Approximation-Aware Design of an Image-Based Control System

SAYANDIP DE, (Graduate Student Member, IEEE), SAJID MOHAMED, (Member, IEEE), DIP GOSWAMI, (Member, IEEE), AND HENK CORPORAAL
Electronic Systems Group, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands
Corresponding author: Sayandip De (sayandip.de@tue.nl)

This work is part of the From the cloud to the edge—smart IntegraTion and OPtimisation Technologies for highly efficient Image and VId eo processing Systems (FitOptiVis) project funded by the ECSEL Joint Undertaking under grant number H2020-ECSEL-2017-2-783162.

ABSTRACT Image-based control (IBC) systems are common in many modern applications. In such systems, image-based sensing imposes massive compute workload, making them challenging to implement on embedded platforms. Approximate image processing is a way to handle this challenge. In essence, approximation reduces the workload at the cost of additional sensor noise. In this work, we propose an approximation-aware design approach for optimizing the energy, memory and performance of an IBC system, making it suitable for embedded implementation. First, we perform compute- and data-centric approximations and evaluate its impact on the energy efficiency, memory utilization and closed-loop quality-of-control (QoC) of the IBC system. We observe that the workload reductions due to approximations allow mapping these lighter approximated IBC tasks to embedded platforms with lower power consumption while still ensuring proper system functionality. Therefore, we explore the interplay between approximations and platform mappings to improve the energy-efficiency of IBC systems. Further, an IBC system operates under several environmental scenarios e.g., weather conditions. We evaluate the sensitivity of the IBC system to our approximation-aware design approach when operated under different scenarios and perform a failure probability (FP) analysis using Monte-Carlo simulations to analyze the robustness of the approximate system. Finally, we design an optimal approximation-aware controller that models the approximation error as sensor noise and show QoC improvements. We demonstrate the effectiveness of our approach using a concrete case-study of a lane keeping assist system (LKAS) using a heterogeneous NVIDIA AGX Xavier embedded platform in a hardware-in-the-loop (HiL) framework. We show energy and memory reduction of up to 92% and 88% respectively, for 44% QoC improvements with respect to the accurate implementation. We show that our approximation-aware design approach has an FP (per km) ≤ 9.6 × 10^{-6}%.

INDEX TERMS Approximate computing, approximate systems, approximation-in-the-loop, energy-aware system design.

I. INTRODUCTION Image-Based Control (IBC) systems are feedback control systems which rely on image-based sensing data obtained from camera sensor(s). Advancements in camera technologies, image processing algorithms and parallel computing heterogeneous platforms have made IBC systems immensely popular in automotive applications [1] like advanced driver assistance systems (ADAS), autonomous driving systems etc. The emergence of electric vehicles (EVs) has increased the need to implement these automotive applications on power-constrained embedded platforms [2]. However, the enormous compute requirements of IBC systems make them challenging to be implemented on such platforms without sacrificing control performance. The focus of this work is to improve the compute, memory and energy efficiency of IBC systems (implemented on embedded platforms) by introducing approximations, without sacrificing the control performance. In essence, approximation reduces the compute workload at the cost of additional sensor noise. However, the inherent error resilience of IBC systems allows the introduction of this additional sensor noise without sacrificing control performance.

A typical IBC system consists of a sensing task \( T_s \), a control task \( T_c \) and an actuation task \( T_a \) (see Fig. 1). \( T_s \) involves pre-processing the image frames obtained from the camera sensor and extracting application-specific features. \( T_c \) applies the control algorithm using these extracted features...
features and $T_a$ implements the control decisions in the environment. A worst-case execution time (WCET) analysis shows that the execution time of $T_s$ takes orders of magnitude higher than that of $T_i$ and $T_a$ [3], thus, resulting in a long sensing-to-actuation delay $\tau$ (delay $(T_i+T_c+T_a)$) (see Fig. 1).

The quality-of-control (QoC) of an IBC system depends on the sensing-to-actuation delay $\tau$. Smaller $\tau$ results in a lower sampling period for the controller, which improves the overall QoC. The sensing task ($T_i$) is the main bottleneck in lowering $\tau$. There are two ways to reduce the WCET of $T_i$. First, prior literature [3] has shown that approximate computing can reduce the WCET of $T_i$ by introducing sensor noise and reducing the compute overhead. This is motivated by the fact that IBC systems are equipped with image signal processing (ISP) pipelines optimized for human visual consumption and control algorithms, which are inherently resilient to sensor noise, do not need the high-quality images produced by these pipelines. This work focuses on system approximation by reducing the compute workload of the ISP and the data transfer traffic to off-chip memory, resulting in better QoC and energy efficiency of the overall system.

Second, the WCET of $T_i$ can also be reduced by leveraging the high parallel computing capabilities of mobile heterogeneous multi-core platforms [5]. This requires application-specific task mappings to different heterogeneous cores of the platform. Reduced compute workloads due to approximations introduce new task mapping opportunities. Approximate tasks can be mapped to slower but less power consuming heterogeneous platform cores, while operating the controller at the same sampling period, guaranteeing proper system functionality. The combined impact of both approximations and platform mappings is not explored in the literature. This work explores the interplay between the degree of approximation and different platform mappings while analyzing their impact on QoC, memory and energy efficiency of the entire IBC system. Additionally, approximation introduces errors in the measurement of states of the system [3]. We propose a method to design an approximation-aware optimal linear-quadratic-gaussian (LQG) controller by modelling the approximation error as sensor noise.

IBC systems operate under different environmental scenarios [6]. For example, image feedback at night requires a different nature of processing compared to the same during the day in ADAS. These scenarios significantly influence the degree of approximation that can be tolerated without destabilizing the system. The robustness study on how approximations impact the IBC performance under different environmental scenarios is essential but is not addressed in the literature. For different environmental conditions, we design different approximation-aware controllers and show that our approach is robust to different environmental scenarios by performing a failure probability (FP) analysis for the approximate system using Monte-Carlo simulations.

In summary, our approach takes into account various artefacts of approximation-in-the-loop in terms of platform mapping and controller design, while considering robustness to failures. We refer to this approach as approximate-aware IBC design. The key contributions of this work are as follows:

1) **Optimized approximate-aware IBC:** The basic idea of approximation in an IBC design was introduced in our previous work [3]. In [3], we illustrated the potential of exploiting the inherent error resilience of IBC systems by performing coarse-grained computation skipping in the ISP (compute-centric approximation). In this work, we introduce two optimizations on top. First, we perform a data-centric approximation by varying the degree of lossy compression post-ISP (Sec. V-B) which has shown memory reduction of up to 88%. Second, we design an approximate-aware LQG controller that models the errors due to approximation as sensor noise (Sec. VI). These optimizations improve the overall QoC up to 44% and energy up to 92% compared to the accurate implementation.

2) **Platform-specific power-awareness:** IBC systems are often deployed on battery-operated edge devices having limited compute and memory capacity. They are power-constrained for achieving longer battery life. In this work, we take into account the platform-specific power-constraints by considering different platform mappings under strict power budgets (5W, 15W, 30W). We evaluate our results by considering an embedded platform, NVIDIA AGX Xavier, in a hardware-in-the-loop (HIL) setup (Sec. VII, III-A).

3) **Scenario-awareness:** Environmental scenarios significantly influence the degree of approximation that can be tolerated without destabilizing an IBC system. While our previous work [3] focused on the day scenario, in this work, we perform scenario-specific optimization considering six different environmental scenarios relevant for LKAS, i.e., day, night, dawn, dusk, fog, snow. We show that scenario-specific approximation decisions improve the overall QoC (Sec. VIII).

