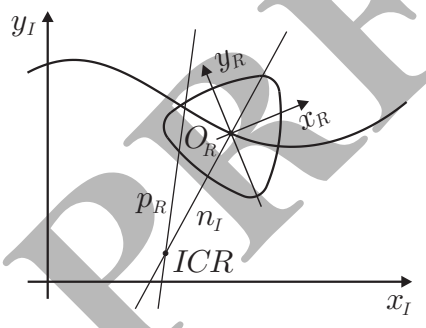

Fig. 6. ICR condition for fault reconfiguration.

V. EXPERIMENTAL RESULTS

For the evaluation of the proposed method we implemented the diagnosis and reconfiguration framework for the robot platform presented in Section II. Moreover, we conduct a series of experiments in order to evaluate the reaction of the proposed system to different fault scenarios.

In order to be able to perform the experiments we extend the firmware of the robot drive with the possibility to inject faults on-line. By sending a special command to the drive we are able to switch each motor independently to three different modes. The possible modes are: the motor works as expected (*OK mode* - m_1), the motor become stuck (*stuck mode* - m_2) and the motor rotates freely without providing any torque (*free mode* - m_3). Both fault scenarios are simulated by permanently breaking the motor or by switching off the power of the motor. The former mode simulates situations like broken wheels or gear boxes. The latter simulates faults in the motor or its control electronics. It has to be noted that all other functions of the drive firmware are not aware of the actual mode or mode changes. Furthermore, the drive is able to provide observations, e.g., encoder ticks of each motor and odometry data, at a variable frame-rate.

The robot performs a so called *eye catcher* task where it moves on a given path (in our experiments a Hermite spline) and has to maintain its orientation towards a given point. We choose this task because it requires all degrees of freedom of our omni-directional drive.

The experiments were conducted in the following way. First we command the robot to traverse the given Hermite spline trajectory and maintain the orientation for the nominal mode where no fault occurs. It has to be noted that we just control the robot on the spline without any global localization feedback. The correct spline is depicted in Fig. 10.

After that run we repeat the same experiment and inject a fault (stuck mode - m_2) halfway through the spline. The robot's control system is not informed about this fact and we further switched off the reconfiguration engine but performed the diagnosis. Obviously, the control system actuates the drive as in nominal mode, but the robot drive does not behave as expected. We record the diagnosis results in order to evaluate the performance of the diagnosis in this uncorrected mode. Finally, we repeat the fault experiment with an operational reconfiguration engine in order to evaluate the capability of the framework to react to an injected fault.

We repeat this set of experiments for single faults in each of the motors M_1, M_2, M_3. During the different runs of the experiments we recorded the commanded motion, the measurement from drive, the global position and orientation and the derived diagnosis.

A. RESULTS OF THE DIAGNOSIS

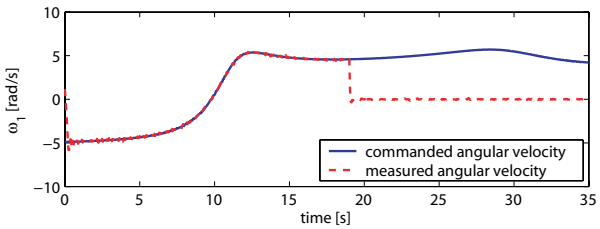

Fig. 7. Angular velocity of motor M_1.

Fig. 7 shows experimental results for a fault in motor M_1 that blocks the wheel. The discrepancy between the desired angular velocity and the actual velocity leads to a detection of this fault after two sampling periods ($Td = 0.05 \ [s]$).

However, we have to keep in mind that our diagnosis engine works in close collaboration with the reconfiguration engine. Fig. 8 shows the same experimental setting but with reconfiguration enabled. Our reconfiguration procedure leads to a corrected actuation that turns off the impaired drive and corrects the actuation of the other drives (Fig. 9). As a consequence, the fault in motor M_1 becomes *unobservable*. The suppressed actuation and no angular velocity is conform with the blocking mode m_2, as well as the nominal operational mode m_1. This example clearly shows that one has to consider diagnosis and reconfiguration together. One cannot assume that a diagnosis engine that works properly alone remains operational in a closed loop system that compensates faults through reconfiguration.

