Designing image-based control systems considering workload variations

Sajid Mohamed, Asad Ullah Awan, Dip Goswami and Twan Basten

***Abstract*— We consider the problem of designing an Image-Based Control (IBC) application mapped to a multiprocessor platform. Sensing in IBC consists of compute-intensive image processing algorithms whose execution times are dependent on image workload. The challenge is that the IBC systems have a high (worst-case) workload with significant workload variations. Designing controllers for such IBC systems typically consider the worst-case workload that results in a long sensing delay with suboptimal quality-of-control (QoC). The challenge is: how to improve the QoC of IBC for a given multiprocessor platform allocation?**

We present a controller synthesis method based on a Markovian jump linear system (MJLS) formulation considering workload variations. Our method assumes that system knowledge is available for modelling the workload variations as a Markov chain. We compare the MJLS-based method with two relevant control paradigms - LQR control considering worst-case workload, and switched linear control - with respect to QoC and available system knowledge. Our results show that taking into account workload variations in controller design benefits QoC. We then provide design guidelines on the control paradigm to choose for an IBC application given the requirements and the system knowledge.

I. Introduction

Image-Based Control (IBC) systems are a class of data-intensive feedback control systems having camera(s) as the sensor (see Fig. 1). IBC has become popular with the advent of efficient image-processing systems and low-cost CMOS cameras with high resolution [1][2]. The combination of the camera and image processing (sensing) gives necessary information on parameters such as relative position, geometry, relative distance, depth perception and tracking of the object-of-interest. This enables the effective use of low-cost camera sensors to enable new functionality or replace expensive sensors in cost-sensitive industries like automotive [1][3][4].

A typical implementation of an IBC system uses linear quadratic regulator (LQR) control [5] and considers the worst-case workload [4]. However, this leads to a long sensing delay, poor effective resource utilisation in the multiprocessor platform, and suboptimal quality-of-control (QoC) [6]. Fig. 2 illustrates these challenges. The camera captures an image stream at a fixed frame rate per second (fps). The execution times of the compute-intensive processing (sensing) of the image stream depend on image workload variations. The workload variations occur due to image content and result in a wide range between best-case and the worst-case image processing times.

This research has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under grant agreement no 674875 (oCPS).

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.

A. U. Awan is with the Department of Electrical and Computer Engineering, Technical University of Munich, D-80290 Munich, Germany. Email: asad.awan@tum.de.

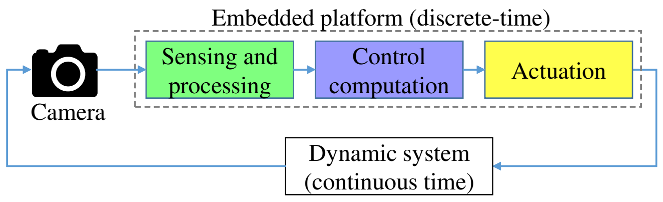

Embedded platform (discrete-time)
Camera
Sensing and processing
Control computation
Actuation
Dynamic system (continuous time)

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

The workload variations can, however, be statistically analysed, e.g. as a PERT distribution [7] or discrete-time Markov chain (DTMC) [8], from observed data and can be modelled as workload scenarios [6]. The workload scenarios can be modelled (e.g. as a task graph or a data flow graph), analysed (for timing) and then mapped to a multiprocessor platform. A system scenario abstracts multiple workload scenarios having the same sampling period as determined by platform constraints. An optimal mapping and controller may then be designed for each system scenario.

For efficiently designing IBC systems, we should consider the workload variations and the given platform allocation. An ideal design approach should: (i) identify, model and characterise the workload scenarios; (ii) find optimal mappings for these workload scenarios for the given platform allocation; (iii) identify optimal system scenarios; and (iv) design a controller with high overall QoC for the chosen system scenarios. One of the critical aspects here is: what is a good metric to define the QoC for the application? A vision-guided braking application requires a fast settling time, whereas an automotive vision-based lateral control [9] application requires to minimise the reference tracking error.

In [6], a scenario- and platform-aware design (SPADe) approach is introduced for designing IBC systems. SPADe characterises a set of frequently occurring workload scenarios, identifies a set of system scenarios that abstract multiple workload scenarios based on platform constraints, and designs a switched linear control system for these system scenarios to improve the settling time. However, a challenge in SPADe is the difficulty to guarantee stability for the resulting switched system [2]. In case of failure to guarantee stability, SPADe would result in LQR control for the worst-case workload scenario.

The contributions of the current paper are as follows:

- We present an alternate controller synthesis method based on a Markovian jump linear system (MJLS) for-

mulation. Our synthesis method involves the following steps. (i) Modelling workload variations as a DTMC, (ii) system scenario identification, and (iii) controller design and implementation. The motivation to choose the MJLS approach [10] over other standard sampled-data linear control design techniques [11] is that it does not require us to know the exact sequence of incoming sample times due to the workload variations apriori.
- We provide design guidelines on the applicability of control design methods for given requirements, implementation constraints and system knowledge. For this, we compare the three control paradigms - optimal control design using LQR, switched linear control design [2] using SPADe [6], and controller synthesis using the MJLS formulation - for IBC system design with respect to QoC while taking into account available system knowledge and implementation constraints, i.e. camera fps, platform allocation and mapping. Note that we cannot compare with adaptive [12] or model predictive control [13] approaches since we do not know the exact sequence of occurrence of incoming sample times due to the workload variations apriori.

