Improvement of an MPC-based Motion Cueing Algorithm with Time-Varying Prediction and Driver Behaviour Estimation

Fabio Maran [1], Mattia Bruschetta [2], Alessandro Beghi [1], Diego Minen [3]

(1) University of Padova, Department of Information Engineering, via Gradenigo 6/B, 35131 Padova, Italy, e-mail: {name.lastname}@dei.unipd.it
(2) University of Padova, Human Inspired Technology Research Center, via Luzzati 4, 35122 Padova, Italy, e-mail: mattia.bruschetta@unipd.it
(3) VI-Grade s.r.l., via G. Galilei 42, 33010 Tavagnacco (UD), Italy, e-mail: diego.minen@vi-grade.com

Abstract – *Dynamic simulators have recently become common devices in the automotive industry, their aim possibly being virtual prototyping, training, medical studies, R&D. The critical point is the capability of providing the final user with a driving feeling as realistic as possible, in order to exploit the device at best. This is the task of the Motion Cueing Algorithm (MCA), which calculates the correct motion displacements to be provided to the motion controller, while assuring not to exceed the working operational area. Model Predictive Control has been successfully applied to MCA design, being well suited for the particular problem, where optimal performance is required while respecting physical boundaries. Nevertheless, the predictive aspect of the algorithm has been developed only in a trivial way, due to the evident difficulties in predicting the behaviour of the whole system, since it includes a human in the loop, and hard real-time requirements. In this paper, we present a solution to apply real-time, realistic prediction to a MCA based on MPC, where the variability introduced by the driver is taken into account. In particular, the proposed technique can effectively handle possible unexpected driver's behaviour and can be adapted to the driving ability of the specific user.*

Keywords: MPC, Time-Varying Prediction, Move Blocking, Motion Cueing, Driver Behaviour

Introduction

The automotive industry is one the fields that is most impacted by continuous technological innovation. In the research and development process, test and validation are crucial phases, in particular with automotive applications where the presence of a human driver in the loop has to be taken very carefully into account. Testing on prototypes is possible but in general not convenient nor desirable, because of their high cost and the risks the driver would be subject to. These tasks are now often performed using *Driving simulators*, and dynamic platforms (where the device itself is able to move within a defined space) are seen with increasing interest in the automotive world. Other than the explained R&D purposes, these platforms come in useful for a variety of other applications, such as physical tests and rehabilitation for injured drivers and elderly people, or training for racing drivers. It is clear that the effectiveness of this particular class of devices is directly related to the quality of the motion sensations the driver would have during the use, so that the point is not to reproduce 1:1 the vehicle behaviour, but rather to provide in the best possible way the *driving feelings* that the user would perceive, and in some case to reproduce the driver's reaction. The procedure responsible for this task goes under the name of *Motion Cueing Algorithm* (MCA). The platform displacements should always be kept within the operative working area, as well as each one of the actuators, in order to prevent unnatural movements or damage to the device and the user itself. This particular aspect is known as *Washout Action*.

During the years different approaches have been applied to the design of MCAs, mainly based on high-pass and low-pass filtering the signals from the computational engine to define which information is more relevant to reproduce the driver feelings. This approach has many limitations, as examined in [Bas11], [Rey00], mostly due to the lack of information about the process inside the controller, and the consequent difficulties in rendering the demands of the driver. Moreover, the constraints cannot be directly taken into account, so that only conservative approaches can be used to perform the Washout Action [Beg12].

Model Predictive Control is instead characterized by many of the features that are useful to the specific problem, and has been used in the last years to design MCAs [Dag09], [Aug09]. In particular, in [Bas11], [Beg12] we proposed an MPC based MCA that has been implemented on a real device. The critical aspects that improve the quality of the MC are, among others

- model-based approach: the procedure integrates a model of the human perceptual system, so the produced signals are reliable in terms of motion sensation. Moreover, if a model of the platform is available it can be integrated in a natural way;
- constrained procedure: MPC takes into account the presence of constraints, so limitations on actuators, working area, thresholds and similar can be explicitly set;
- optimal-control technique: the goal is the minimization of a weighted cost function, so by setting it in the proper way one can reach the desired performance. In particular, by manipulating the weights, different behaviours can be achieved, according to the needs of the user (tuning phase).

The result is a flexible control strategy that is very effective in terms of motion reproduction, working area exploitation, and safety management while preserving the real-time requirement to work on real devices [Beg12]. In that first implementation, the prediction step has been handled in a trivial way, common in industrial MPC application when a reliable reference is not assured: The prediction window is limited in size and the signals are kept constant.

