Expressive Navigation and Local Path-Planning of Independent Steering Autonomous Systems

George Todoran[1], Markus Bader[1]

***Abstract*— This paper presents a novel Local-Path Planning approach for an independent four-wheel steering (I4WS) mobile base. The approach smoothly drives and rotates the platform while still following a given path and avoiding obstacles. I4WS vehicles are omnidirectional vehicles, similar to shopping carts, if one neglects the time needed to steer the wheels. However, due to the danger of mechanical parts breaking while actuating the wheels, they are commonly controlled in a stop-and-go fashion, where the vehicle stops to turn its wheels perpendicular to a desired instantaneous center of curvature (ICC) before starting to move again. The approach overcomes this limitation and ensures an ICC-based kinematic constraint during continuous motion using a flat-input controller running at 100 Hz. Therefore, the trajectory of the ICC is both predictable and suitable for model predictive control (MPC). The MPC implemented generates optimized collision-free trajectories of up to several meters ahead at 10 Hz, given a set of points along a path and laser contour readings. Furthermore, the planner realized here is able to deal with additional constraints such as the vehicle's view direction to focus on attention points while driving. Experimental results highlight the capabilities of the approach on a simulated robot, using GazeboSim, and demonstrate its applicability for the field of service robotics.**

I. INTRODUCTION

Many fields of the application of mobile robotics benefit greatly from holonomic motion capabilities. True holonomicity represents the capability of the system to undergo an arbitrary trajectory (position and orientation) from any starting configuration. This property allows wheeled mobile agents to navigate effectively in cluttered environments such as work-spaces shared with humans or work-spaces restricted with respect to the agent size (e.g. forklifts operating in indoor warehouses). As true holonomic platforms using omni or mecanum wheels tend to be avoided for operation in unstructured environments (due to their generally required wheel maintenance), the typical design used for holonomic motions makes use of independent steering systems.

Fig. 1 – left illustrates *Blue*, an assistive robot possessing an independent four-wheel steering (I4WS) mobile base and represents the motivation for this work. Its mobile platform is classified as an independent steering system, possessing eight actuators (two for each wheel) and the additional constraint of limited wheel orientation (approximately 270°).

Despite their physical robustness and maintenance benefits, independent steering systems impose other difficulties, collocated control and synchronization of the wheels' motors being required in order to make use of their increased maneuverability. Furthermore, nonlinear kinematic and dynamic constraints of such platforms are inherently present.

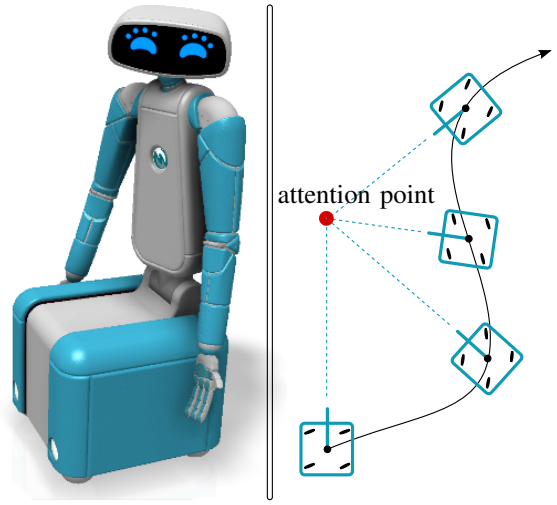


Fig. 1. Left: the robot *Blue*; Right: navigation while focusing on an attention point

The work presented here focuses on a short-term navigation solution for such platforms, aiming to push the limits of achievable platform motions and behaviours. Especially in the field of assistive robotics, expressive, natural motions ease the process of human-robot interaction and increase agent effectiveness. Advancements in research regarding robot attention provide further motivation for a navigation approach that is capable of different (safe) behaviours. For example, Fig. 1 – right illustrates an I4WS platform navigating under an interest point focusing behaviour, representing one of the expressive capabilities of our approach.

Targeting wide contexts of operation and tasks of such a mobile platform, the key points of Local Path-Planning are:

- unified motion parametrization
- safety by design through constraint satisfaction (dynamics, route deviation, collision avoidance)
- encapsulation from higher-level robotics tasks (such as human-robot interaction, grasping, etc.)
- robust trajectory-altering set of parameters (such as minimum distances-to-obstacle, chassis orientation, maximum distance to path, time optimal, energy optimal, maximum velocity, etc.)

Taking into account those considerations, the navigation module proposed here addresses the low-level actuator control as well as local trajectory planning and following, using

*The research leading to these results has received funding from the Austrian Research Promotion Agency (FFG) according to grant agreement No. 852708 (MPCv1), No. 855409 (AutonomousFleet) and No. 854865 (TransportBuddy).

[1] George Todoran and Markus Bader are with the Institute of Computer Aided Automation, Vienna University of Technology, Vienna 1040, Austria [george.todoran],[markus.bader]@tuwien.ac.at

flatness-based control as well as sampling and optimization-based trajectory generation.

This paper is organized as follows: Section II presents the state of the art on the topics of independent steering platforms and their control, Local Path-Planning and trajectory sampling and optimization in navigation contexts. Section III introduces the proposed approach and provides an overview of the underlying modules and their interaction. The main points of the approach are further investigated in the Sections IV – V. Section VI presents simulated results of the navigation module applied to the robot *Blue*. Conclusions are drawn in the Section VII.

II. STATE OF THE ART

Considering the desired holonomic characteristics, *designs* that possess quasi-holonomic motion capabilities are typically classified as independent steering platforms. In such a design, each of the actuated wheels possesses two manners of actuation freedom: wheel rotation and wheel orientation relative to the platform base.

