Comparative Evaluation of a Trajectory Generator for Obstacle Avoidance Guaranteeing Computational Upper Cost

Isao Okawa 1) Kenichiro Nonaka 2)
1) DENSO CORP,
1-1 Showa, Kariya, Aichi, 448-8661, Japan (E-mail: isao.okawa@denso.co.jp)
2) Tokyo City University
1-28-1 Tamazutsumi, Setagaya, Tokyo, 158-8557, Japan

Received on November 15, 2017

ABSTRACT: A novel trajectory generator for obstacle avoidance is proposed and evaluated through numerical simulations with common iterative methods. Since the proposed method generates quasi-optimal trajectories using model predictive control (MPC) theory with a predefined upper bound on computational cost, it makes it easy to guarantee real-time feasibility for autonomous driving and/or driving support systems. Numerical round-robin simulations are conducted for both the proposed method and a comparative method, after which we evaluate the results through statistical analysis and individually analyze several characteristic results. Taken together, the results show that the proposed method generates trajectories that are statistically equivalent to those generated by the comparative method, while guaranteeing that the upper bound of the computational cost is predetermined.

KEY WORDS: Safety, Active safety, Driving support / Autonomous drive, Obstacle avoidance, Trajectory generation [C1]

1. Introduction

1.1. Background

A wide variety of autonomous driving technologies have been under active development by numerous academic and industrial research groups in recent years (1,2). Indeed, highway-capable autonomous driving and/or driver assistance systems now appear to be nearly ready for practical use (3). As a result, there are strong ongoing demands for advances that will close the gap and make such systems ready for use on urban roads.

Obstacle avoidance is an important technical problem that such systems must be able to resolve in urban environments. Since safety assurance is critical in this area, such systems must be able to generate trajectories that guarantee obstacle avoidance. Or, if no secure trajectories are possible, they must be able to immediately detect such scenarios (so that drivers can react appropriately).

In the research area of unmanned ground vehicles (UGVs), numerous methods for generating such trajectories have been proposed. Citing examples for front-wheel steering vehicles like automobiles, methods that utilize graph search algorithms in discretized space (4) have been shown capable of achieving autonomous operation in complicated off-road environments. Furthermore, methods based on model predictive control (MPC) theory (5) have been shown capable of generating steering input trajectories that avoid obstacles considering vehicles’ kinematic constraints.

1.2. Application and implementation of MPC theory

Since a system’s performance is determined primarily by vehicles’ kinematic constraints, consideration of those constraints is a crucial property for an obstacle avoidance. With that point in mind, the methods based on MPC theory have advantages since those methods are capable to consider the constraints explicitly. Thus, numerous method applying MPC for obstacle avoidance have been proposed. Citing examples, a method based on robust nonlinear MPC utilizing a vehicle dynamics model and bounds for tire slip angle as constraints (6) has been shown capable of obstacle avoidance in uncertain road conditions. An another nonlinear MPC utilizing vehicle dynamics models and bounds for steering angle, steering rate, speed, acceleration, and jerk as constraints (7) has been shown capable of obstacle avoidance with high-speed vehicle navigation.

However, since constrained MPC is not analytically solvable and generally requires iterative computation, the methods cited above solve MPC problem using general solvers for optimization. Thus, it is difficult to discern upper computational costs and guarantee the system’s real-time feasibility. Especially for autonomous driving of automobiles, specifications of embedded computer will not be fixed unless upper computational costs are predetermined. Therefore, guaranteeing the upper cost is a crucial demand for popularization of autonomous driving technologies.