This paper is organised as follows: We explain the embedded IBC system setting, the motivating case study, controller implementation and our controller configurations in Sec. II. In Sec. III, modelling the IBC application, system mapping, system scenario identification and configuration switching are discussed. The control problem and the QoC metrics we consider are explained in Sec. IV. Sec. V explains the method for designing the controllers we consider, including the controller synthesis method using the MJLS formulation that we present in this work. Section VI presents the results and observations of our comparison of control paradigms and provides design guidelines. Conclusion and future work are summarised in Section VII.

II. Embedded Image-Based Control

We consider a setting for an IBC system as shown in Fig. 1. Our sensor is the camera module that captures the image stream. The image stream is then fed to an embedded platform, e.g a multiprocessor system-on-chip (MPSoC), at a fixed frame rate per second (fps), e.g. 30 fps. The tasks in our IBC application - compute-intensive image sensing and processing (S), control computation (C) and actuation (A) - are then mapped to run on this MPSoC.

Motivating case study: We consider the motivating case study of vision-based lateral control of a vehicle [9], where the vehicle should follow a lane autonomously (lane-keeping). The image (sensing and) processing algorithm processes the camera frames and computes the lateral deviation at a set look-ahead distance. The controller takes the lateral deviation as the sensor input, computes the steering angle and actuates the steering to follow the lane.

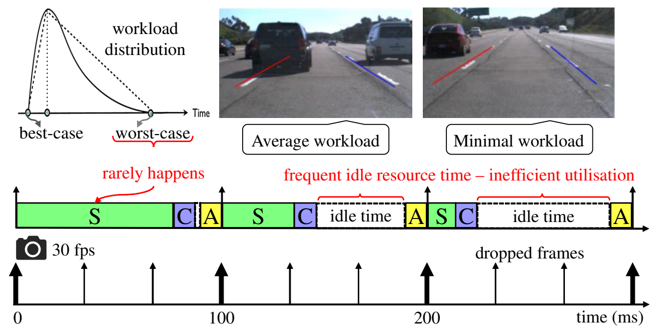


Fig. 2: Illustration of IBC system implementation and challenges for LQR control design considering worst-case workload. (S: sensing and image processing, C: control computation and A: actuation, see Fig. 1.)

A. LTI systems

We consider a linear time-invariant (LTI) system given by:

$$\begin{aligned}\dot{x}_c(t) &= A_c x_c(t) + B_c u(t), \\ y_c(t) &= C_c x_c(t),\end{aligned} \tag{II.1}$$

where $x_c(t) \in \mathbb{R}^n$ represents the *state*, $y_c(t) \in \mathbb{R}$ represents the *output* and $u(t) \in \mathbb{R}$ represents the control *input* of the system at any time $t \in \mathbb{R}_{\geq 0}$. A_c, B_c and C_c represent the state, input and output matrices of the system, respectively. For our case study, we consider the model in [9], where

$$A_c = \begin{bmatrix} -10.06 & -12.99 & 0 & 0 & 0 \\ 1.096 & -11.27 & 0 & 0 & 0 \\ -1.000 & -15.00 & 0 & 15 & 0 \\ 0 & -1.000 & 0 & 0 & 15 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; B_c = \begin{bmatrix} 75.47 \\ 50.14 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The five system states are - lateral velocity, yaw rate of of vehicle, lateral deviation from the desired centerline point at look-ahead distance y_L, the angle between the tangent to the road and vehicle orientation, and the curvature of the road at the look-ahead distance. The control input $u(t)$ is the front wheel steering angle δ_f and the output $y_c(t)$ is the look-ahead distance y_L leading to $C_c = [0\ 0\ 1\ 0\ 0]$.

B. Discrete-time control implementation

Implementation of an IBC system involves the execution of three sequential tasks: *sensing and processing* (S), *control computation* (C) and *actuation* (A). These tasks repeat; let the start and finish times of the k-th instance be given by $t_s(.)$ and $t_f(.)$, respectively. The execution times of S^k, C^k and A^k (the k-th instance) are given by,

$$e^k_T = t_f(T^k) - t_s(T^k)$$

where $T \in \{S, C, A\}$. The interval between two consecutive executions of sensing tasks S^k and S^{k+1} is then the *sampling period* h_k for the k-th instance.

$$h_k = t_s(S^{k+1}) - t_s(S^k)$$

Within each sampling period h_k, the control operations are executed sequentially (i.e., $S^k \to C^k \to A^k$). In addition, the time interval between the starting time of S^k and finishing time of A^k is then the *sensor-to-actuator delay* τ_k for the k-th instance.

$$\tau_k = t_f(A^k) - t_s(S^k).$$

We consider a time-triggered implementation for actuation task A to guarantee constant τ_k. A data-driven implementation approach is considered for S and C. Each workload scenario s_k is annotated with pair (h_k, τ_k) that models the sampling period and delay associated with it. A zero-order sample-and-hold approach can then be used to discretize the system based on the workload scenario s_k. Eq. (II.1) can be reformulated as follows:

$$\begin{aligned} x[k+1] &= A_{s_k}x[k] + B_{0,s_k}u[k] + B_{1,s_k}u[k-1], \\ y[k] &= C_c x[k] \end{aligned} \tag{II.2}$$

where,

$$A_{s_k} = e^{A_c h_k}, \tag{II.3}$$
$$B_{0,s_k} = \int_0^{h_k-\tau_k} e^{A_c s} ds \cdot B_c, \; B_{1,s_k} = \int_{h_k-\tau_k}^{h_k} e^{A_c s} ds \cdot B_c$$