However, such designs are intrinsically over-actuated, as they do not possess the hard mechanical constraints that guarantee the geometrical validity of an instantaneous center of curvature (ICC, depicted in Fig. 3), relying rather on electronic control to achieve geometrically proper configurations. In general, the holonomic definition of such designs is loosened to quasi-holonomicity due to the singularities that are present when the ICC is in the vicinity of a wheel [1], as well as due to additional design constraints (such as use of limited rotation motors for wheel orientation). Also, the geometrical topology of the wheels causes nonlinearities in actuator trajectories as well as in trajectory dynamics constraints.

Low-level control of such platforms can be done using various approaches. Use of discrete motion models that emulate other mobile platforms can be used, the control problem thereby becoming less general [2] and allowing algorithmic reasoning of the motion model in use. At the other end of the spectrum, control can be obtained through optimization of individual actuator states [3]. However, the large input-space as well as the increased temporal frequency of the optimization points makes it infeasible regarding the higher-level path-planning problem of obstacle avoidance, route-following and long-range (to the scale of several *m* for indoor applications). Another approach exploits the differentially flat character of the system, as presented in [4]. The approach proposed here makes use as well of a flat controller; however, it is parametrized in the local frame of the platform and used only for low-level control, the task of navigation being designated to an optimization-based Local Path-Planner.

Local Path-Planning is typically performed by sampling the control input space (also known as the Dynamic Window Approach – DWA) and by applying the best trajectory with respect to cost. However, this method becomes inaccurate and computationally expensive when the evaluated temporal horizon becomes large. An alternative method of state-space sampling [5] can be performed for increased efficiency. However, this requires pre-computation of trajectory primitives, dependent upon platform parameters and defined costs. An alternative to sampling towards an optimum is optimization. A general description of trajectory optimization in the context of autonomous navigation is done by T. Howard in his PhD. thesis [6]. Being accurate and able to cope well with nonlinear constraints, optimization is also used in autonomous vehicle driving [7]. The work proposed here is largely related to optimization. Although previous cycle trajectory is usually a good initialization for optimization, control space DWA is performed only at irregular cycles.

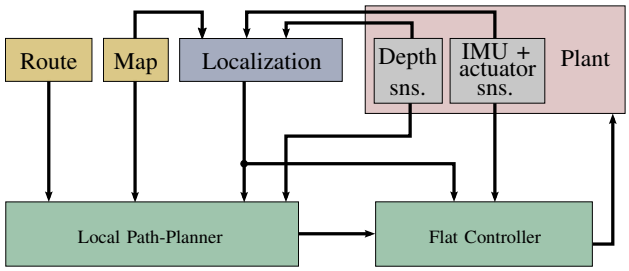


Fig. 2. Overview Diagram

III. APPROACH

Data-flow, the required modules and their interdependence for the proposed Local-Path Planning approach is depicted in Fig. 2. The Local Path-Planner, based on route, map, localization and sensor information computes, in every cycle, an optimal sequence of parametrized control commands with respect to user-defined costs. The refresh rate of the planner is designed to be around 10 Hz. Based on the (time-stamped) sequence of control commands, the low-level controller computes the necessary actuator inputs of the plant at a frequency of approximately 100 Hz. The plant in this case is represented by the I4WS platform and is equipped with actuator sensors (e.g. velocity measurements for wheel revolutions and angular measurements for wheels rotations) and an inertial measurement unit (IMU), which are used in the feedback loop by the flat controller. Additionally, the flat controller requires localization information. The localization module is a topic of research in itself and is beyond the scope of this paper. It is further considered a black box that delivers sufficiently accurate information. The Local Path-Planner requires localization information, depth information for obstacle avoidance (with the possible addition of a-priori maps) as well as a route. The focus of this paper lies on the approach, design and implementation of the Local-Path Planner and the flat controller modules, being sensitive to their interdependency, requirements and inter-connectivity with the other modules that constitute autonomous navigation.

IV. LOW-LEVEL CONTROL

Starting with the geometrical topology and constraints of an I4WS system, this section presents considerations when developing the low-level control as well as possible parametrizations and motion models used to control and simulate the agent motion.

A. Geometrical Considerations

Fig. 3 illustrates the local frame of an I4WS with its ICC at an arbitrary location. It is worth noting that for control and wheel synchronization, the ICC should be considered, in general, to be a local-frame quantity due to present geometrical constraints. In this way, its location is tractable and parametrization is substantially easier as opposed to when considered in a global frame.

Assuming that no drifting motion is desired, wheel orientations are constrained to perpendicular to the ICC and their revolution has to be consistent with the angular velocity caused by the ICC. Thus, the instantaneous kinematic state of an IWS can be described by three abstract parameters $\mathbf{p}(t)$, for example: v_c – velocity of the body center and the position of the ICC represented in polar coordinates (r_c and a_c). However, such a parametrization leads to singularity in the case of parallel wheel placement ($r_c \to ¥$) and therefore r_c is substituted with $r_c = 1=r_c$. In order to be able to perform changes to the ICC from one side of the base to the other, r_c is allowed to be negative.

The orientation $a_w^i(t)$ and rotation $w_w^i(t)$ of wheel i result from geometrical considerations and are presented in Eq. 1 and 2, with x_w^i and y_w^i – the wheel position coordinates in the local frame, r_w – the wheel radius. In the following, for notation simplification, the time dependency of v_c, r_c and a_c is assumed but not denoted.

$$a_w^i(t) = arctan \quad \frac{sin(a_c)=r_c - y_w^i}{cos(a_c)=r_c - x_w^i} \tag{1}$$

$$w_w^i(t) = \frac{-v_c}{r_w} \sqrt{1 + (x_w^{i\,2} + y_w^{i\,2})r_c^2 - 2r_c(x_w^i cos(a_c) + y_w^i sin(a_c))} \tag{2}$$

Note that these equations hold for other kinematic parametrizations as well as for other mobile platforms (such as differential drives or Ackermann drives) through trivial variable substitution.

