A Low-Speed Flatness-Based Path Tracking Control with the Time-Scaling Concept for Different Types of Steering and Drive for the Robot Operating System

Ilja Stasewitsch[1], Tobias Blume[1], Hannes Harms[1], Jan Schattenberg[1], Ludger Frerichs[1]
[1]Institute for Mobile Machines and Commercial Vehicles, TU Braunschweig, Germany
i.stasewitsch@tu-braunschweig.de

Keywords: Robot Operating System, Path Tracking Control, Machine Navigation

Abstract: For applications like parking maneuver, placing a pallet with a forklift or similar applications with a high requirement to the positioning and path tracking it is necessary to use higher control algorithms. A convenient control method is a flatness-based path tracking control. For the separation of the path from the driving velocity the time-scaling can be used. This paper presents the implementation of this control algorithm in the Robot Operating System. In this package different types of steering and drives are considered. Furthermore, the derivation for the control will be outlined and the controller will be validate on different test platforms.

1 INTRODUCTION

The motion control is essentially for a mobile robot or a highly automated mobile machine. It can be separated in two main tasks: longitudinal and lateral control. Longitudinal control deals with velocity of the machine whereas lateral control is concerned with the position and orientation of the machine. Both control tasks can certainly be integrated in one control algorithm. The focus of this paper is lateral control which is usually called path tracking control. In contrast to a trajectory control the time of arrival is irrelevant. In the literature there are a lot of different algorithms for a path tracking control. For instance, Chang et al. (2013) or Normey-Rico et al. (2001) used the modified PID controller as a path tracking control. There are also some self-defined controllers like the Stanley method by Thrun et al. (2006) or the Lyapunov stable path tracking control by Kanayama et al. (1991). Examples for advanced control methods with linearized or non-linearized models for path tracking are the sliding mode control (Solea et al., 2009), the state space control (Divelbiss et al., 1997), the exact feedback linearization (Müller et al., 2008), the fuzzy control (Giap et al., 2008), the gain scheduling, the linear or nonlinear model predictive control (Kühne et al., 2005). This variety is the result of different applications and kinematics of mobile robots and machines. For the parking assistance systems the automotive industry is using a flatness based feedback controller with the time-scaling concept (Kochem et al., 2004), (Müller et al., 2008), (Szádeczky-Kardoss et al., 2009). This approach has the advantage that a linear input-output performance exists. Therefore a linear controller can be used for all system states so that no adaptions of the controller poles are necessary. This controller was implemented in the context of our project. The aim is to develop a mobile robot for removing dung, cleaning cubicle and littering down in dairy stables. An important module in the robot navigation is the path tracking control.

The motivation of this paper is to introduce our software package which contains the flatness based path tracking controller and which is implemented in the Robot Operating System (ROS) (Quigley et al., 2009). ROS is an open-source robotic framework that has been widely applied across academia, industry and militaries around the world. Although, there are several thousand free available packages in the ROS community only few path tracking controllers exist which are not designed for robot manipulators. Furthermore, these controllers are mostly simple and not designed for robots with an Ackermann steering geometry.

The flatness based path tracking control is explained in section 2 of this paper. The target is to show the advantages and the limits of the implemented path tracking controller by deriving the control algorithm. In section 3, the implementation in ROS is presented. There, we explain the ROS interface and the controller settings. Section 4 details experiments on different test platforms. We vary path types, poles and velocities and we show also errors on the wheelbase length and the encoder. Section 5 concludes the paper and details plans for the extending the package.

2 FLATNESS BASED PATH TRACKING CONTROL

The flatness based feedback control rests on the idea to linearize the properties between the input and output of the system so that a linear control can be used. This method was successfully implemented as a parking assistance system in Müller et al. (2007). For this approach a model of the motion dynamic has to be chosen. Due to small velocities, small accelerations and large steering angles kinematic motion models are used in robotics and in this control. In addition, dynamic motion models are too complex and unnecessary due to consideration of the inertia forces, tire forces and vertical dynamic. Simple dynamic models are linearized for small steering angles $\cos\phi \approx 1$, $\sin\phi = 0$, so that large steering angles lead to inaccuracies.