4) **Failure probability analysis:** Applicability of our proposed approach to safety-critical systems (e.g., LKAS) require failure probability (FP) analysis to comply with well-accepted safety margins [7]. In this work, we perform FP analysis based on Monte-Carlo simulations.

Our approximation-aware design framework is open sourced and can be accessed on github: https://github.com/sayandipde/approx_ibr.
to show the robustness of approximate IBC systems designed using our approach (Sec. IX).

II. RELATED WORK

Prior efforts in approximate computing domain can be broadly classified into compute-centric and data-centric approaches.

A. COMPUTE-CENTRIC APPROXIMATIONS

Compute-centric efforts are focused on reducing the compute workload across algorithm, architecture and circuit-levels. Commonly used algorithmic approximations are computation skipping [8], precision scaling [9] and replacing error-resilient compute-intensive functions with neural networks [10]. A similar learning approach to design ISPs for new camera systems is proposed in [11]. Next, at the architecture level, research efforts have focused on both approximating general-purpose processors [12] as well as domain-specific accelerators [13]. At the circuit-level, research efforts focus on manual design techniques for adders and multipliers [14], as well as automated methodologies for designing energy-efficient approximate circuits [15].

B. DATA-CENTRIC APPROXIMATIONS

Data-centric approximations either approximate the memory device that is being accessed or they approximate the value of the data being accessed. Both cases lead to reduced on/off chip data traffic, thereby reducing the required memory bandwidth. Reducing the DRAM refresh rate [16] and load value speculation [17] are examples of approximating the memory location. Approximating data values involves storing/accessing data in a compressed format [18]. Quality-aware memory controllers for directing memory transactions to different compression schemes are proposed in [19].

Both compute and data-centric approaches are focused on approximating individual subsystems. There are major downsides of approximating each subsystem as a stand-alone entity.

1) Individual subsystems are usually a part of bigger IBC systems. Proper interaction between them is key in ensuring system stability. However, approximating a subsystem might result in undesired behaviour in another, thereby resulting in the failure of the entire IBC system.

2) From an energy perspective, these approaches improve the energy efficiency of the targeted subsystem. They leave behind substantial energy reduction opportunities due to lack of intercomponent interactions and proper tradeoff analysis.

To address these limitations, this work proposes a holistic full-system analysis approach wherein different subsystems are approximated together in a compute or data-centric manner and the quality implications are evaluated for the full-system rather than individual subsystems. An overview of prior efforts in full-system approximation analysis highlighting the key differences from our work is given below.

### Table 1. Qualitative comparison with state-of-the-art.

| Contributions | [20] | [21] | [22] | [23] | [3] | This Work |
|---------------|------|------|------|------|-----|-----------|
| compute-centric optimizations | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| data-centric optimizations | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| closed-loop approximations | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| evaluation framework | SiL | HiL | ✔ | ✔ | ✔ | ✔ |
| platform mappings | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| environmental scenarios | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| approximation-aware controller | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| real-system implementation | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| camera sensor | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| computation | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| controller | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| actuation | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

✔ True, ❌ Partially True, △ Modeling or Simulation.

SiL: Software-in-the-loop, HiL: Hardware-in-the-loop

*Closed-loop approximations: approximations in closed-loop feedback systems.

C. FULL SYSTEM APPROXIMATION ANALYSIS

These approaches are targeted at different application domains, however for this study, we focus mainly on camera-based systems. Approximation benefits in a camera-based biometric security system, using an iris recognition application, is showcased in [20]. An approximate smart camera system is introduced in [21], using camera resolution scaling, reducing memory refresh rate and computation skipping. An approximate ISP pipeline tuned for computer vision algorithms is designed in [22], by skipping selected ISP stages. An algorithm-hardware co-designed system is showcased in [23]. It leverages the temporal motion information generated by ISPs to reduce the compute demands of the perception engine, at the cost of accuracy loss.

These research efforts [20]–[23] lack a closed-loop feedback behaviour. Approximation decisions in a closed-loop system have quality implications at a later point in time. Optimising a system while accounting for the temporal approximate behaviour is not explored in [20]–[23], making this a key distinguishable feature of our work. Additionally, looking solely from an approximation perspective, some of the approximation techniques applied in these research efforts can also be applied to our work for additional benefits. For instance, fine-grained computation skipping techniques proposed in [21], [24] can be applied in the ISP, which is an interesting research direction for future. Also, leveraging motion information to relax the number of invocations of the PR stage (as shown in [23]) is another interesting research direction, orthogonal to our work.

Table 1 summarizes the key contributions of this work stacked up against other state-of-the-art full system approximation approaches. It is worth mentioning that a camera-based system is composed of several functional elements (e.g. camera-sensor, computation, controller, actuator) which are either modeled or implemented in real system for

2 Only for closed-loop systems
output validation. For a fair qualitative comparison of existing literature, it is important to highlight these implementation and validation differences. We categorized the literature based on their implementation method and summarized in Table 1.

III. EMBEDDED IMAGE-BASED CONTROL

We consider a motivating case study of a lane-keeping assist system (LKAS) to demonstrate our approximation-aware IBC design approach. In this section, we start by giving a top-level overview of the hardware-in-the-loop (HiL) setup used for our evaluation (Sec. III-A). Then we zoom in on the different stages in the LKAS system from an algorithmic perspective (Sec. III-B) and a hardware mapping perspective (Sec. III-C).

A. HARDWARE-IN-THE-LOOP (HiL) SETUP

Fig. 2 illustrates our HiL simulation setup for LKAS adapted from [25]. It simulates a vehicle with a top look ahead camera using Webots [26] physics simulator engine and interacts with NVIDIA AGX Xavier platform using TCP/IP protocol. The simulator works in a server-client configuration, wherein Webots acts as the server while the NVIDIA platform acts as the client. The server (Webots) progresses simulation in full synchronization with the client (NVIDIA AGX Xavier) [4]. At each simulation step, the camera sensor simulated in Webots generates a raw image containing state information \( x[k] \), that is fed to the NVIDIA platform. It executes the sensing \( T_s \) and control \( T_c \) tasks to generate control input \( u[k] \), which is communicated back to Webots for actuation. After actuation, the simulation progresses to the next step.

![HiL simulation setup for LKAS system.](image)

For our evaluation, the camera sensor in Webots simulator is modelled based on AR1335 CMOS digital image sensor [27] and is set to a resolution of 720p. The camera frame rate is varied between 30fps, 60fps and 120fps, depending on the sampling period of the controller. The actuation dynamics are modelled based on [29]. The vehicle is initially positioned with a fixed bias of 15 cm from the lane centre to test the control performance. Lane width of 3.25 m is considered, as per standard road safety guidelines. The Webots simulation step is set to 1 ms, while the vehicle speed is set to 50 km/hr for all of our evaluations.

B. LKAS IN DETAILS: ALGORITHMIC OVERVIEW

A lane-keeping assist system (LKAS) consists of six main components/stages: Camera Sensor, ISP, data compression, perception (PR), control \( T_c \) and actuate \( T_a \), as shown in Fig. 2. The camera sensor and actuation are modelled and executed in Webots. ISP, data compression, perception (PR) and control \( T_c \) are executed in NVIDIA AGX Xavier platform. Below we provide an overview of these stages.

1) IMAGE SIGNAL PROCESSING (ISP), PERCEPTION (PR)

An ISP pipeline transforms a RAW image in the Bayer domain to pixels in the RGB domain through a series of image enhancing stages. Modern ISPs comprise of hundreds of proprietary stages. However, in this work we consider a set of five essential stages common to all ISP pipelines, demosaic, denoise, color map, gamut map and tone map, as defined in [3], [22]. Fig. 3(a) shows these five stages along with their corresponding outputs. It is worth noting that the RGB output from the ISP pipeline is typically stored in the main memory (off-chip DRAM) due to the large size of the image data. In this work, we consider JPEG compression (see Fig. 3) to reduce the data communication between different processing stages like ISP, PR etc.

