Centralized MPC for autonomous intersection crossing

Markus Carlander, Niklas Lidander, Lea Riegger, Nikolce Murgovski, Jonas Sjöberg

***Abstract*— This paper proposes a method for a safe and autonomous intersection crossing. A centralized system controls autonomous vehicles within a certain surrounding of the intersection and generates optimized trajectories for all vehicles in the area. By formulating a convex quadratic optimization problem in space domain with respect to collision avoidance constraints, a model predictive controller is designed. To validate the controller for a more advanced vehicle model, a complete simulation environment for virtual test driving is generated by combining CarMaker with Matlab/Simulink. A case study shows the effectiveness of the proposed method.**

I. INTRODUCTION

Around the world, traffic accidents that are related to intersections occur all too often. Compiled statistic from several European Countries shows that $43\,\%$ of all road injury accidents can be related to intersections [1]. Statistic conducted in the USA indicates similar numbers [2]. The actual number of related traffic accidents in Sweden is slightly less in total. Of all reported accidents in the year 2014 in Sweden, $24\,\%$ are intersection related. Of those, $91\,\%$ led to severe person injuries [3]. As the statistics show, intersections lead to high risk for accidents. Even though the most dangerous intersections are regulated by traffic lights, signs and road-markings, accidents occur still very frequently. Higher safety in intersection crossing comes at a price in today's society since it limits the traffic flow. This causes bottlenecks in the traffic rhythm which not only wastes a lot of time for travelers but also leads to environment pollution caused by unnecessary accelerations and decelerations.

Today's vehicles are trending to become more and more autonomous. Exclusive benefits as adaptive cruise control, automatic lane change maneuvers, parking assistance are available on the market today. Statistics from USA show that about $96\,\%$ of all intersection related accidents are attributed to drivers [2]. If intersections would be controlled by an autonomous cooperative intersection algorithm that can optimize the crossing sequence for all nearby vehicles, the intersection should be more safe since no human factor would cause accidents. The intersection should also be more efficient in both terms, time and energy consumption, since the algorithm would decide the crossing order and vehicles' speed for maximum efficiency. Aspects as minimizing deceleration, respectively acceleration, as well as reducing the total time for the intersection crossing could be taken into account for computing the optimal crossing sequence. It would also be possible to set different priority orders for different types of vehicles, for example according to their fuel consumption and performance. Emergency vehicles could be given precedence to enter the intersection on call-out.

In this paper, a Model Predictive Controller (MPC) is designed for a centralized system which takes control over autonomous vehicles within a certain surrounding of the intersection. The vehicles are assumed to drive fully autonomously. Among other recent work some have already exploited the idea of using different MPC implementations [4], [5], [6], [7], [8], [9]. In [6] for example, the solution is approximated by a centralized, finite time optimal control problem. In [7], a decentralized approach based on sub-optimal decision-making heuristics is used.

The general optimization algorithm for intersection crossing in this paper is based on the modeling approach of [9] where the problem is formulated in space coordinates and inverse of speed is used as a state variable. For a given crossing sequence, the approach allows the problem to be formulated as a convex program that optimizes vehicles' speed and prevents collisions. This paper has two main contributions. First, an MPC is designed with the convex modeling proposed by [9]. The MPC generates optimized trajectories for all vehicles in the controlled area, such that a cost function is minimized and several constraints are satisfied. The vehicles may differ in speed, acceleration and braking capabilities. Second, this paper implements the MPC in Matlab/Simulink and the simulation tool CarMaker, which provides advanced vehcile models making the simulation more realistic.

The paper is organized as follows: Section II gives an overview of the convex problem formulation in space coordinates. In Section III an MPC is designed. Section IV provides an MPC simulation in CarMaker running under Matlab/Simulink. Section V shows a case study and investigates the controller efficiency in the simulation environment. Section VI closes the paper with final conclusions.

II. PROBLEM FORMULATION

This section presents the modeling approach proposed by [9] and formulates the autonomous intersection crossing as a convex optimization problem.

A. Assumptions

Considering N_v autonomous vehicles in a surrounding of an intersection, each with a predefined path which they are assumed to follow, it is assumed that for each vehicle $i = 1, ...N_v$, the acceleration along its path can be varied. The vehicle dynamics in the control model are simplified to a point mass model.