To address the aforementioned difficulty, numerous algorithms to implement MPC have been also studied. These include Explicit MPC (8), which computes optimal control low in advance using offline optimization in order to achieve high real-time feasibility for small-size MPC problems; CGMRES algorithm (9), which traces optimal solutions via continuation in real-time and is even applicable to nonlinear systems; and the M-matrix based efficient algorithm (10), which generates exact solutions for input constrained linear systems and computes the optimal control input with a predetermined computational upper cost.
However, since these methods are configured to design controllers for ordinary driving scenes, there are some unsuitableness to generate vehicle’s obstacle avoidance trajectory. As an example, supposing the predictive range of MPC as 1.5 ~ 3.0 s and its discrete interval as 0.1 ~ 0.3 s corresponding to human driver’s steering operation, the number of predictive points is about 5 ~ 30 that are not small as number of variables. Although, the constraints at each predictive point are often inactive in ordinal driving scenes. Therefore, an algorithm whose computational costs depend only on the number of active constraints has advantages. While the M-matrix based efficient algorithm (10) have satisfied this feature, it is difficult applying to obstacle avoidance since its scope is limited for input constrained problem.

1.3. Proposed method

In this study, we propose and evaluate a new trajectory generator based on the M-matrix based efficient algorithm (10). The proposed method generates a quasi-optimizable and feasible trajectory with a predetermined computational upper cost. Thus, the computational upper cost is known a priori. In the previous work (11), we evaluated this method through simple bench-mark simulations, such as a double lane change (moose test). In this paper, a numerical round-robin simulation is conducted exhaustively changing the ego vehicle state and obstacle geometry. We then employ a commercially available numerical solver with an interior-point algorithm as a comparative method and conduct the same simulations. All of the simulation results are then comparatively evaluated through statistical analysis with regard to the performance of generated trajectories, and several characteristic results are individually analyzed. The results indicate that the proposed method shows almost equivalent performance with the optimal one even though the computational upper cost is guaranteed.

2. Problem Definition (11)

2.1. Modeling vehicle motion and obstacle avoidance

Fig. 1 depicts the definition of words and variables. We define the lateral offset $y$ as the relative distance between the ego vehicle and the nearest point on the lane center. Similarly, the angular offset $\psi$ is defined as the relative angle between the vehicle movement direction and the lane center tangent at the nearest point. The angular offset is also represented as the sum of relative yaw angle $\theta$ and vehicle slip angle $\beta$. Then, $\kappa$ is the curvature of lane and $s$ is the travel distance along the lane center. The ego vehicle moves at speed $V$ and yaw rate $\gamma$, with front tire angle $\delta$. The ego vehicle mass is $M$, while $l_f$ and $l_r$ are the distances between center of gravity and front and rear axle, respectively.

Next, we model the variation rate of lateral offset $y$ and $\psi$ with respect to the travel distance $s$. The variation rate of $s$, $y$, and $\psi$ with respect to the time are modeled as follows:

$$\frac{ds}{dt} = \frac{V \cos \psi}{1 - y \kappa},$$

$$\frac{dy}{dt} = \frac{V \sin \psi}{1 - y \kappa},$$

$$\frac{d\psi}{dt} = \kappa - \left( a_\beta + a_\gamma + b_\delta \right),$$

where $a_\beta$, $a_\gamma$, and $b_\delta$ represent the model coefficients for vehicle slip angle $\beta$. These coefficients are defined as

$$a_\beta := \frac{2(K_f + K_r)}{MV},$$

$$a_\gamma := \frac{2(l_f K_f - l_r K_r)}{MV^2},$$

$$b_\delta := \frac{2K_f}{MV},$$

where $K_f$ and $K_r$ are the cornering power of front and rear tires, respectively. Dividing the model (1-2) and (1-3) by (1-1), the variation rate of $y$ and $\psi$ with respect to the travel distance $s$ are modeled:

$$\frac{d}{ds} \left[ \begin{array}{c} y \\ \psi \end{array} \right] = f(y, \psi) + g(y, \psi) \delta,$$

$$f(y, \psi) := \left[ \begin{array}{c} 1 - y \kappa \\ \frac{1 - y \kappa}{V \cos \psi} (a_\beta + (1 + a_\gamma) y) - \kappa \end{array} \right],$$

$$g(y, \psi) := \left[ \begin{array}{c} \frac{0}{V \cos \psi} \end{array} \right].$$