Like usual in the robotics we are using a skid steering wheeled robot (see fig. 1) in our project. The

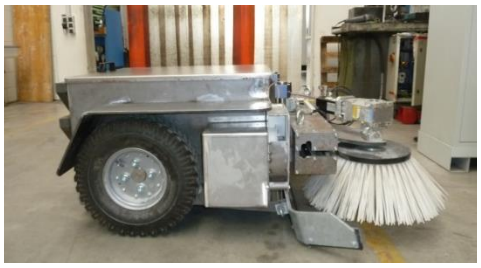
Figure 1: Project test platform

kinematic unicycle model for such a robot is

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}v\ \cos\theta\\ v\ \sin\theta\\ \omega\end{bmatrix} = \begin{bmatrix}v\ \cos\theta\\ v\ \sin\theta\\ \kappa v\end{bmatrix} \tag{1}$$

where the state vector is $x = [x, y, \theta]^T$, the input vector is $u = [v, \omega]^T$ and the curvature is $\kappa = \frac{\dot{x}\,\ddot{y} - \ddot{x}\,\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}}$.

To separate the driving velocity from the reference path the system model (1) can be transformed by the time scaling concept, so that the differential equations are not time dependent but dependent from the arc-length of the reference path $s_d(t)$ (Szádeczky-Kardoss et al., 2009). This can be done by extending (1) with $\frac{ds_d}{ds_d}$ and introducing the new control input $u_2 = \frac{v}{\dot{s}_d}$:

$$\begin{bmatrix}x'(s_d)\\ y'(s_d)\\ \theta'(s_d)\end{bmatrix} = \begin{bmatrix}u_2\ \cos\theta(s_d)\\ u_2\ \sin\theta\,(s_d)\\ u_2\kappa(s_d)\end{bmatrix}. \tag{2}$$

A flat output of the system (3) is

$$y_f = \begin{bmatrix}x(s_d)\\ y(s_d)\end{bmatrix} \tag{3}$$

because the state

$$\theta(s_d) = \arctan\frac{y'}{x'}, \tag{4}$$

the new velocity control input

$$u_2(s_d) = \sqrt{x'^2 + y'^2} \tag{5}$$

and the rotational velocity control input

$$u_1(s_d) = \omega(s_d) = \frac{x'y'' - x''y'}{\sqrt{x'^2 + y'^2}} \tag{6}$$

are depend on the flat output and their derivations. To transform the system (2) into the Brunovský normal form, the dynamic feedback extension is applied so that instead of the input u_2 the derivation of u_2 is used as the system control input:

$$u_{2e}(s_d) = u_2'(s_d) = \frac{x'y'' + x''y'}{\sqrt{x'^2 + y'^2}}. \tag{7}$$

The transformation into the Brunovský coordinates can be done by

$$z = \begin{bmatrix} z_1^1 \\ z_2^1 \\ z_1^2 \\ z_2^2 \end{bmatrix} = \begin{bmatrix} x \\ x' \\ y \\ y' \end{bmatrix} = \begin{bmatrix} x \\ u_2 \cos\theta \\ y \\ u_2 \sin\theta \end{bmatrix}. \qquad (8)$$

Due to the linear system behavior between the input and output the following linear tracking control is chosen

$$\begin{aligned} v_1 &= x_d'' - a_1^1(z_2^1 - x_d') - a_0^1(z_1^1 - x_d) \\ v_2 &= y_d'' - a_1^2(z_2^1 - x_d') - a_0^2(z_1^1 - x_d) \end{aligned} \qquad (9)$$

where v_1 and v_2 are the control inputs of the exact linearized system. The equations (9) can be interpreted as error dynamics for the coordinates (3). Their dynamic can be defined by the parameters a_0^1, a_1^1, a_0^2 and a_1^2. The parameters should be chosen so that the aperiodic boundary case exists. Additionally, same dynamics should be chosen for both coordinates. These facts lead to the determination of one real pole for the linear tracking control. The control inputs (9) are used to calculate the system inputs (6) and (7):

$$\begin{aligned} u_1(s_d) &= \theta'(s_d) = \frac{z_2^1 v_2 - v_1 z_2^2}{\sqrt{{z_2^1}^2 + {z_2^2}^2}} \\ u_{2e}(s_d) &= u_2'(s_d) = \frac{z_2^1 v_2 + v_1 z_2^2}{\sqrt{{z_2^1}^2 + {z_2^2}^2}}. \end{aligned} \qquad (10)$$