The presented intersection crossing scenario is limited to one vehicle per lane which means that the scenario where vehicles are following each other on the same lane is not studied. Nevertheless, the controller is designed so that an adaptation to enable this is possible.

B. Problem statement

For a safe intersection crossing, autonomous vehicles within a certain surrounding of the intersection shall be controlled by a centralized system. An illustration of an intersection is shown in Fig. 1. The red color indicates the critical set, where never more than one vehicle should be due to safety reasons. The yellow color shows the enlarged set. The light blue color indicates the start and end of the control region. The centralized system controls only vehicles in this region. The position where vehicle k enters respectively exits the critical set is called L_k respectively H_k. The order in which the vehicles cross the intersection is called crossing sequence and the matrix which contains all the possible crossing sequences is called permutation matrix Ω.

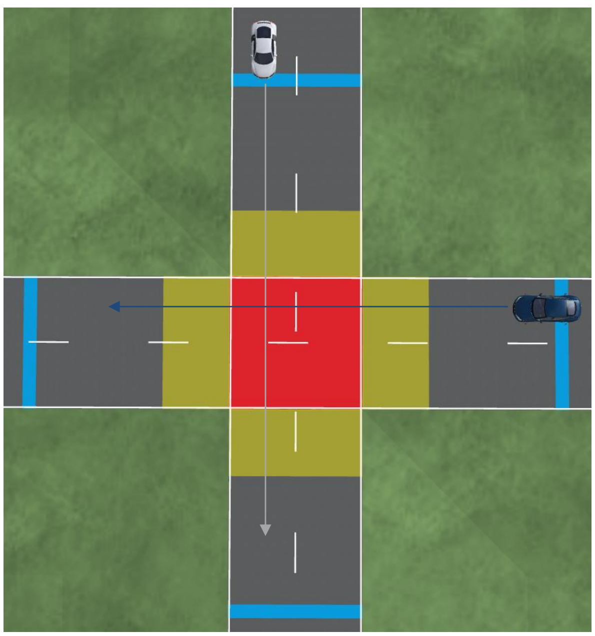

Fig. 1. Illustration of an intersection showing the critical set (red), the enlarged set (yellow) and the control region (light blue).

C. Convex problem statement

To formulate the optimization problem in a convex form, it is formulated in space coordinates, rather than time, according to [9]. With the spatial coordinate p, the linear state space model

$$\kappa_i'(p) = \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}}_{A} \kappa_i(p) + \underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{B} u_i(p) \tag{1}$$

is used to model each vehicle i, where $\kappa_i = \begin{pmatrix} t_i(p) & z_i(p) \end{pmatrix}^T$ is the state vector and $\kappa_i' = d\kappa/dp$ is the derivative with respect to the spatial distance p. The *lethargy* $z_i(p) = 1/v_i(p)$ indicates the slowness of the system and $t_i(p)$ is the time needed to reach position p. Input $u_i(p) = z_i'(p)$ is the spatial derivative of $z_i(p)$.

1) Cost function: The cumulative cost function is the sum of cost functions for each vehicle i

$$\min_{u_i(p)} \sum_{i=1}^{N_v} J_i(\kappa_i(p), u_i(p), u_i'(p), \kappa_i(p_{if})) \tag{2}$$

where p_{if} is the final position after leaving the intersection. The convex cost function for each vehicle i is

$$J_i = J_{i1} + J_{i2} + J_{i3}. \tag{3}$$

The first terms

$$J_{i1} = w_{i1}\bar{v}_{ir}^3 \int_0^{p_{if}} \left(z_i(p) - \frac{1}{v_{ir}(p)} \right)^2 dp \tag{4a}$$

penalize the deviation from the reference velocity, where w_{ij} [$j = 1$ in (4a)] are weighting factors. The mean of the reference velocity $v_{ir}(p)$ for each car i is $\bar{v}_{ir}$. The term $J_{i2} + J_{i3}$ with

$$J_{i2} = w_{i2}\bar{v}_{ir}^5 \int_0^{p_{if}} u_i^2(p)dp, \tag{4b}$$

$$J_{i3} = w_{i3}\bar{v}_{ir}^7 \int_0^{p_{if}} u_i'^2(p)dp \tag{4c}$$

penalizes high longitudinal acceleration and jerk to guarantee a comfortable drive and limited actuator usage. The unconventially appearence of J_{i2} and J_{i3} with powers of the mean velocity is due to that the problem is described in space coordinates. In [9] it is shown how these are obtained from quadratic penalties in the time domain.

