Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-2 (2020) 15146–15153

Control of interacting vehicles using model-predictive control, generalized Nash equilibrium problems, and dynamic inversion

Andreas Britzelmeier * **Matthias Gerdts** **
Thomas Rottmann ***

* *Bundeswehr University, Munich, 85577 Germany (e-mail: andreas.britzelmeier@unibw.de).*
** *Bundeswehr University, Munich, 85577 Germany (e-mail: matthias.gerdts@unibw.de).*
*** *Bundeswehr University, Munich, 85577 Germany (e-mail: thomas.rottmann@unibw.de).*

Abstract: The paper presents a control concept for interacting vehicles in a road network. The approach combines a high-level controller for the generation of collision-free trajectories and a low-level dynamic inversion controller for path tracking. The high-level controller uses model-predictive control for generalized Nash equilibrium problems, which are used to coordinate the vehicles. The control concept was implemented and validated on scale robots and experimental results are discussed.

Keywords: multi-vehicle systems, motion planning and decision making, dynamic games, model predictive and optimization-based control, trajectory tracking and path following

1. INTRODUCTION

Numerous applications arise from the task of conflict avoidance employing multiple autonomous robotic vehicles. This includes intersection management, platooning or general path planning. The common ground of these problems motivates the formulation of coordination and path-following problems, where the realization of the numerical solutions on real robots introduce additional difficulties regarding communication and imperfect information. To this end, the task is to design a control concept which is able to coordinate the vehicles on their respective tracks and to resolve conflicts. In addition, a reliable and stable controller is required to track paths in combination with a communication topology that ensures the distribution of dependable information among the vehicles with low latencies.

There are various ways to approach this class of problems or parts of it. In Alessandretti and Aguiar (2019) the coordination problem is tackled with a model predictive control approach, where a coordinated output regulation problem is solved to meet a predefined consensus. Further, the vehicles do not share their predictive trajectories but their coordination vectors. However, no implementation and tests are conducted on real nonholonomic robots. Hult et al. (2018) propose a mixed integer approach to solve the scheduling problem occuring at intersections by computing time slots for each vehicle. Further, in Zanon et al. (2017) an algorithm with asynchronous sensitivity updates to minimise the time lost in communication is presented. Another approach to solve the intersection management problem is to compute state dependent hierarchies for the conflicting vehicles by employing a model predictive control approach, see Britzelmeier and Gerdts (2018).

Most of the aforementioned approaches use predefined hierarchies. An alternative concept is motivated by game theory. In this setting, the vehicles are considered as individual players trying to achieve an equilibrium. The resulting differential game usually is one of three types: cooperative, antagonistic, or a leader-follower game. A game theoretic approach is associated with a more dynamic behaviour and greater efficiency compared to a priori defined rules. In this paper we employ the decomposition algorithm proposed in Britzelmeier et al. (2019); Britzelmeier and Dreves (2019). The vehicles exchange information about their planned paths. Then, the conflict is resolved by formalising a nonlinear model predictive control framework, where in each step every player has to solve a generalized Nash equilibrium problem (GNEP). However, the problems are coupled by a nonconvex anti-collision constraint. The coupled problem is reformulated into a generalized potential game (GPG) by exploiting a penalty approach. An extensive convergence proof can be found in Britzelmeier and Dreves (2019). For further information on GNEP and potential games, we refer the interested reader to Ba and Pang (2018); Facchinei et al. (2011); Facchinei and Kanzow (2010); von Heusinger and Kanzow (2009); Monderer and Shapley (1996); Rosenthal (1973); Rosen (1965). For implementation and experimental validation we use nonholonomic robots equipped with a microcomputer. For the positioning we employ the HTC Vive System. The information exchange is designed as a vehicle-in-the-cloud system, where information and plan-

10.1016/j.ifacol.2020.12.2080

ning is executed online in a central cloud. Furthermore, we propose an improved version of a dynamic inversion controller, which was primarily introduced in Burger and Gerdts (2019), to track the planned paths of the vehicles. We further provide a stability proof of this controller.

The paper is structured as follows. In Section 2 we introduce the problem setting and formalise the GNEP as well as the GPG. Section 3 gives a short overview of the decomposition algorithm to solve the conflict problem. The dynamic inversion controller is presented in Section 4, as well as the stability proofs of the error dynamics and the internal dynamics. The implementation and the experimental setup is explained in Section 5, followed by a comparison of computed and measured results in Section 6.

2. SETTING

We model a common traffic scenario, where we consider a given road network on which N cars aim to reach their desired destination, compare Figure 7 in Section 6. Herein, the objective of each car is to reach the respective destination in minimal time, whilst avoiding collisions with other traffic participants. As in usual traffic each car has a point of departure and a point of destination, which are connect by a route, acquired from, e.g, a navigation system. Hence, we a priori assign a path to each vehicle on which it moves towards to its destination. These paths are modelled as cubic periodic splines,

$$\gamma^\nu : [0, L^\nu] \mapsto \mathbb{R}^2, \quad \gamma^\nu(s) \coloneqq \begin{pmatrix} x^\nu(s) \\ y^\nu(s) \end{pmatrix}$$

where $\nu = 1, \dots, N$, L^ν is the path's length and s designates the arclength along the path. The coordination of the vehicles' motion, i.e., the choice of their velocity profiles on the paths in the road network, is resolved by a non-linear model-predictive control (NMPC) approach with an embedded GNEP on time horizons of length T. Herein, the optimal controls are computed as a solution to the GNEP, i.e., for each player on the stated time horizon. Then the controls are applied for some interval $\tau < T$. Further, the time horizon is shifted by τ and the GNEP is solved again for the shifted time horizon $[\tau, \tau + T]$. This way the computed velocity profiles can be adapted in every NMPC step and a high-level feedback controller is realized.