Figure 2 illustrates the control loop of the path tracking algorithm. It shows that the robot velocity is calculating the velocity of the arc $\dot{s}_d$ which is used to

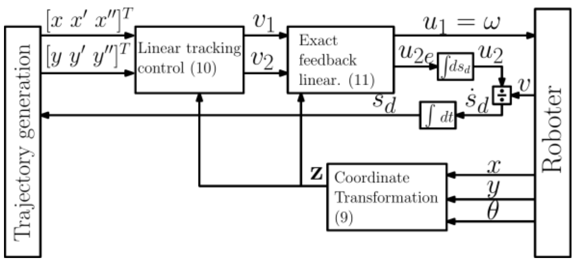

Figure 2: Block diagram of the path tracking control

determine the path reference point. This implies that an error in the measurement of velocity is effecting the control quality. In addition, the control input u_2 is used to calculate the velocity of the arc length $\dot{s}_d$. This means, if the path error is getting higher, the velocity of arc length will be reduced by the controller input u_{2e} in a way that the convenient reference point will be determined.

3 IMPLEMENTATION IN THE ROBOT OPERATING SYSTEM

The framework ROS is a robotic middleware and based on the idea that a standalone program, called node, executes a specific task. The aim is to have small programs which can be reused. Nodes can communicate with each other through messages in three different types: topics, services and actions. With topics every node can subscribe, publish and process a message. With services every node can call the service of another node by a message and get the result of the calculation or execution. An action is similar to a service but it can be cancelled or it aborts the action due to errors. And it sends a feedback to the exclaimed node frequently. In our ROS package we chose an action for the communication between our control node and the exclaimed node, because it is possible to cancel the action by the exclaimed node anytime. It is, furthermore, not necessarily to be done permanent and an action can send frequently a feedback. The Table 1 shows that in our package all usual combinations of steering and drive types in mobile machines and robotics are implemented. For every combination the system model (1) has to be adjusted so that other control inputs (10) are used. Machines with four-wheeled steering have the restriction that both axes have to be steered with the same angle. Omnidirectional steering is not considered because such machines are holonomic which can be controlled easily with PID controllers.

Table 1 : Implemented drive and steering combinations

		Drive		
		Front	Rear	All
Steering	Front	✓	✓	✓
	Rear	✓	✓	✓
	Four-wheeled	✓	✓	✓
	Articulated	✓	✓	✓
	Skid	-	-	✓

Figure 3 shows the implementation in ROS. A node which is acting as the action client sends an action message to the action server. The message contains the path, the desired velocity and the pole for the error dynamic (9). This means that control performance can be influenced by the path curve and the determined pole. As a feedback the action server sends frequently the degree of fulfillment which is the driven distance. If the robot achieve the goal, the fulfillment status is done. The control node subscribes the topic "robot_vel" and needs the transformation from the global frame to the base frame to calculate the Brunovský coordinates (8). The node is publishing the topic "cmd_vel" with a message which contains the translation velocity and rotational velocity or the steering angle for an Ackermann steering vehicle. With a launch file, parameters like the steering and drive type can be chosen. The controller frequency depends on the frequency of the published topic "robot_vel" so that the current velocity is used for each iteration.