B. Feed-forward & State-feedback Control

Even though it suffices to model the actuators as linear, the topology of a I4WS presents unavoidable nonlinearities as well as overactuation. Thus, the control scheme employed uses feed-forward control by making use of the differentially flat parameter set $\mathbf{p}(t)$ and applies control allocation to cope with overactuation. State-feedback is then performed on the actuators in the form of a linear quadratic regulator (LQG, [8]) with a Kalman-filter based state observer (through the actuator sensors and an inertial measurement unit – IMU).

Keeping in mind the quasi-linearity present in many actuator designs with the motor-shaft torque, the wheel state required for the controller is thus defined as:

$$\mathbf{x}_w^i(t) = \begin{bmatrix} a_w^i(t) & \dot{a}_w^i(t) & t_a^i(t) & w_w^i(t) & t_w^i(t) \end{bmatrix}^T \tag{3}$$

with $t_a^i(t)$ and $t_w^i(t)$ representing the feed-forward actuator torques that control the orientation and the respective rotation of each wheel.

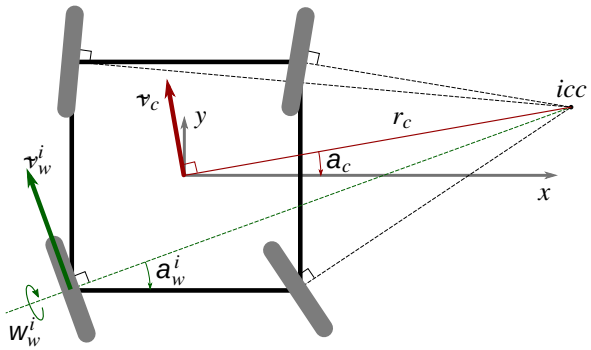


Fig. 3. Geometric model

$a_w^i(t)$ and $w_w^i(t)$ result from geometrical considerations and are presented in Eq. 1 and 2 (x_w^i and y_w^i represent the wheel position coordinates in the local frame, r_w the wheels' radius).

Assuming independence in the wheels' orientation actuation and modelling the actuators with viscous friction (parameters m_a and m_w), the wheel orientation torque becomes (I_a – inertial mass of a wheel):

$$t_a^i(t) = I_a \ddot{a}_w^i(t) + m_a \dot{a}_w^i(t) \tag{4}$$

Regarding the wheel rotation actuators, their interconnection cannot be neglected. Imposing force and torque equilibrium in the body (Eq. 5 – 8), the problem becomes solving the undetermined system (eq. 9) for computing the required torque of the wheels (m_b, I_b – platform mass and inertial mass respectively).

$$t_{w_0}(t) = \begin{bmatrix} t_{w_0}^0(t) & t_{w_0}^1(t) & t_{w_0}^2(t) & t_{w_0}^3(t) \end{bmatrix}^T \tag{5}$$

$$\mathbf{a}_i(t) = \begin{bmatrix} cos(a_w^i(t)) \\ sin(a_w^i(t)) \\ x_w^i sin(a_w^i(t)) - y_w^i cos(a_w^i(t)) \end{bmatrix} \tag{6}$$

$$\mathbf{A}(t) = \begin{bmatrix} \mathbf{a}_0(t) & \mathbf{a}_1(t) & \mathbf{a}_2(t) & \mathbf{a}_3(t) \end{bmatrix} \tag{7}$$

$$\mathbf{b}(t) = -r_w \frac{d}{dt} \begin{bmatrix} m_b v_c cos(a_c) \\ m_b v_c sin(a_c) \\ I_b w_c \end{bmatrix} \tag{8}$$

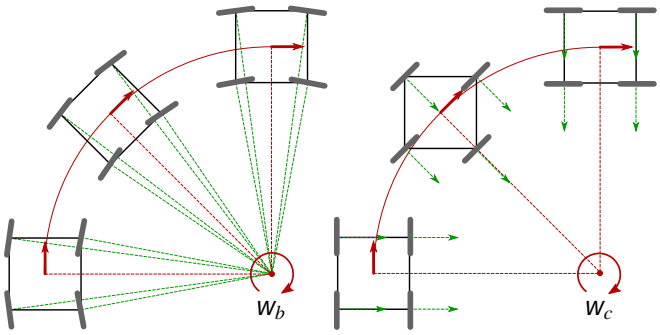
$$\mathbf{A}(t) t_{w_0}(t) = \mathbf{b}(t) \tag{9}$$

With similar modelling of the actuator, the rotational actuator torque then becomes:

$$t_w^i(t) = t_{w_0}^i(t) + m_w w_w^i(t) \tag{10}$$

C. Parametrizations and Motion Models

Due to the nonlinear nature of the geometrical configuration of an IWS, several parametrizations of its kinematic state are possible and advisable depending on further application. For example, the parametrization presented in Section IV has the advantage of remaining well defined when the base velocity tends toward 0, simplifying initial conditions and parametrization switches in the higher-level modules. However, the trade-off lies on the fact that a singularity is present when performing pure rotation. An alternative



wheel geometry ICC induced ICC

Fig. 4. Global Frame considerations; the vehicles body can rotate as well.

would be to use $\mathbf{p} = [v_c; w_b; a_c]$, ($w_b = -v_c \text{=} r_c$) and avoid the previous singularity. However, this parametrization is not defined at $v_c \to 0$. This can be exploited by the fact that when the platform stands still, the wheels do not have to be synchronized in order to achieve a different initial state and thus can rotate at full speed. However, this generally complicates the control scheme.