It is clear that having information about the future trajectories of the system could increase the overall performance, in particular in terms of improving the working space management with immediate benefit for motion reproduction. As an example, if the algorithm is given the information that a braking will occur, it will move the platform to gain as much linear displacement as possible, so that the specific action is reproduced with a greater amplitude. Tilt coordination (namely, using slow rotations to exploit gravity to provide low-frequency, “false” linear accelerations [Beg12]) can also be significantly improved by a smart management of the future reference. However, the use of non-trivial references requires a wider prediction window, hence an increase in the size of the optimization problem and the computational burden as well, thus making the fulfillment of the real-time constraint more complicated. Moreover, the derivation of reliable references is by itself a challenging task: even in the (common) case of driving on a closed track, repeatability of the driver actions is an issue. Only the most skilled drivers have repeatable enough driving behaviour to ensure that a single telemetry can be taken as a valid reference signal, while a less experienced or a non professional driver is likely to vary the driving behaviour significantly between subsequent laps. In [Beg13] we proposed solutions for both issues. To ease the computational complexity a *move-blocking* technique [Lon11] has been applied, where the calculation of a smaller, sub-optimal control sequence is computed at each time step, by dividing the control window in segments each one with constant control signal. The second problem has been addressed by considering a professional driver, hence with a high reliability in terms of repeatability, and using a benchmark trajectory with small real-time modifications to adapt to the actual driving behaviour.

In the present work, we aim to extend the prediction capabilities of the algorithm, by considering more generic users, hence with possibility of handling large difference in trajectories (positions, velocities, accelerations) between two consecutive laps. One of the key features of the proposed approach is the capability of switching in real-time from the use of time-varying prediction to a more conservative approach with constant references on a small window. This is required to obtain an algorithm that can be implemented on a real device for a non-racing context: the actual behaviour is compared to the current tracking signals, and if the error is too large, the prediction strategy is switched to a conservative one, very close to the non-time varying, small prediction window case. The switching has to be performed by respecting some requirements associated with the use in real-time of the optimisation algorithm [Fer14], such as avoiding a full re-initialization of the quadratic programming problem and the consequent real-time constraint violation. To this aim, some parameters have to be kept constant, such as the length of the prediction window and the weights of the cost function.

The simulator

The proposed algorithm has been tested on a specific device, *Driver in Motion* (DiM) by VI-Grade & Saginomiya (www.driverinmotion.com). This new concept platform privileges the DOFs of interest for automotive purposes, i.e. longitudinal and lateral displacements and yaw rotations. The mechanical structure (Fig. 1) consists of a hexapodal structure mounted on a tripod frame sliding on special air/mag pads on an extremely even and stiff steel surface. The system allows for satisfactory results in physical simulation with a relatively small size assembly. Great part of longitudinal, lateral, and yaw movements are performed by the tripod, while the hexapod is used for pitch, roll and vertical ones, together with small longitudinal, lateral, and yaw movements. Such redundancy increases the overall bandwidth and motion envelope, while maintaining compact dimensions.

The simulated vehicle dynamics physical engine is considered the state-of-the-art in the field and provides a highly reliable representation of the real vehicle behaviour. The screen covers over 200deg angle and the projected image moves in proper coordination with the platform to guarantee full immersion of the driver in the virtual environment. Force feedback on the steering wheel and the braking system improves the driver's feeling of the vehicle behaviour. Tab. 1 highlights the platform dynamic performance and operational space.

The MC strategy has to provide the displacement references to the control system of the platform, which is assumed to be able to perfectly track the signals with a fixed time delay.

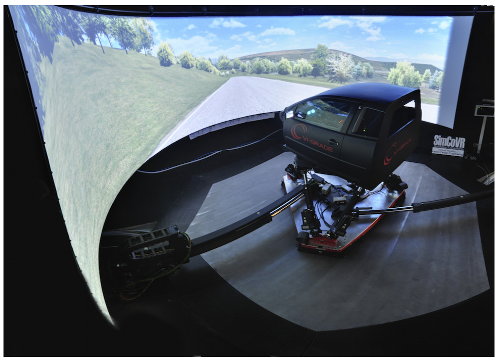


Figure 1. Driver in Motion simulator

Table 1. Platform Tripod (t) and Hexapod (h) performance

DOF	Position	Velocity	Acceleration
x_t	±0.8 m	1.7 m/s	12 m/s^2
y_t	±0.75 m	1.5 m/s	10 m/s^2
Yaw, ϕ_t	±25 deg	165 deg/s	900 deg/s^2
x_h	±0.28 m	2 m/s	25 m/s^2
y_h	±0.25 m	1.7 m/s	25 m/s^2
z_h	±0.22 m	1.6 m/s	35 m/s^2
Roll, ψ_h	±20 deg	135 deg/s	2500 deg/s^2
Pitch, θ_h	±20 deg	130 deg/s	2000 deg/s^2
Yaw, ϕ_h	±20 deg	135 deg/s	3000 deg/s^2

Move blocking strategy

The many advantages of applying MPC to the Motion Cueing problem have been thoroughly analyzed, and insights and results can be found in our previous works [Bas11], [Beg12]. In brief, the idea is to use a linear mathematical model of the human perceptive (*vestibular*) system together with a simple model for the mechanics in order to derive reliable perception signals hence improving the tracking action. Moreover, constraints are specified in regards to both perceptive aspects (e.g. perception thresholds [Hou05], motion inversion limitation) and the platform structure (e.g. limits on the working area and actuators capabilities).