In Eq. (II.2), we assume that $u[-1] = 0$ for $k = 0$. We define new system states $z[k] = \left[\begin{array}{cc} x[k] & u[k-1] \end{array} \right]^T$ with $z[0] = \left[\begin{array}{cc} x[0] & 0 \end{array} \right]^T$ to obtain a higher-order augmented system as follows:

$$z[k+1] = A_{aug,s_k} z[k] + B_{aug,s_k} u[k] \tag{II.4}$$

where,

$$A_{aug,s_k} = \begin{bmatrix} A_{s_k} & B_{1,s_k} \\ 0 & 0 \end{bmatrix}, \; B_{aug,s_k} = \begin{bmatrix} B_{0,s_k} \\ I \end{bmatrix}. \tag{II.5}$$

0 and I represent the zero and identity matrices of appropriate dimensions. A check for controllability [5] is done for this augmented system. If the system is not controllable, controllability decomposition is done to obtain a controllable subsystem.

C. Control law and control configurations

The control input $u[k]$ is a *state feedback* controller of the following form,

$$u[k] = F_{s_k} z[k] + F_{f,s_k} r_{ref} \tag{II.6}$$

where F_{s_k} is the state feedback gain and F_{f,s_k} is the feedforward gain both designed for the workload scenario s_k. r_{ref} is the reference value for the controller. The approaches we use for designing the gains are explained in Sec. V.

For each workload scenario s_k, we then define a control configuration χ_{s_k} as a tuple $\chi_{s_k} = (h_{s_k}, \tau_{s_k}, F_{s_k}, F_{f,s_k})$.

III. IBC MODEL, MAPPING AND CONFIGURATIONS

In this section, we explain how we model our IBC application, characterise the workload, map our application to platform and identify our system scenarios. We also explain why there is a switching behaviour.

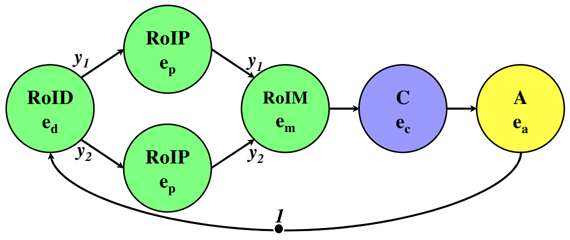


Fig. 3: IBC application SADF graph. Assuming two allocated processors and hence two RoIP actors.

A. IBC Application Model

A typical IBC application model derived from [6] is shown in Fig. 3. The sensing and processing algorithm receives the camera image frames and detects the regions-of-interest (RoID) in the frames. The detected regions-of-interest (RoI) can be processed in parallel on a multiprocessor platform. The number of allocated processors for our application determines the number of RoI processing (RoIP) nodes (or *actors*) in our model. In this case, we have two allocated processors and hence two RoIP actors. The total number of RoI detected ($= y_1 + y_2$) by RoID determines the workload. The parameters y_1 and y_2 determine how many RoI need to be allocated to the individual processors. Note that the sensor-to-actuator delay and sampling period vary based on the choice of y_1 and y_2. After processing the RoI, the data is merged and the controller state (e.g. the lateral deviation y_L in our case study, see Sec. II-A) is computed by the RoI merging (RoIM) task. The control algorithm (C) then computes the controller input $u[k]$ (e.g. steering angle δ_f in our case study, see Sec. II-A) and feeds it to the actuation (A) task. The execution times for the tasks RoID, RoIP, RoIM, C and A are e_d, e_p, e_m, e_c and e_a respectively. A scenario-aware dataflow graph [14] (see Fig. 3) is composed of: (i) a set of scenarios that model the workload variations (and are hence called workload scenarios); and (ii) a language that describes a set of infinite scenario (switching) sequences. Each scenario represents a workload situation and is modelled as a synchronous dataflow graph (SDF). An SDF graph instance of Fig. 3 is obtained by assigning values to parameters e_i and y_i. E.g. in Fig. 4(d), assigning $y_1 = 2$, $y_2 = 3$, $e_d = 5$, $e_p = 10$, $e_m = 7 \times (y_1 + y_2) = 35$, $e_c = 2$, $e_a = 2$ gives the SDF graph for a workload of 5 RoI and for mapping to two processors. The approach in [6] uses SADF as a model-of-computation for an IBC application.

In [6], the workload variations are characterised using a PERT distribution [7] (see distribution in Fig. 2). Using this the probability of frequently occurring workload scenarios are characterised. However, information regarding scenario transitions is not captured. This allows any arbitrary order for scenario switching sequences to be considered in the language for the SADF.

In the MJLS-based approach, the workload variations are characterised using a DTMC that takes into account the scenario transition probabilities along with the scenario probabilities. The states of the DTMC model the workload scenarios (see Sec. V-A) and the transitions in the DTMC model the scenario transitions. This means that a DTMC can wholly capture the language of the SADF [15].

B. System mapping and timing parameters

In this section, we describe the mapping and scheduling of our IBC application model to the platform. Fig. 4 illustrates three workload scenarios and its possible platform mapping. Fig. 4 (a), (c), and (e) model the data flow graphs for different workloads and Fig. 4 (b), (d) and (f) show their corresponding mappings on two or three processor tiles P_i. Having more processor tiles means that we can reduce h and τ by parallel execution of the sensing tasks.