The perception (PR) stage calculates the lateral deviation of the vehicle from the centre of the lane by performing preprocessing, feature extraction and inference steps on the decompressed ISP output. During preprocessing, first, the region of interest (ROI) is selected based on the scene. A perspective transform is then performed on the ROI to get a bird’s eye view of the lane ahead (see Fig. 3(b) row 2). During feature extraction, the candidate lane pixels are extracted from the bird’s eye view image. For this, the bird’s eye view image is converted to grayscale, and subjected binarization using static thresholding (see Fig. 3(b) row 3). Finally, candidate lane pixels are obtained using sliding windows ranging from bottom to top of the image. During inference, first, the previously identified lane positions markers are fit to a second-order polynomial. Then these polynomials are used to calculate the centre of the lane at a look-ahead (LL) distance. The centre of the image in the x-direction is considered as the vehicle’s current position. Using these two metrics, the lateral deviation in the transformed domain \( y_{LP} \) is calculated.

3We observe that state-of-the-art lane detection algorithms [28] operate on low-resolution images. So, we perform our evaluation using downscaled (512 x 256) sensor images. We believe our approach is also effective for high-res images required in applications like object detection.
We consider the bicycle model derived from [30] for simulating the LKAS\(^4\) and it is described as follows,

\[
\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t),
\]

where the state vector \(x(t) = [v_x, \psi, y_L, \epsilon_L]\); the measured output \(y(t) = y_L\), the lateral deviation from the desired centerline point at look-ahead distance; the control input \(u(t)\) as \(\delta_f\) which is the steering angle; \(v_x\) and \(v_y\) are the longitudinal and lateral velocity in m/s; \(\psi\) is the vehicle’s yaw rate in rad/s; \(\epsilon_L\) is the angle between the tangent to lane centerline and the vehicle orientation in rad; \(l_f, l_r\) (\(=1.6975\) and 1.2975 m respectively) denote distance of the front and rear axles from the center of gravity (CoG); \(I_{\psi} \approx 6337.74\) kg·m\(^2\) is the total inertia of the vehicle around its CoG; \(c_f, c_r\) (\(=2 \times 60000\) N/рад) denote cornering stiffness of the front and rear tires; and the total vehicle mass \(m = 2000\) kg.

We guarantee constant sensing-to-actuation delay \(\tau\) by enforcing an implementation with time-triggered activation of tasks. An implementation is annotated with a pair \((h_i, \tau_i)\) that models the sampling period and delay associated with it. The zero-order hold (ZOH) method is used to discretize the system [31] with the annotated \((h_i, \tau_i)\) to obtain an augmented system of the form: 
\[
z[k+1] = A^d_kz[k] + B^d_ku[k],
\]
where \(A^d_k\) and \(B^d_k\) are discretized matrices and the augmented system states 
\[
z[k] = [x[k] u[k-1]]^T.
\]

Control Law: The control input \(u[k]\) is a state feedback controller of the form given below:

\[
u[k] = Kz[k],
\]

where \(K\) is the state feedback gain. We design \(K\) using the optimal linear quadratic regulator (LQR) [31]. The control objective is for the output \(y[k] \to 0\), when \(k \to \infty\).

C. HARDWARE SUPPORT FOR LKAS

1) PLATFORM OVERVIEW

An industrial embedded heterogeneous platform NVIDIA AGX Xavier [4] is considered for LKAS implementation. It consists of an 8-core NVIDIA Carmel ARMv8.2 CPU and an integrated 512-core NVIDIA Volta GPU, along with 16GB of LPDDR4x off-chip DRAM memory, as shown in Fig. 4(a). It is noted that Fig. 4(a) only shows the IPs used in this work. Balancing performance and energy/power requirements is key to designing compute- and energy-efficient IBC systems. To address this, the considered platform allows software-controlled power-gating of the CPU cores. The default configuration considers all 8 CPU cores to be online and has a maximum power budget of 30W [4].

2) INITIAL TASK MAPPING

Main tasks in LKAS that are executed in the NVIDIA platform are ISP, jpeg encode/decode, PR and control \((T_c)\). As an initial step, we map all the tasks to an 8-core CPU only configuration. The measured runtimes are shown for image workloads of \((512 \times 256)\) resolution.
The ISP is optimized using Halide [32] domain-specific language with GPU as backend. Additionally, we also map parts of JPEG encode/decode and PR to the GPU. Fig. 4(c) gives a detailed mapping overview. Task offloading from CPU memory to GPU memory is a major bottleneck. We make use of the unified memory (a single memory address space accessible from any processor in a system) support in NVIDIA Volta GPUs to optimize our mappings further. The control task \( T_c \) is light in compute, so we map it to the CPU. Our initial task mapping gives a runtime speedup of \( 4.7 \times \) over an 8-core CPU only mapping (see Fig. 4(b)). We consider this as a baseline for exploring approximation opportunities in LKAS.

IV. DESIGN AND EVALUATION STRATEGIES

In this section, we first outline our approximation-aware design strategies for optimizing IBC performance (QoC), memory utilization and energy efficiency. We determine what to approximate by identifying the error-resilient stages which can give maximum benefits on approximation (Sec. IV-A). Then we explain how to approximate by summarizing the main steps of our approximation-aware IBC design (Sec. IV-B). Then, we show how to interpret the outcomes by introducing an energy optimal and a QoC optimal mode for approximated IBC systems (Sec. IV-C). Finally, we describe the quality metrics considered for evaluation (Sec. IV-D).

A. WHAT SHOULD WE APPROXIMATE?

The first challenge is to identify the error-resilient as well as compute-heavy stages in LKAS. These are target candidates which can give maximum gains. LKAS consists of six different stages as shown in Fig. 2. Actuation cannot be approximated as it depends on vehicle dynamics. However, prior literature has shown that rest of the stages (camera sensor [22], ISP [10], data compression [21], PR [13], control [33]) can be approximated. So, to figure out the best approximation opportunities, we perform time and energy-aware profiling of LKAS.

Time and energy profiling results for LKAS are shown in Fig. 5. Profiling is performed using default configuration of NVIDIA AGX Xavier (see Fig. 4(a)). The system is running Ubuntu 18.04. For time profiling, we execute each stage in LKAS 100 times and for 100 different images to reduce sensitivity to access locality. For each stage, we consider the maximum of all such execution runs to get the WCET. For actuation, a WCET of 0.5 ms is considered [29]. For energy profiling, we need the execution time (obtained from time profiling) and the average power consumption per stage. The latter is obtained using the on-board power monitors (Texas Instruments INA3221 [4]) present in the platform.

From Fig. 5, it is evident that ISP, which consumes 83% of the total runtime and 94% of the total energy, is the main target candidate for approximation. Additionally, the off-chip data transfer can also be optimized to obtain added gains. So, in the rest of the paper, we confine our scope to optimizing the ISP and the off-chip data transfer.

B. HOW TO APPROXIMATE AND OPTIMIZE LKAS?

In this work, we propose an approximation-aware design approach shown in Fig. 6. Each of the contributions presented in this work is marked.

First, we perform coarse-grained approximations to the ISP (Optim 1) by skipping one or more sub-stages within the pipeline (see Fig. 3(a)). This is a compute-centric approximation approach focused on reducing the compute workload of the ISP. Then we perform data-centric approximations by varying the degree of lossy JPEG compression (Optim 2). The goal is to reduce the data transfer traffic between the processor and off-chip DRAM, as accessing DRAM is both slow as well as expensive in terms of energy. The Q-parameter in the JPEG algorithm decides the degree of lossy compression. A smaller value of Q denotes larger numerical values in the quantization matrix, thereby leading to a higher level of compression. This comes at the cost of larger errors in the decompressed image. In case of Optim 1, we choose the highest value of Q (\( Q = 100 \)) and keep it constant which results in a lower degree of lossy compression. However, in the case of Optim 2, we use the Q-parameter as a quality control knob and reduce it in discrete steps of 10 from Q = 100 to Q = 10.