One of the key aspect of MPC applied to Motion Cueing problem is the definition of the cost function *J(t)* in the optimisation step. In the situation at hand, a quadratic cost function is chosen as follows

$$J(t) = \sum_{j=1}^{N_P} \delta(j)\big(y(t+j|t) - r(t+j)\big)^2 + \sum_{j=0}^{N_C-1} \lambda(j)u(t+j)^2 + \sum_{j=0}^{N_C-1} \gamma(j)\Delta u(t+j)^2 \qquad (1)$$

where $y(t+j \mid t)$ is the predicted trajectory, $r(t+j)$ is the future reference (the perceived accelerations and rotational velocities) over the prediction window of size N_P, $u(t+j)$ and $\Delta u(t+j)$ are the future inputs (the vehicle accelerations and rotations) and *input difference*, respectively, over a control window of size N_C; δ, λ, γ are weight values.

The improvement given by using a non-constant reference signal becomes relevant if the prediction window length is large enough to allow for a significantly better exploitation of working area. Therefore, a larger number of samples N_P has to be considered and consequently a larger optimal input sequence has to be found, so that the size of the optimization problem is increased, directly affecting the resolution time which typically grows exponentially in the instance size [Boy04]. Hence, on the one hand the prediction window length has to be as large as possible for the non-constant predictive action to be effective, on the other hand, the number of samples can not be too large to preserve real-time capabilities. To overcome this limitation, a *move-blocking* strategy is considered. This is a common solution in the literature [Lon11], [Cag04], based on the idea of limiting the size of the optimization problem by reducing the number of *free inputs*, obtained via the use of different sampling frequencies that decrease while moving away from the initial condition and keeping the values between two steps constant. In particular, by applying this strategy on $\Delta u(t)$, the corresponding input $u(t)$ will have a first-order interpolation. From a mathematical point of view, the N_C variables of the original problem are reduced to $N'_C < N_C$ possible different values via a Kronecker product

$$\Delta U(t)' = (T \otimes I_M)\Delta U(t), \qquad (2)$$

with $\Delta U(t)' = [\Delta u(t)', \ldots, \Delta u(t+N'_C-1)]^T$ and analogously for $\Delta U(t)$. T is a matrix of ones and zeros only, with each row containing exactly one non-zero element disposed in “stairs” structure [Cag04]: e.g., to reduce a problem from $N_C = 5$ to $N'_C = 3$ with the last three samples equal among each other, T is given by

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The problem is modified by substituting Eq. 2 in Eq. 1, and adding Eq. 2 as a constraint.

The use of move-blocking may have an important draw- back in terms of problem feasibility, namely, the calculated optimal sequence is not optimal for the original problem, possibly leading to unfeasibility in the subsequent optimization steps. In particular, when using a hot-start based solver, the information derived from the previously computed solution is exploited to solve the new iteration. Given that the sequence in Eq. 2 is an approximation of the exact solution, the risk of unfeasibility increases. The regulation of the constraints and weights of the cost function, that is the *tuning* of the algorithm, has to be carried out very carefully in order to preserve the feasibility of the solution.

Prediction Strategy

To be effective, prediction-based control strategies require the availability of reliable reference signals, that may be hard to obtain when the system involves fast dynamics and signals variability is high. In the current application, moreover, the presence of a human in the loop obviously introduces potentially unpredictable behaviours. When driving on closed circuits or more generally on pre-defined tracks (as is often the case in vehicle development or driver training applications) the presence of repetitive patterns helps in defining suitable reference trajectories. In such cases, professional or highly skilled drivers operate the platform so that the difference between two consecutive laps is small enough that a pre-defined benchmark signal can be used as a reliable reference for prediction [Beg13].

If this is not the case, it is necessary to devise appropriate strategies for making effective use of suitable prediction signals. In the present work, a generic procedure for managing the tracking signal is introduced, with the specific task of providing an algorithm that is reliable and efficient in terms of safety, motion envelope exploitation, tracking quality, and computational burden. In particular, the use of prediction in a MCA should naturally introduce *prepositioning* strategies in the algorithm, i.e. if it is known beforehand what the next maneuver will be, it is possible to anticipate the next movement of the vehicle and move the platform to a point that will increase the workspace in the required direction. As an example, if it is know that within the next *n* seconds a strong braking will occur, the device could be slowly moved forward combining the movement with tilt coordination to assure that at the time of braking, all the available linear space would be used. Tilt coordination itself could be more effectively used by exploiting anticipated knowledge of the future actions.