In the game theoretical sense, the vehicles are considered as players and the coordination problem is the game, in which these players participate. Let us define the GNEP, for simplicity, on the interval $[0, T]$ for players $\nu = 1, \dots, N$. The controls (accelerations) $u = (u^1, \dots, u^N)$, with $u^\nu : [0, T] \to \mathbb{R}, \nu = 1, \dots, N$, are measurable functions. Further, we define the velocity $v^\nu : [0, T] \to \mathbb{R}$, and the arclength, $s^\nu : [0, T] \to \mathbb{R}$, for all $, \nu = 1, \dots, N$, which describe our states and are absolutely continuous functions in $W^{1,1}([0, T])$. Let $\mathcal{J}^\nu$ denote the index set of all players $\mu \in \{1, \dots, N\} \backslash \nu$, which have intersecting paths with player ν. For $\mu \in \mathcal{J}^\nu$ let $P^{\nu\mu}$ denote the set of tupels (p, q), where p and q are the arclengths relative to the paths of player ν and μ, respectively, for a common intersection point of the two paths. Since the paths are known a priori, the intersection points and their respective arclengths in $P^{\nu\mu}$ can be precomputed. Let us introduce the function

$$g_{\nu\mu}(a, b) \coloneqq d_{\nu\mu} - \max\{|a|, |b|\}$$

with a safety radius $d_{\nu\mu} > 0$ for $\mu \in J^\nu$.

Each player $\nu = 1, \dots, N$ then has to solve the following optimal control problem on the time horizon $[0, T]$ to find a velocity profile:

$$\begin{aligned} \min_{(u^\nu, v^\nu, s^\nu)} \; & -s^\nu(T) \\ \text{s.t.} \; & v^\nu(0) = v_0^\nu, \quad s^\nu(0) = s_0^\nu, \\ & (v^\nu)'(t) = u^\nu(t) - c_1 v^\nu(t) - c_2 v^\nu(t)^2, \\ & (s^\nu)'(t) = v^\nu(t), \\ & v^\nu(t) \in [v_{\min}^\nu, v_{\max}^\nu], \\ & u^\nu(t) \in [u_{\min}^\nu, u_{\max}^\nu], \\ & v^\nu(t) \leqslant v_{\max}^\nu(s^\nu(t)), \\ & g_{\nu\mu}(s^\nu(t) - p, s^\mu(t) - q) \leqslant 0, \; \forall (p, q) \in P^{\nu\mu}, \mu \in J^\nu. \end{aligned} \tag{GNEP}$$

Herein, $c_1 \geqslant 0$ and $c_2 \geqslant 0$ are given constants. Let $v = (v^1, \dots, v^N)$, $s = (s^1, \dots, s^N)$, $u = (u^1, \dots, u^N)$ be combined in $z = (z^1, \dots, z^N)$ with $z^\nu = (v^\nu, s^\nu, u^\nu)$, $\nu = 1, \dots, N$.

In the NMPC, (GNEP) is discretized by applying an explicit Euler method with step-size $h > 0$, such that $t_i = i \cdot h$ with $i \in \mathbb{N}$. Throughout the paper the index h is used to emphasize the discretized system. Then the strategy space for all players $\nu = 1, \dots, N$ is

$$\begin{aligned} \Xi_\nu^h \coloneqq \{ & (u_h^\nu, v_h^\nu, s_h^\nu) \mid v_h^\nu(0) = v_0^\nu, \quad s_h^\nu(0) = s_0^\nu, \\ & \text{and for all } i = 0, \dots, M-1 \text{ and } t \in [t_i, t_{i+1}] : \\ & v_h^\nu(t) = v_h^\nu(t_i) + (t - t_i)(u_h^\nu(t_i) - c_1 v_h^\nu(t_i) - c_2 v_h^\nu(t_i)^2) \\ & s_h^\nu(t) = s_h^\nu(t_i) + (t - t_i) v_h^\nu(t_i) \\ & v_h^\nu(t_{i+1}) \in [v_{\min}^\nu, v_{\max}^\nu], \\ & u_h^\nu(t_i) \in [u_{\min}^\nu, u_{\max}^\nu] \cap \{u_{\min}^\nu + j h_u^\nu \mid j = 0, \dots, M_u^\nu\}\}. \end{aligned} \tag{1}$$

We introduce a generalized potential game. To this end, we define a penalty function, where the penalty term holds the non-convex constraints, as well as the velocity constraints,

$$\begin{aligned} \Lambda_h(v_h, s_h) \coloneqq \sum_{\nu=1}^{N} \sum_{i=0}^{M-1} \Bigg(& \max\{0, v_h^\nu(t_{i+1}) - v_{\max}^{\nu,h}(s_h^\nu(t_i))\} \\ & + \frac{1}{2} \sum_{\substack{(p,q) \in P^{\nu\mu} \\ \mu \in J^\nu}} \max\{0, g_{\nu\mu}(s_h^\nu(t_{i+1}) - p, s_h^\mu(t_{i+1}) - q)\} \Bigg). \end{aligned} \tag{2}$$

With (1) and (2), we define the discrete generalized potential game

$$\begin{aligned} \min_{(u_h, v_h, s_h)} \; & -\sum_{\nu=1}^{N} s_h^\nu(T) + \rho \, \Lambda_h(v_h, s_h) \\ \text{s.t.} \; & (u_h, v_h, s_h) \in \bigtimes_{\nu=1}^{N} \Xi_\nu^h. \end{aligned} \tag{GPG$_\text{h}$}$$

where $\rho = \frac{1}{(h + h_u)^\beta}$, with h_u the step-size of the discretized control and some constant $\beta < 1$. We further extent the problem by adding a regularization term for the control, to acquire applicable and smooth trajectories. Hence, (GPG$_\text{h}$) becomes,

$$\begin{aligned} \min_{(u_h, v_h, s_h)} \; & -\sum_{\nu=1}^{N} s_h^\nu(T) + \rho \, \Lambda_h(v_h, s_h) + \alpha \sum_{\nu=1}^{N} \sum_{i=0}^{M-1} u_h^\nu(t_i)^2 \\ \text{s.t.} \; & (u_h, v_h, s_h) \in \bigtimes_{\nu=1}^{N} \Xi_\nu^h. \end{aligned} \tag{eGPG$_h$}$$

A theoretical investigation without the regularization term can be found in Britzelmeier and Dreves (2019). Finally, (eGPG$_h$) can also be considered as a multi-objective optimal control problem and applying the method in Kaya and Maurer (2014) yields,

$$\min_{(u_h,v_h,s_h)} \max\{-\gamma_1 \sum_{\nu=1}^{N} s_h^\nu(T), \gamma_2 \Lambda_h(v_h,s_h), \gamma_3 \sigma(u)\}$$
$$\text{s.t.} \quad (u_h,v_h,s_h) \in \bigtimes_{\nu=1}^{N} \Xi_\nu^h, \tag{MOOCP}$$

with $\gamma_1+\gamma_2+\gamma_3=1$, $\gamma_1,\gamma_2,\gamma_3 \in [0,1]$, and

$$\sigma(u_h) \coloneqq \sum_{\nu=1}^{N} \sum_{i=0}^{M-1} u_h^\nu(t_i)^2.$$

We observed that this formulation leads to trajectories at the limit of the constraints. This shall be investigated numerically and practically in this paper.