2) Constraints: In addition to the equality constraint (1), the problem includes inequality constraints on the states κ_i and the input u_i as well as initial and final state constraints

$$\kappa_i'(p) = A\kappa_i(p) + Bu_i(p) \tag{5a}$$

$$\kappa_i(p) \in [\kappa_{imin}(p), \kappa_{imax}(p)] \tag{5b}$$

$$u_i(p) \in [u_{imin}(p, z_i(p)), u_{imax}(p, z_i(p))] \tag{5c}$$

$$\kappa_i(0) = \kappa_{i0} = \begin{pmatrix} 0 & 1/v_{i0} \end{pmatrix}^T \tag{5d}$$

$$\kappa_i(p_{if}) = \kappa_{if} = \begin{pmatrix} \text{free} & 1/v_{if} \end{pmatrix}^T, \tag{5e}$$

where the limits (5c) are linear functions of z_i and represent a linearized inner approximation of constant acceleration limits [9]. To avoid collisions, a final constraint is needed, which guarantees that a vehicle can enter a certain critical set at the center of the intersection, only when the previous vehicle has left the critical set, i.e.

$$t_k(H_k) \leq t_l(L_l),\ k = \Omega_{m,n},\ l = \Omega_{m,n+1}, \quad n = 1, \dots, N_v - 1, \tag{5f}$$

where k and l are indices of consecutive vehicles in a given crossing sequence m of the permutation matrix Ω, which contains all possible crossing sequences. Position H_k is the point when vehicle k exits the critical set and L_l is the entry point for vehicle l. For a given crossing sequence, the optimization problem is a convex quadratic program (QP).

More detailed explanations about the convex modeling of the problem can be found in [9].

III. MPC DESIGN

This section presents the optimization problem in a discrete space coordinate, proposes and extended cost function and rewrites the problem as a standard QP suitable for MPC implementation.

A. Discrete state space model

In order to implement the controller in Matlab, a discrete version of the model (1) with a sampling interval d_s is derived with Forward Euler. The result is the following discrete state space representation

$$\kappa_i(p+1) = A_d\kappa_i(p) + B_d u_i(p), \tag{6}$$

with the discrete matrices expressed as

$$\begin{aligned} A_d &= I_2 + d_s A = \begin{pmatrix} 1 & d_s \\ 0 & 1 \end{pmatrix} \\ B_d &= d_s B = \begin{pmatrix} 0 \\ d_s \end{pmatrix}. \end{aligned} \tag{7}$$

In order to guarantee stability of the discretized state space representation, the discretization step must fulfill the criterion,

$$|I_2 + d_s A| \leq 1 \tag{8}$$

Since the eigenvalues of the discrete state space representation $\lambda_{1,2} = 1$ are mapped on the border of the unit circle, stability is guaranteed.

B. Extended cost function

The constraint (5f) prevents the vehicles to collide, but it allows the vehicles to come very close to each other. In this case, this can lead to an infeasible solution in a later sampling instant since there is no margin left. Hence, constraint (5f) is modified so that there is some margin between the vehicles. This is done by introducing slack variables. The constraint is modeled by adding a slack variable s_j for each consecutive vehicle pair inside the control region. The variable s_j expresses the time difference between the first vehicle leaving the intersection and the second vehicle entering the intersection. By defining Δt as the desired time difference, the cost function (2) can be extended to

$$\min_{u_i, s_j} \sum_{i=1}^{N_v} J_i(\cdot) + \sum_{j=1}^{N_v-1} w_j \max(0, \Delta t - s_j)^2 \tag{9a}$$

and the constraint (5f) can be replaced by

$$s_j = t_l(L_l) - t_k(H_k), \quad s_j \geq 0. \tag{9b}$$

The maximization in (9a) is a convex function of s_j and Δt, where $t_l(L_l) - t_k(H_k) < \Delta t$. Vehicle pairs in which the vehicles are far apart are not forced to have a predefined time difference in the crossing. The extended cost function (9a) together with constraint (9b) effectively introduces an additional enlarged region, which is illustrated in Fig. 1. In comparison to the critical region, multiple vehicles may reside within the enlarged region, as long as they are outside the critical region.