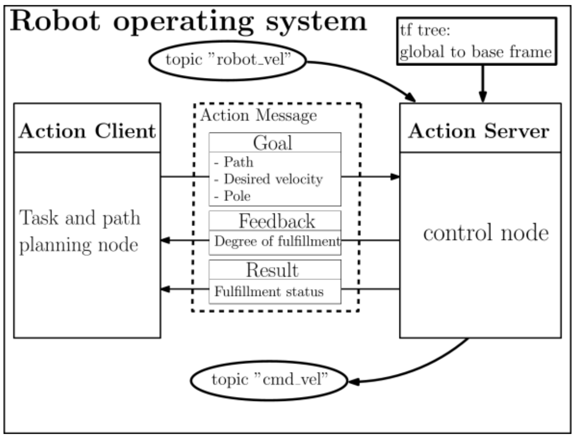

Figure 3: Implementation of the control node in the Robot Operating System

4 VALIDATION OF THE IMPLEMENTATION

We designed and executed experiments with our test platforms to validate and evaluate the performance and implementation of our control node. Our main test platform is the in-house developed mobile robot comRoBS. The comRoBS (see fig. 4) has a four-wheel drive due to a central stepping motor which is connected with each axis through a cardan shaft and a differential gear.The goal of the design was to get a high stiffness and small dead zones and hysteresis in the mechanical transmission. Otherwise, these facts would lead to additional nonlinearities which impact the control performance. The encoder of the stepping motor can be used to get the robot velocity. For the steering, there is a servomotor on each axis so that it is possible to choose front wheel steering, back wheel steering or four-wheel steering. Furthermore, the control is tested on a Fendt 724 Vario and in the simulation with a Robotnik SUMMIT XL which has a skid steering. We used the SUMMIT XL because our project robot (see fig. 1) was not available during the experiment executions due to modification work.

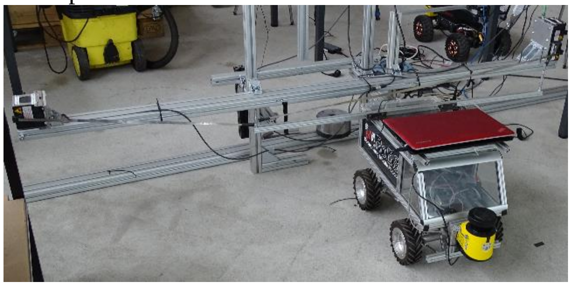

Figure 4 : Validation of the control with the robot comRoBS

4.1 Experiment Setup

A local positioning system was built for the validation. The System consists of three string potentiometers to get the 3DOF. The potentiometers are mounted on a fixed bar and the strings are attached to a bar at the robot on two points. Figure 5 shows the principle of the measuring configuration with the length of the string potentiometers l_1, l_2 and l_3. These three lengths are used to calculate the robot pose $[x_R \; y_R \; \theta_R]^T$ using trigonometry. String potentiometers are highly precise. The used sensors ASM POSITAPE® WB85 have only a linearity error of $\pm 0.1\%$. Because of this high accuracy all significant errors in the pose determination results of manufacturing and assembly errors.

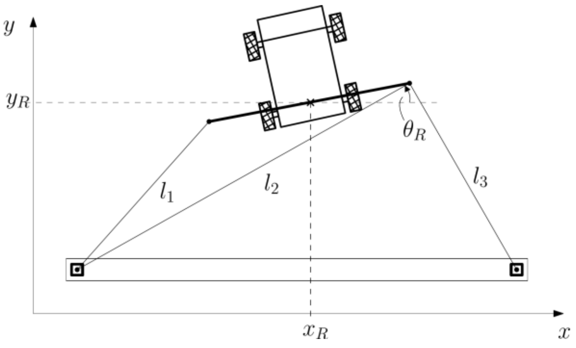

Figure 5 : Schematic diagram of the global positioning systems

4.2 Experiment Execution

For the main experiments the comRoBS is set to front wheel steering. The first experiment examines the impact of the curve geometry. For path planning, in robotics Dubins path, clothoid and splines are used usually. The issue of Dubins path is that the derivation are no continuous functions which are required for the error dynamics (10). Figure 6 reveals,