3. DECOMPOSITION ALGORITHM

In this section, we briefly outline the decomposition algorithm which is employed to solve (GPG$_h$). An extensive convergence proof for this problem as well as for the algorithm can be found in Britzelmeier and Dreves (2019). We do not impose a prescribed hierarchy of players, which is usually introduced by employing iterative methods to solve GPGs. Solving the generalized potential game comes down to solving a penalized standard Nash game. Due to the non-convexities moved into the penalty term, we can avert difficulties, which would arise from nonempty feasible sets.

To simplify the notation for the algorithm, we drop the fixed h in our notation. We further set $f_\nu(z^\nu) = -s_h^\nu(T)$ and let the strategies of all opposing players be $z^{-\nu}$.

Decomposition Algorithm with Penalty Selection [DAPS]

(S.0) Choose a starting point $z_0 = (z_0^1,\dots,z_0^N) \in \bigtimes_{\nu=1}^{N} \Xi_\nu$ and set $k := 0$,
$$(c(1),\dots,c(N)) = (0,\dots,0),$$
$$(q(1),\dots,q(N)) = (0,\dots,0).$$

(S.1) If $(q(1),\dots,q(N)) = (1,\dots,1)$: STOP.

(S.2) Compute
$$A_k := \{\nu \in \{1,\dots,N\} \mid q(\nu) \neq 1, \; \Lambda^\nu(z_k) \geqslant \Lambda^\mu(z_k) \quad \forall \mu = 1,\dots,N\}.$$
Choose a $\nu_k := \operatorname{argmax}_{\nu \in A_k} c(\nu)$.

(S.3) For $\nu = \nu_k$ compute a global solution z_{k+1}^ν of (GPG$_h$)

(S.4) Set
$$z_{k+1} = \begin{cases} z_k, & \text{if } f_\nu(z_k^\nu) + \rho\Lambda^\nu(z_k^\nu, z_k^{-\nu}) \\ & = f_\nu(z_{k+1}^\nu) + \rho\Lambda^\nu(z_{k+1}^\nu, z_k^{-\nu}), \\ (z_{k+1}^\nu, z_k^{-\nu}), & \text{else.} \end{cases}$$

(S.5) If $z_{k+1} \neq z_k$, set $(q(1),\dots,q(N)) = (0,\dots,0)$. Set $q(\nu_k) = 1$,
$$c(\nu) = \begin{cases} c(\nu)+1, & \text{for all } \nu \neq \nu_k, \\ 0, & \text{for } \nu = \nu_k, \end{cases}$$
$k \leftarrow k+1$, and go to (S.1).

In (S.3) the selected player solves his optimization problem by utilization of a dynamic programming approach, which has the advantage of giving us global optimal solutions. Then, if the acquired solution constitutes an improvement with respect to the penalty contribution, the new solution z_{k+1} is adopted. If the contrary is the case, the previous iterate remains unchanged, i.e., $z_{k+1} = z_k$. The player to optimize is chosen by his contribution to the sum of all penalties. Hence, the one who contributes the most is selected. Moreover, the players who optimized are tracked by a logical index set q, where the index of the previously optimizing player is set to $q(\nu_k) = 1$. Eventually, the player which was chosen in the previous iteration is exempt in this turns selection process. If an improvement is made in one iteration the tracking set q is reset in (S.5). In (S.2), if A_k is no singleton, then the player with the most idle turns is selected. This information is stored in the counter set c, which at the end of each iteration increments all players indices by one, except the optimizing players one, which is reset to zero. The algorithm stops, if after $N-1$ iterations no improvements are made, i.e., in a finite number of iterations.

For (MOOCP) we proceed analogously, where the terms of the penalized objective functions in (S.4) are replaced by the respective maximum terms.

4. DYNAMIC INVERSION CONTROLLER

So far, the NMPC-GNEP approach serves as a high-level controller that coordinates the motion of the vehicles on their prescribed paths. For a practical realization we need a tracking controller that keeps the vehicles on said paths. The proposed controller operates on a lower level than the NMPC-GNEP based planning algorithm. The lower level controller is designed to track the curvature of the spline curves characterizing these paths. The control law is based on the inversion of the dynamics of a point mass vehicle model, defined in curvilinear coordinates, see Lot and Biral (2014). Due to the limited scope of this paper we refer the interested reader to Rotella et al. (2001); Martin et al. (2002) and Fliess et al. (1995) for further information on flatness based controllers. Let us introduce the point mass motion model in curvilinear coordinates relative to a reference path given by a periodic cubic spline $\gamma_m : [0,L] \mapsto \mathbb{R}^2$, compare Figure 1:

$$\begin{aligned} s'(t) &= \frac{v(t)\cos\chi(t)}{1-r(t)\cdot\kappa_m(s(t))}, \\ r'(t) &= v(t)\sin\chi(t), \\ \chi'(t) &= \psi'(t) - \frac{\kappa_m(s(t))v(t)\cos\chi(t)}{1-r(t)\cdot\kappa_m(s(t))} \end{aligned} \tag{3}$$

with $\chi(t) = \psi(t) - \psi_m(t)$, $\psi'(t) = v(t)\kappa(t)$. Herein, κ denotes the curvature of the driven path, κ_m is the curvature of a given reference path, ψ is the yaw angle of the vehicle's driven path, and ψ_m is the yaw angle of the reference path. The offset of the vehicle to γ_m is denoted by r at the arclength s. Throughout, the index m refers to a reference value, and the index d to a desired value. In our experiments we use a mobile robot with two driven wheels on the left and the right, compare Figure 3. Hence, we have

$$v(t) = \frac{v_r(t)+v_l(t)}{2}, \quad \psi'(t) = \frac{v_r(t)-v_l(t)}{w_c},$$

with $v_l(t)$, $v_r(t)$ the velocities of the left and right wheel and w_c the width of the robot.