Future reference generation and switching strategy

It is trivial to understand that when the actual behaviour is different from the expected one, e.g. due to delays or different magnitude of the signals, the output of the MPC is not acceptable, and a more conservative, *Non-Look-Ahead* (NLA) strategy with constant and small-windowed prediction

is to be preferred. This situation can be worsened by the possible loss of optimality introduced by move blocking. Starting from this concept, we use as tracking signal *r(t)* defined as

$$r(t) = \begin{cases} \bar{r}, & t \le T_{LA} \\ k_{LA} \cdot r_b(t) & t > T_{LA} \end{cases} \tag{4}$$

where $\bar{r}$ is a constant reference vector, correspondent to the current outputs of the system, $r_b(t)$ is the future benchmark and k_{LA} (LA stands for *Look-Ahead*) is a coefficient. In this way, a natural balance arises between what the driver is doing and what it is expected from them. The cost function is modified in the appropriate way by acting on the *weights* of the tracking errors, that have to be regulated differently depending on which part of the reference they are influencing. A crucial role is clearly played by the value of the parameter T_{LA}.

If, however, the driver is acting in a very different way with respect to the expected behaviour, it is preferable not to use prediction at all, so that a switching strategy between the time-varying and constant reference cases has to be included in the algorithm. In this regard, it is important to observe that a linear MPC with a quadratic cost function is typically reformulated as a constrained Quadratic Programming problem (QP) [Wan09], [Bas11] to be solved by dedicated solvers. In [Bas11], [Beg12] it is proposed to use `qpOASES`, an efficient and open-source real-time solver implemented in `C++` [Fer14] and based on an on-line, hot start Active-Set strategy. Given the increased complexity and computational burden needed to solve the LA strategy, this kind of resolution approach is even more crucial to reach real-time performance. `qpOASES` solves constrained QP problems in standard form

$$\min_\xi \xi^T H \xi + \xi^T g \tag{5}$$

$$\text{s.t.} \quad b_{LB} \le A\xi \le b_{UB} \tag{6}$$

and the hot-start strategy requires that only parameter g can be modified on-line in order to minimize the computational time. All the other parameters in Eq. 5, which values depend on the weights of the original cost function (Eq. 1), are "hardcoded" so the trivial solution of switching from the LA strategy to the NLA one is not "real-time" feasible, asking for a full restart of the problem. The only variables that can be manipulated are the ones fed to the algorithm, in particular the reference signals themselves.

The switching strategy is then implemented as follows. The coefficient k_{LA} is set to zero, so that only the constant component of the reference, $\bar{r}$ is "active". A number of tests are then performed to find the set of *weight values* that drive the system performance as close as possible to the NLA case. Note that the two considered procedures are intrinsically different, i.e. it is impossible to find an equivalent tuning configuration. In the application at hand, we consider as "*standard*" the case detailed in [Beg12] where the prediction is constant for 0.3s, and try to find the best match by acting at the same time on the weights of the cost function and the value of T_{LA}. The results in Fig. 2 illustrate the analysis conducted along the longitudinal/pitch DOF (other DOFs are omitted due to the lack of space), where the LA case is tuned to be as close as possible to the *standard* one by varying T_{LA}, the weight values on the prediction window.

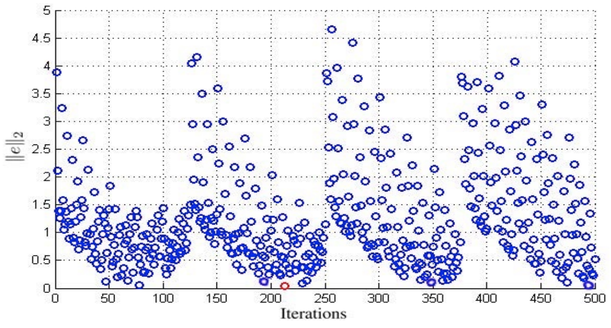
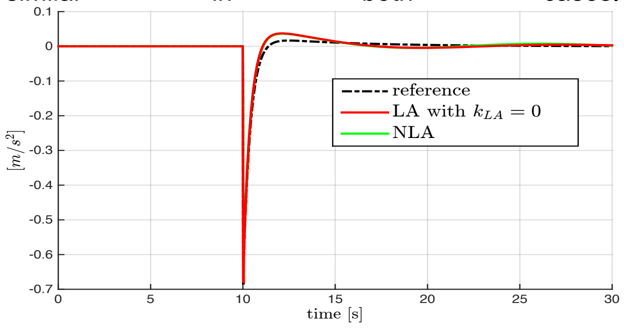