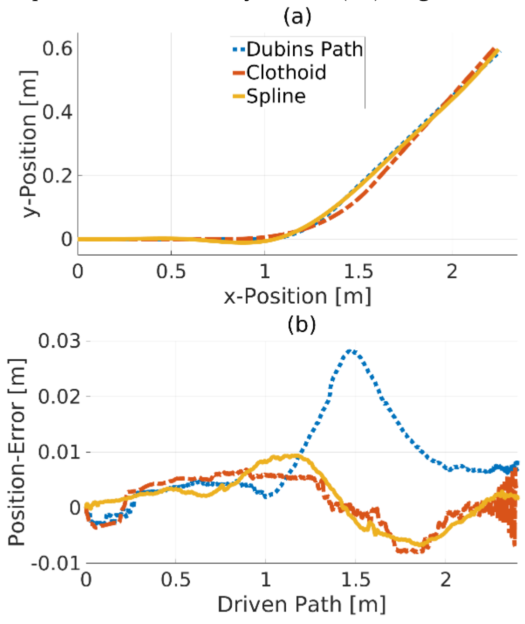

Figure 6 : Impact of the curve (a) desired trajectory (b) resulting position error

that the position error for the Dubins path is much higher while the position error is more or less the same for the clothoid and the spline. For this reason a spline will be used for the other tests. The next experiment shows effects of an error in the length of the wheelbase which certainly occur only for machines with Ackermann steering. Figure 7 shows, that the controller has acceptable position errors regarding to large errors in the wheelbase length. Therefore, small errors do not affect the control significant. As already mentioned in section 2 the velocity of the robot has an impact on the control performance. Errors in the velocity can be slippage, encoder offset or unknown transmission parameters in the powertrain. For the experiment displayed in figure 8, a constant offset is added to the velocity to analyze this impact. The position error is rising during the drive as well as in the endpoint because the wrong desired position is chosen from the trajectory (see fig. 2). For our test platform comRoBS, there are small errors in the wheelbase length and in the robot velocity. But the impact is so small, that it does not affect the following experiments. In figure 9 and 10 the velocity and the pole are varied. It is obvious that poles have an impact on the control performance and higher velocities need a lower pole otherwise it will occur an oscillation.