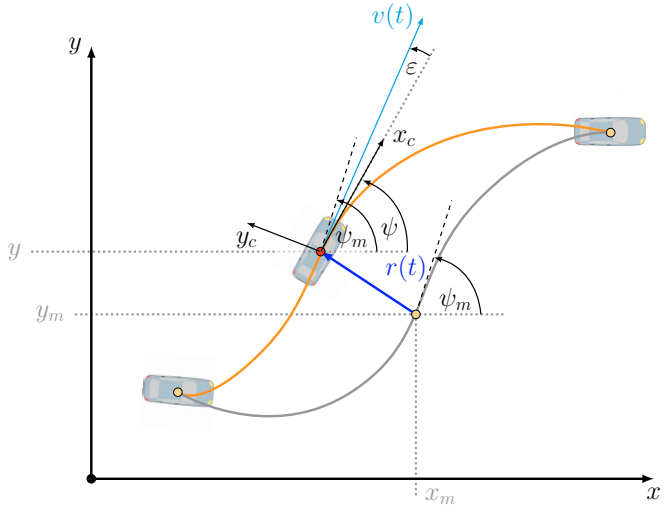

Fig. 1. Curvilinear coordinates, defined with respect to spline trajectory $\gamma_m(s)$.

Now, we aim to track the reference path γ_m with offset $r(t) \equiv 0$. Differentiating twice yields,

$$\begin{aligned} 0 = r''(t) &= v'(t) \cdot \sin\chi(t) + v(t) \cdot \chi'(t) \cdot \cos\chi(t) \\ &= v'(t) \cdot \sin\chi(t) \\ &\quad + v(t)^2 \left(\kappa(t) - \frac{\kappa_m(s(t)) \cos\chi(t)}{1 - r \cdot \kappa_m(s(t))} \right) \cdot \cos\chi(t). \end{aligned}$$

Consider $v \neq 0$, then the first equation yields $\sin\chi(t) = 0$ and the second equation is satisfied for,

$$\kappa(t) = \frac{\kappa_m(s(t)) \cdot \cos\chi(t)}{1 - r(t) \cdot \kappa_m(s(t))} = \kappa_m(s(t)) \cdot \cos\chi(t).$$

With this we can derive a feedback control law that guides the system back to the desired reference trajectory, for which $r_d \equiv 0, \chi_d \equiv 0$ and $s_d'(t) = v_d(t)$ holds. Therefore we want to construct a second order control law. Hence, let us introduce the following definitions,

$$\begin{aligned} y &\coloneqq r \\ y' &\coloneqq r' = v(t) \sin\chi(t) \\ y'' &\coloneqq r'' = v'(t) \cdot \sin\chi(t) \\ &\quad + v(t)^2 \left(\kappa(t) - \frac{\kappa_m(s(t)) \cos\chi(t)}{1 - r \cdot \kappa_m(s(t))} \right) \cdot \cos\chi(t). \end{aligned}$$

Then we can write the desired error as,

$$e = y - y_d = r - r_d. \tag{4}$$

Hence, we derivate until the control appears,

$$\begin{aligned} \dot{e} &= y' - y_d' = r' - r_d' \\ \ddot{e} &= y'' - y_d'' = r'' - r_d'' \end{aligned}$$

with $r_d \equiv 0$ we also get $r_d' = 0, r_d'' = 0$. Employing the following second order error function,

$$\ddot{e} + k_1 \dot{e} + k_2 e = 0. \tag{5}$$

Inserting (4) yields,

$$r'' + k_1 r' + k_2 r = 0.$$

Finally with the derivatives for r we get,

$$\begin{aligned} &v'(t) \sin\chi(t) + v^2(t) \left(\kappa(t) - \frac{\kappa_m(s(t)) \cos\chi(t)}{1 - r\kappa_m(s(t))} \right) \cos\chi_d(t) \\ &\quad + k_1 r' + k_2 r = 0 \end{aligned}$$

and solving for $\kappa(t)$ with $\cos\chi_d = 1, \sin\chi_d = 0$

$$\begin{aligned} \kappa(t) = \frac{1}{v^2(t)\cos\chi(t)} &\left(-k_1 r' - k_2 r - v'(t) \sin\chi(t) \right) \\ &+ \frac{\kappa_m(s(t)) \cos\chi(t)}{1 - r\kappa_m(s(t))}. \end{aligned} \tag{6}$$

With this we can derive the feedback control law for a mobile robot,

$$\begin{aligned} \triangle v = \; & v_r - v_l = w_c \cdot v \cdot \kappa \\ = & w_c \cdot v \left[\frac{1}{v^2 \cos\chi} \left(-k_1 r' - k_2 r - v' \sin\chi \right) \right. \\ & \left. + \frac{\kappa_m(s) \cos\chi}{1 - r\kappa_m(s)} \right]. \end{aligned}$$

And finally for the left and right wheel speed,

$$v_{r|l} = v_d \pm \frac{\triangle v}{2}.$$

4.1 Stability Analysis

We analyse the error dynamics stability and the stability of the internal dynamics.

Error Dynamics

Lemma 4.1. Consider (5) with parameters $k_1, k_2 > 0$. Then the error dynamics are asymptotically stable.

Proof. Rewriting the second order error dynamics in (5) as a first order system yields

$$\dot{x} = \begin{pmatrix} \dot{e} \\ \dot{\xi} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -k_2 & -k_1 \end{pmatrix} \begin{pmatrix} e \\ \xi \end{pmatrix}.$$

The eigenvalues are

$$\lambda_{1/2} = \frac{1}{2} \left(-k_1 \pm \sqrt{k_1^2 - 4k_2} \right) \tag{7}$$

and their real parts need to be negative for the system to be asymptotically stable.

