2020 59th IEEE Conference on Decision and Control (CDC)
Jeju Island, Republic of Korea, December 14-18, 2020

Adaptive predictive control for pipelined multiprocessor image-based control systems considering workload variations

Sajid Mohamed, Nilay Saraf, Daniele Bernardini, Dip Goswami, Twan Basten and Alberto Bemporad

***Abstract*— Image-based control (IBC) systems have a long sensing delay. The advent of multiprocessor platforms helps to cope with this delay by pipelining of the sensing task. However, existing pipelined IBC system designs are based on linear time-invariant models and do not consider constraint satisfaction, system nonlinearities, workload variations and/or given inter-frame dependencies which are crucial for practical implementation. A pipelined IBC system implementation using a model predictive control (MPC) approach that can address these limitations making a step forward towards real-life adaptation is thus promising. We present an adaptive MPC formulation based on linear parameter-varying input/output models for a pipelined implementation of IBC systems. The proposed method maximizes quality-of-control by taking into account workload variations in the image processing for individual pipes in the sensing pipeline in order to exploit the latest measurements, besides explicitly considering given inter-frame dependencies, system nonlinearities and constraints on system variables. The practical benefits are highlighted through simulations using vision-based vehicle lateral control as a case study.**

I. Introduction

Image-based control (IBC) systems refer to a class of data-intensive feedback control systems whose feedback is provided by camera sensor(s) (see Figure 1). The combination of camera sensor(s) and image processing algorithms is capable of detecting a rich set of features in an image that can be used to compute the states of the system such as relative position or distance, depth perception, and tracking of the object-of-interest [1]. The challenge, however, is that there is an inherent long sensing delay due to compute-intensive image processing algorithms [2].

A typical implementation of an IBC system uses an optimal linear quadratic regulator (LQR) [3] considering the worst-case image workload and thus has worst-case sensing delay [2] (illustrated in Figure 2). However, this results in poor effective resource utilisation in a multiprocessor platform, and suboptimal quality-of-control (QoC) [4], [5]. Multiprocessor platforms with high processing power that allow parallel and pipelined executions can be used to cope

This research has received partial funding from the EU's H2020 framework programme under grant agreement no 674875 (oCPS) and the FitOptiVis project funded by the ECSEL Joint Undertaking under grant number H2020-ECSEL-2017-2-783162.

S. Mohamed, D. Goswami and T. Basten are with the Electronic Systems group, Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands. Email: {s.mohamed,d.goswami,a.a.basten}@tue.nl.

N. Saraf and A. Bemporad is with IMT School for Advanced Studies Lucca, Piazza San Francesco 19, Lucca, 55100 LU Italy. Email: {nilay.saraf@alumni.,alberto.bemporad@}imtlucca.it.

D. Bernardini is with ODYS S.r.l., Via A. Passaglia 185, Lucca, 55100 LU Italy. Email: daniele.bernardini@odys.it.

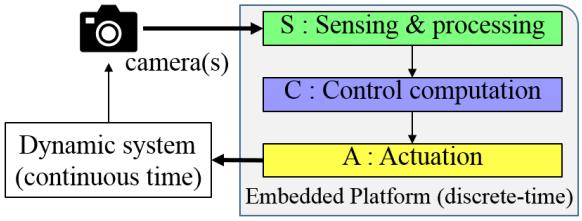

Fig. 1: An image-based control (IBC) system: block diagram

with this long worst-case sensing delay. The sensing algorithms may be parallelised (whenever possible, but limited by the degree of application parallelism) if the algorithmic structure is known (white/grey box) and thus reduce the worst-case sensing delay [2], [6]. Pipelined control implementation executes the sensing algorithm in a pipelined fashion with the worst-case sensing delay and thus reduces the effective actuation and sampling rate [7], [8]. The advantage of a pipelined implementation is that it is applicable even if the application algorithm is a black box. Note that for nominal control implementation, the sensor-to-actuator delay τ is at most equal to the sampling period h, whereas for a pipelined control implementation τ is greater than h.

However, pipelining is limited by inter-frame dependencies, i.e. the data or algorithmic dependencies between consecutive frame processing, e.g. due to video coding [9] or visual tracking [10]. Inter-frame dependence time (denoted by f_d) can be quantified for the current image frame as the maximum time required to complete the processing of (parts of) the algorithm the subsequent image frame processing depends on. Alternatively, f_d is the maximum time required to wait between processing consecutive image frames.

The sensing delay due to the compute-intensive processing (sensing) of the image stream is dependent on image workload variations, which occur due to image content and result in a wide range between best-case and worst-case image-processing times. It is known from [4], [5] that explicitly taking into account workload variations in controller design improves the QoC. Workload variations are typically considered as a variable delay or stochastic delay in standard sampled-data linear control design techniques [11].