Our application runs on a multiprocessor platform with a tile-based architecture [16]. Each tile has a processor, a memory, a communication assist and a network interface. We assume that the platform is predictable (like [17], [18]) so that the worst-case execution times (WCETs) of tasks can be bounded. A scheduler performs (re)configuration and time-triggered task execution.

System mapping refers to the mapping of application tasks (modelled as an SADF graph) to the platform. Each application has a given platform allocation that determines the resources to which it can be mapped. A platform allocation includes: i) number of processors (could also be a part of a processor, e.g. a processing tile); ii) memory size; and iii) network bandwidth. An application can have multiple mapping options for a given platform allocation. E.g. in Fig. 4 (c) and (e), the given platform allocation is two and three processor tiles respectively (visible in the number of RoIP actors) for the same workload (5 RoI).

The timing parameters for the three mapped workload scenarios in Fig. 4 are obtained as follows:

$$\tau_i = e_d + (\max_i^p y_i) \times e_p + e_m + e_c + e_a, \;\; h_i = \lceil \frac{\tau_i}{f_h} \rceil \times f_h,$$

where f_h represents the interval between two consecutive frame arrivals ($f_h = \frac{1}{30}s$ for a camera with 30 fps) and $e_m = 7 \times (\sum_i^p y_i)$ where p represents the number of allocated (or given) processors. Cost of communicating data between processors is assumed to be part of the actor execution times e_i; if meaningful, such cost could be made explicit, but for simplicity, we do not do so. For our example shown in Fig. 4:

$$\begin{aligned} \tau_1 &= 5 + 1 \times 10 + 7 \times (1+1) + 2 + 2 = 33ms, \\ \tau_2 &= 5 + 3 \times 10 + 7 \times (2+3) + 2 + 2 = 74ms, \\ \tau_3 &= 5 + 2 \times 10 + 7 \times (2+1+2) + 2 + 2 = 64ms, \end{aligned}$$

and $h_1 = f_h, \; h_2 = 3f_h, \; h_3 = 2f_h$. The timing parameters are then used for the discrete-time controller implementation as described in Sec. II-B and for designing the controller gains as explained in Sec. V. Further, the timing parameters are a part of the control configuration as defined in Sec. II-C.

C. System-scenario identification

It is possible for multiple workload scenarios to have the same sampling period due to implementation constraints like platform allocation and camera frame rate. E.g. for the workload scenario represented in Fig 4 (a) with (h_1, τ_1), the number of RoI, #RoI $= 2$. However, even for the workload scenario with #RoI $= 1$ mapped to two processors, we would have the same timing parameters (h_1, τ_1) since the tasks would have to execute sequentially on one processor. Similarly, for the workload scenario in Fig 4 (c), we would have the same timing parameters for #RoI 5 and 6.

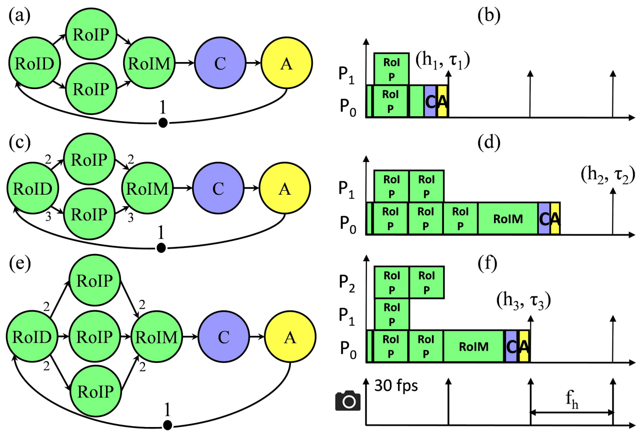


Fig. 4: Illustration of workload variations and platform mapping.

A system scenario s_k abstracts multiple workload scenarios s_i such that for $h_k = n \times f_h$ for some $n > 0$, $(h_k - f_h) < h_i \leq h_k$ and $\tau_i \leq \tau_k$, where $f_h = \frac{1}{30}s$ for a camera frame rate of 30 fps. Only system scenarios are then considered for defining the control configuration and for platform implementation [6].

D. Control configuration switching

Workload variations happen during runtime. This means that the number of RoI in every camera frame can be varying. The number of RoI in the frame determines the shortest sensor-to-actuator delay and sampling period possible for that control instance as explained in Sec. III-B and as illustrated in Fig. 4. Note that it is always possible to delay the execution of the actuation task since we have a time-triggered implementation in order to guarantee control design constraints like a constant sensor-to-actuator delay.

If our goal is to improve the control performance, we should sample as fast as possible. However, sampling as fast as possible considering workload variations results in runtime switching between different control configurations χ_{s_k} (defined in Sec. II-C). Fig. 4 illustrates three control configurations for scenarios $s_i, \; i \in \{1, 2, 3\}$, depending on the workload variations and chosen system mapping. This closed-loop system can have a switching behaviour as follows:

$$\chi_{s_1} \rightarrow \chi_{s_2} \cdots \rightarrow \chi_{s_1} \cdots$$

IV. Control problem and QoC metrics

We consider a regulation problem for the IBC system. That is, the control objective is to design $u[k]$ such that $y[k] \rightarrow r_{ref}$ (a constant reference) as $k \rightarrow \infty$. The control objectives can be performance-oriented, control effort/energy-oriented or a combination of both. The control performance quantifies, in essence, how fast the output $y[k]$ reaches the reference r_{ref}. The control effort is the amount of energy or power necessary for the controller to perform regulation. The control performance and effort are parameters that can be tuned in the cost function for the LQR design and MJLS synthesis using the state and input weights. We evaluate QoC of an IBC application considering the following metrics: two commonly used control performance metrics - mean square error (MSE) and settling time (ST); and two commonly used metrics to evaluate control effort/energy - power spectral density (PSD) and maximum control effort (MCE).