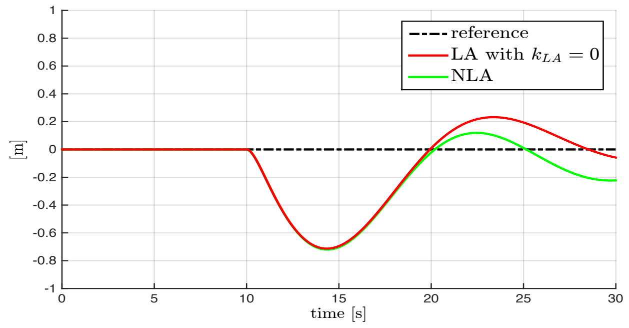
Figure 2. Analysis of position errors with different configuration of parameters in the LA case w.r.t. the *standard* one

The aim is to maximize the value of the weight on the perceived acceleration $\hat{w}_x$ (to increase the performance when $k_{LA} > 0$), while minimizing the 2-norm of the position error. More than 500 iterations have been performed to choose the ideal setup (the red circle in Fig. 2). The resulting setup is called *equivalent tuning*.

In Fig. 3 the comparison between the algorithm behaviour in the standard and LA case is depicted, showing that the performance of the system is very similar in both cases.



(a) Longitudinal acceleration



(b) Longitudinal position

Figure 3. Motion reproduction and platform exploitation with *equivalent tuning* configuration of parameters

The reported analysis indicates that there exist some weight combinations that allow the system to perform in a conservative way, hence it is possible, if required by critical situations, to switch to a safer configuration while guaranteeing the continuity of the procedure, that is, without stopping the algorithm letting the driver be always in full control while carrying on the dynamic simulation. The only parameter to act on for the switch is the gain of the LA part of the reference, hence the transition between the two strategies is immediate and simple. Moreover, acting on different values of k_{LA} can modify the reference to be easy adapted to each driver, as will be illustrated in the next paragraph.

Reference signal management

The reference $r(t)$ has to be provided, in the specific application, as indicated in Eq. 4. It is important to evaluate how the MCA performs when the actual driver behaviour will determine discrepancies with respect to the benchmark. In particular, the critical case are the delays (possibly positive or negative) that can affect a specific maneuver. For instance, considering a braking or a turn, a non-expert driver could anticipate or defer the action introducing misalignments that could result in unsatisfactory performances. As a first compensation, an online delay management is introduced, which compares the current behaviour with the benchmark signal and, by evaluating the 2-norm error, derives the most appropriate time-shift to be applied. Nevertheless, the presence of a safety procedure is still required. The proposed switching-strategy assures a recovery action by nullifying the time-varying part of $r(t)$. Similarly, it is expected that manipulating the scale factor k_{LA} will have an effect on the overall performance, providing somehow an adaptation to different levels of precision in driving. This aspect will be analysed and discussed in the following Section.

Results

To identify different sets of parameters that influence the behaviour of the system when prediction is active, depending on the skill of the user, a set of simulations have been performed. As a performance index, the variation of the input gain G on vehicle accelerations with respect to its value G_0 in the standard case has been chosen. To motivate such choice, let us consider the standard case with constant prediction on a small time window. The original signal coming from the simulated vehicle must be necessarily scaled before being fed to the algorithm, to match the platform capabilities. If the algorithm cannot provide a reliable prepositioning strategy, i.e. a reliable prediction, the value of G has to be reduced until it is granted that the platform will not reach its limits, to assure continuity of motion and prevent inconsistent behaviours. The more efficient the predictive action is, the less the input signals have to be scaled down, hence improving the device working space exploitation. Therefore, to assess the effect of variation of k_{LA} in the time-varying prediction strategy, we evaluate the maximum value that G can take with respect to its optimal value G_0 in the standard case described in [Beg12], expressed as percentage:

$$G_{\%} = \frac{G - G_0}{G_0} \cdot 100.$$

Moreover, we are interested in assessing the effects of errors introduced by anticipations or delays of the current action with respect to the benchmark trajectory. So the analysis are conducted by considering different values of the time-translation Δp.