In current literature, workload variations are typically considered only for sequential IBC implementation [5], [6], [12] and not for pipelined implementation [7], [8] (see Figure 3). In [13], pipelining is considered along with variable delay but the inter-frame dependencies are neglected. Further, these approaches do not consider system nonlinearities, i.e. the variations in system dynamics, and constraints imposed on the system variables, which can be crucial when considering

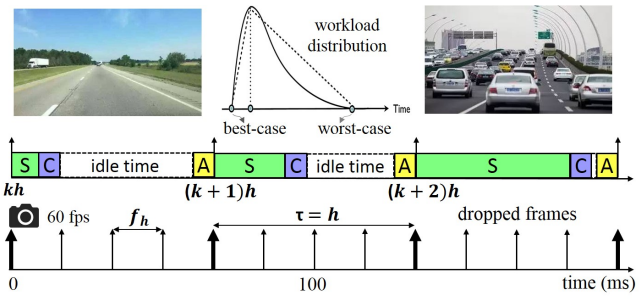

Fig. 2: Illustration of a workload distribution and a sequential IBC implementation considering worst-case image workload.

a practical implementation. E.g. the maximum steering angle of the Udacity self-driving car is set to +/- 25 degree [14] and the vehicle velocity is kept constant for the simulations in [4], [15].

The main motivation of this paper is to address limitations of the state-of-the-art pipelined multiprocessor IBC approaches which do not take into account inter-frame dependencies, system constraints and nonlinearities for the application of interest. These limitations make it difficult for these approaches to be realised in real systems.

Contribution: We present an adaptive predictive control formulation based on linear parameter-varying (LPV) input/output (I/O) models for a pipelined multiprocessor implementation of IBC systems while considering workload variations, inter-frame dependencies, system nonlinearities and constraints, and thus makes a step forward towards real-life adaptation. Further, we compare the proposed formulation with the state-of-the-art multiprocessor IBC system implementations.

Recent advances in numerical optimization for MPC have enabled safety-critical applications on embedded platforms, such as engine control and powertrain coordination in the automotive domain [16], [17]. Moreover, latest methods such as those recently reported in [18] suggest that the model and MPC tuning parameters can be adapted at run time without reconstructing the optimization problem. This allows implementing adaptive MPC with the same computational complexity as the non-adaptive case. An MPC formulation is thus advantageous for use in image-based control systems where, due to constraints, nonlinearities and workload variations, an adaptive control method that maximizes control performance is desirable.

The paper is organized as follows. Section II describes the application of interest i.e. pipelined image-based control systems. It details how inter-frame dependencies, the available number of processing cores and workload variations influence control requirements and performance. Next, a modelling approach for such systems is described in Section III. Section IV includes details on the proposed MPC formulation. The proposed method is then tested in simulation on a vision-based lateral control system, including a comparison with a baseline approach based on a worst-case estimate for sensing delay. The experimental simulation results and comparison with the state-of-the-art are discussed in Section V before concluding the paper in Section VI.

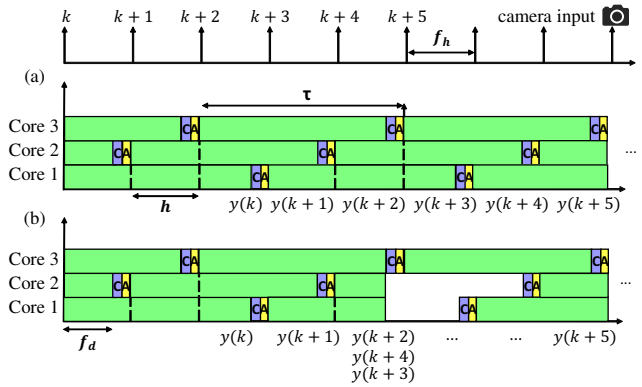

Fig. 3: Illustration of pipelined IBC system implementation with constant sampling period h and: (a) with constant worst-case sensing delay; (b) considering workload variations.

II. Pipelined IBC system implementation

We consider a typical setting for an IBC system as shown in Figure 1 having the workload distribution as illustrated in Figure 2. The main sensor is a camera module that captures the image stream. The image stream is then fed to an embedded multiprocessor platform at a fixed frame rate per second (fps), e.g. 60 fps, and image arrival period f_h is given by $f_h = 1/60 = 16.67$ ms. The tasks in such an application primarily include compute-intensive image sensing and processing (S), control computation (C) and actuation (A) which are then mapped to run on a multiprocessor platform.

In a pipelined control implementation (see Figure 3), the sensing operations to read and process the system states start periodically at $t_S = kh$, where k is a non-negative integer. The sampling period h is the interval between two consecutive activations of the sensing operation that require image frames for processing. We align $t_S = kh$ with the availability of the frames as can be seen in Figure 3 and hence, h is an integer multiple of f_h.

Sensing and processing is followed by control computation and actuation operations, which generally take short and nearly constant time for execution. A sensing operation takes much longer time due to compute heavy processing, i.e., $\tau_S \gg \tau_C + \tau_A$, where τ_S, τ_C and τ_A are the worst-case execution times of sensing and data processing, control computation and actuation tasks, respectively. The total (worst-case) execution time of a loop is thus given by $\tau = \tau_S + \tau_C + \tau_A$.

For a pipelined implementation, $\tau > h$ and it can be represented as [19]

$$\tau = (n_f - 1)f_h + \tau_f, \text{ where } 0 < \tau_f \leq f_h, \; n_f = \left\lceil \frac{\tau}{f_h} \right\rceil.$$

The number of frames arriving within τ is n_f. An assumption we make, for the scope of this work, is that each pipe in the pipeline is implemented on one processing core.