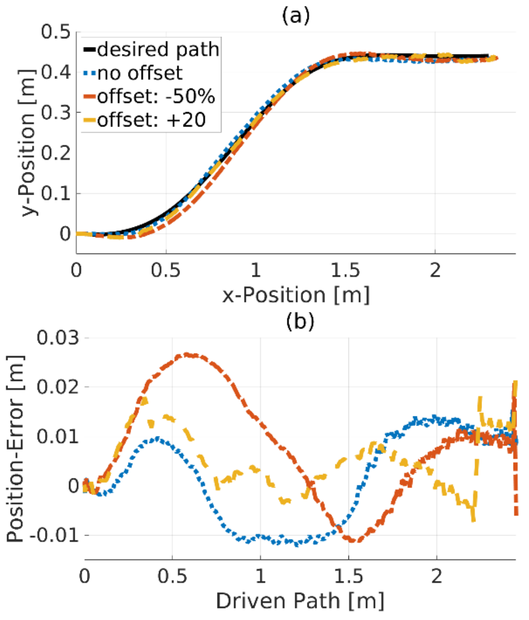

Figure 7 : Impact of the wheelbase length (a) trajectory (b) position error

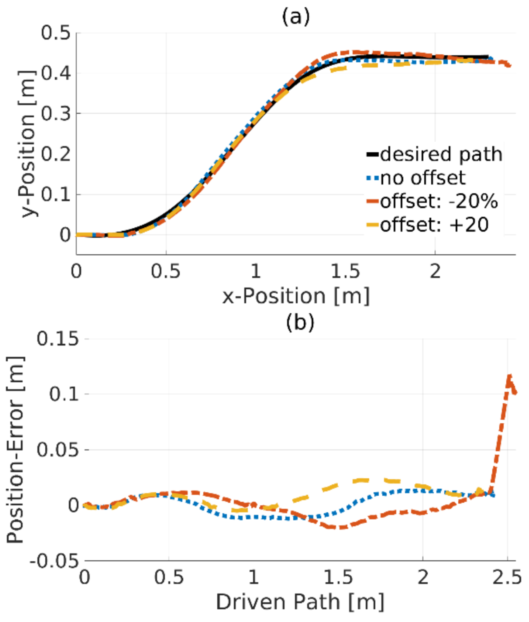

Figure 8: Impact of a velocity error (a) trajectory (b) position error

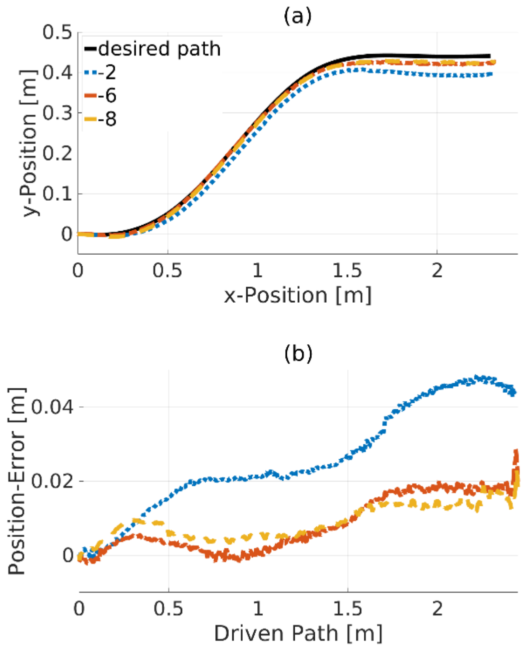

Figure 9: Impact of the pole for the error dynamic (a) trajectory (b) position error

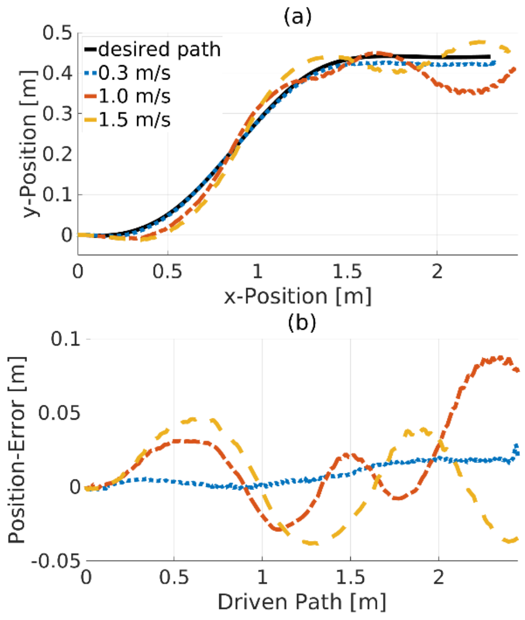

Figure 10 : Impact of the desired velocity (a) trajectory (b) position error

As previously mentioned, the control is also validated with other platforms. We did our tests with the skid steered robot Robotnik SUMMIT XL in the simulation environment Gazebo. The control performance in figure 11 is sufficient for such a difficult trajectory. Due to errors in the simulation, the drive of the robot reacts only above a quite high threshold. Furthermore, the comRoBS was used to test the four-wheel steering control. The results can be seen in figure 12. The control has a good performance except of the oscillation in the end of the path. For testing the control on large machine we chose a Fendt 724 Vario which has a wheelbase about 2.8 m. Figure 13 displays applicability on such machines. Although, the reference path does not look difficult for the control, it is comparable with the test on the other test platforms due to the large turning circle.

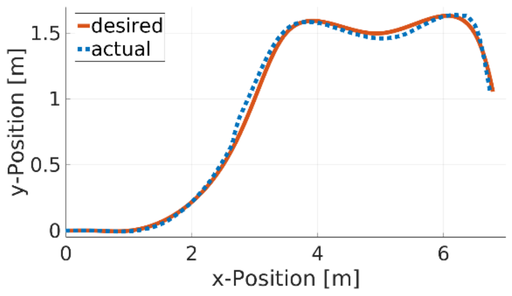

Figure 11 : Validation of the control for a skid steered robot with the Robotnik SUMMIT XL

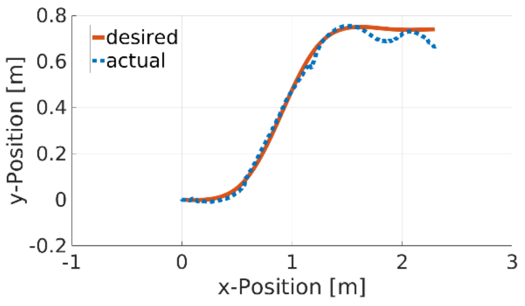

Figure 12 : Validation of the control for a four-wheel steered and all-wheel drive robot with the comRoBS