5Note that this compute-centric approach has no impact on the data transfer traffic as the degree of lossy compression performed in the JPEG encoder post-ISP is fixed to the minimum.
This reduces the off-chip data transfer traffic and impacts the QoC, energy and memory footprint of LKAS.

From a control design perspective, approximations performed in Optim 1 and Optim 2 introduce errors in the state of the system. The controller is unaware of this added error. So, we design an approximation-aware LQG controller (Optim 3) by modelling the error as sensor noise. Profiling results in Fig. 5 show that LKAS is compute-bound. So, we expect most gains from Optim 1 as it reduces the compute workload of the system. We start by evaluating Optim 1. Then we incrementally add Optim 2 and Optim 3. It is worth mentioning that Optim 1 and 2 are not novel approximation strategies. However, the focus of this work is to evaluate the combined impact of these optimizations (Optim 1, Optim 2 and Optim 3) on the closed-loop QoC, memory and energy of LKAS, which is not explored in the literature.

The workload reductions obtained due to these approximations allow mapping the approximated tasks to slower but low power consuming embedded platforms. We explore this approximation-mapping interaction to further optimize LKAS for energy and QoC. Four different configurations of the NVIDIA AGX Xavier platform is considered.

The workload reductions obtained due to these approximations allow mapping the approximated tasks to slower but low power consuming embedded platforms. We explore this approximation-mapping interaction to further optimize LKAS for energy and QoC. Four different configurations of the NVIDIA AGX Xavier platform is considered.

LKAS operates under different environmental scenarios. We design approximation-aware controllers for each scenario and evaluate the sensitivity of LKAS to Optim 1, Optim 2 and Optim 3 when operated under these scenarios. We set up six different environmental scenarios (day, night, dawn, dusk, fog and snow) in Webots for analyzing the sensitivity of LKAS to approximation errors. LKAS is a safety-critical application which requires an evaluation of its robustness to approximation error. So, we perform a failure probability (FP) analysis of approximate LKAS using HiL based Monte-Carlo simulation.

C. HOW TO ANALYZE APPROXIMATION-AWARE DESIGN OUTCOMES?

To properly analyze the design points obtained from our approximation-aware design approach, we consider two different modes: energy-optimal mode and QoC-optimal mode. Fig. 7 shows a snapshot of LKAS operating in these modes over a fixed evaluation time window (t_{ETW}). In energy-optimal mode, we consider a reduced sensing-to-actuation delay τ obtained due to approximations. However, the sampling period (h) is kept constant. This results in a longer processor idle time. So, we expect better energy efficiency when operating in this mode. We can reduce energy further by applying voltage-scaling to take advantage of the idle time. Considered NVIDIA platform does not support this. So, we keep it out of the scope of this work. In QoC-optimal mode, both reduced sensing-to-actuation delay τ as well as reduced sampling period (h) are considered. This allows a higher no. of frames to be processed over the same t_{ETW}. As a result, we expect a higher energy consumption but a lower QoC for LKAS.

D. QUALITY METRICS

Image quality degradation due to Optim 1 and Optim 2 is evaluated using the Structural Similarity (SSIM) index. The SSIM index for two images m, n is defined as:

\[
SSIM(m, n) = \frac{(2\mu_m\mu_n + C_1)(2\sigma_{mn} + C_2)}{\mu_m^2 + \mu_n^2 + C_1(\sigma_m^2 + \sigma_n^2 + C_2)}
\]

(3)

where \(\mu_m\), \(\mu_n\), \(\sigma_m\), \(\sigma_n\) and \(\sigma_{mn}\) are the local means, standard deviations, and cross-covariance for images m, n, C1, C2 are constants. High SSIM loss denotes images with higher visual difference. QoC evaluation of the proposed IBC system is performed using the following four metrics:

1) Mean Absolute Error (MAE): mean of the cumulative sum of absolute errors, i.e.

\[
MAE = \frac{1}{n} \sum_{i=1}^{n} |y[k] - \text{reference}|
\]

(4)

where n is the no. of samples and y[k] is the value of the kth sample. A lower MAE implies a better QoC.

2) Settling Time (ST): time needed for y[k] to reach and stay within 5% of the final reference value. A lower ST implies a better QoC.

3) Power Spectral Density (PSD): power distribution of the signal over frequency. A lower PSD for control input u[k] implies that less energy is required and hence a better QoC.

4) Maximum Control Effort (MCE): maximum control effort is max_k ||u[k]||. Lower MCE implies better QoC.

These metrics are explored in [5], [34] and consider both control performance and control energy. For Pareto analysis between QoC and energy, we consider MAE as the default QoC metric as it gives a good indication of the steady-state performance. For all evaluation, we consider the following as defaults, unless otherwise mentioned: (a) Power Budget: 30W (b) Platform Config: CPU_8C + GPU (see Fig. 4(a) (c) Scenario: day.
V. COMPUTE- & DATA-CENTRIC APPROXIMATIONS

In this section, we first evaluate the impact of approximations on the QoC, energy and memory of LKAS. We perform both compute-centric and data-centric approximation as detailed below. Our goal is to optimize the control performance as well as energy-efficiency of IBC system. So, we explore the trade-offs between QoC and energy efficiency using Pareto-analysis.

TABLE 2. Coarse-grained approximation settings in Optim 1.

| Setting | ISP Stages | Description |
|---------|------------|-------------|
| S0      | DM, DN, CM, GM, TM | Accurate (all stages included) |
| S1      | DM, CM, GM, TM | Skip Denoising |
| S2      | DM, DN, GM, TM | Skip Color Mapping |
| S3      | DM, DN, CM, TM | Skip Gamut Mapping |
| S4      | DM, DN, CM, GM | Skip Tone Mapping |
| S5      | DM, DN      | Keep only Denoising |
| S6      | DM, CM      | Keep only Color Mapping |
| S7      | DM, GM      | Keep only Gamut Mapping |
| S8      | DM, TM      | Keep only Tone Mapping |

DM: Demosaic, DN: Denoise, CM: Color Mapping, GM: Gamut Mapping, TM: Tone Mapping
Note: We will refer to S0 as the accurate setting as no stages are skipped.
In all other settings, S1-S8, certain stages are skipped (approximate settings).

A. OPTIM 1: COARSE-GRAINED ISP APPROXIMATIONS

Here, the ISP is approximated in a coarse-grained manner by skipping one or more sub-stages within the pipeline (see Fig. 3(a)). Testing all possible combinations for skipping sub-stages is not feasible due to high compute overheads. So, for our analysis, we consider 9 different approximation settings adopted from [3], as shown in Table 2. Settings S1-S4 are obtained by skipping one stage at a time, while settings S5-S8 are obtained by keeping one stage and disabling rest of the pipeline. It is noticed that skipping the demosaic stage results in an LKAS failure. This is because PR algorithms operate in the RGB domain and they do not work for Bayer domain. So, demosaicing is essential for proper LKAS operation. Additionally, it needs to be mentioned that certain approximation settings initially led to LKAS failure. Minor modifications to the PR stage to handle these errors made it working (explained in Sec. V-C).

Skipping one or more sub-stages in the ISP has both positive and negative impacts on the QoC of LKAS. The loss in image quality due to approximation settings (S1-S8) may degrade the QoC. However, the reduced sensing-to-actuation delay ($\tau$) allows faster sampling of the controller, thereby improving the QoC. Balancing this interaction is essential in determining if we gain or lose in final QoC. We evaluate the former by operating LKAS in energy-optimal mode (without considering faster sampling) and the latter by operating LKAS in QoC-optimal mode (considering faster sampling).