A. Adaptation with inter-frame dependencies

Inter-frame dependencies capture the data or algorithmic dependencies between consecutive frame processing. Considering inter-frame dependencies is crucial for practical

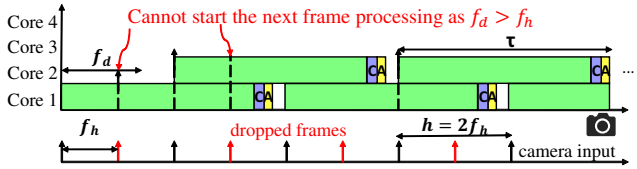

Fig. 4: Illustration of inter-frame dependencies with $f_d > f_h$. Note: 1) Even if more cores are available they cannot be used due to inter-frame dependencies; 2) Our method as compared to [7] does not restrict τ to be an integral multiple of f_h; 3) If $f_d \leq f_h$ then $h = f_h$ using all four processing cores.

real-life implementation. Inter-frame dependence time f_d is the maximum time required to wait between processing consecutive image frames due to inter-frame dependencies. Figure 4 illustrates the impact of inter-frame dependence time on sampling period. In a pipelined implementation, considering inter-frame dependencies means that strictly $h \geq f_d$. Further, h should be an integer multiple of f_h. We assume that f_d is known or can be computed.

Inter-frame dependencies mean that sometimes a number of image frames have to be skipped for processing with respect to the given image arrival period f_h and the sampling period h. Skipping a frame means that h increases and thus degrades the control performance [4], [7]. The number of frames that has to be skipped after processing every frame is $n_s - 1$ as illustrated in Figure 4 where $n_s = 2$ and one frame is skipped after every frame processing. The effective image arrival period or the minimum possible sampling period h_{min} we can then have is

$$h_{min} = n_s \times f_h, \text{ where } n_s = \left\lceil \frac{f_d}{f_h} \right\rceil.$$

The assumption here is that a sufficient number of processing cores n_c is available for pipelining.

B. Adaptation with the available number of processing cores

Another crucial aspect to consider for practical implementation is the number of available processing cores. A maximal pipelined implementation is defined as the pipelined implementation without skipping or dropping feasible image frames considering inter-frame dependencies and camera frame rate. A maximal pipelined implementation is achieved when the realisable periodic sampling period $h = h_{min}$. E.g. Figure 4 illustrates a maximal pipelined implementation with $n_s = 2$. The number of processing cores needed to realise the maximal pipelined implementation n_c^{max} and the effective sampling period h considering n_c^{avl} are defined as follows:

$$n_c^{max} = \left\lceil \frac{n_f}{n_s} \right\rceil, \; h = \left\lceil \frac{n_c^{max}}{n_c^{avl}} n_s \right\rceil \times f_h, \text{ if } n_c^{avl} < n_c^{max},$$
$$= n_s \times f_h, \text{ otherwise.}$$

where n_c^{avl} is the available number of processing cores. Having more cores, i.e. $n_c^{avl} > n_c^{max}$, does not help as there are no more frames available for pipelining. However, if $n_c^{avl} < n_c^{max}$, there are not enough cores available to process the arriving frames n_f and thus h has to be increased and a maximal pipelined implementation cannot be achieved.

C. Workload variations in pipelined IBC system

The workload variations occur due to varying features in image content (see Figure 2). When we do not consider workload variations, a pipelined implementation results in constant τ and h, as illustrated in Figure 3(a). Notice that here we measure the outputs $y(k+i)$ for the input image frame at $k+i$ with a constant worst-case sensing delay of τ, where for simplicity of notation, by $k+i$ we denote the time instant $(k+i)h$ with i an integer.

Considering workload variations in a pipelined implementation of an IBC system implies that we would have varying sensing delays, e.g. as illustrated in Figure 3(b). For this example, notice that the camera input frame at $k+4$ has a sensing delay of one frame ($\tau_1 = h$), at $k+3$ has a sensing delay of two frames ($\tau_2 = 2h$) and all other frames have a sensing delay of three frames ($\tau = 3h$). This scenario results in multiple sensing and image processing (S) tasks completing their execution at the same time. What this means is that multiple output measurements $y(k+2)$, $y(k+3)$ and $y(k+4)$ are available for control computation task C at the same time instance and no measurements arrive at the next two sampling instances.

Thus, the main challenge for the pipelined IBC system design to maximize performance, i.e. QoC, is to effectively use the sensor measurements as early as possible for control computation without any unnecessary idling and to predict the system state when there are no sensor measurements available. Modelling this behaviour is far from trivial. We define QoC as the inverse of root mean square error (RMSE), i.e. QoC=(RMSE)$^{-1}$.

III. Modelling and Discretization

In this paper, we consider a broad class of systems that can be described via linear parameter-varying (LPV) models for the predictive control approach. Specifically, this section first describes continuous-time state-space linear models which are typically obtained from first-principles. Next, a discretization scheme is described followed by details on transformation of the linear model to (I/O) form which is more suitable for the proposed control method considering the varying sensor-to-actuator delay.