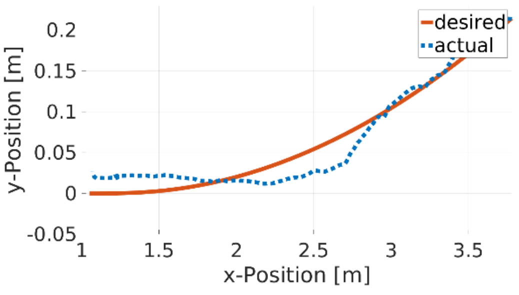

Figure 13: Validation of the control for a front-wheel steered and rear-wheel drive robot with the Fendt 724 Vario

4 CONCLUSION AND FUTURE WORK

In this paper, we introduced a control algorithm for the ROS framework. It supports all usual types of drive and steering in the autonomous machine guidance and robotics and it is easy to tune due to only one parameter. Because of the chosen model in section 2 it is only useful for low speed applications. The focus was to validate the developed control on different test platforms. It turns out that clothoids and splines should be used as paths and small errors on the wheelbase length and velocity do not affect the control performance significantly. Furthermore, the impact of the error dynamic and the desired velocity was outlined.

In future work will extend the control algorithm with motion models which consider the steering dynamic by a first order lag element. This extension will yield a higher accuracy. The development of other advanced control algorithm is also conceivable. But the next step is to release this algorithm as a package for the ROS community. It is the authors' hope that this package will be a benefit for the ROS community.

ACKNOWLEDGEMENTS

The project is supported by funds of the German Government's Special Purpose Fund held at Landwirtschaftliche Rentenbank.

rentenbank

REFERENCES

Chang, H. and Jin, T., 2013, September. Adaptive Tracking Controller Based on the PID for Mobile Robot Path Tracking. *In International Conference on Intelligent Robotics and Applications* (pp. 540-549). Springer Berlin Heidelberg.

Divelbiss, A.W. and Wen, J.T., 1997. Trajectory tracking control of a car-trailer system. *IEEE Transactions on Control Systems Technology*, 5(3), pp.269-278.

Giap, N.H., Shin, J.H. and Kim, W.H., 2008. Adaptive robust fuzzy control for path tracking of a wheeled mobile robot. *Artificial Life and Robotics*, 13(1), pp.134-138.

Kanayama, Y., Kimura, Y., Miyazaki, F. and Noguchi, T., 1991, November. A stable tracking control method for a non-holonomic mobile robot. In *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91*. IEEE/RSJ International Workshop on (pp. 1236-1241). IEEE.

Kochem, M. and Isermann, R., 2004. Nonlinear path following control of non-holonomic vehicles with the human as actuator in a parking application. *International Journal of Robust and Nonlinear Control*, 14(6), pp.513-524.

Künhe, F., Gomes, J. and Fetter, W., 2005, September. Mobile robot trajectory tracking using model predictive control. In *II IEEE latin-american robotics symposium*.

Müller, B. and Deutscher, J., 2007, July. Orbital tracking control for car parking via control of the clock using a nonlinear reduced order steering-angle observer. In *Control Conference (ECC), 2007 European* (pp. 1917-1924). IEEE.

Normey-Rico, J.E., Alcala, I., Gómez-Ortega, J. and Camacho, E.F., 2001. Mobile robot path tracking using a robust PID controller. *Control Engineering Practice*, 9(11), pp.1209-1214.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A.Y., 2009, May. ROS: an open-source Robot Operating System. *In ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).

Solea, R., Filipescu, A. and Nunes, U., 2009, August. Sliding-mode control for trajectory-tracking of a wheeled mobile robot in presence of uncertainties. In *Proceedings of the 7th Asian Control Conference* (pp. 1701-1706).

Szádeczky-Kardoss, E. and Kiss, B., 2009. On-line trajectory time-scaling to reduce tracking error. In *Intelligent Engineering Systems and Computational Cybernetics* (pp. 3-14). Springer Netherlands.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G. and Lau, K., 2006. Stanley: The robot that won the DARPA Grand Challenge. *Journal of field Robotics*, 23(9), pp.661-692.