Moreover, the model (3-1) – (3-3) can be transformed linearly without any approximation. Upon applying a nonlinear transformation, local feedback, and discretization, we get the following transformed model:

$$x[k + 1] = Ax[k] + bu[k],$$

$$A := \left[ \begin{array}{c} 1 \\ 0 \end{array} \right],$$

$$b := \left[ \begin{array}{c} \Delta s^2/2 \\ \Delta s \end{array} \right].$$

Fig. 1 Definition of words and variables
where \( x := [y, (1 - y)x\tan \psi] \), \( u \) represents a linearized steering input, and \( ds \) is a discrete interval of the travel distance \( s \).

Fig. 2 depicts a procedure that is used to consider obstacles as constraints for vehicle state. First, the risk area that must not be approached is measured and settled. Next, we set predictive points at regular intervals \( \Delta s \) on the lane center. The interval \( \Delta s \) corresponds to a discrete interval of model (4-1) – (4-3), and the number of the points is \( N \). For each predictive point, lateral offset margins for the boundary of the risk area are calculated with respect to the left and the right side of the lane, and are represented as \( y_{[1]} \) … \( y_{[N]} \), \( y_{[1]} \) … \( y_{[N]} \), respectively. Using these margins, the constraints for obstacles are represented by the following vehicle lateral offset inequality:

\[
y_{[k]} \leq y[k] \leq y'_{[k]} \quad \text{for} \quad k = 1, 2, \ldots, N.
\]

2.2. Formulation to a model predictive control problem

Next, the trajectory generation for obstacle avoidance is formulated into a MPC problem. The index function that represents the quality of trajectory is defined as follows:

\[
J := \phi(x[N]) + \sum_{k=0}^{N-1} L(x[k], u[k]),
\]

\[
\phi(x[N]) := \frac{1}{2} x[N]\text{T}Qx[N],
\]

\[
L(x[k], u[k]) := \frac{1}{2} x[k]\text{T}Qx[k] + \frac{1}{2} Ru[k]\text{T}
\]

where \( Q := \text{diag}(q_1, q_2) \) and \( R \) are weight matrix and scalar, respectively. Since this function consists of an offset term and an inequality \( (5) \) is also reconstructed as follows:

\[
\frac{1}{2} U^T H U + U^T F x[0]
\]

where \( H \) and \( F \) are constant matrices comprised of \( A, b, Q, \) and \( R \). In the same way, inequality \( (5) \) is also reconstructed as follows:

\[
\tilde{A} U \preceq \tilde{b}(x[0], Y_f, Y_u)
\]

where \( \preceq \) indicates element-wise inequality. The matrix \( \tilde{A} \) is constant, while the matrix \( \tilde{b} \) is variable with \( x[0] \), \( Y_f \) and \( Y_u \).

The optimal steering input vector \( U^* \), which traces the lane center and avoids obstacles, is obtained by solving the MPC problem below. This representation is also called quadratic programming (QP).

\[
\min_U J \quad \text{subject to} \quad \tilde{A} U \preceq \tilde{b}(x[0], Y_f, Y_u)
\]

By simulating model (4-1) – (4-3) using the elements of the obtained input vector, the optimal state vector \( Y^* \) for obstacle avoidance is generated. In addition, if there were no obstacles (constraints), the input vector will be calculated analytically as follows:

\[
U^*_i = -H^{-1}F x[0]
\]

Thus, the state vector \( Y^*_i \) is also calculated analytically.

3. Proposed Method

3.1. Outline

In this section, the proposed trajectory generation method is presented. The proposed method solves MPC (15) within a predetermined upper iteration count and thus allows the computational upper cost for trajectory generation to be predetermined. The method reformulates MPC (15) into a linear complementarity problem (LCP) according to the M-matrix based efficient algorithm \( (10) \). This reformulation is described in subsection 3.2. Then, in subsection 3.3, we describe the algorithm used to generate the trajectory. The proposed method decomposes the problem into two sub-problems, which is similar to the M-matrix based efficient algorithm \( (10) \), but these sub-problems are used in a different and original way. In subsection 3.4, the computational cost of the method is analyzed.