Fig. 8 shows a comparison between image degradation and QoC of LKAS. The QoC metrics (MAE, ST, MCE and PSD) on the left x-axis are normalized to the accurate setting (S0). The SSIM loss (right x-axis) shows the degree of image degradation. First, it is evident that the different settings (S1-S8) have varying impact on image quality. S1 (skipping denoise) performs the worst in terms of both image quality and QoC. We conclude that skipping denoising while keeping the rest of the stages (S1) is not a suitable candidate for getting better QoC. We also observe that settings (S3, S4, S5, S7, S8) perform relatively similar to the baseline. This shows that ISP pipelines optimized for human vision...
FIGURE 11. Energy improvements from coarse-grained ISP approximation (Optim 1). Both modes of operation for LKAS are shown with $t_{ETW} = 25$ ms.

are overkill for LKAS. There is no one-to-one correlation between image degradation and QoC. For instance, settings S4, S5 and S7 have high SSIM loss (more degradation) but they perform similar to S0 in terms of QoC, with S4 being slightly better. Contrarily, S2 has low SSIM loss but high MAE (worse QoC). This is explained by the fact that the performance of control algorithms depend on the presence of an essential feature in the image (lane markings in this case).

Image degradation metrics like SSIM loss fail to identify this. This non-correlation shows that approximating different subsystems individually without considering its impact on the bigger closed-loop system can lead to sub-optimal results.

Fig. 9 shows that lowered compute workloads due to approximations (S1-S8) result in up to 76% reduction (S5) in the sensing-to-actuation delay ($\tau$). This allows faster sampling of the controller which is taken into account in Fig. 10. A detailed explanation of GPU schedules (and resulting memory access pattern) obtained for S0-S8 in Fig. 9 is given in the Appendix. We observe a significant impact of reduced sampling period on the QoC. Faster control sampling in S3-S8 due to reduced $\tau$ overshadows the negative impact of image degradation on QoC. We observe up to 63% (S7), 35% (S4), 10% (S8), 40% (S8) improvements in MAE, ST, MCI and PSD respectively. However, in settings S1 and S2, the minor reduction in $\tau$ do not allow a faster sampling. So, the negative impact of image degradation is not balanced, resulting in worsened QoC. From Fig. 8, we observe that skipping only gamut mapping (S3) and keeping only gamut mapping (S7) have opposite impacts on the visual quality of the image [22]. This is evident in Fig. 10 where both settings S3 and S7 perform similarly in terms of QoC (0.38 and 0.36 respectively) with S7 being slightly better than S3.

Fig. 11 summarizes the implications of coarse-grained ISP approximations (Optim 1) on the energy of LKAS. A stage-wise energy breakdown is presented which shows that the energy efficiency of the ISP stage is significantly improved. This is expected as Optim 1 is targeted to optimizing the ISP stage. The energy evaluation is done over a time window ($t_{ETW}$) of 25 ms. The left y-axis reports the energy normalized to S0. The right y-axis reports the processed frame rate (no. of frames consumed for processing per second) as blue dots. For the chosen $t_{ETW}$, only one frame is processed across all settings in energy-optimal mode. In QoC-optimal mode, one frame is processed for S0-S2 and three frames are processed for S3-S8. This explains the increased energy for S3-S8 in QoC-optimal mode compared to energy-optimal mode. S6 is the most energy efficient setting. For S6, upto 92%, 78% energy reduction is observed for energy-optimal and QoC-optimal mode respectively.

Looking into QoC and energy separately gives us a partial picture. We are more interested in analysing the best QoC that can be achieved for a given energy budget. This is shown using a QoC-energy Pareto analysis in Fig. 12. Only the Pareto-optimal points are highlighted for each mode. Operating in energy-optimal mode results in up to 91% energy savings with only 4% degradation in QoC performance (S8). Operating in QoC-optimal mode results in up to 63% improvements in QoC while also improving energy by

6This is different from a camera frame rate which can be tuned only to 30, 60 and 120 fps. To give an example, for setting S0, the sensor-to-actuation delay $\tau$ allows a maximum of 40 frame per second. So, tuning the camera to 30fps results in stalling of the pipeline, while tuning it to 40fps results in frame drops. In our analysis, we consider frame drops.
An interesting observation is that S3, which gives the best visual quality among all approximate settings (see Fig. 8), is dominated by other settings and is not included in the Pareto-front in Fig. 12. This shows that visual quality is not paramount for designing better IBC systems. Fig. 13 reports the memory improvements in LKAS due to Optim 1. Lack of visual quality in the approximated images due to Optim 1 explains the lower memory footprint. However, this high visual quality is an overkill for LKAS, already shown in Fig. 8. Up to 69% reduction (S5) in memory traffic is obtained from Optim 1.

**B. OPTIM 2: LOSSY COMPRESSION USING VARIABLE Q-PARAMETER**

Here, we vary the Q-parameter in the JPEG algorithm to control the degree of lossy compression to reduce the off-chip memory traffic. In the evaluation of Optim 1, a fixed Q-parameter (\(=100\)) is considered. For Optim 2, the Q-parameter is modulated in fixed steps from Q10 to Q90 in addition to Q100. Fig. 14 shows the off-chip memory traffic reduction for the different approximation settings while varying Q-parameter. All values are normalized to S0Optim 1. S5 is the most memory-efficient setting across all Q-parameters. For S5, additional memory reductions of up to 81% (Q10) is obtained over Optim 1. However, these reductions come at a cost, they introduce more noise to the system. To gain on QoC for the entire system, there must be significant runtime reductions in \(\tau\) to allow for faster control sampling. We observe that for low values of Q (Q < 70), the runtime reductions are insignificant (see Fig. 14). Lower Q-parameter leads to more aggressive quantization during compression. However, for smaller pixel values (more common in approximated images), values are rounded to zero resulting in no additional reduction in memory traffic, thereby no runtime improvements. So, for the QoC-energy trade-offs, we consider only higher values of Q (\(\geq 70\)).

Fig. 15 shows the QoC-energy trade-offs for Optim 2. Three different values for Q (\(=100, 90, 70\)) are considered. We obtain new Pareto non-dominated points for Q = 90. These points have a better QoC as the runtime reductions result in a lower \(\tau\) which in turn results in reduced sampling period for the control. Also, the positive impact on QoC due to faster sampling overshadows the negative impact due to excess noise for Q = 90. The area under the Pareto curve for Optim 2 is improved by 10.3% over Optim 1, indicating better performance in terms of both QoC and energy. As mentioned earlier, for better LKAS design (in terms of energy and QoC), the impact of runtime improvements must overshadow the image degradation. For Q = 70, just 2% reduction in \(\tau\) is achieved compared to that of Q = 90, which cannot overshadow the extra approximation noise, resulting in no Pareto-optimal points.

**C. INTERSTAGE EFFECTS OF APPROXIMATIONS**

It is observed that both the coarse-grained approximations in Optim 1 and the lossy compression in Optim 2, have a quality impact on the subsequent stages, especially the PR stage. However, the impact of Optim 1 on the PR stage is critical, as it results in LKAS failure in some cases. The PR stage does not detect the lanes properly for certain approximated streams obtained from S1-S8. The problem is traced back to the static image thresholding step in PR (see Fig. 3(b)), which fails to identify the lane markers from the grayscale bird’s eye view image due to incorrect choice of threshold. To counter this, Otsu’s binarization algorithm is used, which dynamically identifies the optimal threshold. Otsu’s algorithm brings an additional computational complexity of...
$MN + 7L^2 + 5L - 12$, for a $M \times N$ image with $L$ grey levels. Depending on the approximation settings, dynamic thresholding is either performed on the grayscale bird’s eye view image or RGB bird’s eye view image. This results in desired LKAS behaviour across all approximation settings. It is noted that performing dynamic thresholding on the RGB image has thrice the computational complexity, which has been taken into account in our evaluation results.