Our diagnosis model (Fig. 4) accounts for this fact and allows the 'repair' mode-transition $m_2 \rightarrow m_1$ only through an explicit 'reset' command that is initiated through the reconfiguration unit whenever it reverts to nominal operation. This strategy is desirable whenever a blocking motor can overheat and thus should be switched off. However, one

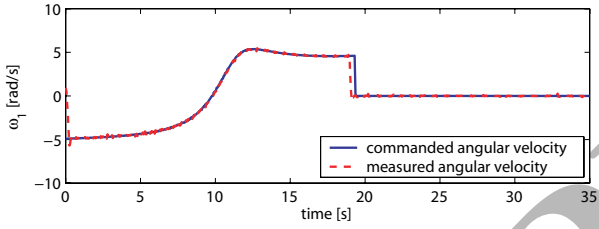

Fig. 8. Angular velocity of motor M_1 with reconfiguration enabled.

might want a different strategy to deal with *temporarily blocked* motors. Instead of using the adapted drive command $\dot{\xi}_R^*$ of (17) we can modify the command in a way that leads to the same angular velocities for the operational motors, but to a *non-zero* velocity for the impaired one. As a consequence, we instantly recognize whether the motor is still blocked or not and can automatically revert to nominal operation of the drive.

B. RESULTS OF THE RECONFIGURATION

Fig. 10 shows the results of the reconfiguration experiments. The three figures depict the series of experiments for faulty motors M_1 (left), M_2 (center) and M_3 (right). The solid blue line denotes the reference path for the fault-free robot drive (the arrows depict the global orientation of the robot).

Without reconfiguration, we obtain an unpredictable path of the robot once the fault occurs (green - dashed). The robot shows quite different faulty trajectories depending on which of the three motors failed. Obviously, the robot is neither able to stay on the desired path nor it is able to maintain the desired global orientation.

The red dash-dotted line depicts the traversed trajectory for the case that the reconfiguration is switched on. The drive's reconfiguration takes place immediately after the robot diagnoses the fault. The recorded trajectories clearly

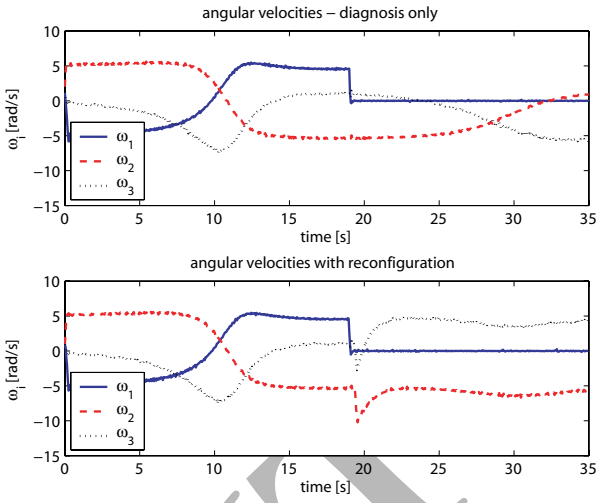

Fig. 9. Angular velocities of all 3 motors.

show that the robot is now able to follow the desired path in an adequate quality[2] even if a fault occurs in one of the motors. However, it is evident that the robot cannot maintain the eye-catcher orientation.