A. Mean square error (MSE)

The MSE is the mean of the cumulative sum of the squared errors, i.e.:

$$MSE = \frac{1}{n}\sum_{k=1}^{n}(x[k] - r_{ref})^2$$

where n is the number of observations, $x[k]$ is the value of the k^{th} observation and r_{ref} is the reference value. A lower MSE implies a better QoC.

B. Settling time (ST)

The settling time is defined as the time required for the output $y[k]$ to reach and stay within a range of a certain percentage (usually 5% or 2%) of the final (reference) value r_{ref} for ever without external disturbances.

C. Power spectral density (PSD)

The PSD of a signal describes the power present in the signal as a function of frequency, per unit frequency. It tells us where the average power is distributed as a function of frequency. We use Welch's overlapped segment averaging spectral estimation method [19] to compute PSD of our control input. Lower PSD for the control input signal implies that the energy required is less and hence QoC is better.

D. Maximum control effort (MCE)

We define the maximum control effort as $\max_k \|u[k]\|$. A lower MCE means better QoC. MCE can also be used to identify input saturation, if any.

V. Control Design

The control design technique we choose decides the controller feedback and feedforward gains F and F_f for the control law defined in Sec. II-C. To design a controller we assume that the sampling period h_k and sensor-to-actuator delay τ_k are known.

A. MJLS synthesis

In this section, we characterise the workload variations as a discrete-time Markov chain (DTMC). The states of a DTMC model the workload scenarios and the transitions model the scenario switching. We aggregate workload scenarios into system scenarios as explained in Sec. III-C, and then recompute transition and steady-state probabilities for the DTMC. This results in a DTMC with number of states equal to the number of identified system scenarios, with each state representing a system scenario. We assume that the switching between the different control configurations is governed by this DTMC and show how the system in (II.1) can be re-cast as a Markov jump linear system (MJLS) [10]. A Markov chain consists of a tuple $\mathcal{M} = (X, P)$ where X represents the state space, and P represents the one-step transition probability matrix. In our context of system scenarios annotated with sampling period and sensor-to-actuator delay, the state-space of $\mathcal{M}$ is given by $X = \{s_1, s_2, s_3\}$, where $s_i = (h_i, \tau_i), i \in \{1, 2, 3\}$, and

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}.$$

Associated with $\mathcal{M}$ is a discrete-time stochastic process $\theta : \mathbb{N} \to X$ such that for all times sampling instances $k \in \mathbb{N}$, and states $s_i, s_j \in X$, $i, j \in \{1, 2, 3\}$, one has:

$$\mathsf{Prob}(\theta[k+1] = s_j \quad | \quad \theta[k] = s_i) = p_{ij},$$

i.e. p_{ij} represents the probability of transitioning from state s_i to s_j. We assume that the initial condition of the stochastic process i.e. $\theta[0]$ is deterministic. The states here represent the system scenarios and the transitions represent the switching between system scenarios based on the workload variations. For the sake of simplicity, we illustrate the formulation using only three states in the DTMC representing three system scenarios. Note, however, that it is applicable to any number of identified system scenarios (as explained in Sec. III-C).

Using a zero order sample-and-hold approach, it can be readily shown that Eq. (II.1) can be re-formulated into an MJLS governed by the following stochastic difference equations:

$$\begin{aligned} x[k+1] &= A_{\theta[k]}x[k] + B_{0,\theta[k]}u[k] + B_{1,\theta[k]}u[k-1], \\ y[k] &= C_c x[k] \end{aligned} \tag{V.1}$$

where for each $s_i \in X$, $i \in \{1, 2, 3\}$: A_{s_i}, B_{0,s_i}, and B_{1,s_i} are computed using Eq. (II.3). In Eq.(V.1) we assume that $u[-1] = 0$ for $k = 0$. We define the new system states $z[k] = \begin{bmatrix} x[k] & u[k-1] \end{bmatrix}^T$ with $z[0] = \begin{bmatrix} x[0] & 0 \end{bmatrix}^T$ to obtain a higher-order augmented system as follows:

$$\begin{aligned} z[k+1] &= A_{aug,\theta[k]}z[k] + B_{aug,\theta[k]}u[k], \\ y_z[k] &= C_{aug}z[k] \end{aligned}$$

where for each $s_i \in X, i \in \{1, 2, 3\}$: A_{aug,s_i} and B_{aug,s_i} are computed using Eq. (II.5) and $C_{aug} = \begin{bmatrix} C_c & 0 \end{bmatrix}$.

Infinite horizon quadratic optimal controller: Here, we present the control law design for the MJLS (V.1). We design a controller to minimize the infinite horizon cost given by

$$J(\theta[0], z[0], u[0]) = \sum_{k=0}^{\infty} \mathbb{E}[z[k]^T C_{aug}^T C_{aug} z[k] + d_u^2 |u[k]|^2],$$

where $d_u \in \mathbb{R}_{>0}$ represents the input weight and the notation $\mathbb{E}[X]$ represents the expected value of a random variable X.