Other parametrizations can be used by exploiting global frame characteristics of the kinematics. More intuitively, consider the two situations presented in Fig. 4. On the left, the global frame motion of the platform with a fixed wheel geometry ICC is shown. The same trajectory can be achieved by modifying a_c while the wheels stay parallel. This creates an induced ICC and the total trajectory motion curvature can be interpreted as a superposition of the two: $w_{traj} = w_b + w_c$. However, the orientation of the platform is influenced only by w_b. This motivates the parametrization $\mathbf{p} = [v_c; w_{traj}; w_b]$ due to the parameter decoupling between trajectory and orientation of the platform, a fact that is further exploited in the Local Path-Planner in order to achieve trajectories with the platform orientation following a focal point.

The differential equation that models the kinematics of the platform is presented in Eq. 11, where all variables can be time-dependent. According to the desired accuracy, discretization is achieved through various methods (e.g. Euler or Runge-Kutta).

$$\begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{q}_b \\ \dot{r}_c \\ \dot{a}_c \end{bmatrix} = \begin{bmatrix} v_c cos(q_b + a_c) \\ v_c sin(q_b + a_c) \\ w_b \\ -v_c \text{=} w_b \\ w_c \end{bmatrix} \tag{11}$$

Considering that the optimization problem requires numerical evaluation of gradients, the model numerical stability is of importance. It is worthy to note that discretized motion models based on circular considerations are not advised, as they present numerical instability when $w \to 0$.

Having had an analysis of the I4WS platform and its low-level control, the next Section will discuss the considerations of trajectory planning using MPC.

V. LOCAL PATH-PLANNING

Especially due to the increase of the control parameter $\mathbf{p}$ space to 3, sole DWA-based sampling approaches become too expensive to evaluate, together with their reduced accuracy. This motivates the main approach towards Local Path-Planning to be based on optimization i.e. MPC. However, as described in the following, trajectory sampling still has a role in the planner in order to provide an initial estimate of the trajectory when optimization reaches a local-minima. As many of the approach choices are made while being sensitive to the optimization requirements and bottlenecks, the optimization problem will be discussed first, followed by key elements of the Local Path-Planner.

A. The optimization problem

Theoretically, finding an optimal local trajectory with respect to a cost and constraints represents a dynamical optimization problem [9], as presented in canonical form:

$$\begin{aligned} \min_{\mathbf{u}(\cdot)} \quad & J(\mathbf{u}(\cdot)) = j\,(\mathbf{x}(t_1); t_1) + \int_{t_0}^{t_1} l(\mathbf{x}(t); \mathbf{u}(t); t)dt \\ \text{subject to:} \quad & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t); \mathbf{u}(t); t); \quad \mathbf{x}(t_0) = \mathbf{x}_0 \\ & y\,(\mathbf{x}(t_1); t_1) = \mathbf{0} \\ & \mathbf{h}(\mathbf{x}(t); \mathbf{u}(t); t) \succeq \mathbf{0}; \quad \forall t \in [t_0; t_1] \end{aligned} \tag{12}$$

with the cost functional $J(\mathbf{u}(\cdot))$, the terminal cost $j\,(\mathbf{x}(t_1); t_1)$, the Lagrange density function $l(\mathbf{x}(t); \mathbf{u}(t); t)$ and the state transition function $\mathbf{f}(\mathbf{x}(t); \mathbf{u}(t); t)$. In practice, the problem is discretized, resulting in static optimization over finite space:

$$\min_{\mathbf{u} \in U} \quad f(\mathbf{x}; \mathbf{u}) \tag{13}$$

$$\text{subject to} \quad \mathbf{g}(\mathbf{x}; \mathbf{u}) \text{=} \mathbf{0} \tag{14}$$

$$\mathbf{h}(\mathbf{x}; \mathbf{u}) \succeq \mathbf{0} \tag{15}$$

with the cost function f, equality constraints $\mathbf{g}$ and inequality constraints $\mathbf{h}$.

This introduces another dimension to $\mathbf{p}$, its arc parametrization (Eq. 17). A popular choice for this parameter is time and is usually assumed constant (e.g. time between control-points). As the optimization problem requires at least the numerical evaluation of gradients, our approach has opted towards reduction of the number of parametrized control points by optimizing the trajectory arc-length parameter. Even though most of the approaches take time as a parametrization of choice [9], a considerable portion of the optimization problem for navigation relies on spatial information. By choosing sparse arc-length parametrization (i.e. distance along trajectory), optimization convergence speed is significantly ($\sim 100\%$) increased.

Considering the piecewise continuous differential character of the system required in the optimization problem of the Local Path-Planner, the kinematics of the parametrized state suffice a constant parameter velocity model (Eq. 16). Even though higher order splines represent an expressive method of parametrizing the control points, due to their difficulty of arc-length computation, trapezoidal interpolation is used (Fig. 5 – left).

$$\mathbf{p}^i(t) = \mathbf{p}_0^i + \dot{\mathbf{p}}^i\, t; \quad t - \text{bound by } p_{arc}^i \tag{16}$$

$$\mathbf{u} = \begin{bmatrix} p_0 & p_1 & p_2 & p_{arc} \end{bmatrix}^T; \quad p_{arc} \in \{\Delta t; \Delta s\} \tag{17}$$