Further, the min/max function in (9a) can be written without the max term as

$$\min_{s_j} \sum_{j=1} w_j q_j^2 \tag{10a}$$

$$\text{subject to: } q_j \geq \Delta t - s_j, \quad q_j \geq 0 \tag{10b}$$

where q_j are additional optimization variables.

C. Transformation to a standard QP

In this section, the problem is transformed into the standard QP form

$$\min_x \frac{1}{2} x^T H x + f^T x \tag{11a}$$

$$\text{subject to: } A_{eq} x = b_{eq}, \tag{11b}$$

$$A_{in} x \leq b_{in}. \tag{11c}$$

where x is the vector of optimization variables, H the Hessian matrix and f the remaining linear terms in the objective. The constraints are also transformed to fit the formulation in (11b)-(11c).

1) Cost function: The cost function (3) is transformed in a quadratic form for the MPC by writing the $x = \begin{pmatrix} x_1 & \dots & x_{N_v} \end{pmatrix}^T$ vector in (11a) as

$$x_i = \begin{pmatrix} K_i \\ U_i \\ U_i' \\ S_j \\ Q_j \end{pmatrix} \tag{12}$$

where $K_i = [\kappa_i(1), \dots, \kappa_i(N)]^T$, $U_i = [u_i(0), \dots, u_i(N-1)]^T$, $U_i' = [u_i'(0), \dots, u_i'(N-1)]^T$, $S_j = [s_j(1), \dots, s_j(N)]^T$, $Q_j = [q_j(1), \dots, q_j(N)]^T$ for each vehicle i and vehicle pair j in the control region. The vector x_i involves the states, the control input, the derivative of the control input and the slack variable. The prediction and control horizon are both equal to N. The Hessian matrix H_i for each vehicle i results in

$$H_i = 2 \begin{pmatrix} Q_{i1} & 0 & 0 & 0 & 0 \\ 0 & Q_{i2} & 0 & 0 & 0 \\ 0 & 0 & Q_{i3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_{i4} \end{pmatrix}, \tag{13}$$

where the matrices Q_{i1}, Q_{i2}, Q_{i3} and the scalar Q_{i4} are equal to

$$Q_{i1} = w_{i1}\bar{v}_{ir}^3 C^T C I_N, \tag{14a}$$

$$Q_{i2} = w_{i2}\bar{v}_{ir}^5 I_N, \tag{14b}$$

$$Q_{i3} = w_{i3}\bar{v}_{ir}^7 I_N, \tag{14c}$$

$$Q_{i4} = w_j, \tag{14d}$$

where I_N is the identity matrix with N rows and $C = [0 \;\; 1]$. The f_i vector containing the remaining non-quadratic terms for each vehicle i is

$$f_i^T = -2w_{i1}\bar{v}_{ir}^3 \frac{1}{v_{ir}(p)} C \begin{pmatrix} 1 & \dots & 1 & 0 & \dots & 0 \end{pmatrix}. \tag{15}$$

2) Constraints: The state constraint of the model described in (5a) has to be reformulated in a matrix form as shown in (11b) with the discrete model described in (7). Furthermore, the constraints (5b)-(5c) limiting the state variables and the acceleration as well as the collision avoidance constraint (5f) can be written into new inequality constraints (11c). The two constraints (5b)-(5c) have to be split up into two constraints in order to be implemented in a matrix form.

In order to control N_v vehicles, the cost function (11a) needs to be extended. The new x vector and Hessian matrix for N_v cars are then according to (11a)

$$x = \left(x_{1:N_v}^T\right)^T, \tag{16}$$
$$H = diag\left(H_{1:N_v}\right), \tag{17}$$

and the f vector is changed to the following

$$f^T = \left(f_{1:N_v}^T\right). \tag{18}$$

Analogeus, the constraints also change in the same manner. Using the x vector (16), the equality and inequality constraints for all N_v vehicles can be written as

$$A_{eq}^T = \left(A_{eq,1:N_v}^T\right), \tag{19}$$
$$b_{eq}^T = \left(b_{eq,1:N_v}^T\right), \tag{20}$$
$$A_{in}^T = \left(A_{in,1:N_v}^T\right), \tag{21}$$
$$b_{in}^T = \left(b_{in,1:N_v}^T\right). \tag{22}$$

D. Control area and optimization horizon

Whether a vehicle is included in the MPC computation depends on its distance to the intersection. The control area of the centralized controller is defined for a certain surrounding of the intersection. Before entering the control area, it is assumed that the car drives with its reference speed. The control area is re-scanned in every time step searching for new arriving or leaving cars. The controller re-optimizes in every time step and takes therefore only the vehicles in the control area into account. The re-optimization is needed because the vehicles do not follow exactly the desired acceleration computed by the controller due to the simplified point mass model for the vehicle dynamics in the controller.