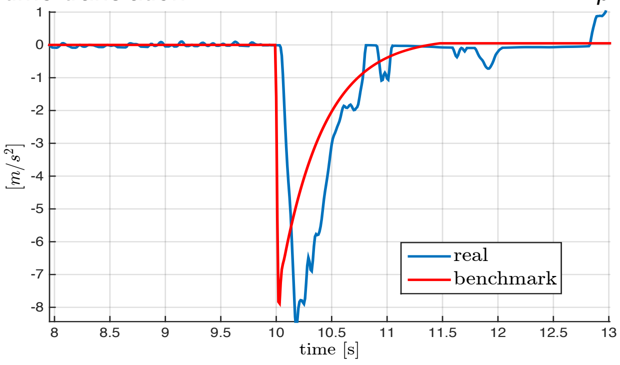


Figure 4. Benchmark acceleration signal vs. real one

We report only results regarding the coupled longitudinal/pitch DOF, although the analysis has been carried out for all the DOFs, since it is a particularly significant one, being associated with the management of braking and accelerating maneuvers. The benchmark signal is similar to a real braking one and it is shown in Fig. 4. The parameters of interest vary in given intervals, i.e. $k_{LA} \in [0, 1]$ and $\Delta p \in [-1, 1]$ seconds. The chosen prediction (and control) window is 8s, since it has been tested to be the maximum prediction horizon that maintains the feasibility of the required real-time constraints (100 Hz control frequency). Accordingly, the move blocking strategy uses a 100 Hz sampling in the first 0.5s of the prediction/control horizon, and 10 Hz for the remaining range.

Figure 5 shows the variation of $G_{\%}$ as a function of Δp for different values of k_{LA}. The influence of the variation of Δp on the final result is evident: with $k_{LA} = 1$ and Δp next to 0, the overall performance is noticeably increased by approximately 250%.

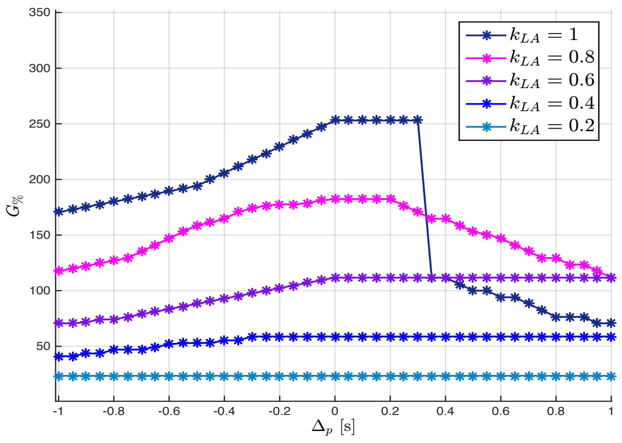


Figure 5. Performance analysis of equivalent tuning, different values of k_{LA}

The results also show how the effect of Δp is not symmetric, with a very abrupt degradation in performance with $k_{LA} = 1$ for a delay greater than 0.3s. The Figure enlightens another crucial aspect. If $k_{LA} = 0.8$ we have a high efficiency of the algorithm with greater delays. This indicates the effect of a reduced amplitude of the expected behaviour in the described framework is a suitable setup for a wider set of users, even less skilled, provided that the advantage in using prediction is reduced. Fig. 6 shows how different set of tuning parameters can be classified depending on driver's skills.

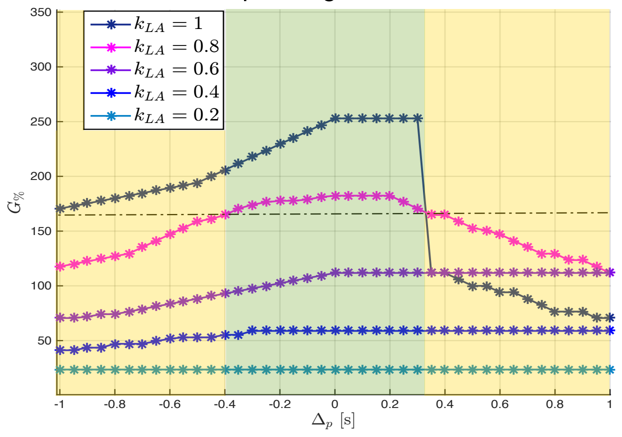


Figure 6. User classification for equivalent tuning: green area corresponds to more expert driver

In this sense, the indicated procedure can adapt in an easy way to the user that will perform the simulation and its capabilities. More precisely we can say that the advantage in using prediction is an increment of 165% compared to the standard approach if the driver is capable to guarantee repeatability within -0.4 and +0.3 seconds.

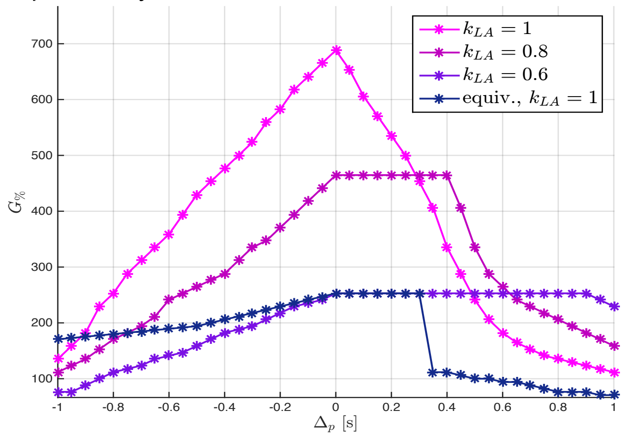


Figure 7. Performance analysis of optimal tuning, different values of k_{LA}. Comparison with equivalent tuning is reported