It is shown in [10] that the solution to the above infinite horizon optimal control problem can be obtained by solving the *coupled algebraic Riccati (matrix) equations* (CARE)

$$\Gamma_i = A_{aug,s_i}^T \mathcal{E}_i(\Gamma) A_{aug,s_i} + C_c^T C_c$$
$$- A_{aug,s_i}^T \mathcal{E}_i(\Gamma) B_{aug,s_i} (B_{aug,si}^T \mathcal{E}_i B_{aug,s_i})^{-1} B_{aug,s_i}^T \mathcal{E}_i(\Gamma) A_{aug,s_i}$$

where
$$\mathcal{E}_i(\Gamma) = \sum_{j=0}^{3} p_{ij} \Gamma_j$$

where $i \in \{1, 2, 3\}$ and $\Gamma = \{\Gamma_1, \Gamma_2, \Gamma_3\}$ are the unknown matrices to be solved for. The mean-square stabilizing optimal control law is then given by

$$u[k] = F_{\theta[k]}(\Gamma) x[k] + F_{f,\theta[k]} r_{ref},$$

where

$$F_{s_i} = -(B_{aug,s_i}^T \mathcal{E}_i(\Gamma) B_{aug,s_i})^{-1} B_{aug,s_i}^T \mathcal{E}_i(\Gamma) A_{aug,s_i},$$
$$F_{f,s_i} = \frac{1}{C_{aug}(I - (A_{aug,s_i} - F_{s_i} B_{aug,s_i}))^{-1} B_{aug,s_i}},$$

where $i \in \{1, 2, 3\}$. It is shown in Theorem A.12 in [10] the above CARE can be solved by solving a related convex optimization problem, the details of which are omitted due to lack of space.

B. LQR design with worst-case workload

We consider the system scenario for the worst-case workload s_{wc} having the worst-case period and delay (h_{wc}, τ_{wc}) (one of the scenarios as explained in Sec. III-C) for designing the control law. We design a controller to minimize the following cost function

$$J(u) = \sum_{k=0}^{\infty} z[k]^T d_s C_{aug}^T C_{aug} z[k] + d_u^2 |u[k]|^2,$$

where $d_u, d_s \in \mathbb{R}_{>0}$ represents the input weight and the state weights respectively. The weights are optimized for the considered QoC metric. Typically, $d_u \ll d_s$ so as to optimise for control performance and $d_u \gg d_s$ to optimise for control energy (see Sec. VI-B).

C. Switched linear control design

The switched linear control design we consider is explained in [6]. Frequently occurring workload scenarios are characterised using the the PERT distribution. A set of optimal system scenarios are identified. LQR controllers are designed for each of these scenarios s_i with (h_i, τ_i) that minimises the cost function given in Sec. V-B. Further, the stability of this switched system is analysed by deriving linear matrix inequalities (LMIs) that check for the existence of a common quadratic Lyapunov function (CQLF).

VI. Results, Observations and Guidelines

We consider the case-study of vision-based lateral control for a vehicle (explained in Sec. II) for comparison of the three approaches for control design with respect to the QoC metrics described earlier. The controllers for LQR and switched linear control are tuned for the corresponding QoC metric evaluation by adjusting the input and state weights.

A. Simulation setup

We illustrate an instance of our simulation that compares the three control paradigms: (i) for a reference output profile (for the control state y_L) shown in Fig. 5 (a) - we see that the switched linear control system (SLC) settles faster than both MJLS and LQR design; and (ii) for control input shown in Fig. 5 (b) - we see that minimum control effort is needed for LQR. SLC needs maximum control effort and might violate the input saturation requirements, if any. The control metrics PSD and MCE are derived from the control input $u[k]$ plots (e.g. see Fig. 5 (b)) and focus on minimizing the control effort or energy, whereas the control metrics MSE and ST are derived from the considered control output y_L plots (e.g. see Fig. 5 (a)) and focus on improving the performance of the system states.

We consider for the above simulation instance a frame rate of 30 fps, i.e. $f_h = \frac{1}{30} = 33.33\ ms$ and an allocation of two processors. Then, we characterise the workload variations of a synthetic data set using a DTMC model. We notice that the (h_i, τ_i) for the worst-case workload for this allocation is $(100, 74)\ ms$. We then identify the three possible system scenarios $\{s_1 = (f_h, 33),\ s_2 = (2f_h, 57),\ s_3 = (3f_h, 74)\}\ ms$. The transition probability matrix of the DTMC model considered in this case for three scenarios is:

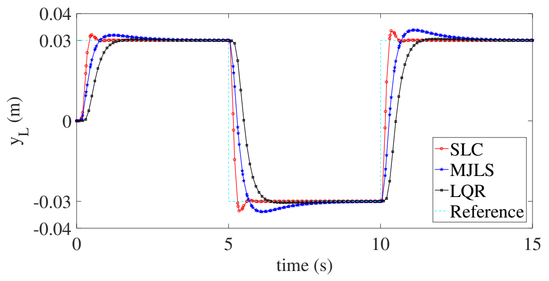
$$P = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0.5 \end{bmatrix};$$

The three controllers are then designed for the above scenarios as explained in Sec. V using the input weight $d_u = 1$.