If the eigenvalues are complex, i.e. $k_1^2 - 4k_2 < 0$, then the real part $-k_1/2$ is negative iff $k_1 > 0$.

If the eigenvalues are real-valued, i.e. $k_1^2 - 4k_2 \geqslant 0$, then both eigenvalues are negative iff $k_2 > 0$.

This yields the assertion.

Internal Dynamics Now we look into the stability of the internal dynamics of the controller. Let a desried velocity profile v_d be given (from the NMPC-GNEP). We consider the first order dynamics

$$v' = \frac{v_d - v}{\delta}$$

for the velocity with some constant $\delta > 0$. The feedback controlled system reads $x' = f(x)$ with $x = (r, \chi, v)^\top$ and

$$f(x) = \begin{pmatrix} v \sin\chi \\ \frac{1}{v\cos\chi} \left(-k_1 v \sin\chi - k_2 r - \frac{v_d - v}{\delta} \sin\chi \right) \\ \frac{v_d - v}{\delta} \end{pmatrix}. \tag{8}$$

Herein, we introduced (6) into (3) and exploited the differential equations for r and v.

Lemma 4.2. The internal dynamics in (8), with parameters $k_1, k_2 > 0$ are locally asymptotically stable at $r_d = 0$, $\chi_d = 0$, v_d.

Proof. The Jacobian of f at $r_d = 0, \chi_d = 0$ and v_d is given by

$$f'_x(r_d, \chi_d, v_d) = \begin{pmatrix} 0 & v_d & 0 \\ -\frac{k_2}{v_d} & -k_1 & 0 \\ 0 & 0 & -\frac{1}{\delta} \end{pmatrix}.$$

The eigenvalues are given by

$$\lambda_1 = -\frac{1}{\delta} < 0$$

$$\lambda_{2|3} = \frac{1}{2}\left(-k_1 \pm \sqrt{k_1^2 - 4k_2}\right).$$

With the proof of Theorem 4.1 we see that all eigenvalues have negative real part iff $k_1 > 0$ and $k_2 > 0$ and $\delta > 0$. This yields the assertion.

5. IMPLEMENTATION

To validate the overall algorithm consisting of the NMPC-GNEP high-level controller in Section 3 and the low-level tracking controller in Section 4 with multiple vehicles, a laboratory testing environment was set up. Currently it is possible to handle up to four vehicles on a testing area of 3.5 m by 3.2 m. To track the position and velocity of the vehicles, a HTC Vive VR System was installed. For this purpose two HTC Vive Lighthouses rev. 2.0 were mounted facing each other in opposite corners, Figure 2. Each vehicle carries a HTC Vive Tracker rev.2.0 on top of the body as shown in Figure 3.

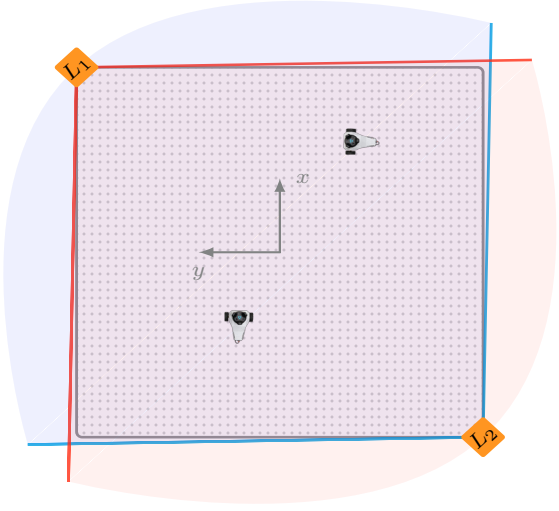

Fig. 2. Experimental setup, with the testing area (dotted), vehicles, the lighthouses L_1, L_2, their respective measuring cones and orientation.

According to Niehorster et al. (2017) recorded tracking data is accurate up to 1 cm in x and y direction. The measurement and control of the system is managed on the implemented Vehicle Cloud. Internally the Vehicle Cloud is interfaced with SteamVR, as well as OpenVR to acquire the tracking information in a frequency of 50 Hz. Further, there are two TCP server implemented, one to control and manage the vehicles and a second providing and handling the data . In particular, the primary server is used to handle the status communication of each vehicle, i.e., the registration and connection status, or to control the vehicle during the tests. This service could get adapted and extended depending on future requirements. The secondary server is used for data distribution and acquisition. If a vehicle is registered and requests a data set, the secondary server provides the measured position, angles, velocities as well as the desired control input, i.e., the offline computed velocity profiles, in the response. The communication stream for data distribution was separated from the state communication to be flexible in future use cases, e.g., change protocol from TCP to UDP. So it's possible to emulate a local position service that could be replaced by other position services like GPS or Galileo.

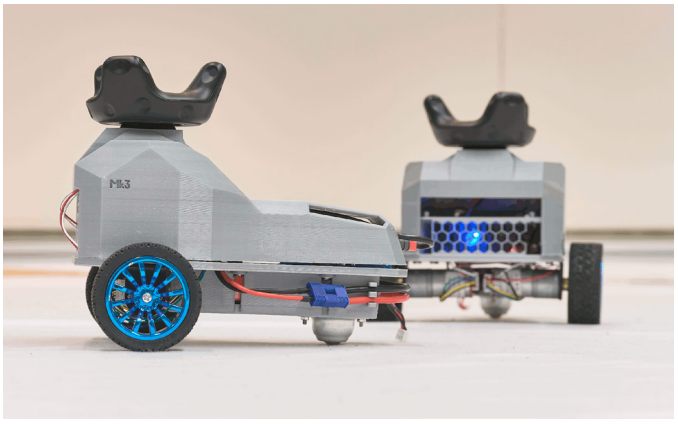

Fig. 3. Small vehicles on test track, vehicle 1 in foreground

5.1 The vehicle

For the test purposes a small vehicle was developed. It is driven by two 6 V DC motors with a transmission. Each motor is equipped with a rotary sensor, which later can be used for fusion of sensor data to improve the positioning. However, this functionality was not implemented for this paper. The computing unit is a *Raspberry Pi* 3*B*+. It is used in combination with two different motor controller boards. The first is a *Phidget* DC Motor controller board, which is connected by USB to the Raspberry Pi. The second is an *Adafruit* DC Stepper Motor HAT, which is connected via GPIO Pins and uses the I^2C-bus. Two different motor controller boards were used to reach different driving characteristics. Vehicle 1 is equipped with the Phidget and vehicle 2 is equipped with the Adafruit board. Vehicle 1 accelerates and decelerates with at most $1\,\mathrm{m\,s^{-2}}$. The measured accelerations for vehicle 2 are $4\,\mathrm{m\,s^{-2}}$ in acceleration and $10\,\mathrm{m\,s^{-2}}$ in deceleration respectively. The dynamic inversion controller described in Section 4 is used to calculate the speed difference for the left and right wheel. The desired velocity is controlled by a proportional controller.