A. Continuous-time model with input delay

The continuous-time LPV model we consider can mathematically be described at time t as

$$\dot{x}(t) = A(p)x(t) + B(p)u(t-\tau) \quad (1a)$$
$$y(t) = C(p)x(t) \quad (1b)$$

where $x \in \mathbb{R}^{n_x}$ denotes the state vector, $y \in \mathbb{R}^{n_y}$ contains measured outputs and $u \in \mathbb{R}^{n_u}$ is the vector of control inputs. Vector $p \in \mathbb{R}^{n_p}$ contains the scheduling parameters which determine the model coefficients in matrices A, B and C as shown in (1). The sensor-to-actuator delay is denoted by τ and $\tau > 0$. Continuous-time models are useful for an accurate simulation of the system under study; however, for computer-based control, it is necessary to have a discrete-time model considering sampled signals.

B. Discrete-time model

1) State-space description: Based on a zero-order hold (ZOH) approximation where we assume the input signal to be constant over each sampling interval, we can use the methods described in [19] to obtain the discrete-time LPV model

$$x(kh+h) = \Phi(p)x(kh) + \Gamma_0(p,\tau')u(kh-n_dh) + \Gamma_1(p,\tau')u(kh-n_dh-h) \quad (2)$$

where h is the sampling period, k is an integer indicating the time step, and

$$\Phi(p) = e^{A(p)\cdot h} \quad (3a)$$

$$\Gamma_0(p,\tau') = \int_{\tau'}^{h} e^{A(p)\cdot(h-s)}\mathrm{d}s \quad (3b)$$

$$\Gamma_1(p,\tau') = \int_{0}^{\tau'} e^{A(p)\cdot(h-s)}\mathrm{d}s \quad (3c)$$

such that the total delay τ can be expressed in multiples of the sampling period as $\tau = n_dh+\tau'$, where the remainder τ' is $0 \leq \tau' < h$. In practice, only a numerical approximation of the matrix exponential is used to compute the model matrices in (3), for which several methods exist. Specifically, for the example discussed in Section V, we approximate the matrix exponential by using its 12th degree Taylor polynomial.

When some states do not need to be controlled, the size of the control problem may unnecessarily become large especially if the number of output variables is relatively small. This motivates the use of I/O models for control, which also allow an easy incorporation of delay as shown in Sections III-B.2-III-B.3. In the linear model case, the I/O equations may simply be derived from the equivalent transfer function of the state-space model. Note also that linear models obtained from data-based system identification methods are often parameterized in I/O form.

2) Input/output difference equations: The state-space model (2) can be written as the following I/O model

$$y(k) = C(p)\left(q\boldsymbol{I} - \Phi(p)\right)^{-1}\Gamma_0(p,\tau')u(k-n_d) + C(p)\left(q\boldsymbol{I} - \Phi(p)\right)^{-1}\Gamma_1(p,\tau')u(k-n_d-1)$$

where q denotes the forward shift operator such that $qy(k) = y(k+1)$ and $q^{-1}y(k) = y(k-1)$ and for ease of notation we dropped h by assuming the time scale in terms of sampling period. The symbol $\boldsymbol{I}$ denotes an identity matrix of appropriate size. On simplification of the above difference equations, the multivariable LPV model can be rewritten in the noise-free auto-regressive exogenous (ARX) form as

$$y(k) = \sum_{j=1}^{n_x} A_j(p)y(k-j) + \sum_{j=1}^{n_x+n_d+1} B_j(p,\tau')u(k-j) \quad (4)$$

where the coefficient matrices $A_j(p)$ are derived by evaluating the determinant of $(q\boldsymbol{I} - \Phi(p))$ whereas entries of $B_j(p,\tau')$ are derived from its adjugate matrix, $C(p)$, Γ_0 and Γ_1.

3) Adapting the I/O model with time delay: Observing (4) it is clear that for $j = \{1,\ldots,n_d\}$, $B_j = \mathbf{0}$, where $\mathbf{0}$ is a zero matrix. Therefore, an increase in delay n_d implies an according zeroing of the foremost input coefficients, without a change in the size of the MPC problem as clarified in Section IV. This also allows fixing the memory allocation for the model and the control algorithm based on the worst-case delay which can reasonably be assumed to be known. For further simplicity in the design and implementation of the controller, we assume τ' and h to be constant. The delay τ' can be set to zero, especially when it varies, by a unit increment in n_d to simplify the model evaluation as Γ_1 becomes a zero matrix referring (3c). The influence of this simplification is negligible when h is sufficiently small.

IV. Predictive control strategy

This section discusses the details on formulating the adaptive predictive control problem based on the LPV model (4) with varying time delay. The reader is referred to [20] for basic terminology and details related to MPC.