Since there is no point in controlling the vehicles after they have passed the control radius, the optimization horizon N_i for each vehicle $i = 1, \ldots, N_v$, is not moving along the vehicles as they advance. Instead the horizon is fixed to the end of the control radius. In this way, the optimization horizon is shrinking as the cars are progressing through the control area. This lowers the computation time of the MPC algorithm in each position step of the vehicles in the control area.

IV. Simulation

The designed MPC controller is based on a simple point mass model. In this section the controller is validated using an advanced vehicle model using CarMaker.

For the simulation, a combination of Matlab/Simulink and CarMaker is used. In this section, the focus is on connecting the MPC developed in Matlab/Simulink to a traffic and vehicle model created with CarMaker. The MPC is implemented as a Matlab Function block in Simulink, see Fig. 2. The simulation tool IPG CarMaker serves as a simulation environment to design a traffic model, containing an intersection and traffic flow. A detailed vehicle model is also provided by CarMaker. The detailed model is used in simulation to test if collisions are avoided. It is also possible in CarMaker to change the type or only some parameters of a car. Furthermore, CarMaker provides a video animation by the plug in program IPG Movie for visualizing the simulation results. This simplifies the validation and representation of the simulation results.

A. Restrictions of CarMaker 5.0.2

In CarMaker 5.0.2, only one host car can be simulated as an advanced car model. It is not possible to control several vehicles with detailed vehicle dynamics. Except of the host car, all other cars can only be modeled as traffic objects, which implies static objects without dynamics. They are simulated as 3-D boxes with a certain initial position and acceleration. For the acceleration of each traffic object, the corresponding output of the MPC can be used. Nevertheless, it should be noted that these cars have the same dynamics as the point mass model used for the controller design.

B. Simulation environment design

A CarMaker road is constructed from start to stop by a list of road segments, each one only connected with the previous and following segment [10]. For simplicity, an intersection can be constructed by making a turn and letting the road cross itself (see Fig. 3). This causes a limitation to the movements of the vehicles because the motion of the traffic objects is connected to the definition of the road. Thus, the traffic objects have to pass straight through the intersection. A turn in such intersection is not possible.

After the creation of the intersection, traffic is added to the model. For each vehicle, the host car as well as the traffic objects, the vehicle type, the reference starting point and reference speed are preset. Since the host car starts always with zero velocity, the road model should be designed in a way such that the host car has reached its desired speed before entering the controlled intersection surrounding.

C. Adding the MPC

The CarMaker simulation model in Simulink consists of a chain of individual subsystem blocks. These blocks can not be removed, but their functionality can be changed by overwriting their input or output signals.

The MPC is added by hijacking the gas and brake signals in the *Vehicle Control* block in Simulink as seen in Fig. 2, where drivers wish is overwritten by the calculated values from the MPC algorithm. The gas and brake signals from the driver model are only used for the vehicle with dynamics, where the control signal calculated by the MPC is converted

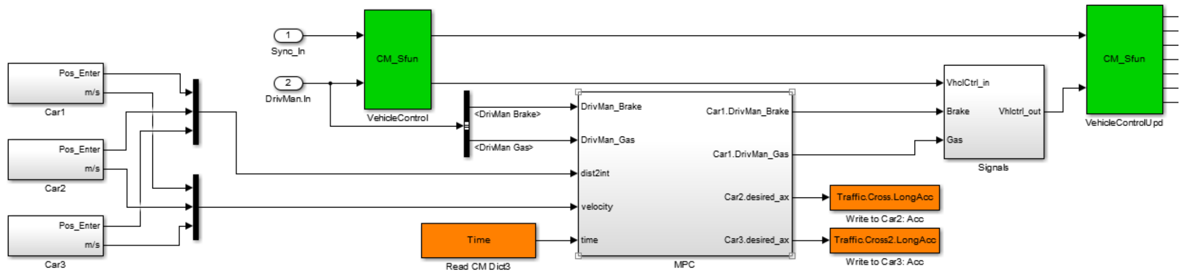

Fig. 2. Implementation of the MPC in Simulink combined with CarMaker's Simulink environment. Green color indicates CarMaker's environment and orange color indicate signals taken from CarMaker. The MPC is added in between the two green boxes, where signals, e.g brake and gas are 'hijacked' from CarMaker and used in the MPC algorithm. The output of the MPC with the manipulated signals are afterwards reconnected to CarMaker.