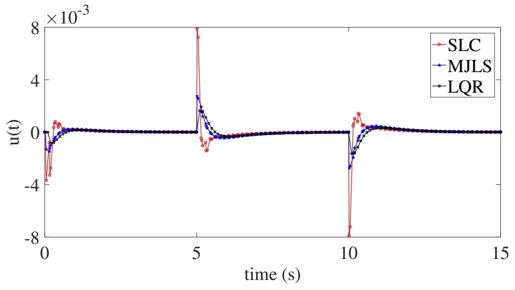
The control performance is then evaluated for the QoC metrics (defined in Sec. IV) - MSE, PSD, MCE, ST and combinations of MSE/ST with PSD/MCE. The combinations of MSE/ST with PSD/MCE is considered as they are contradictory in nature and optimising both together is a challenge and often times necessary. The above QoC metrics' empirical cost for each control technique is evaluated over multiple runs of the simulation. Each simulation run generates different workload scenario sequences that satisfy the modelled DTMC. These scenario sequences determine the switching sequence for both SLC and MJLS.

B. Observations

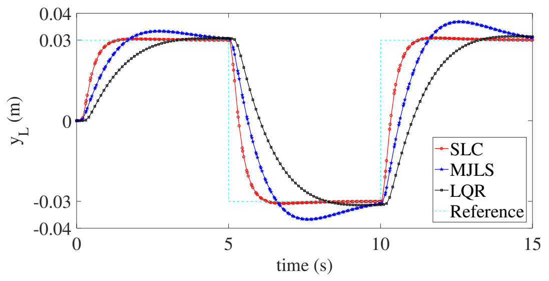
In order to provide design guidelines, we consider and vary the following different parameters: number of scenarios - we consider 3, 4 and 6 system scenarios; camera frame rates - 30 and 60 fps; input and state weights used for tuning the controllers; given platform allocation - 1, 2, 3, 4, 5, and 6 processors; and available system knowledge. We



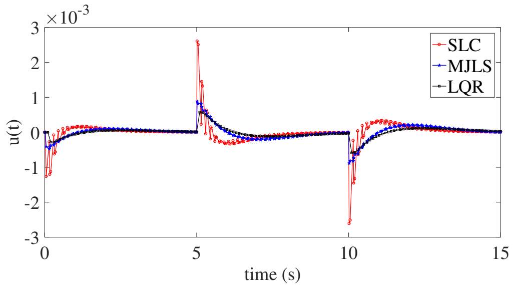
(a) Output y_L when input weight $d_u = 1$.



(b) Input $u[k]$ when input weight $d_u = 1$.



(c) Output y_L when input weight $d_u = 10$.



(d) Input $u[k]$ when input weight $d_u = 10$.

Fig. 5: Comparison between controller synthesis method based on MJLS formulation (Sec. V-A), LQR design (Sec. V-B), and switched linear control (SLC) system design (Sec. V-C).

then evaluate QoC metrics' empirical cost for each control technique over multiple runs of the simulation. The effects of varying these parameters are explained as follows:

1) The number of states in the DTMC model, the corresponding transition probability matrix and the number of control configurations change proportionally with the number of scenarios we consider. Only the SLC and MJLS system are affected by this parameter as the LQR approach always (and only) considers the worst-case workload scenario;
2) Changing the camera frame rate affects the total number of feasible scenarios we can have. As the maximum number of scenarios we can have, intuitively, is equal to $\lceil \frac{\tau_{wc}}{f_h} \rceil$, where τ_{wc} is the sensor-to-actuator delay for the worst-case workload (for a given platform allocation) and $f_h = \frac{1}{\text{camera fps}}$;
3) The input and state weights directly affect the control performance. E.g. for the simulation instance considered before, if we set the input weight $d_u = 10$, we obtain the plots shown in Fig. 5 (c) and Fig. 5 (d). What we observe is a similar overall trend for the considered control paradigms. However, we see poorer QoC metrics for MSE and ST and better QoC metrics for PSD and MCE, when compared to $d_u = 1$ and all other parameters remaining the same;
4) The given platform allocation directly affects the timing parameters for a scenario s_i, i.e. (h_i, τ_i). A higher number of available processors mean that we could execute more tasks in parallel and reduce the (h_i, τ_i) (even) for the worst-case workload scenario. This means that we could possibly reduce the total number of scenarios as well;
5) System knowledge is an important parameter that determines which control design techniques can be used. An optimal control design using LQR only requires the worst-case (workload) timing information. However, for designing an SLC system requires information regarding frequently occurring workloads as well and for MJLS synthesis approach we need both the frequently occurring workloads and their transition probabilities.

C. Design guidelines

The design guidelines we have inferred from observing our simulations for choosing the control design techniques for different QoC metrics and available system knowledge are listed in Table. I. The cases we see in the table are explained below.

- *Only worst-case workload information is known:* This situation is quite common for a control designer. Worst-case response time or delay of the algorithms can be analysed (where often times are pessimistic) through profiling and/or analytical methods [4]. The control designer is then given only the worst-case timing information and is asked to design a controller with a QoC requirement. In this case, the SLC and MJLS approach are not applicable and only the optimal LQR design approach can be used.
- *PERT distribution is known:* Here, we assume that the information with respect to frequently occurring workloads is known and are characterised analytically as a PERT distribution [7]. In this case, SLC wins for performance-oriented metrics - MSE and ST, and LQR wins for control effort or energy-oriented metrics - PSD and MCE. For jointly optimising performance and energy, there is no clear winner as it depends mainly on which of the two metrics is more important. If performance is important, SLC is preferred and if energy is important, LQR should be chosen. The MJLS approach

TABLE I: Guidelines for choosing the control design techniques: MJS (Sec. V-A), LQR (Sec. V-B), SLC (Sec. V-C).