A. Optimization problem formulation

The MPC problem is formulated based on a performance index which reflects the control objectives and a set of constraints including the equalities due to the LPV prediction model (4). We consider a quadratic performance index J that penalizes output tracking error and deviation of inputs from steady-state targets, i.e.,

$$J(k) = \sum_{j=1}^{N_\mathrm{p}(k)} \frac{1}{2}\|W_y(k+j)\cdot(y(k+j) - y_r(k+j))\|_2^2 + \sum_{j=0}^{N_\mathrm{p}(k)-1} \frac{1}{2}\|W_u(k+j)\cdot(u(k+j) - u_r(k+j))\|_2^2$$

where $W_y(k)$ and $W_u(k)$ denote weights on output and input respectively at time step k whereas y_r and u_r denote their target values. The prediction horizon $N_\mathrm{p}(k)$ determines the number of decision variables i.e., the inputs and outputs in prediction to be optimized. Considering simple bounds on the decision variables, the MPC problem to be solved at each time step is the constrained optimization problem

$$\min_{u(\cdot),y(\cdot)} J(k) \quad (5a)$$

$$\text{s.t.} \quad y(k+l) = \sum_{j=1}^{n_x} A_j(p)y(k+l-j) + \sum_{j=1}^{n_x+n_d(k)+1} B_j(p)u(k+l-j)$$

$$= M(p,n_d(k))\cdot\phi(k+l-1), \forall l \in \{1,\ldots,N_\mathrm{p}\} \quad (5b)$$

$$u(k+j) = u(k+N_\mathrm{u}-1), \forall j \in \{N_\mathrm{u},\ldots,N_\mathrm{p}\} \quad (5c)$$

$$y(k) = y_0 \quad (5d)$$

$$\phi(k) = \phi_0 \quad (5e)$$

$$y_\mathrm{l}(k+j) \leq y(k+j) \leq y_\mathrm{u}(k+j), \forall j \in \{1,\ldots,N_\mathrm{p}\} \quad (5f)$$

$$u_\mathrm{l}(k+j) \leq u(k+j) \leq u_\mathrm{u}(k+j), \forall j \in \{1,\ldots,N_\mathrm{u}\} \quad (5g)$$

where M in (5b) is the matrix of parameter-dependent model coefficients such that $\phi(k) = [y^\top(k), y^\top(k-1), \ldots, y^\top(k-$

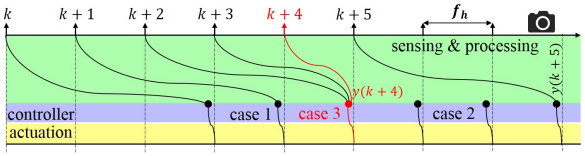

Fig. 5: Illustration of cases described in Section IV-B. The samples $k+3$ and $k+4$ have lower workloads and thus the latest output measurement $y(k+4)$ is available within one f_h. Note that for case 2, the latest measurements are not available and thus $u(k+i)$ is computed based on the MPC prediction model.

$n_x), u^\top(k-1), u^\top(k-2), \ldots, u^\top(k-n_x-n_d(k)-1)]^\top$. The upper and lower bounds on any variable z are denoted as z_{u} and z_{l} respectively. The initial condition (5d)-(5e) in the I/O case is provided via the measured output feedback y_0, and vector ϕ_0 which also includes the known past sequence of inputs and outputs. The input sequence that is optimized is typically restricted to fewer variables for trading-off control performance with computations via the control horizon (N_{u}) constraint (5c) i.e., by tuning the parameter N_{u} such that $1 \leq N_{\mathrm{u}} < N_{\mathrm{p}}$, where $N_{\mathrm{p}} > n_d$. Note that the future values of parameters p can be incorporated as they may be known and in that case p represents $p(k+l)$ in the LPV model (5b).

B. Adaptation with workload variations

In IBC, besides control computation and actuation time, the sensor-to-actuator delay mainly includes the sensing time. We assume that the delay due to control computation and actuation are fixed, but the sensing delay may vary as explained in Section II-C. We propose to adapt the controller at run time to make immediate use of the latest available measurement in order to maximize QoC. The basic idea is to have the time delay as a variable parameter based on which the model is adapted as explained in Section III-B.3. The varying parameter (n_d) is then kept constant in prediction as shown in (5b). Since the actuation rate can be constant thanks to pipelining, a new control action is computed at each time step with a ZOH during the sampling period.

The following three cases may occur at each time step due to varying sensing delay (illustrated in Figure 5):- case 1) the new measurement is available with the same sensing delay as in the previous step: in this case the parameter n_d is kept constant and the initial condition for computing the new control input is updated as usual i.e., $\phi = [y^\top(k-1), \ldots, y^\top(k-n_x), \cdots]^\top$ becomes $\phi = [y(k), \ldots, y^\top(k-n_x+1), \cdots]^\top$; case 2) the sensing delay is increased compared to the previous step as the latest measurement is not available: in this case, the prediction is done starting from the old measurement, i.e., the delay parameter n_d is incremented by 1 and the stack of past outputs in ϕ is not updated. case 3) the sensing delay is reduced by one or more steps: when multiple pipes finish processing a corresponding sequence of frames, both the latest measurement(s) along with the past measurements now available are fed to the controller and n_d is accordingly reduced where, from an implementation perspective, the output stack in the initial condition ϕ is updated as in case 1 discussed above and the same procedure is then repeated as many times as the reduction in n_d before computing the next control input.