to the gas and brake signals. For the other vehicles which are static traffic objects, there is no driver model, due to the limitation of only one car with full dynamics. Therefore, the calculated control signals for these two vehicles overwrite directly the acceleration signal without a conversion to gas and brake signals.

V. Simulation Evaluation with Three Vehicles

In the selected case study, the presented MPC is tested for three cars approaching an intersection as shown in Fig. 3. In Table I, the parameters for the cars $i = 1, 2, 3$ can

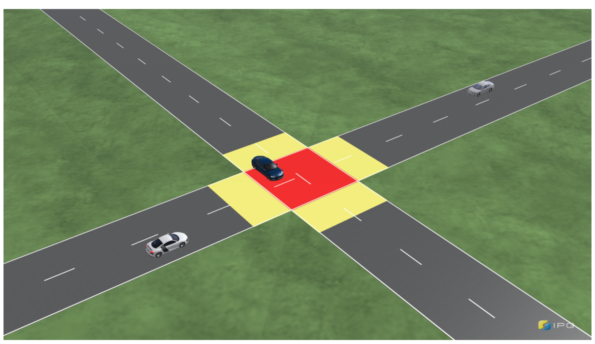

Fig. 3. Created intersection crossing in CarMaker environment where the three case are approaching the intersection.

TABLE I
PROBLEM DATA OF THE CASE STUDY.

Parameter	Values
$\begin{pmatrix} v_{1r} & v_{2r} & v_{3r} \end{pmatrix}$	$\begin{pmatrix} 47\,\mathrm{km/h} & 48\,\mathrm{km/h} & 50\,\mathrm{km/h} \end{pmatrix}$
$\begin{pmatrix} v_{imin} & v_{imax} \end{pmatrix}$	$\begin{pmatrix} 30\,\mathrm{km/h} & 90\,\mathrm{km/h} \end{pmatrix}$
$\begin{pmatrix} a_{imin} & a_{imax} \end{pmatrix}$	$\begin{pmatrix} -3\,\mathrm{m/s^2} & 3\,\mathrm{m/s^2} \end{pmatrix}$
$\begin{pmatrix} w_{i1} & w_{i2} & w_{i3} & w_{i4} \end{pmatrix}$	$\begin{pmatrix} 1 & 1100 & 23 & 10000 \end{pmatrix}$
Δt	0.6 s
d_s	4 m

be found. The speed and acceleration limits as well as the weights for the cost function are selected identically for all three vehicles. The desired crossing sequence is chosen to $1, 2, 3$. The vehicles start with different initial speed and distances from the intersection. Without a controller, car 1 and 2 would collide in the intersection. This is also illustrated in the following video animation, created with Carmaker: `https://youtu.be/LKcXflY6Mtw` .The MPC takes control over a vehicle when it has entered the control radius of the intersection, which is 60 meters before and 60 meters after the intersection. The critical region, where no collisions are allowed, is the $15 \cdot 15\ m^2$ intersection crossing area. The vehicles do not turn left or right in the intersection, instead they only drive straight forward. The initial optimization horizon for every car entering the control area is N=135 meters.

Gathered data from the sensors of the vehicles via CarMaker is shown in Fig. 4 and a video animation can be found on `https://youtu.be/VV36-eJ0tEw`. When vehicle 1 comes closer to the intersection it can be observed that it starts to accelerate and vehicle 2 starts to decelerate in order to avoid a collision. The third vehicle slows down to avoid a collision between the second and third vehicle. Furthermore, by looking at the last plot, it is evident that the MPC controller efficiently prohibits collisions between the vehicles. An upward-pointing triangle means the time where the vehicle is entering the intersection and a downward-pointing triangle shows the time when it is leaving. Since there is no vertical alignment between an downward-pointing triangle and an upward-pointing triangle, there are no collisions. CarMaker provides also a video animation for visualizing the simulation results. In the video, it can be observed that the cars drive in a smooth way following the given crossing sequence and there is never more than one car in the intersection.