**VI. APPROXIMATION-AWARE CONTROL DESIGN**

Here, we quantify the errors introduced due to coarse-grained approximations as well as variable lossy compression and use it to design an approximation-aware controller. Initially, we need to identify the system state parameter(s) affected by the approximations. For LKAS, the lateral deviation $y_L$ is affected by approximation. We quantify the error ($e_i$) due to approximation for the setting $S_i$ as the covariance of the calculated $y_L^i$ for the approximation setting $S_i$ with respect to the calculated $y_L^0$ for the accurate setting $S_0$, i.e. $e_i = \frac{1}{n} \sum_{j=1}^{n} (y_L^i - y_L^0)^2$.

We use the optimal LQG control design [31] technique to design the approximation-aware controller for the system defined as follows:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) =Cx(t) + e_i,$$

where $e_i$ models the error due to approximation as measurement noise for the output.

**FIGURE 16.** QoC-Energy trade-offs for Optim 3. Pareto front improvements over Optim 1 and Optim 2 are shown.

Fig. 16 shows the QoC-Energy trade-offs for Optim 3. We observe that the area under the Pareto curve for Optim 3 improves by 22% and 15% over Optim 1 and 2 respectively. This means better trade-offs in terms of QoC and energy of LKAS. It is important to mention that Optim 3 has QoC improvements over Optim 2 but no energy improvements as it does not influence the compute-intensive sensing stage ($T_s$). This explains the Pareto front only moving towards the left. Finally, the improvements from Optim 3 in energy-optimal mode are higher than in QoC-optimal mode. This is because, in energy-optimal mode, faster sampling is not considered. So, control decisions at each actuation are valid longer. So, better decisions taken by the LQG controller are more profound in energy-optimal mode compared to QoC-optimal mode.

**VII. APPROXIMATION AND MAPPING INTERPLAY**

In Sec.V & VI, we mapped all the approximate tasks to a default (CPU_8C + GPU) platform configuration (see Fig. 4(a)). In an embedded setting, we are often strictly power-constrained to extend the battery lifetime of the system/device. Mapping tasks to a GPU has runtime benefits, but it is extremely power hungry. One approach to optimize for power is to take advantage of the reduced compute workloads of the approximate tasks by mapping them to slower but low power CPU-cores. Only approximate tasks with significant compute reductions can benefit from this, otherwise increased runtimes due to slower CPU-cores can lead to system failure.

The no. of CPUs online is controlled using software-controlled power gating in NVIDIA AGX Xavier. We use this to introduce three new platform configurations (CPU_1C, CPU_4C, CPU_8C) in Fig. 17. For our evaluation, we consider three different power budgets (5W, 15W and 30 W) common in industrial embedded platforms [4]. We report our mapping results considering design points obtained by combined application of Optim 1, 2 and 3 as these are Pareto-optimal points as shown in Fig. 16. Fig. 18 shows the peak power consumption for the approximate tasks under different platform mappings. As expected, we see that mapping tasks to GPU have high power requirements than CPU only mappings. For maximum power budget $\leq$ 5W, only
CPU_1C can be used. For maximum power budgets ≤ 15W, all CPU only mappings can be used. While for maximum power budgets ≤ 30W, all four can be used.

Fig. 19(a) shows a snapshot for the timing implications of different task mappings for all approximate settings S0-S8. Sensing-to-actuation delays τ are shown on the y-axis. We have observed in our experiments that for proper LKAS operation at a vehicle speed of 50 kmph, a controller sampling period ≤ 125 ms is required. So, mapping the accurate setting S0 to CPU_1C and CPU_4C result in a vehicle crash. However, mapping approximate settings S3, S5, S6 and S8 to CPU_1C and CPU_4C, result in desired LKAS behaviour (in QoC-optimal mode). This shows that a combination of approximation and platform mappings can improve power efficiency while guaranteeing system functionality. Fig. 19(b) shows the energy consumption of different approximate settings (S0-S8) with respect to different platform mappings. The left y-axis reports the energy normalized to S0_CPU_8C+GPU (baseline). The right y-axis reports the processed frame rate (no. of frames consumed for processing per second) as yellow dots. As mentioned earlier, a minimum frame rate of 8 fps (period ≤ 125 ms) is required for proper LKAS operation. In energy-optimal mode, for each mapping, settings S1-S8 operate at the same frame rate as S0 of the corresponding mapping. Thus, all the settings result in LKAS failure for CPU_1C and CPU_4C. For CPU_8C, we obtain up to 12% (S6) more energy reduction over CPU_8C+GPU while guaranteeing desired LKAS functionality. In QoC-Optimal mode, settings S3, S5, S6, S8 mapped to CPU_1C and CPU_4C result in desired behaviour due to improved sampling period. Higher energy consumption is observed due to increased no. of frames being processed compared to energy-optimal mode. For a lower power budget (≤ 15 W), due to approximations, we are able to operate at the same fps (= 40 fps) as the baseline with up to 74% (S5_CPU_4C) energy reduction.

Fig. 20 shows the QoC-energy trade-offs for settings S0-S8 under different platform mappings. Mapping. However, for strict power budgets (≤ 15W, ≤ 5W), we can explore approximation-mapping interplay to get better trade-offs. For instance, we can obtain the same QoC (= 1) as baseline using S8_CPU_4C and S5_CPU_8C+GPU. S8_CPU_4C can operate in a power budget ≤ 15 W, while S5_CPU_8C+GPU cannot.

VIII. SENSITIVITY TO ENVIRONMENTAL SCENARIOS
IBC systems operate under different environmental scenarios. In this section, we evaluate the impact of approximation noise on IBC performance when operating under these scenarios. Fig. 21 shows the six different environmental scenarios considered for our evaluation. These are commonly encountered driving conditions relevant for LKAS. It needs to be emphasized that the energy consumption of the system is not affected by changing these scenarios as the default image sensor resolution (720p) and platform config (CPU_8C + GPU) are not changed.

Fig. 22 shows the QoC sensitivity for LKAS to approximation settings (S0-S8) when operated under different environmental scenarios. All values are normalized to S0 (baseline). It is observed that the choice of approximation (S1-S8) is extremely critical for better QoC. To get the best QoC, different approximation settings should be chosen based on the
FIGURE 21. Different environmental scenarios considered in this work.

FIGURE 22. Sensitivity of QoC to different approximation settings operating under different environmental scenarios. Results are for Optim 1.

We consider six different environmental scenarios most relevant to LKAS. It is worth noting that our approach is applicable to a combination of these scenarios as well. For instance, a segment of road with multiple tunnels can be handled by dynamically switching from a day to a night scenario and vice-versa. An extensive study of different scenario combinations is out of the scope of this work and will be done in future.

IX. ROBUSTNESS OF APPROXIMATE IBC

IBC system considered in this work is safety-critical which requires a robustness study when subjected to approximations. For this, we evaluate the failure probability of LKAS when subjected to different approximate settings (S1-S8). First, we perform Monte-Carlo simulations of the entire system while taking into account different approximate settings and environmental scenarios. Then, we obtain the percentage of lane misprediction and closed-loop failure probability of LKAS.