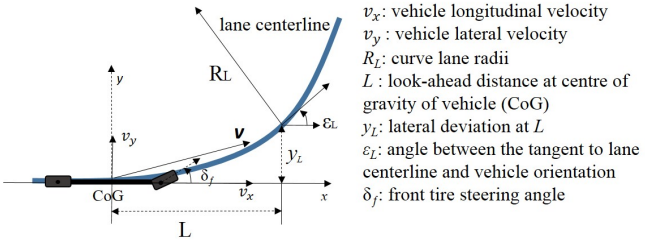

Fig. 6: Vision-based lateral control system derived from [15].

Since a new control input is applied at each step, the stack of past inputs in the initial condition vector ϕ is always updated by a unit shift in all the three cases, i.e., in $\phi = [\cdots, u^\top(k-1), \ldots, u^\top(k-n)]^\top$ becomes $\phi = [\cdots, u^\top(k), \ldots, u^\top(k-n+1)]^\top$. In conclusion, until the next measurement is available, the proposed controller compensates for the delay by making use of the prediction model to implicitly estimate the current status of the system (without a separate open-loop estimator) while accordingly computing an optimal action that satisfies given constraints.

The ordering of measurements is important in the current MPC implementation since we do not want to allow discarding measurements (as the application we consider, in Section V-A, is safety-critical). This can be considered as a limitation of the current approach. Ordering means that the latest measurement $y(k+i)$ is updated in the output stack ϕ iff the previous measurements $y(k+j),\ j<i$, were already updated in ϕ.

V. Simulation Results and Comparison

A. Case study: Vision-based lateral control system

We consider the bicycle model derived from [15] (illustrated in Fig. 6) for simulating the vision-based lateral control of a vehicle[1] on a straight road and it is described as follows,

$$\dot{x}(t) = A(v_x)x(t) + Bu(t-\tau),\ y(t) = Cx(t), \qquad (6)$$

$$A(v_x) = \begin{bmatrix} -\frac{c_f+c_r}{mv_x} & \frac{-mv_x^2+c_rl_r-c_fl_f}{mv_x} & 0 & 0 \\ \frac{-l_fc_f+l_rc_r}{I_\psi v_x} & -\frac{l_f^2c_f+l_r^2c_r}{I_\psi v_x} & 0 & 0 \\ -1 & -L & 0 & v_x \\ 0 & -1 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} \frac{c_f}{m} & \frac{l_fc_f}{I_\psi} & 0 & 0 \end{bmatrix}^\top,\ C = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix},$$

where, referring to Figure 6 and (6), we define the state vector $x(t) = [v_y, \dot{\psi}, y_L, \varepsilon_L]$; the measured output $y(t)$ as y_L; the control input $u(t)$ as δ_f which is the steering angle constrained to have a maximum magnitude of 0.5 radians; $\dot{\psi}$ is the vehicle's yaw rate in rad/s; the velocity components v_x and v_y are in m/s; l_f, l_r (= 1.22 and 1.62 m respectively) denote distance of the front and rear axles from the center of gravity (CoG); I_ψ (= 2920 kg·m^2) is the total inertia of the vehicle around its CoG; c_f, c_r ($= 1.2 \times 10^5$ N/rad) denote cornering stiffness of the front and rear tires; and the total mass of the vehicle is m (= 1590 kg).

[1]The vehicle parameters are those specified in [15] for Honda Accord.

B. MPC implementation

Based on a discretized version[2] of (6) using methods discussed in Section III-B, the optimization problem (5) can be formulated for MPC with longitudinal velocity v_x and delay parameter n_d as the scheduling variables. Constraints are imposed on the control input δ_f. In (5b), v_x may be assumed either to be a constant in prediction or a variable, as described in Section IV-A. Since for the considered lateral control system, problem (5) has a quadratic cost function subject to linear constraints, it can be solved using any quadratic programming (QP) algorithm. We use the methods described in [18] for an efficient implementation. Specifically, the QP problem (5) is transformed to the following box-constrained least-squares problem by eliminating equality constraints through quadratic penalties with large weight ρ, i.e.,

$$\min_{u(\cdot),y(\cdot)} J(k) + \rho \sum_{l=1}^{N_\mathrm{p}} \|y(k+l) - M(p, n_d(k)) \cdot \phi(k+l-1)\|^2$$
$$\text{s.t. (5f)-(5g)}, \tag{7}$$

and by substituting (5c)-(5e) in the cost function. Following [18], we implement the optimization algorithm such that it is not only able to automatically adapt to changes in parameters p including n_d but also MPC parameters such as the horizons N_u, N_p, and the tuning weights through which the controller can be re-tuned at run time. Besides this, as problem (7) is always solvable, this method has a practical benefit as it is also able to deal with situations under which the constrained QP (5) might become infeasible to solve due to model mismatch and unmeasured disturbances.

C. Controller performance evaluation

This section includes simulation results for our case study. We consider two scenarios: 1) the adaptive MPC algorithm is run with a constant worst-case delay, i.e. neglecting workload variations, and 2) with delay as a variable to explicitly consider workload variations, without changing any tuning parameters. The purpose of this simulation is to highlight the benefits of the control design that also adapts well with workload variations. The influence of delay is apparent with model mismatch and unmeasured disturbances. Since the true system considered is the continuous-time model (6) and discretization errors are negligible, in order to emulate the influence of realistic model mismatch and unmeasured disturbances, we provide the output reference for vehicle lateral control along with the output measurement, i.e., with a varying delay. Note that this is done only for this particular simulation scenario and is not the case in practice where the reference is already known.