VI. Conclusions

This paper provides a centralized MPC for optimal control of autonomous vehicles in the control area of an intersection. The problem is formulated as a convex quadratic problem such that the MPC equations can be solved efficiently. The controller is tested for an advanced vehicle model using the simulation tool CarMaker. The simulation results are shown in a case study that verifies that the MPC works efficiently for the advanced CarMaker vehicle model.

Compared to the solution presented in [9], the computation time for the convex problem is decreased by using a QP solver instead of the generalized second order cone program (SOCP) solver. Using the quadratic form for the MPC, the

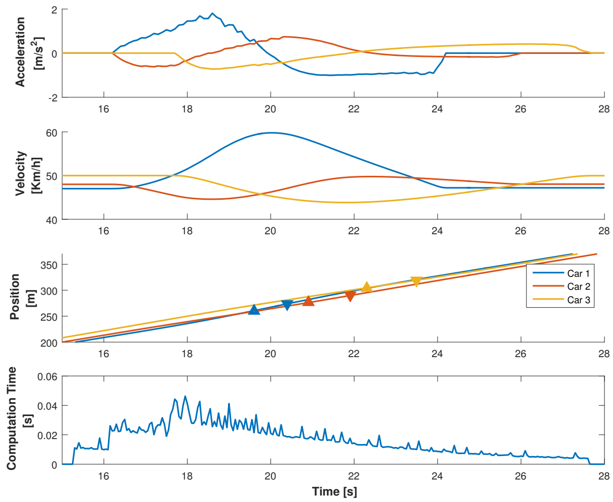

Fig. 4. Trajectories from the MPC controller. The upper three subplots show velocity, acceleration and position for each vehicle. The very last plot shows the computation time of the quadratic programming solver.

problem can be solved very fast in every time step. The computation time is related to the distance to the intersection of each vehicle. Since it depends on the number of cars in the control area and the length of their prediction horizon, the computation time is decreased by using a shrinking prediction horizon as can be seen in Fig. 4. The Simulation was preformed on a computer with (Intel(R) Core(TM) i7-3520M) processor and 8 GB RAM.

The MPC works efficiently for the virtual test drive, but using CarMaker as a simulation environment turned out to have some drawbacks. For a more extended case study, e.g. with more cars in the control area, CarMaker 5.0.2 combined with Matlab/Simulink can therefore be considered as ineffective as a virtual test basis.

REFERENCES

[1] M. A. Martinez. Accident causation and pre-accidental driving situations. [Online]. Available: https://dspace.lboro.ac.uk/2134/8434, 2008

[2] National Highway Traffic Safety Administration. Crash factors in intersection-related crashes: An on-scene perspective. [Online]. Available: http://www-nrd.nhtsa.dot.gov/Pubs/811366.pdf,2010

[3] Sveriges Officiella Statestik Trafik Analys. Vägtrafikskador 2014. road traffic injuries 2014. [Online]. Available: http://www-nrd.nhtsa.dot.gov/Pubs/811366.pdf

[4] K.-D. Kim, "Collision free autonomous ground traffic: A model predictive control approach," in *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2013, pp. 51–60.

[5] K.-D. Kim and P. Kumar, "An mpc-based approach to provable system-wide safety and liveness of autonomous ground traffic," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3341–3356, 2014.

[6] R. Hult, G. R. de Campos, P. Falcone, and H. Wymeersch, "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections," in *Proceedings of the American Control Conference*, 2015, pp. 763–768.

[7] G. R. de Campos, P. Falcone, and J. Sjöberg, "Autonomous cooperative driving: A velocity-based negotiation approach for intersection crossing," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct 2013, pp. 1456–1461.

[8] G. R. de Campos, P. Falcone, H. Wymeersch, R. Hult, and J. Sjöberg, "Cooperative receding horizon conflict resolution at traffic intersections," in *IEEE 53rd Annual Conference on Decision and Control (CDC)*, Dec 2014, pp. 2932–2937.

[9] N. Murgovski, G. R. de Campos, and J. Sjöberg, "Convex modeling of conflict resolution at traffic intersections," in *Conference on Decision and Control, Osaka, Japan*, 2015.

[10] IPG CarMaker, *User's guide version 5.0.2.* IPG Automotive, Karlsruhe, 2015.