A. MONTE-CARLO SIMULATION

For Monte-Carlo simulation, we sweep different LKAS parameters in Webots using the HiL setup (Sec. III-A) and obtain various system models for simulation. The considered parameters are initial starting position or initial lateral deviation of the vehicle, different weather conditions and road surface types. From these simulations, the camera frames obtained by driving the vehicle are extracted and stored as a dataset for further analysis. For each frame in the dataset, the lane markings are annotated as ground truth (information extracted from the accurate image).
B. LANE MIS_PREDICTION (LM)
Correct lane prediction is essential for proper LKAS operation. So, in this analysis, we study the sensitivity of the PR stage to different approximate settings by calculating the increase in lane misprediction (LM) compared to the predictions using ground truth. We use the dataset obtained using Monte-Carlo simulations. Lane misprediction (LM) is calculated as shown below [35]:

\[
LM = 1 - \frac{1}{n} \sum_{i=1}^{n} \frac{C_i}{S_i}
\]

where \(C_i\) is the number of correct predicted lane points per frame and \(S_i\) is the number of ground truth points per frame. A prediction is considered correct if the difference between a ground truth point and the predicted point is less than a certain threshold. \(n\) is the total number of frames considered. In this work, \(S_i = 512\) and \(n = 3000\). Fig. 24 shows that the PR stage is robust to errors from approximation settings S2-S7 with average LM \(\leq 1\%\). We also see that image frames subjected to approximate settings S4-S7 have high visual changes (Fig. 8) compared to the accurate one, but they still have low LM. This is because essential lane markings are not affected by these visual changes. Skipping denoising (S1) has a high negative impact on lane detection (PR) with average LM = 11.7\%. Thus, skipping denoising (S1) while keeping the rest of the stages is not suitable for proper LKAS performance. We also observe that keeping only tone mapping (S8) works for all scenarios except snow (LM = 18\%) which shows that scenario-based approximation selection is needed. We determine the worst-case approximation error (Sec. VI) per setting considering the scenario with the highest LM for that setting. This error is used for designing different LQG designs for failure probability (FP) analysis.

\[
S = \sum_{i=1}^{m} F_i
\]

where \(F_i\) is 1 if the number of times the vehicle goes out of the lane markings during the simulation interval is one or more, otherwise 0. \(m\) is the number of Monte-Carlo simulations performed for each approximate setting. For our experiments, we consider \(m = 50000\). \(l\) is the lifetime of the vehicle. The typical lifetime of a conventional vehicle is 150,000 mi or 241,401.6 km [7].

C. FAILURE PROBABILITY (FP)
We calculate the failure probability of LKAS for different approximate settings (S1-S8) using Monte-Carlo simulations of the entire system. For this analysis, we use the worst case approximation error (explained in Sec. IX-B) to design an optimal LQG controller for each approximate setting. There are two questions which we need to address here: (a) Is the designed controller stable and robust? (b) Is there LKAS failure, i.e., whether the vehicle goes out of the lane?

To evaluate stability robustness of the designed LQG controller, we calculate the stability radius \(r\) which is the radius of the largest ball centered at the \((-1, 0)\) point to loop transfer function of the system in the nyquist plot [36]. \(r\) is calculated as follows:

\[
r = \frac{1}{||S||_\infty}
\]

where \(S\) is the sensitivity function calculated at the output of PR for the proposed LKAS model and \(0 \leq r \leq 1\). A higher value of \(r\) means more stability robustness to sensor noise. The \(r\) values obtained for S0-S8 are 0.8787, 0.9800, 0.9129, 0.9906, 0.9900, 0.9982, 0.9898, 0.9982 and 0.9999 respectively. All the \(r\) values are close to 1 which shows the stability of designed LQG controllers. It is noted that the \(r\) values for the approximate settings (S1-S8) are higher than the accurate (S0) one. This means that for the approximate cases, LQG is designed to sustain a higher sensor noise margin. This has an impact on the LQG performance which is evaluated in terms of failure probability (FP).

Failure probability of LKAS is calculated as shown below:

\[
FP_{\text{per km}} = \frac{1}{m \times l} \sum_{i=1}^{m} F_i
\]

where \(F_i\) is 1 if the number of times the vehicle goes out of the lane during the simulation interval is one or more, otherwise 0. \(m\) is the number of Monte-Carlo simulations performed for each approximate setting. For our experiments, we consider \(m = 50000\). \(l\) is the lifetime of the vehicle. The typical lifetime of a conventional vehicle is 150,000 mi or 241,401.6 km [7].
FIGURE 26. Comparative study showing LKAS performance (QoC) versus failure probability for the different approximate settings when operating under different environmental scenarios.

failure probabilities of the hardware and the communication subsystems.\(^7\)

Considering failure probability (FP) in a standalone manner paints a partial picture. We are more interested in designs which not only have low FP but also perform better than the actual setting (S0). Fig. 26 shows this comparative study between LKAS performance (QoC) and FP per km. QoC values are normalized to the baseline (S0) for each scenario. S0 has the lowest FP as expected. We start by comparing the four static approximate settings S2, S3, S4 and S6 which have low FP across all scenarios. We observe that S2 has the lowest average FP compared to S3, S4 and S6, but its closed-loop QoC is worse than baseline (S0) across all scenarios (see Fig. 26, green dotted line). Statically choosing S3, S4 and S6 across all scenarios shows improved QoC compared to baseline for some scenarios and degraded QoC for others (see Fig. 26, blue, yellow and black dotted lines). This motivates dynamically selecting the approximate settings for each scenario which results in either improved or same QoC as that of the baseline (S0). This is shown by the red dotted line in Fig. 26.

X. DISCUSSIONS

In this section, we provide a summary and give insights on some key aspects of our design approach in terms of dynamic configuration overheads, possible gain in energy-critical domains like electric vehicles and applicability in safety-critical systems. We also discuss the generality as well as the modularity of our design approach while highlighting the ease of switching to newer design models.

A. RESULT SUMMARY AND INSIGHTS

In this section, we summarize the results presented in this work by looking from three different perspectives: (a) exploiting error-resilience in IBC systems, (b) enabling applicability in edge devices, and (c) sensitivity and robustness. A detailed list of parameters, optimizations and simulation settings considered for evaluation of our results is shown in Table 3.

1) EXPLOITING ERROR-RESILIENCE IN IBC SYSTEMS

IBC systems like LKAS have intrinsic error resilience and visual quality is not paramount for such applications. We exploit this by performing coarse-grained computation skipping in the ISP as well as variable lossy compression post-ISP. This significantly reduces the sensing-to-actuation delay \(\tau\) at the cost of loss in image quality. Reduced \(\tau\) enables faster sampling of the controller which not only negates the negative impact of image degradation on QoC but in turn improves QoC as shown in Sec. V-A & V-B. We model this error as sensor noise in the approximation-aware LQG controller which improves the overall QoC further as shown in Sec. VI. Reduced compute workloads due to approximate ISP and a higher degree of lossy compression gives significant energy and memory improvements as well. This work allows optimizing for either QoC (QoC-optimal mode) or energy (energy-optimal mode). In energy-optimal mode, we obtain 92% improvements in energy efficiency for 88% reduction in memory footprint and 44% improvements in QoC compared to accurate implementation (see Fig. 16, S6_{Optim 1+2+3} in energy-optimal mode and Fig. 14, S6 in Q90). In QoC-optimal mode, we obtain 78% improvements in QoC for 77% improvements in energy efficiency and 89% efficiencies.