6. RESULTS

6.1 Controller evaluation

To evaluate the low-level tracking controller of Section 4 both vehicles described in Section 5 were tested on the

track shown in Figure 4 and Figure 5. The figures show the measured paths and the given track.

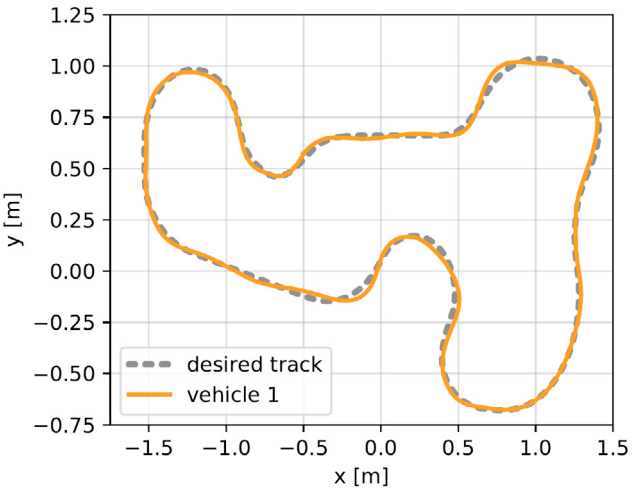

Fig. 4. Measured driving path of vehicle 1 using the dynamic inversion tracking controller.

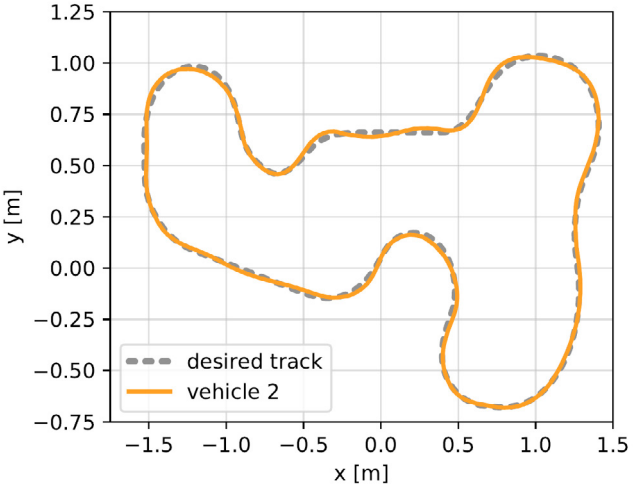

Fig. 5. Measured driving path of vehicle 2 using the dynamic inversion tracking controller.

To quantify the deviation from the track, the offset r was analysed over time in Figure 6. The upper picture shows the deviation r from the reference track over time. The lower provides a statistical evaluation for both vehicles. A maximum deviation of 3 cm for vehicle 1 and 2.6 cm for vehicle 2 was observed. In comparison to a vehicle width of 20 cm and a measurement error of about 1 mm the controller performs accurately. The parameters with which this test was conducted are listed in Table 1.

Table 1. Controller parameters for evaluation

	vehicle 1	vehicle 2
k1	6.0	6.0
k2	2.0	3.0
preview in m	0.0	0.02

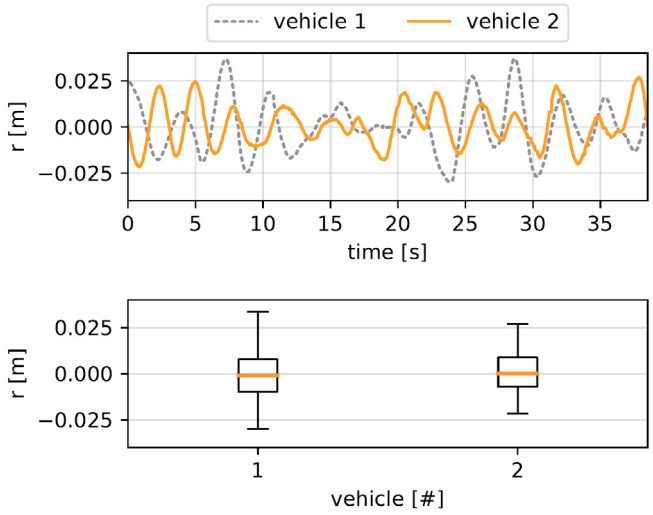

Fig. 6. Deviation between measured and ideal trajectory. In the lower plot, 50% of the measurements reside within the box, the orange line indicates the median and the minimal and maximal values are visualized.

6.2 Coordination of vehicles and collision avoidance

The NMPC-GNEP high-level controller with collision avoidance introduced in Section 2 was tested in two different configurations. So far, the controller was only used to generate paths offline. Its online realization is a future task, which is currently under construction.
The first configuration was setup with both vehicles described in Section 5, where vehicle 1 uses track one and vehicle 2 track two.