As a further remark, it is interesting to evaluate the potential of the algorithm when no switching-strategy is adopted, hence with a tuning conceived to achieve the best possible performance with active LA. This is to understand the full potential of applying a reliable prediction to the specific problem. Fig. 7 shows how the quality of motion reproduction would dramatically increase with a full exploitation of prediction: the *optimal tuning* provides an increase in performance that reaches the 700% compared to the standard case. The *equivalent tuning* is reported for comparison with $k_{LA} = 1$. Note that, as expected, this setup is more susceptible to driver errors: the gain is maximum only for $\Delta p = 0$ and decreases with non-null values more rapidly than in the previous case. Similarly to the equivalent tuning case, different classes can be defined from this analysis for the k_{LA} value, depending on the skills of the user (Fig. 8). These results suggest that, accepting the risk of an abrupt stop of the motion when a reliable driver's behavior is not available, the performance can be increased considerably.

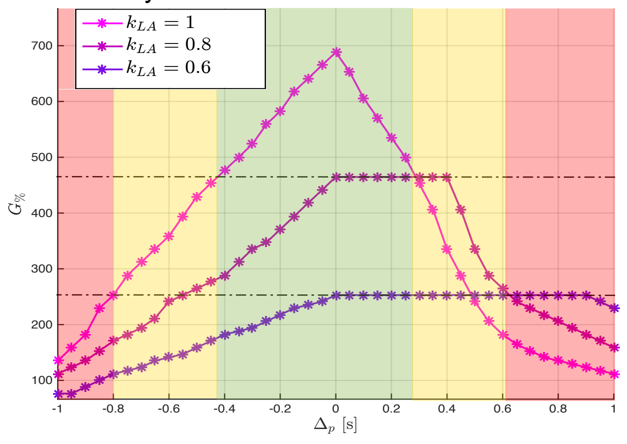


Figure 8. User classification for optimal tuning: green area corresponds to a decrease in driver's ability from the more to the less expert

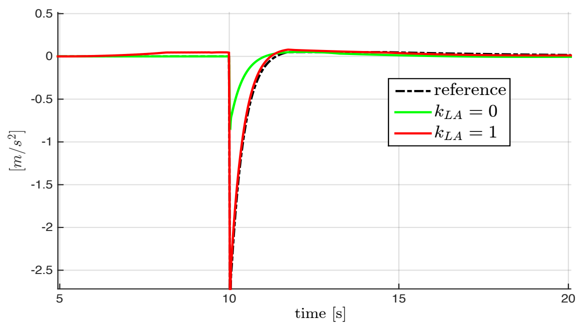
This optimal configuration can be considered a development of the classical prepositioning, given that it adapts in a natural way to slight modifications on driver's action.

Finally, Fig. 9 illustrates the effect of prediction in terms of motion reproduction and working area usage by comparing the equivalent tuning behaviour with $k_{LA} = 0$ and $k_{LA} = 1$. In the latter case, the acceleration signal is tracked almost perfectly since the algorithm exploits the information known in advance to slowly move the platform in the direction opposite to the one of the braking. It is important to remark that such a big advantage in using prediction can be explained by a combination of factors. Time-varying prediction, besides providing prepositioning, makes the tilt coordination more effective (in Fig. 8(d) the pitch angle corresponding to the tilt-coordination action is reported in the two cases). Moreover a proper velocity is reached synchronously and consistently with the expected event, allowing a greater acceleration within the same working area. This is obtained while maintaining the false cues under the perception threshold: Fig. 8(c) shows the pitch velocity corresponding to the tilt action, which is maintained well below the perception threshold of 0.05 rad/s [Hou05].

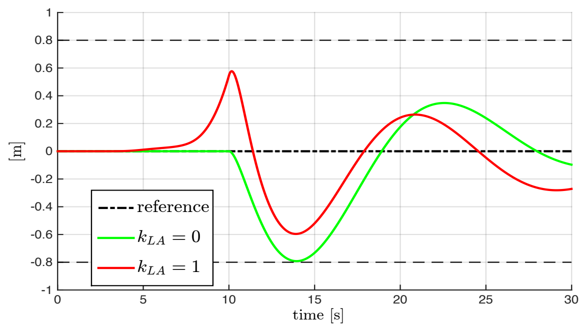
From a computational point of view, the simulations have been performed within MatLab environment, on a common PC with i3 processor, 4 GB RAM and Microsoft Windows as OS. Considering a telemetry of 80s, we have that with a prediction window of 8s and a control frequency of 100 Hz, the time required for a complete run is of 1743.8s, i.e. approximately 21.8× the desired real-time constraint. By using the described move-blocking, i.e. 0.5s at 100 Hz and the remains 7.5s at 10 Hz frequency, the whole computation is reduced to 41.18s, that is 0.005s ca. for each step, safely below the required maximum time of 0.01s per step.