We assume the camera frame rate of 60 fps i.e. $f_h = 1/60$ s. Simulation length is 5000 time steps i.e. $T = 83.33$ s. Unit weights are imposed on all I/O variables for MPC, while $N_\mathrm{p} = N_\mathrm{u} = 10$ time steps. The steady-state input reference $u_r(t) = 0$ whereas the output reference profile is set to a sinusoidal signal such that at time step k, $y_r(k) = 2.5 \cdot \sin(5kh\pi/T)$ m. The longitudinal velocity is a ramp signal such that $v_x(0) = 45$ and $v_x(T) = 80$ km/h, which we assume to be known in prediction for the controller.

[2]The symbolic math toolbox of MATLAB R2015b was used for obtaining the required discrete-time LPV model from (6)

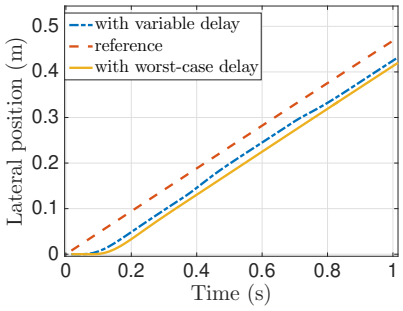

Fig. 7: Lateral position of the vehicle w.r.t. the road centre.

The mean runtime for the control algorithm was 2.2 ms while solving the optimization problems with 20 decision variables in MATLAB (on a computer equipped with a 2.6GHz processor). Referring to the reported runtimes in [18] for control problems of comparable size, we expect a similar efficient embedded implementation of the algorithms with a C backend to significantly reduce these runtimes (roughly by 20x, to about 0.1 ms) on a processor with comparable specifications. Assuming that the target platform is around 60 times slower, the WCET of tasks S, C and A are τ_S= 60 ms [6], τ_C= 6 ms [18] and τ_A= 0.5 ms, respectively. This results in worst-case delay $\tau = 66.5$ ms and $n_f = 4$.

We assume that the inter-frame dependence time f_d is given and $f_d = 15$ ms. So $n_s = 1$ and the maximum number of cores needed $n_c^{max} = 4$. We assume that $n_c^{avl} = 4$ and thus $h = 1/60$ s. To simulate the workload variations for the variable delay scenario, we consider the delay to be a random signal where the delay may take any value $n_d h$ which is held constant for every 20 frames such that $n_d \in \{1, 2, 3, 4\}$ with the corresponding probability distribution of occurrence as $\{0.2435, 0.3534, 0.3073, 0.0958\}$. Such a probability distribution for characterising workload variations can be statistically analysed from observed data [4], [5].

Based on the aforementioned simulation implemented in MATLAB on a computer equipped with a 2.6GHz processor, the results obtained show that an RMSE of 2.98 cm from the output reference is achieved using MPC considering a variable delay. For MPC based on constant (worst-case) delay the RMSE increases by 26.85% to 3.78 cm, as is clearly seen in Figure 7. The improvement by handling workload variations is expected to be higher when the worst-case delay is considerably larger than its mean.

In the simulation, we considered the following aspects which are crucial for real-life practical implementation. 1) inter-frame dependencies: $f_d = 15$ ms; 2) system nonlinearities: v_x is a ramp signal; 3) constraints: δ_f constrained to maximum magnitude of 0.5 radians; and 4) workload variations as a variable delay based on probability distribution. Future work includes validating our case study in a hardware-in-the-loop setting using [21].

TABLE I: Comparing the proposed MPC approach with the state-of-the-art multiprocessor IBC system implementations

Criteria	Proposed pipelined MPC	Pipelined		Parallelisable sequential [4]
		constant delay [7], [8]	variable delay [13]	
Inter-frame dependencies	explicitly considered	not considered	not considered	independent
System nonlinearities	explicitly considered	not considered	not considered	not considered
Constraints on variables	can be strictly imposed	cannot be imposed	cannot be imposed	cannot be imposed
Control computation time	high (worst-case up to 15x greater than [4])	low	medium (a delay predictor needed)	low (feedback gain matrix multiplication)
Algorithm	white/gray/black box	white/gray/black box	white/gray/black box	white/gray box
Parallelisation potential	independent	independent	independent	should be high
Workload variations	explicitly considered in design	not considered	indirectly considered as variable delay	explicitly considered in design
Platform	can be adapted for all	suitable for homogeneous	can be adapted for all	directly applicable for all
Restrictions on h[1]	strictly periodic; $h < \tau_{wc}$	strictly periodic; $h < \tau$	strictly periodic; $h < \tau_{wc}$	switched system possible
Restrictions on τ	strictly[2] $\tau_{wc} > h$	strictly $\tau > h$; in [7], τ is strictly a multiple of h	strictly[2] $\tau_{wc} > h$	$\tau_{wc} \leq h$

τ_{wc}: worst-case delay; [1] If camera frame arrival period f_h is considered, always h is a multiple of f_h; [2] if $\tau \leq h$ design reverts to sequential;

D. Comparison with the state-of-the-art

We compare the proposed MPC formulation for pipelined IBC implementation with state-of-the-art multiprocessor IBC design techniques in Table I. For brevity, we only compare with multiprocessor IBC system implementations and not with traditional sequential control design techniques based on the worst-case sensing delay, as it has already been shown in [4], [5] that considering workload variations is beneficial for optimizing control performance. The multiprocessor implementations can be classified into pipelined [7], [8] with constant delay, pipelined with variable delay [13] and sequential implementation with parallelisable sensing [4], [6], [12]. The camera frame rate, however, is not explicitly considered in [8].