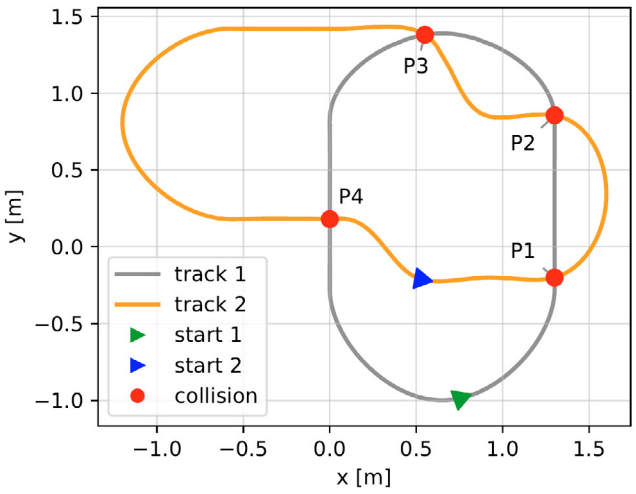

Fig. 7. Tracks of the vehicles with intersection points P1-P4 for the first scenario

Figure 7 shows both tracks, the starting positions, and the intersection points. The safety distance $d_{\nu\mu}$ was set to 20 cm with a defined constant speed of 0.2 m s^{-1} for both vehicles and with a step-size in t of $h = 0.1$ s. The time horizon of the NMPC was set to 3.0 s. Table 2 summarizes the parameters for the velocity profiles shown in Figure 8. It can be nicely observed that the second vehicle stops,

such that vehicle 1 can pass. Afterwards, the second vehicle accelerates again. This is one possible generalized Nash equilibrium.

Table 2. Parameters for scenario 1: desired velocity v_d and initial arclength parameter s_0 for the vehicles.

	vehicle 1	vehicle 2
v_d in m	0.2	0.2
s_0 in m	0.0	2.0

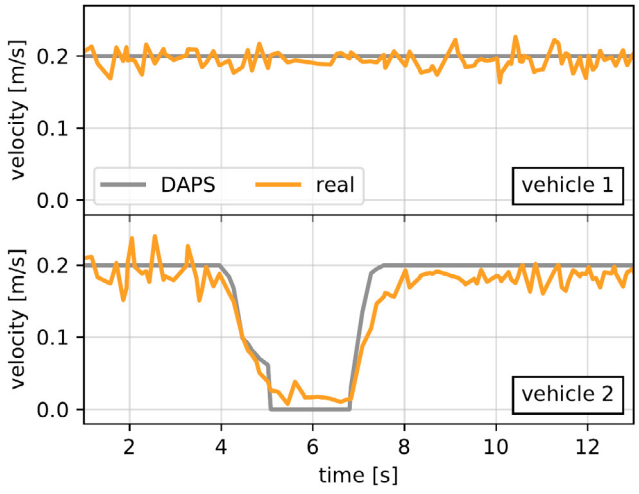

Fig. 8. Velocity profiles for the first scenario: Computed profiles and tracking results.

Figure 9 shows the distance to point $P1$ in terms of arclength for each trajectory respectively. A deviation at the start of vehicle 2 compared to the computed solution can be observed. This deviation is a second reason for the smaller distance between the measured vehicle data at the collision point $P1$ and the computed solution. However, the expected safety distance of 30 cm is upheld. Further, a slight delay in the real driving paths compared to the desired trajectory can be observed, due to inaccurate measurements of velocity and position.

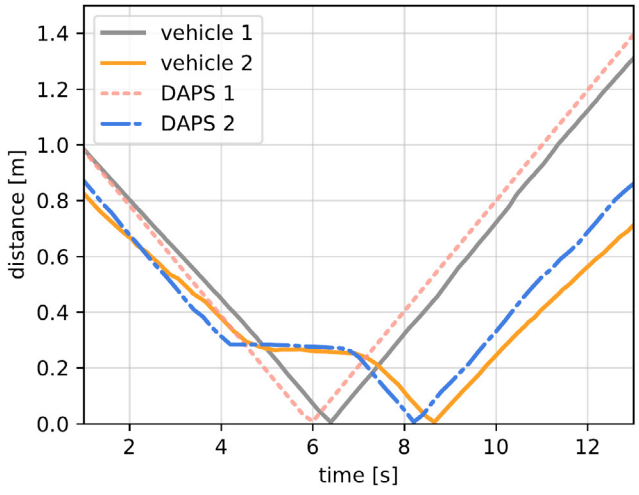

Fig. 9. Distance in arc length to collision point P1 for the first scenario.

A second scenario was designed with three vehicles. Vehicle 3 was a virtual vehicle with the same configuration as vehicle 1. To get a possible collision at point $P1$ the starting points were updated as depicted in Figure 10. The desired values for velocity and safety distance were the same as in scenario 1. Table 3 shows the parameters used for the second scenario. The calculated and measured velocities for this scenario are shown in Figure 11. Herein, vehicle 3 has the same velocity profile as vehicle 2. Vehicle 1 accelerates to increase the distance to vehicle 3. To avoid the collision with vehicle 2, the first vehicle stops and waits until vehicle 2 passes. By improving the velocity measurement and integration of rotary sensors, the safety distance could be reduced further.

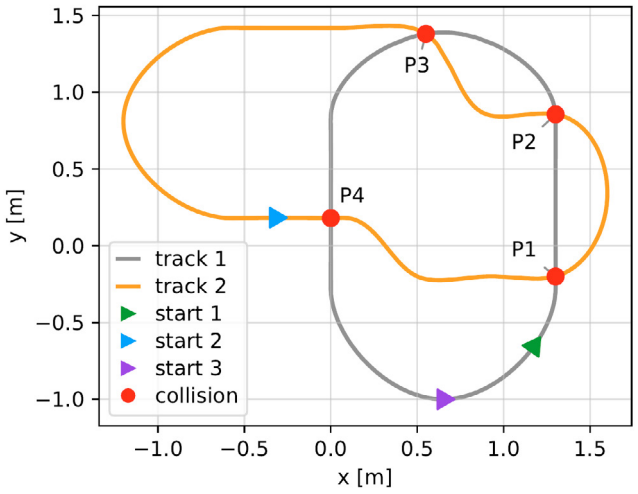

Fig. 10. Road network with intersection points for the second scenario.