3.2. Reformulation to a linear complementarity problem

To simplify the computational cost discussion, we reformulate the problem expressed in MPC into an LCP and introduce a new vector that represents the margin for the risk area boundaries.
\[ Z := \bar{b} - \tilde{\alpha}U = \begin{bmatrix} Y - Y_t \\ Y_u - Y_t \end{bmatrix} \geq 0. \] (17)

Considering the necessary condition of MPC optimality (15), the problem is reformulated as a following LCP while \( \Lambda \) is a Lagrange multiplier vector:

Find \( \Lambda \) such that
\[ Z = M\Lambda + q, \] (18)
\[ \Lambda^T Z = 0, \] (19)
\[ \Lambda \geq 0, \] (20)
\[ Z \geq 0. \] (21)

The matrix \( M \) and the vector \( q \) are defined as follows, while \( M_j \) is a constant symmetric positive definite matrix:

\[ M := \begin{bmatrix} M_j & -M_j \\ -M_j & M_j \end{bmatrix}, \] (22)
\[ q := \begin{bmatrix} Y_t - Y_f \\ Y_u - Y_f \end{bmatrix}. \] (23)

When \( \Lambda \) is obtained, the optimal input vector \( U^* \) is calculated as follows:

\[ U^* = U_j^* - H^{-1}\tilde{\alpha}^T\Lambda. \] (24)

3.3. Algorithm

We decompose the LCP into two sub-LCPs, which allows considering \( Y_t \) or \( Y_u \) separately. This operation also decomposes equation (18) into the following equations:

- **Sub-LCP: only lower bound**
  \[ [Y - Y_t] = M_j \Lambda_j + [Y_u - Y_t], \] (25)

- **Sub-LCP: only upper bound**
  \[ [Y_u - Y] = M_j \Lambda_u + [Y_u - Y_u^*]. \] (26)

Solving these sub-LCPs indicates that, at every predictive point, the trajectory without any constraint \( Y_u^* \) is modified by \( \Lambda_j \) or \( \Lambda_u \) in order to retain the margins \( Z_j \) or \( Z_u \) for a corresponding risk area.

The proposed method modifies the trajectory sequentially, reciprocating between the two sub-LCPs. Fig. 3 shows how the modification is performed and the trajectory is generated. At the first step of the algorithm (a), the trajectory without any constraints is analytically generated. At the second step (b), the trajectory is modified to satisfy one side constraint (either the lower or upper bound). Then, at the third step (c), using the trajectory modified in the second step as the new \( Y_u^* \), the trajectory is modified to satisfy the constraint of the other side while keeping the modified point in the second step if all constraints were satisfied, the process terminates. If not, the algorithm returns to the second step and repeats the process using modified trajectory as the new \( Y_u^* \). If all constraints are then satisfied, the algorithm returns \( U^* \) and terminates. However, if all the points have been fixed while any constraint remains unsatisfied, the algorithm returns “no secure trajectory” and terminates.

3.4. Computational complexity analysis

From equations (19) - (21), corresponding elements between \( \Lambda \) and \( Z \) are complementary. Thus, the LCP is a combination problem for use when choosing the modification pattern. The total number of sub-LCP combinations (25), (26) is \( 2^N - 1 \). The computational upper cost appears when the algorithm returns “no secure trajectory”, while each of the predictive points are fixed one-by-one at each sub-LCP. In this case, the maximum iteration count of the algorithm is computed as follows:

\[ (\text{Max. iteration}) = \left( 2^N - 1 \right) + \left( 2^N - 1 \right) + \ldots + \left( 2^N - 1 \right) \]
\[ = 3 \times 2^N - 3. \] (27)

Note that the maximum iteration count grows exponentially with the number of predictive points.