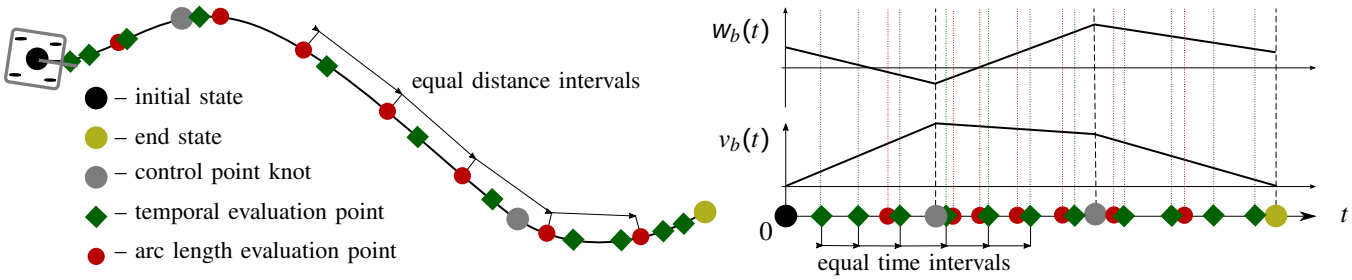


Fig. 5. Trajectory parametrization

B. Trajectory simulation and evaluation

As most of the motion models used in navigation are nonlinear, evaluation of trajectories implies numerical simulation. Because of this, trajectory simulation (which is used in most of the path-planning approaches [10]) represents a considerable bottle-neck. Thus, a critical point to developing an efficient path planner revolves in efficiently computing and evaluating trajectories as well as balancing desired runtime versus simulation accuracy.

Due to the model kinematics nature, the typical constraints are nonlinear and in general finding a piecewise closed-form approximation of them is unfeasible. Because of this, a relaxed assumption of sufficiently small evaluation intervals is typically used. Fig. 5 – right illustrates a simulated trajectory and its relevant evaluation points. As costs and constraints present in the problem have dynamic as well as spatial nature (e.g. dynamic motor constraints or distance to obstacles), two evaluation lattices are used, one at constant time intervals and one at constant arc-length. This way, together with an analysis of the constraint nature, constraint satisfaction becomes tractable. Good definitions of cost-functions influence heavily the characteristics of the generated trajectories and can lead to unexpected local minima if not chosen carefully. For example, penalizing distance to the end-point might seem intuitive, however it will get stuck at sharp corners of the route boundary. A better approach is to weight the distance along path (or traveled distance), especially for configurations with relatively small route margins. To summarize, the main costs used in this approach due to the varied characteristics and non-intrusion (they do not typically form local-minima between themselves) are:

- distance to route (integral/power cost)
- traveled distance (integral/power cost)
- base orientation (integral/power cost)
- trajectory time or average velocity (end cost)
- distance to desired end-pose (end cost, small weight)

Most of the constraints in this approach are inequality constraints (the internal kinematic equality constraints are not needed due to the lower-level parametrization). The only equality constraint enforces zero base velocity at the end of the optimized trajectory. This way, the platform is guaranteed to not collide with obstacles even when moving at high speeds. As previously mentioned, the inequality constraints are of different natures:

- maximum distance to route (spatial, nonlinear)
- minimum distance to obstacles (spatial, nonlinear)
- actuator dynamics (temporal, nonlinear for IWS)
- base dynamics (temporal, varies with parametrization)
- maximum prediction time (affine)
- arc-parametrization non-negativity (affine)

Having already introduced motion-models and costs, algorithm 1 presents the pseudo-code of the trajectory simulator with on-line evaluation. Starting from the initial state $\mathbf{x}_0$, the simulation computes the time interval until the next evaluation point and advances according to the used motion model (implemented in $sim\_step(\mathbf{x};\mathbf{u}[];dt)$). Depending on the type of the evaluation point, the cost/constraint evaluator computes and stores the afferent data.

Especially in the case of non-linear constraints, the number of evaluations per constraint type has to be sufficiently large (otherwise one might be unable to evaluate a high constraint violation). An alternative for reduced optimization constraint space is to perform soft-min of the constraint costs over a larger interval (e.g. between two parametric knots) and provide only one entry per interval in the optimization formulation. Note that the piece-wise character of the used parametric functions can furthermore motivate placing cost evaluation and constraints at their knots (one might identify piecewise convex formulation of costs/constraints).

When reaching the end of simulation, the partial precomputed costs/constraints are finalized. Note that such an approach is not only parallelizable with respect to the simulated trajectories but also allows immediate interruption in case of constraint violation, action that is used for speedup and efficient pruning of trajectories sampling, as future described in Subsection V-C.

Algorithm 1 $simulate\_trajectory(\mathbf{x}_0;\mathbf{u}[])$

```
t ← 0; x ← x_0; evaluator.init()
while t < t_end do
    (dt; eval_type) ← find_next_step(u[]; t)
    x ← sim_step(x; u[]; dt);  t ← t + dt
    switch (eval_type)
    case dt_gitter:  evaluator.evaluate@partial_time(x; t)
    case ds_gitter:  evaluator.evaluate@partial_dist(x; t)
    case u_i_end:  evaluator.evaluate_last_u_interval(x; t)
    case sim_end:  evaluator.finish_evaluation(x; t)
    end switch
end while
```

Room for computational cost improvement is present also in the computation of objective function and constraints

gradients, by beginning the simulation and evaluation only from the modified region.

C. Trajectory sampling

Considering that optimization typically requires an initial estimate, the planner design includes sampling as a preliminary step to optimization. This is performed to cope with the prowess of optimization-based approaches to local-minima. Even though during most of the cycles, the previous trajectory is sufficient as a starting solution, sampling can also provide a layer of redundancy by sampling at every cycle until a safe trajectory to a platform stop is found. However, total space sampling is infeasible especially for I4WS as the parameter space is cubic.