The proposed approach is advantageous to others with respect to: 1) considering inter-frame dependencies; 2) modelling and considering system nonlinearities; and 3) strictly imposing constraints on the system variables. These aspects are crucial for practical implementation and explicitly considering them helps in making a step forward towards real-life adaptation. The proposed approach, however, requires higher worst-case control computation time τ_C (up to 15x greater than in [4]) due to solving the online optimization problem. Note that τ_C is not yet significant compared to the sensing workload, i.e. $\tau_S \gg \tau_C + \tau_A$. Future work includes identifying a case where τ_C can be significant compared to the sensing workload and adapting our method for it.

VI. Conclusion

We presented a pipelined, multiprocessor, adaptive MPC formulation for IBC systems while considering workload variations, inter-frame dependencies, system nonlinearities and constraints on system variables. The proposed approach aims to reduce the gap between current state-of-the-art multiprocessor IBC approaches and practical control requirements while optimizing quality-of-control and making a step forward towards real-life adaptation. First results based on simulations suggest that by using the proposed method one can address practical implementation challenges not directly dealt with in the past approaches, which either did not consider variations in sensing delay, or inter-frame dependencies, presence of nonlinearities in system dynamics, or practical constraints on system variables. Prospective work aims to bring the proposed methods closer to practical implementation through a hardware-in-the-loop implementation of a vehicle lateral control system.

References

[1] S. D. Pendleton *et al.*, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[2] S. Saidi *et al.*, "Future automotive systems design: research challenges and opportunities: special session," in *CODES+ISSS*, 2018.

[3] R. C. Dorf and R. H. Bishop, *Modern control systems*. Pearson, 2011.

[4] S. Mohamed *et al.*, "Designing image-based control systems considering workload variations," in *CDC*, 2019.

[5] D. Fontantelli, L. Palopoli, and L. Greco, "Optimal cpu allocation to a set of control tasks with soft real–time execution constraints," in *HSCC*, 2013, pp. 233–242.

[6] S. Mohamed *et al.*, "A scenario-and platform-aware design flow for image-based control systems," *MICPRO*, 2020.

[7] R. Medina *et al.*, "Designing a controller with image-based pipelined sensing and additive uncertainties," *TCPS*, 2019.

[8] P. Krautgartner and M. Vincze, "Performance evaluation of vision-based control tasks," in *ICRA*, vol. 3. IEEE, 1998, pp. 2315–2320.

[9] S. Li *et al.*, "Lagrangian multiplier adaptation for rate-distortion optimization with inter-frame dependency," *TCSVT*, 2015.

[10] A. W. Smeulders *et al.*, "Visual tracking: An experimental survey," *IEEE Trans. PAMI*, vol. 36, no. 7, pp. 1442–1468, 2013.

[11] K. Ogata *et al.*, *Discrete-time control systems*. Prentice Hall Englewood Cliffs, NJ, 1995, vol. 2.

[12] S. Mohamed *et al.*, "Optimising quality-of-control for data-intensive multiprocessor image-based control systems considering workload variations," in *DSD*, 2018.

[13] R. Medina *et al.*, "Implementation-aware design of image-based control with on-line measurable variable-delay," in *DATE*, 2019.

[14] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *ICSE*. ACM, 2018.

[15] J. Kosecka, R. Blasi, C. Taylor, and J. Malik, "Vision-based lateral control of vehicles," in *ITSC*. IEEE, 1997, pp. 900–905.

[16] A. Bemporad, D. Bernardini, R. Long, and J. Verdejo, "Model predictive control of turbocharged gasoline engines for mass production," *SAE Technical Paper 2018-01-0875*, 2018.

[17] A. Bemporad, D. Bernardini, M. Livshiz, and B. Pattipati, "Supervisory model predictive control of a powertrain with a continuously variable transmission," *SAE Technical Paper 2018-01-0860*, 2018.

[18] N. Saraf and A. Bemporad, "An efficient non-condensed approach for linear and nonlinear model predictive control with bounded variables," in arXiv preprint arXiv:1908.07247, 2019.

[19] K.J. Åström and B. Wittenmark, *Computer-controlled Systems: Theory and Design (2Nd Ed.)*. Prentice-Hall, Inc., 1990.

[20] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[21] S. Mohamed, S. De, K. Bimpisidis, V. Nathan, D. Goswami, H. Corporaal, and T. Basten, "IMACS: a framework for performance evaluation of image approximation in a closed-loop system," in *MECO*, 2019.