On the other hand, interior point method algorithm has been used for many years when solving both QP (15) and LCP related processes (12). While there are many variations of the algorithm, the best computational complexity order is known to be \( O(\sqrt{NL}) \), where \( L \) is the data length. However, while the interior point method is clearly advantageous for solving large-size problems, these algorithms require the initial value of \( U \). Thus, it can be seen that the overall computational cost required to solve a problem depends on the suitability of the initial \( U \). Since the environment surrounding the ego vehicle changes innumerable, it is difficult to determine the worst initial \( U \) for the computational cost.

In contrast to above, since the proposed method does not require any initial value, the overall computational cost is predetermined by the number of predictive points \( N \). Furthermore, since the proposed algorithm terminates if all the constraints have
been satisfied, the practical computational cost will be small under ordinary driving scenarios where the number of simultaneously active constraints is expected to be limited. Therefore, providing the size of problem is sufficiently small, the proposed method would be practical for structuring real-time systems.

4. Numerical Simulation

4.1. Configuration

The proposed method is evaluated through simulating trajectory generations that avoid obstacles while remaining within the present lane. Fig. 4 provides an illustration of this configuration. Fig. 4 (a) depicts the geometric configuration. The lane width and the ego vehicle’s width are set at 3.0 m and 1.85 m, respectively. The minimum width of the risk area, which includes half the vehicle’s width and an additional safety margin, is 1.0 m. At that point, the number of the predictive points is set at five, while the interval between each point is set at 3.0 m.

Fig. 4 (b) shows a sample simulation scene. To examine every possible combination, an exhaustive round-robin simulation that covers all the possible scenarios is conducted. The ego vehicle’s initial state, and the width of risk area are quantized to make the combinations finite. The details are listed in Table. 1. Then simulations for every possible combination of the quantized parameters are conducted, eliminating combinations that satisfy $y_i[k] \leq y_i[k]$ since those results clearly have “no secure trajectory”. In this configuration, the total number of combinations is represented as follows:

$$5 \times 5 \times (4 + 3 + 2 + 1)^5 = 2500000$$  \hspace{1cm} (28)

The resulting population is sufficient for statistical analysis. The weight parameters for the index function of MPC (6) - (8) are set as $q_1 = 1.0 \times 10^{-2}$, $q_2 = 1.0 \times 10^{-1}$ and $r = 1.0$. These are adjusted in advance through another simulation without obstacles. Then, the MATLAB (MathWorks Inc.; Natick, Ma.) "quadprog" solver is employed to resolve QP (15) using the interior point method to perform comparative evaluations with the proposed method. If the trajectories generated by the proposed method are sufficiently close to those generated by the MATLAB solver, it is proven that the proposed method has clear utility due to its guaranteed upper computational costs.

4.2. Results

Beginning with the assumption that the [mm] order of variance is to be ignored, the proposed and the comparative method successfully generated trajectories for the entire simulation scenes. Several samples have been extracted and are shown in Fig. 5. In Fig. 5 (a), it can be seen that the vehicle that is initially oriented to left avoids the left obstacle successfully, while gradually changing its orientation. The vehicle can also pass through the gap between multiple obstacles, as shown in Fig. 5 (b). In Fig. 5 (c), it can be seen that the vehicle is able to pass through the narrow gap successfully by using large input to change its orientation. Fig. 5 (d) shows a case where the trajectory of the proposed method differs from that of the comparative method, which will be discussed in section 4.4. Then the trajectory shape is quantified by square root of the index function (13) value. Fig. 6 depicts the distributions of quantified trajectories generated by the proposed (a) and comparative (b) methods. Since the range of the quantified value was 0.00 - 1.89, the threshold value used to decide whether different trajectories had been generated was defined as 0.01 (less than 1% of the range). From these results, we can see that the trajectories generated by the proposed method were equivalent to those generated by the comparative method at a 98.4 % concordance rate. Regarding the computational cost, the proposed method required 1.9 ms on average and its worst time was 12.5 ms, while the comparative method required 3.3 ms on average and its worst time was 34.0 ms. These simulations were conducted on a personal computer with an Intel Core i7 3.4 GHz CPU and 16.0 GB RAM.