Conclusions

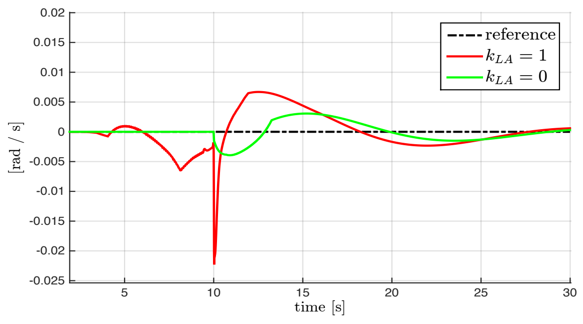
In this work an evolution of a time-varying prediction strategy for a linear MPC-based MCA is presented. The prediction is designed so as to provide a compromise solution between the current driver behaviour and a benchmark one. A thorough analysis has been conducted to study the effect of the key tuning parameters on the algorithm performance while granting a smooth transition between the Look-Ahead and Non-Look-Ahead case. The solution has real-time capabilities and does not require stopping the simulation and re-instantiating the procedure. Finally, the increase in performance has been tested in terms of working area exploitation and motion reproduction on a representative case for the coupled longitudinal/pitch DOFs. An interesting remark is



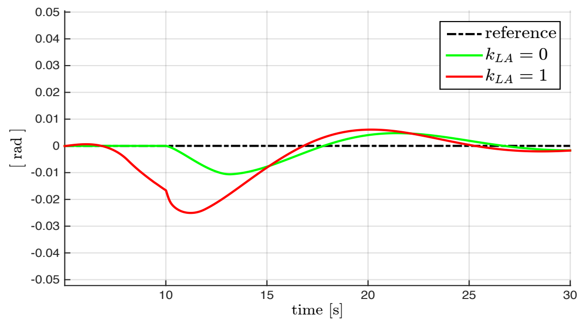
(a) Longitudinal acceleration



(b) Longitudinal position



(c) Pitch velocity



(d) Pitch position

Figure 9. Improvement in device exploitation using prediction (k_{LA} = 1) vs. The conservative case (k_{LA} = 0)

that the algorithm can be adapted to the driver skills.

Future work will focus on improving the transition between the LA and the safer case exploiting the recognition of predetermined events, on improving the management of errors in repeatability (e.g. considering scaling of dilatation of the signals), and introducing a stochastic approach that could take care of the errors both in the references and the model.

References

Augusto B. and Loureiro R., **Motion cueing in the chalmers driving simulator: A model predictive control approach**, *Master's thesis, Chalmers University of technology*, 2009.

Baseggio M., Beghi A., Bruschetta M., Maran F., and Minen D., **An mpc approach to the design of motion cueing algorithms for driving simulators**, in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, oct. 2011, pp. 692 –697.

Beghi A., Bruschetta M., and Maran F., **A real time implementation of mpc based motion cueing strategy for driving simulators**, in *Decision and Control (CDC), 2012 51st International IEEE Conference on*, dec. 2012.

Beghi A., Bruschetta M., and Maran F., **A real-time implementation of an mpc-based motion cueing strategy with time-varying prediction**, in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4149–4154.

Boyd S. and Vandenberghe L., **Convex Optimization**. New York, NY, USA*: Cambridge University Press*, 2004.

Cagienard R., Grieder P., Kerrigan E., and Morari M., **Move blocking strategies in receding horizon control**, in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 2, Dec 2004, pp. 2023–2028.

Dagdelen M., Reymond G., Kemeny A., Bordier M., and Maïzi N., **Model-based predictive motion cueing strategy for vehicle driving simulators**, *Control Engineering Practice*, vol. 17, no. 9, pp. 995– 1003, 2009.

Ferreau H., Kirches C., Potschka A., Bock H., and Diehl M., **qpOASES: A parametric active-set algorithm for quadratic programming**, *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

Houck J. A., Telban R. J., and Cardullo F. M., **Motion cueing algorithm development: Human-centered linear and nonlinear approaches**, *NASACR*, vol. 213747, no. May, 2005.

Longo S., Kerrigan E., Ling K. V., and Constantinides G. A., **A parallel formulation for predictive control with nonuniform hold constraints**, *Annual Reviews in Control*, vol. 35, no. 2, pp. 207 – 214, 2011.

Reymond G. and Kemeny A., **Motion cueing in the renault driving simulator**, *Vehicle System Dynamics*, vol. 34, no. 4, pp. 249–259, 2000.

Wang L., **Model Predictive Control System Design and Implementation Using MATLAB**, 1st ed. *Springer Publishing Company*, Incorporated, 2009.