In a similar fashion with the traditional DWA approach, the parameter space is sampled. However, non-expressive parameter combinations are to be avoided as most of the computation time is reserved for optimization. Another benefit of the parametrization $\mathbf{p} = [v_c; w_{traj}; w_b]$ lies on the fact that if not taking into account constraints, only two parameters are influencing the base trajectory, effectively reducing the sample space for emergency situations.

D. Preprocessing of external data

As previously presented in Section III, the Local Path-Planner requires external data in the form of a route, environment information from a depth-sensor (and additionally static information from a map) as well as the localization information of the agent. The methods of obtaining a global route as well as the localization information are a field of research themselves and are beyond the purpose of this paper. Regarding localization, the proposed Path-Planner assumes it is sufficiently accurate but not necessarily temporally synchronized with the rest of the data (except being time-stamped).

Most on-line global route planners return the shortest euclidean path to the goal. Keeping in mind the dynamics and constraints of mobile agents as well as their desired trajectory behaviours, accurate following of such paths becomes contradictory. As a solution, an additional parameter is defined, which specifies the maximum allowed deviation from the route. This addition not only increases the search-for-optimum space but also allows for a dramatic reduction of the required vertices of the route graph (envision an A* algorithm generated route diagonally on a coarse grid). Especially for mapped areas, image processing techniques for space skeletization using line-of-sight heuristics/techniques could be implemented for efficient and flexible route planning [11]. The route is thus consisted of line-connected consecutive points of arbitrary distance, the desired end-pose being represented by the last point and its orientation.

As we are considering (sufficiently) planar surface navigation, the depth sensor information as well as the map is converted to a 2D representation. In practice, distance fields of the relevant spatial information are generated and stored in separate layered maps. This way, data evaluation can be done in constant time, a critical factor in the terms of the run-time of the optimizer. As piece-wise continuity and differentiability is required in nonlinear optimization, it is also advised to perform interpolation on the maps data (e.g. bi-linear).

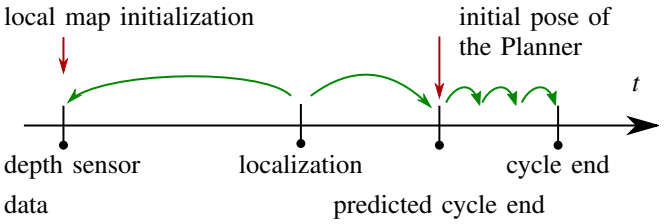


Fig. 6. Pose prediction during one cycle of the planner

By imposing the route maximum deviation constraint, a big part of the obstacle data becomes redundant and is excluded from all the other computations (no distance fields are created for obstacles far away from the route).

E. Temporal synchronization

Especially in asynchronous communication systems such as Robotics Operating System (ROS), proper data synchronisation is vital and must be accounted for. Fig. 6 presents the general temporal sequence of data acquisition and processing. In principle, given the previous sequence of control commands sent by the planner to the low-level controller, the pose of the agent is temporally synchronized with various data. Initially, the time delay between the depth sensor measurement and the localization measurement is compensated by simulating the platform state to the time-stamp of the sensor data. Notice that this can be also done in reversed time.

Regarding the communication with the low-level controller, an important point represents the usage of parametrized control commands arrays. This way, the planner sends to the low-level controller its computed controls for the next n - seconds, enabling non-deterministic cycle times to be feasible as well as potentially increased safety with respect to planner run-time malfunction.

As the planner cannot influence any platform physical state until its cycle is over, its pose is predicted from the beginning to that temporal point (which varies typically between 0.1s - 1s). Even though the optimizer is inherently iterative, in many situations it is preferred to overshoot the desired cycle time for achieving optimization convergence. In this sense, assuming that the time interval of an optimization cycle is relatively small with respect to the platform motion, additional correction through forward prediction of the initial pose before each optimization iteration can be achieved.

VI. EXPERIMENTAL RESULTS

After an introduction to the implementation specifics, the modelling and simulation of the robot "*Blue*" is being introduced. Simulation results are then presented, showcasing the quality of the developed low-level control scheme as well as the capabilities of the proposed Local Path-Planner.

The implementation of the presented work has been done in C++, making use of GazeboSim as simulation environment and the Robotics Operating System (ROS). Considering the

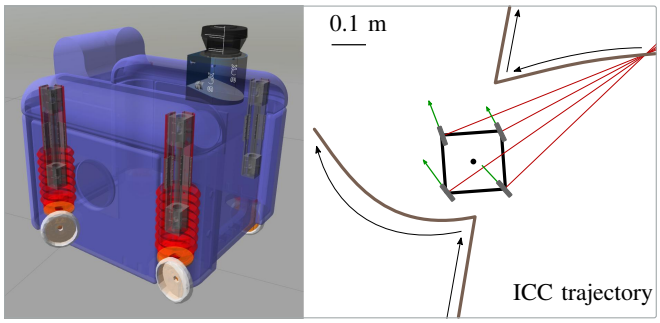


Fig. 7. Left: Simulated base of robot *Blue*, Right: ICC trajectory in local frame of the platform

development of a low-level controller, increased attention has to be payed to the simulated system. Fig. 7 – left illustrates the simulation of "*Blue*"s mobile base, making use of CAx models of the components as specified by the designer, together with component inertial information. General purpose surface friction parameters have been applied accordingly. Without loss of generality, the actuators are modeled taking into account damping and the interface has been designed to apply shaft torques as input. As 3D physics engines implemented in Gazebo (such as ODE, Bullet or Dart) perform typically poor in over-constrained closed kinematic chains (such as a I4WS) when force control is used, increased simulation accuracy is achieved through parameter-tuning of the engine (increased iteration count as well as relaxation of the stiffness of the formed contact joints). Additionally, quantisation noise is modelled for the angular sensors of the actuators as well as Gaussian noise of the depth sensor.