VI. RELATED RESEARCH

Diagnosis and automatic reconfiguration of autonomous systems has been in the focus of research for decades. There are a wide range of different approaches for diagnosis and reconfiguration. These approaches differ mainly in the used type of models (qualitative, quantitative or hybrid) and their deduction process (probabilistic state estimation, rule-based systems or logic inference). For example, the Livingstone architecture proposed by Williams et al. [11] was used by the space probe *Deep Space One* to diagnose failures in its hardware and to recover from them. The fault diagnosis and recovery are based on model-based reasoning. Dearden et al. [12] and Verma et al. [13] used particle filter techniques to estimate the state of the robot and its environment. These estimations together with a model of the robot were used to diagnose faults. The advantage of this approach is that it accounts uncertainties of the robot's sensing and acting and in its environment because the most probable state is derived from unreliable measurements. But so far it was only applied for diagnosis of faults in the robots hardware and no repair actions are derived. Rule-based approaches were proposed by Murphy and Hershberger [14] to detect failures in sensing and to recover from them. Additional sensor information were used to generate and test hypotheses to explain symptoms resulting from sensing failures.

VII. CONCLUSION

Our paper presented an approach for fault detection, diagnosis and reconfiguration for a mobile robot. We solved this task through a close interaction between a diagnosis and

[2]Our described compensation method acts as feed-forward disturbance compensation so that an additional closed loop path control can improve the resulting trajectory even further.