Available system knowledge	QoC metrics				
	Performance		Control energy		Performance and Energy
	MSE	ST	MCE	PSD	
Only worst-case workload information	LQR	LQR	LQR	LQR	LQR
Frequently occurring workloads as a PERT	SLC	SLC	LQR	LQR	SLC/ LQR
Frequently occurring workloads and their transition probabilities as a DTMC	SLC/ MJS	SLC/ MJS	MJS/ LQR	MJS/ LQR	MJS

is not applicable as more information is needed.

- *DTMC model is known:* Information regarding frequently occurring workloads and their transition probabilities are needed for modelling a DTMC. These can be estimated from observed workload variations data [8]. Intuitively, this means that we can predict the possible (workload) scenario switching sequences for the control design. However, for the above two cases the switching sequence is assumed to be arbitrary and not known. In this case, for performance metrics, MJLS wins when the input weight d_u is very small (since SLC tends to oscillate before settling). However, for a large value of d_u, there is no clear winner between SLC and MJLS and it depends on the application and chosen parameters. Please note, however, that a challenge of SLC is to prove the stability of the designed system. MJLS is a synthesis method and the design, if any, is stable by construction. If we consider control effort or energy metrics, LQR wins when the input weight d_u is small and there is no clear winner between LQR and MJLS for a large input weight d_u as the results are similar and depends on the application and chosen parameters. MJLS is the clear winner if we consider a joint optimisation for performance and energy QoC metrics.

VII. Conclusion

We present a MJLS formulation for controller synthesis for image-based control systems considering workload variations and platform implementation constraints. Further, we compare our method with two relevant control paradigms: LQR and switched linear control system design. We also provide design guidelines on the control technique to use for given constraints on the system knowledge, the QoC and the implementation.

The synthesis method assumes that the workload variations can be characterised as a DTMC. A DTMC is sensitive to the data used for its modelling. As a future work, sensitivity analysis of the DTMC towards the QoC needs to be evaluated. Further, the current design guidelines are provided based on multiple empirical simulation runs of the controller for varying workloads, number of scenarios, camera frame rate and given platform allocation. A formal mathematical analysis would strengthen our design guidelines and is planned as future work. The challenge for a formal analysis of the control design is that we do not know the exact sequence of occurrence of the workload variations apriori.

References

[1] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised.* Springer, 2017, vol. 118.

[2] E. van Horssen, "Data-intensive feedback control: switched systems analysis and design," Ph.D. dissertation, Eindhoven University of Technology, 2018.

[3] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[4] S. Saidi, S. Steinhorst, A. Hamann, D. Ziegenbein, and M. Wolf, "Future automotive systems design: research challenges and opportunities: special session," in *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, 2018.

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

[6] S. Mohamed, D. Zhu, D. Goswami, and T. Basten, "Optimising quality-of-control for data-intensive multiprocessor image-based control systems considering workload variations," in *21st Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 320–327.

[7] S. Adyanthaya, Z. Zhang, M. Geilen, J. Voeten, T. Basten, and R. Schiffelers, "Robustness analysis of multiprocessor schedules," in *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, 2014, pp. 9–17.

[8] N. J. Welton and A. Ades, "Estimation of markov chain transition probabilities and rates from fully and partially observed data: uncertainty propagation, evidence synthesis, and model calibration," *Medical Decision Making*, vol. 25, no. 6, pp. 633–645, 2005.

[9] J. Kosecka, R. Blasi, C. Taylor, and J. Malik, "Vision-based lateral control of vehicles," in *Proceedings of Conference on Intelligent Transportation Systems.* IEEE, 1997, pp. 900–905.

[10] O. L. V. Costa, M. D. Fragoso, and R. P. Marques, *Discrete-time Markov jump linear systems.* Springer, 2006.

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

[12] G. Goodwin, P. Ramadge, and P. Caines, "Discrete-time multivariable adaptive control," *IEEE Transactions on Automatic Control*, vol. 25, no. 3, pp. 449–456, 1980.

[13] A. Bemporad, F. Borrelli, M. Morari *et al.*, "Model predictive control based on linear programming-the explicit solution," *IEEE transactions on automatic control*, vol. 47, no. 12, pp. 1974–1985, 2002.

[14] H. Alizadeh Ara, A. Behrouzian, M. Hendriks, M. Geilen, D. Goswami, and T. Basten, "Scalable analysis for multi-scale dataflow models," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 17, no. 4, p. 80, 2018.

[15] B. D. Theelen, M. Geilen, T. Basten, J. Voeten, S. V. Gheorghita, and S. Stuijk, "A scenario-aware data flow model for combined long-run average and worst-case performance analysis," in *Formal Methods and Models for Co-Design (MEMOCODE)*, 2006, pp. 185–194.

[16] S. Stuijk, "Predictable mapping of streaming applications on multiprocessors," Ph.D. dissertation, Eindhoven University of Technology, 2007.

[17] A. Hansson, K. Goossens, M. Bekooij, and J. Huisken, "CoMPSoC: A template for composable and predictable multi-processor system on chips," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14, no. 1, p. 2, 2009.

[18] S. A. Edwards and E. A. Lee, "The case for the precision timed (PRET) machine," in *Design Automation Conference (DAC).* IEEE, 2007, pp. 264–265.

[19] P. Welch, "The use of FFT for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Trans. on audio & electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967.