The developed feed-forward control-scheme, together with the LQG feed-back has been implemented as a ROS-nodelet and has been tested at a frequency of 100 Hz, using simulation ground-truth for validation. For simplicity of data interpretation, the parametrization $\mathbf{p} = [v_c \;\; r_c \;\; a_c]$ is used. Fig. 7 – right presents the local view of the platform as it performs a sequence of control commands. For visualization purposes, the entire parameter trajectory has been computed initially; however, each cycle computation is performed online during normal operation. Desired versus applied parameter and wheel velocity trajectories are illustrated in Fig. 8 – left, showcasing the accuracy of the proposed controller. In Fig. 8 – right, the temporal evolution of the orientation as well as rotation speed of the top-right wheel is illustrated. As expected, when the ICC revolutes at a small radius relative to a wheel (big values of r_c), the nonlinear character of the wheel motion becomes significant.

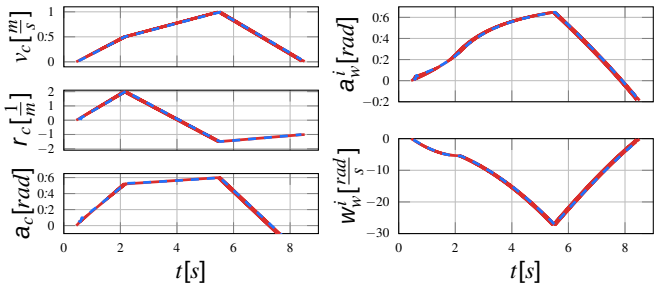


Fig. 8. Desired (red) vs. Measured (blue) data: left - ICC parameters, right - orientation and velocity a wheel

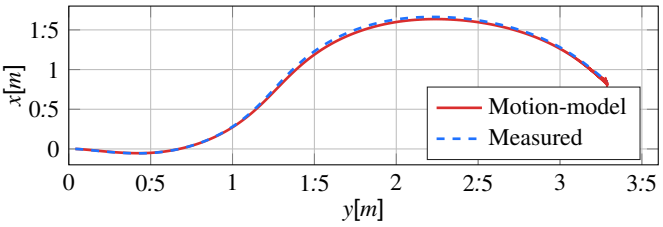
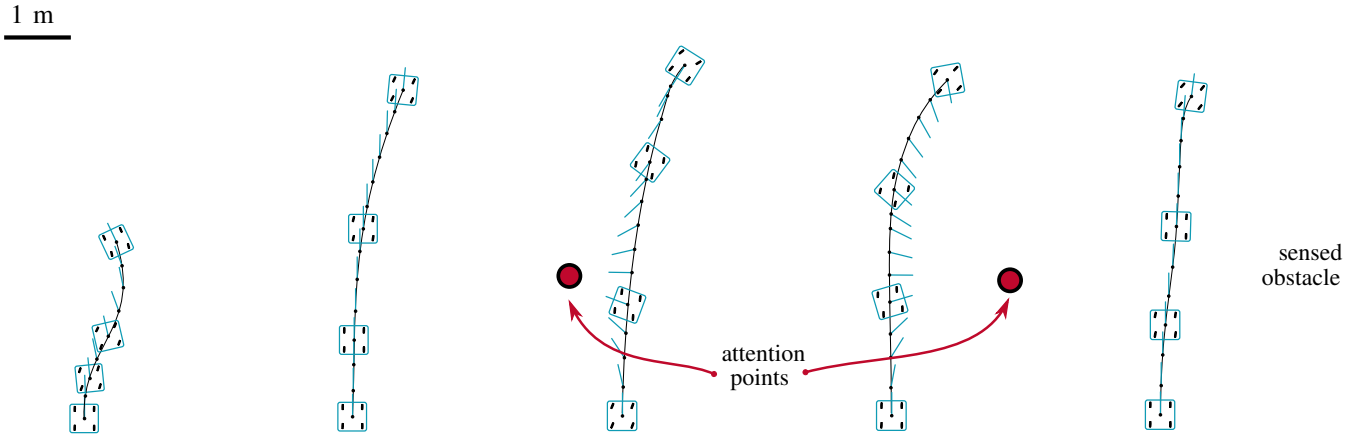


Fig. 9. Global frame trajectory of the platform

Even though the results in the local frame are accurate, the global-frame motion of the platform compared with the motion-model used in the Local Path-Planning module is of importance. The presented motion model has been discretized using trapeze interpolation and evaluated every 10 Hz. As it can be seen in Fig. 9, the motion-model used by the Local Path-Planner accurately predicts the motion that is eventually undertaken by the agent up to several meters.

The Local Path-Planner has been implemented in C++, being dependant only on the low-level controller, ROS (for communication purposes), OpenCV as well as the optimization library *Optizelle*. The motivation behind the choice of the optimization library is its efficiency, State of the Art implemented algorithms as well as matrix-free design. In practice, the optimization problem is solved using the Interior-Point Method with SR1 Hessian estimation, Krylov sub-problem solver and affine line search for the inequality constraints (through problem reformulation). The planner typically runs stable at a frequency of 10 Hz on single core of an Intel i7 processor. However, as discussed, the approach is designed to be robust at lower cycle-frequencies as well.