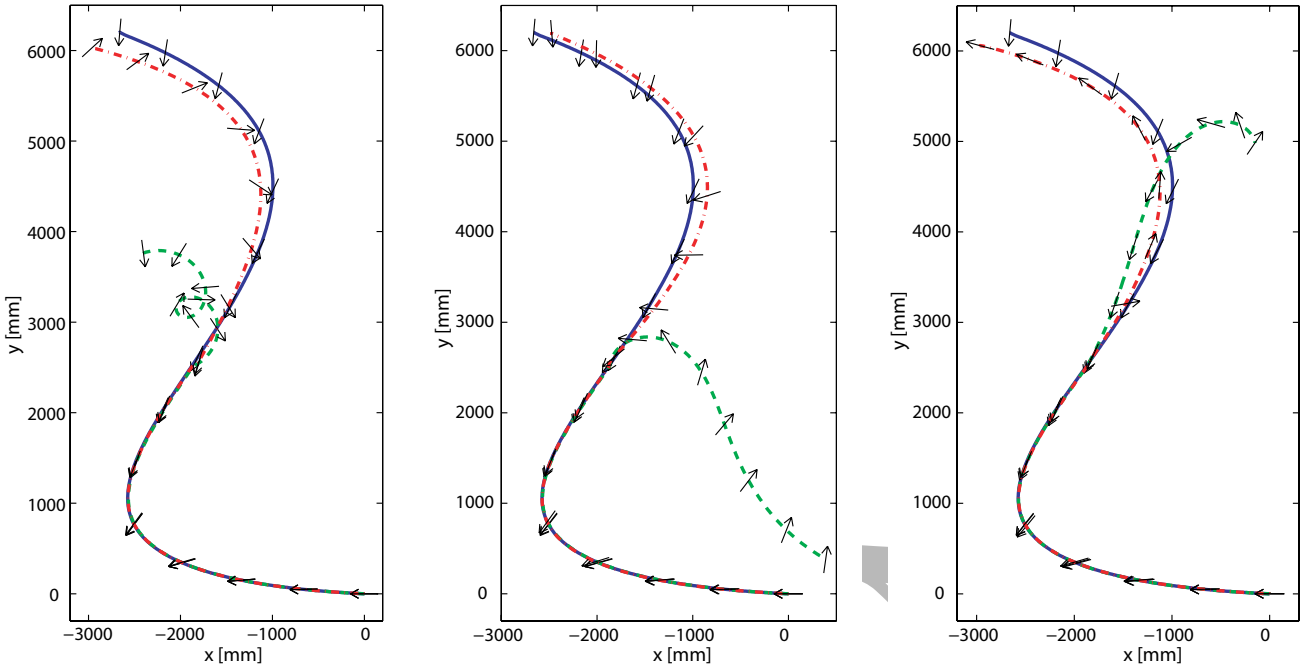

Fig. 10. The figure shows the reconfiguration experiments for a fault in motor M_1 (left), a fault in motor M_2 (center) and a fault in motor M_3 (right). The blue solid line depicts the desired path recorded without any faults. The arrows depict the actual orientation of the robot at that position. Moreover, the robot on that path has to look always towards the position [-3000,0]. The green dashed line depicts the path if on motor sticks after 50 % of the path has been traversed. The red chained dotted line depicts the path if the failure occurs and the fault is detected and the appropriate reconfiguration takes place. These path clearly shows that the robot is able to traverse the desired path also if a fault occurs but is not able to maintain the desired orientation anymore.

a reconfiguration engine that takes into account the dynamics and the kinematics of the drive. An experimental evaluation demonstrates our concept with an omni-directional drive that we use in our RoboCup robots.

This paper presents a specific realization and implementation of our general model-based control framework that we develop for adaptable wheeled drives. This control framework is capable of reconfiguring the functionality of individual drive units and handles various drive geometries. For this purpose we develop on-line algorithms for hybrid estimation and diagnosis, kinematic reasoning and automated reconfiguration. This will provide a solid and flexible basis for a number of future fault-tolerant drives of autonomous mobile robots.

References

[1] Jennifer Carlson and Robin R. Murphy. How UGVs physically fail in the field. *IEEE Transactions on Robotics*, 21(3):423–437, 2005.

[2] John Wright, Frank Hartman, Brian Cooper, Scott Maxwell, Jeng Yen, and Jack Morrison. Driving on Mars with RSVP. *IEEE Robotics & Automation Magazine*, 13(2):37–45, 2006.

[3] Michael Hofbaur, Johannes Köb, Gerald Steinbauer, and Franz Wotawa. Improving robustness of mobile robots using model-based reasoning. *Journal of Intelligent and Robotic Systems*, 48(1):37–54, 2007.

[4] Gerald Steinbauer, Mathias Brandstötter, Martin Buchleitner, Stefan Galler, Simon Jantscher, Gerald Krammer, Martin Mörth, Jörg Weber, and Martin Weiglhofer. Mostly Harmless Team Description 2006 - Robust Control of Mobile Robots. In *Proceedings of the International RoboCup Symposium*, 2006.

[5] Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.

[6] M. W. Hofbaur and B. C. Williams. Mode estimation of probabilistic hybrid systems. In C.J. Tomlin and M.R. Greenstreet, editors, *Hybrid Systems: Computation and Control, HSCC 2002*, volume 2289 of *Lecture Notes in Computer Science*, pages 253–266. Springer, 2002.

[7] Michael Hofbaur. *Hybrid Estimation of Complex Systems*, volume 319 of *Lecture Notes in Control and Information Sciences*. Springer, 2005.

[8] B. Anderson and J. Moore. *Optimal Filtering*. Information and System Sciences Series. Prentice Hall, 1979.

[9] W. Hamscher, Luca Console, and Johan de Kleer, editors. *Readings in Model-Based Diagnosis*. Morgan Kaufmann Publishers, Inc., 1992.

[10] Johannes Köb. Modellbasierte Regelung eines Roboterfahrwerkes. Master's thesis, Institute for Automation and Control, Graz University of Technology, 2005.

[11] Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian C. Williams. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1-2):5–48, August 1998.

[12] Richard Dearden and Dan Clancy. Particle filters for real-time fault detection in planetary rovers. In *Proceedings of the Thirteenth International Workshop on Principles of Diagnosis*, pages 1 – 6, 2002.

[13] Vandi Verma, Geoff Gordon, Reid Simmons, and Sebastion Thrun. Real-time fault diagnosis. *IEEE Robotics & Automation Magazine*, 11(2):56 – 66, 2004.

[14] Robin R. Murphy and David Hershberger. Classifying and recovering from sensing failures in autonomous mobile robots. In *AAAI/IAAI, Vol. 2*, pages 922–929, 1996.