4.3. Statistical analysis for all of the results

Since the generated trajectories satisfied the obstacle constraint, meaning that there is no significant difference in terms of obstacle avoidance performance, an analysis was conducted to determine whether the residual difference in trajectory shape is statistically significant. While the distributions depicted in Fig. 6 clearly do not follow normal distributions, a non-parametric test was conducted to confirm the median and variance by employing the Wilcoxon rank sum test and Ansari-Bradley test, respectively. The confidence interval for both sides was defined as 1%. Since the results showed that there were no significant median and variance differences, it can be concluded that the proposed method generates trajectories that are statistically equivalent to those generated by the comparative method.
4.4. Individual analysis for characteristic results

Next, an analysis was conducted to determine the main factor for the residual trajectory difference by focusing on the result depicted in Fig. 5 (d) and tracing trajectory modification procedure shown in Fig. 7. In this analysis, the trajectory was modified and fixed at predictive point, in the sequence of (a) lower-3rd, (b) upper-1st and upper-4th, (c) lower-2nd, and (d) upper-5th. Since the proposed method modifies the trajectory reciprocating between lower and upper bound of sub-LCPs (25) and (26), the lower and upper modifications were also performed alternately. According to this method property, the predictive points that are located farthest from the ego vehicle tend to be fixed in an earlier phase of the algorithm. Since the trajectory’s flexibility decreases in this case, the trajectory difference is expected to increase.

As an approach for improvement, the algorithm was revised using active constraint selection. More specifically, the revised algorithm selects an applied sub-LCP checking which side of constraints can be expected to be active in the near side of the vehicle. Then, an additional one-off simulation, based on the scene
shown in Fig. 5 (d), was conducted. The results showed that generated trajectory matched the one generated by the comparative solver as shown in Fig. 8. However, since the concordance rate increment was suppressed to 98.7%, further study will be needed in the future.

4.5. Computational cost analysis

Since the upper bound of the computational complexity of the proposed method is $O(2^N)$, the computational cost was checked with respect to the number of predictive points $N$. A one-off simulation was subsequently conducted using the winding narrow pathway like Fig. 5 (c), since it was expected that this scenario would require numerous operations for the proposed method. The number of predictive points was set at 5, 10, 15, 20, 25, and 30 incrementally. Then, the computational time required by the proposed and comparative methods was verified, with respect to corresponding number of points.

Next, the practical computational costs were checked since the practical cost of the proposed method is determined by the number of active constraints. Additional simulations were conducted changing the ratio of active constraints and the number of predictive points. Defining $M$ as the number of active constraints, the ratio for the number of predictive points is $r := M/N \times 100$ [%]. These simulations were conducted for continuously winding narrow pathways using the ratios 0%, 20%, 40%, 60%, 80% and 100%, while varying the predictive points at 5, 10, 15, 20, 25, and 30, respectively.

Fig. 9 depicts the required computational time at each predictive point number with respect to active ratio of 40% and 100% using the same computer environment described in subsection 4.2. In the case of 100%, around $N = 10$, the computational time required by the proposed method overtakes the comparative solver using an interior point method. However, the computational time is still sufficiently small, even at $N = 30$, when the update cycle of general external field sensors for automobiles is taken into consideration.

On the other hand, the computational time of the proposed method is better than the conventional method for smaller $r$ and $N$. To verify the relationship between the computational time and the ratio of active constraints, we listed all the results in Table 2, where “C” refers to the “comparative” and “P” means the “proposed” method, respectively. Note that, in this table, bold font is used for the items where the proposed method performs better.
Here, it can be seen that the proposed method needs less computational time in the configuration with smaller $r$ and $N$, while the computational time of the comparative method increases linearly with respect to $N$ and is roughly constant with respect to $r$. On the other hand, the computational time of the proposed method principally increases with respect to $r$. Thus, it can be said that the proposed method is superior to the conventional method in environments with smaller numbers of predictive points and active constraints, which reflects situations that are often encountered in ordinary driving scenarios.