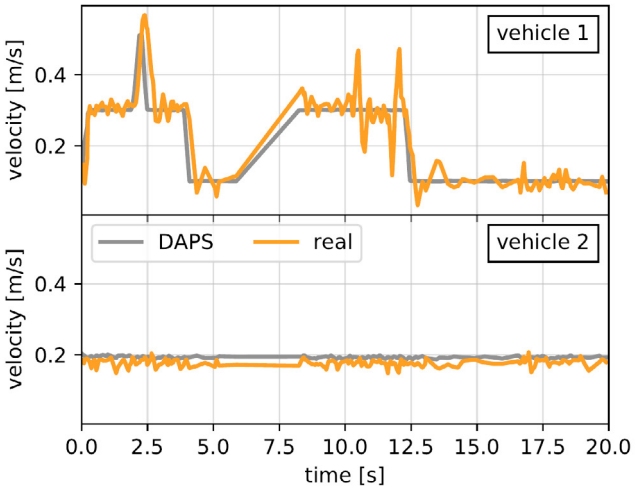

Fig. 11. Velocity profiles of all vehicles in scenario 2.

Table 3. Parameters for scenario 2: desired velocity v_d and initial arclength parameter s_0 for the vehicles.

	vehicle 1	vehicle 2	vehicle 3
v_d in m	0.2	0.2	0.2
s_0 in m	0.0	1.0	0.6

It can be observed that the minimal distance between both vehicle is much higher as in the first scenario. We can observe that the results of MOOCP approach the safety distance closer than the results of eGPG_h. This is due to the choice of penalty parameters and is desired to ensure the safety of the vehicles. The distance of the vehicles to the intersection point $P1$ on their respective paths is shown in Figure 12 in terms of arclength.

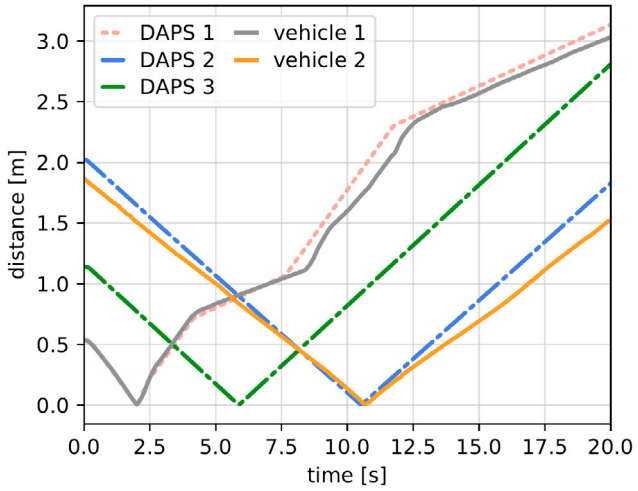

Fig. 12. Distance in arc length to collision point P1 in scenario 2.

7. CONCLUSION

The implemented dynamic inversion controller yields good results with minor deviations from the desired track. The correct synchronization of the offline computed desired paths is difficult and it would be desirable to run the NMPC-GNEP high-level controller online as well. This is subject to current work. Nevertheless, the results show that the method is capable of providing reasonable collision-free paths for the coordinated motion of several vehicles in a road network.

REFERENCES

Alessandretti, A. and Aguiar, A.P. (2019). An optimization-based cooperative path-following framework for multiple robotic vehicles. *ArXiv*, abs/1907.08531.

Ba, Q. and Pang, J. (2018). Exact penalization of generalized Nash equilibrium problems. *CoRR*, abs/1811.10674.

Britzelmeier, A. and Dreves, A. (2019). A decomposition algorithm for Nash equilibria in intersection management. *Optimization*. Submitted.

Britzelmeier, A., Dreves, A., and Gerdts, M. (2019). *Numerical solution of potential games arising in the control of cooperative automatic vehicles*, 38–45.

Britzelmeier, A. and Gerdts, M. (2018). Non-linear model predictive control of connected, automatic cars in a road network using optimal control methods. *IFAC-PapersOnLine*, 51(2), 168 – 173. 9th Vienna International Conference on Mathematical Modelling.

Burger, M. and Gerdts, M. (2019). *DAE Aspects in Vehicle Dynamics and Mobile Robotics*, 37–80. Differential-Algebraic Equations Forum. Springer International Publishing, Cham.

Facchinei, F. and Kanzow, C. (2010). Generalized Nash equilibrium problems. *Annals of Operations Research*, 175(1), 177–211.

Facchinei, F., Piccialli, V., and Sciandrone, M. (2011). Decomposition algorithms for generalized potential games. *Computational Optimization and Applications*, 50(2), 237–262.

Fliess, M., Lévine, J., Martin, P., and Rouchon, P. (1995). Flatness and defect of non-linear systems: introductory theory and examples. *International Journal of Control*, 61(6), 1327–1361.

Hult, R., Zanon, M., Gros, S., and Falcone, P. (2018). An miqp-based heuristic for optimal coordination of vehicles at intersections. *2018 IEEE Conference on Decision and Control (CDC)*, 2783–2790.

Kaya, C. and Maurer, H. (2014). A numerical method for nonconvex multi-objective optimal control problems. *Computational Optimization and Applications*, 57, 685–702.

Lot, R. and Biral, F. (2014). A curvilinear abscissa approach for the lap time optimization of racing vehicles.

Martin, P.A., Murray, R.M., and Rouchon, P. (2002). Flatness based design. In *Control Systems, Robotics and Automation*, volume XIII.

Monderer, D. and Shapley, L.S. (1996). Potential games. *Games and Economic Behavior*, 14(1), 124 – 143.

Niehorster, D.C., Li, L., and Lappe, M. (2017). The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research. *i-Perception*.

Rosen, J.B. (1965). Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica*, 33(3), 520–534.

Rosenthal, R.W. (1973). A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1), 65–67.

Rotella, F., Carillo, F.J., and Ayadi, M. (2001). Polynomial controller design based on flatness. *Kybernetika*, 38, 571–584.

von Heusinger, A. and Kanzow, C. (2009). Optimization reformulations of the generalized Nash equilibrium problem using Bikaido-Isoda-type functions. *Computational Optimization and Applications*, 43, 353–377.

Zanon, M., Gros, S., Wymeersch, H., and Falcone, P. (2017). An asynchronous algorithm for optimal vehicle coordination at traffic intersections. *IFAC-PapersOnLine*, 50(1), 12008 – 12014. 20th IFAC World Congress.