Fig. 10 presents some of the behaviours that can be obtained by the Local Path-Planner using only three control points. The active route for this instance is represented by the green dotted line and its afferent distance-field is created. The route far away is not considered locally and is in this instant inactive (blue). In the presented scenarios, the parametrization $\mathbf{p} = [v_c \;\; w_{base} \;\; w_c]$ has been used as it performs superior to other parametrizations when attention-focus points are targeted. The agent has an initial velocity of 0.1 m/s and for all trajectories, the end-constraint of 0 base velocity is imposed. Furthermore, wheel actuator orientation, velocity and acceleration as well as platform base linear and lateral acceleration constraints have been enforced. Fig. 10 *a*) presents the scenario where route accuracy is desired. As the number of control points is significantly low, high path-following accuracy implies a reduction of the trajectory length. Fig. 10 – *b*) presents the generated trajectory with dominant costs towards time-optimality. Notice that the obstacle represents the only active spatial constraint. Fig. 10 – *c*) and – *d*) showcase the same scenario when an attention-focus point (to the left and right respectively) is targeted. The increased duration of the generated trajectory *d*) is mostly due to the internal actuator constraints of such motion. Lastly, Fig. 10 – *e*) illustrates the situation where maximum path deviation is reduced. Notice here the sensed obstacle distance field kernelization further away from the path, as the path constraint makes that region automatically infeasible.



a) close follow
t = 4:76s; s = 3:31m

b) fast follow
t = 4:822s; s = 5:851m

c) attention point
t = 8:432s; s = 6:121m

d) attention point
t = 10:06s; s = 5:913m

e) reduced path width
t = 4:726s; s = 5:431m

Fig. 10. Generated trajectories of the Local Path-Planner under different dominating weights

VII. CONCLUSIONS

This work presented a two-layer control framework for an I4WS platform capable to smoothly control a vehicle along a given path while enforcing constrains of various natures, including platform body orientation.

Initially, an overview of the addressed modules within the autonomous navigation context has been presented. Afterwards, local and global frame analysis of I4WS kinematics and dynamics has been done in order to identify beneficial local frame parametrizations of I4WS motion. To this end, a flatness-based low-level control has been designed.

In the second part of this paper, Local-Path Planning within the context of autonomous navigation has been addressed. The proposed optimization-based approach is able to deal with various constraints of different natures (spatial, temporal, dynamic) and showcases different motion behaviours, intended to be of use to higher-level robotics tasks while maintaining simplicity of interaction (through a small set of parameters). Moreover, by careful design that targets bottle-necks, physics-engine simulations prove the capabilities and efficiency of the approach, constituting together with the low-level controller a Local Path-Planning Framework for I4WS systems.

On the research side, further investigation is intended on the topic of behaviour control on higher level modules. A generalized approach towards active use of expressive navigation will address many inconsistencies and difficulties, especially in the field of assistive robotics. Also, the mixture of optimization-based Path-Planning and layered maps with fast queries motivates further investigation of multi-agent planning, potentially providing an elegant solution especially in the context where agents posses inter-communication [12].

On the application side, the model of the robot is planned to be published to the community, providing a carefully designed model of I4WS for further development on related topics. Last but not least, the presented work is to be implemented in the robot *Blue*, enabling extensive testing in demanding operating conditions towards smoother and more natural Human-Robot interaction.

REFERENCES

[1] C. P. Connette, C. Parlitz, M. Hagele, and A. Verl, "Singularity avoidance for over-actuated, pseudo-omnidirectional, wheeled mobile robots," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 4124–4130.

[2] C.-J. Lin, S.-M. Hsiao, Y.-H. Wang, C.-H. Yeh, C.-F. Huang, and T. H. S. Li, "Design and implementation of a 4ws4wd mobile robot and its control applications," in *System Science and Engineering (ICSSE), 2013 International Conference on*, July 2013, pp. 235–240.

[3] P. Shen, Y. Fang, and X. Zhang, "Trajectory planning of omnidirectional mobile robots with active casters: Multi-motor coordination and singularity avoidance," in *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on*, June 2015, pp. 151–156.

[4] S. Y. Jiang and K. T. Song, "Differential flatness-based motion control of a steer-and-drive omnidirectional mobile robot," in *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on*, Aug 2013, pp. 1167–1172.

[5] T. Howard, C. Green, D. Ferguson , and A. Kelly , "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 325–345, June 2008.

[6] T. Howard, "Adaptive model-predictive motion planning for navigation in complex environments," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2009.

[7] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 2061–2067.

[8] P. Dorato, V. Cerone, and C. Abdallah, *Linear Quadratic Control: An Introduction*. Melbourne, FL, USA: Krieger Publishing Co., Inc., 2000.

[9] B. Chachuat, *Nonlinear and Dynamic Optimization: From Theory to Practice - IC-32: Spring Term 2009*, ser. Polycopiés de l'EPFL. EPFL, 2009. [Online]. Available: https://books.google.at/books?id=_JOHYgEACAAJ

[10] M. B. Markus Suchi and M. Vincze, "Meta-Heuristic search strategies for Local Path-Planning to find collision free trajectories," in *Proceedings of the Austrian Robotics Workshop (ARW-14)*, May 2014, pp. 36–41.

[11] Q. Wang, M. Wulfmeier, and B. Wagner, "Voronoi-based heuristic for nonholonomic search-based path planning," in *Intelligent Autonomous Systems 13*, ser. Advances in Intelligent Systems and Computing, E. Menegatti, N. Michael, K. Berns, and H. Yamaguchi, Eds. Springer International Publishing, 2016, vol. 302, pp. 445–458.

[12] M. Bader, A. Richtsfeld, M. Suchi, G. Todoran, W. Holl, and M. Vincze, "Balancing Centralised Control with Vehicle Autonomy in AGV Systems for Industrial Acceptance," in *Proceeding of the Eleventh International Conference on Autonomic and Autonomous Systems (ICAS 2015)*, 2015.