Based on all of the above, it can be concluded that the computational cost of the proposed method is sufficiently suitable for use in structuring real-time systems.

5. Conclusion

Herein, a novel trajectory generator for obstacle avoidance, which is based on MPC theory and its computational upper cost is predetermined, was proposed. Although the generated trajectories are quasi-optimal, in order to ensure that the performance is competitive, a comparative round-robin simulation was conducted using the interior point method to perform comparative evaluations with the proposed method. From the simulation results, generated trajectories for the two methods were equivalent at 98.4 % concordance rate and showed no statistical significant differences. Then, an analysis for the main factor for the residual difference were conducted. The revised algorithm that recovers the performance was additionally proposed and the differences were improved, although further study about this revision will be a task for the future.

Next, we investigated the computational cost property with respect to the number of predictive points and the ratio of active constraints. It was shown that the computational cost is sufficiently small in environments which reflects situations that are often encountered in ordinary driving scenarios, when the update cycle of general external field sensors for automobiles is taken into consideration. Therefore, the proposed method is competitive for use in structuring real-time systems and real-world applications.

This paper is written based on a proceeding presented at JSAE FAST-zero ’17 Meeting.

### References

1. Ziegler, Bender, Schreiber et al.: Making Bertha Drive – An Autonomous Journey on a Historic Route, IEEE Intelligent Transportation Systems Magazine, Summer, pp.8-20 (2014).

2. Nilsson, Silvin, Brannstrom et al.: If, When, and How to Perform Lane Change Maneuvers on Highway, IEEE Intelligent Transportation Systems Magazine, Winter, pp.68-78 (2016).

3. Aeberhard, Rauch, Bahram et al.: Experience, Results and Lessons Learned from Automated Driving on Germany's Highways, IEEE Intelligent Transportation Systems Magazine, Spring, pp.68-78 (2015).

4. Kelly, Stenz, Amidi et al.: Toward Reliable Off Road Autonomous Vehicles Operating in Challenging Environments,
(5) Yoon, Shin, Kim et al.: Model-predictive active steering and obstacle avoidance for autonomous ground vehicles, Control Engineering Practice, Vol.17, pp.741-750 (2009).

(6) Gao, Gray, Tseng et al.: A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles, Vehicle System Dynamics, Vol.52, No.6, pp.802-823 (2014).

(7) Liu, Jayakumar, Stein et al.: Combined Speed and Steering Control in High-Speed Autonomous Ground Vehicles for Obstacle Avoidance Using Model Predictive Control, IEEE Transactions on Vehicular Technology, Vol.66, No.10, pp.8746-8747 (2017).

(8) Bemporad, Morari, Dua, Pistikopoulos: The explicit linear quadratic regulator for constrained systems, Automatica, Vol.38, pp.3-20 (2002).

(9) Ohtsuka: A continuation/GMRES method for fast computation of nonlinear receding horizon control, Automatica, Vol.40, pp.563-574 (2004).

(10) Saitou, Imura: An M-matrix based Efficient Algorithm for Model Predictive Control of Input Constrained Linear Systems, Transactions of the Society of Instrument and Control Engineers, Vol.38, No.9, pp.906-914 (2004).

(11) Okawa, Nonaka: Quasi-optimal Trajectory Generation for Vehicle Obstacle Avoidance with State Box-constraint, Proceedings of The 4th Multi-symposium on Control Systems, Okayama, Japan (2017).

(12) Kojima, Mizuno, Yoshise: An $O(\sqrt{nL})$ iteration potential reduction algorithm for linear complementarity problems, Mathematical Programming, Vol.50, pp.331-342 (1991).