TABLE 3. Summarizing details of evaluation parameters considered.

| Sec. | Optimizations Considered | Platform Mappings | Environmental Scenarios | Controller Type |
|------|--------------------------|-------------------|--------------------------|-----------------|
| V-A  | Optim 1                  | CPU_8C+GPU        | day                       | LQR             |
| V-B  | Optim 1 + 2              | CPU_8C+GPU        | day                       | LQR             |
| VI   | Optim 1 + 2 + 3          | CPU_8C+GPU        | day                       | LQG*            |
| VII  | Optim 1 + 2 + 3          | CPU_8C+GPU        | day                       | LQG*            |
| VIII | Optim 1 + 2 + 3          | CPU_8C+GPU+CPU_4C | day, night                | LQG*            |
| IX   | Optim 1 + 2 + 3          | CPU_8C+GPU+CPU_1C | day, night                | LQG*            |

\(^7\)Software, hardware and communication subsystems are responsible for the majority of the autonomous vehicle failures related to vehicular components as reported in [7]. Approximations proposed in this work contribute to software failure due to increased lane misprediction (LM). So, we report our results along with failure contributions due to discrepancies in hardware and communication subsystems to get a better perspective.
reduction in memory footprint (see Fig. 16, $S_{7_{\text{Optim}}}^{1+2+3}$ in QoC-optimal mode and Fig. 14, $S_7$ in Q90).

2) ENABLING APPLICABILITY IN EDGE DEVICES

Edge devices have limited compute and memory capacity. They are battery operated and are often power-constrained for achieving longer battery life. Our design methodology introduces approximation to reduce the massive computing demands of IBC systems while maintaining proper IBC performance. Also, the memory footprint of the application is reduced due to extra lossy compression. This makes approximate IBC systems designed in this work, a suitable candidate for edge computing. To validate this claim, we consider strict power budgets of 5W and 15W in Sec. VII, which is common for edge devices. For both these power budgets, the accurate setting ($S_0$) results in LKAS failure due to long sampling period (see Fig. 19 (a)). But proper LKAS operation can be ensured by introducing approximations using our design approach. For 5W power budget, we obtain proper LKAS operation but performance is not as good as the accurate setting ($S_0$) in 30W budget (see Fig. 20, $S_5$ in QoC-optimal mode). However, for 15W budget, we obtain better performance than the accurate setting ($S_0$) in 30W mode (see Fig. 20, $S_8$ in QoC-optimal mode).

3) SENSITIVITY AND ROBUSTNESS

For proper on-field deployment of approximate IBC systems, a wide range of commonly encountered environmental scenarios are tested in Sec. VIII. It is observed that the choice of approximation is extremely critical and different approximation settings should be chosen for different scenarios to get the best performance (see Fig. 22, 23). However, the goal is not only to get the best performance per scenario but also to design a robust IBC system with a low failure probability (FP) as explored in Sec. IX. We observe that settings $S_2$, $S_3$, $S_4$ and $S_6$ have a low FP across all scenarios (see Fig. 25). However, statically choosing $S_2$, $S_3$, $S_4$ or $S_6$ does not give us the best performance and motivates dynamically switching between settings based on the operating scenario (see Fig. 26).

Summing it all up, we observe that choosing $S_7$ for day & dusk, $S_3$ for night & fog, $S_6$ for dawn and $S_0$ for snow gives us the best balance between performance and robustness (see Fig. 26). For this solution, we obtain average improvements of 51% in QoC, 64% in energy and 83% in memory, with the worst-case FP (per km) of $9.6 \times 10^{-6}\%$. Thus, our design approach provides robust approximate IBC designs (with FP below the hardware and communication subsystems) with significant QoC, energy and memory improvements.

B. OVERHEAD OF DYNAMIC APPROXIMATION SELECTION

In Sec. VIII & Sec. IX, we highlight the benefits of dynamic selection of approximation settings for different environmental scenarios. In this section, we discuss the additional overheads of such an approach. First, we classify the different environment scenarios using a state-of-the-art CNN classifier, ResNet-50 [38]. ResNet-50 has a classification accuracy of 94.75% on the ImageNet 2012 classification dataset that consists of 1000 classes with 1.28 million training images, 50K validation images and 100k test images [38]. We choose ResNet-50 pre-trained on ImageNet and train it on our dataset (see Sec. IX) using transfer learning. On our dataset, it achieves a classification accuracy of 99.72%. Higher classification accuracy is due to the fewer number of output classes compared to ImageNet (6 considered in this work). ResNet-50 has a runtime penalty of 1.5 ms on the NVIDIA AGX Xavier [39] which is 6% of the overall runtime of LKAS (considering $S_0$). Additionally, ResNet-50 has an energy overhead of 4.2mJ which is 0.88% of the total energy consumption of LKAS. The higher energy efficiency comes from offloading ResNet-50 to the deep learning accelerator (DLA) in NVIDIA AGX Xavier.

Proposed scenario classifier is invoked every frame to perform classification (every 25 ms for $S_0$-$S_2$, every 8.3 ms for $S_3$-$S_8$). But in real driving conditions, the transition between different weather scenarios is less frequent. So, we believe that the frequency at which scenario classifier is invoked can be relaxed, thus, reducing the overhead of dynamic selection. This is out of the scope of this work and will be studied in future. Also, the runtime overhead of the scenario classifier can be reduced by performing latency hiding. The scenario classifier can be scheduled on the DLA in parallel with the PR stage to get lower sensor-to-actuation delay ($\tau$).

C. RESULT ANALYSIS IN THE CONTEXT OF ELECTRIC VEHICLES

Modern electric vehicles (EVs) consist of multiple of IBC systems. We are interested in understanding how the driving range (DR) of these EVs improve if our approximation-aware design approach is applied. To evaluate this, we consider a commercial EV (Tesla Model 3). It consists of 8 camera sensors for performing different control operations like traffic-aware cruise control, lane-keeping assist, auto lane change, auto park etc. We consider two cases - when only LKAS is active (3 cameras) and when all control operations are active simultaneously (8 cameras). The processed frames are $1280 \times 960$ in resolution. The energy consumption for the vehicle is 15.1 KWh/100 km [40]. Fig. 27 shows the driving range (DR) improvements obtained from our

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig27.png}
\caption{Driving range (DR) improvements in EVs obtained from our approach for the different active duration of IBC.}
\end{figure}

8Commercial edge-devices like NVIDIA Jetson Nano, Jetson TX2, Xavier NX operate in power budgets of 5-15W [37]. So, we perform our experiments by power-constraining NVIDIA AGX Xavier in 5-15W range.

9Tesla auto-pilot uses 3 cameras for LKAS
approach. We consider S6Optim1+2+3 as the design point which achieves 92%, 44% improvements in energy and QoC respectively. For this analysis, we scale the energy consumption linearly with the no. of processed pixels. For longer active periods of IBC (= 4 hr), we see the driving range (DR) is increased up to 13 km (for 3 cameras) and up to 35 km (for 8 cameras). This analysis is done assuming our approach is applied to other IBCs as well.

D. APPROXIMATIONS IN SAFETY-CRITICAL SYSTEMS
We consider a lane-keeping assist system (LKAS) which is a safety-critical system. Approximations are not intuitive solutions in safety-critical systems. But recent efforts like NVIDIA PilotNet [41], [42] and LaneNet [43] have shown that neural network-based algorithmic approximation approaches can be used for safety-critical systems. In this work, we show that compute and data-centric approximation approaches can be applied to a safety-critical system like LKAS when it comes with in-depth analysis and proper evaluation. We show that the quality of the overall system improves despite errors being introduced in the intermediate sub-systems. Additionally, failure probability analysis of the approximate system shows that failure rates are bounded within acceptable limits [7].

E. GENERALITY AND MODULARITY
1) GENERALITY
Proposed work applies to any feedback control system with camera-based sensing. These systems have massive compute-workload which can be reduced by approximating the sensing stage, as shown in this work. In essence, approximations improve the timing, energy and memory simultaneously at the cost of sensor noise. However, this additional sensor noise can be tolerated due to the inherent error resilience of closed-loop IBC systems, thus, improving the overall QoC. Although the gains reported in this work are application- and platform-specific, but the general idea is applicable to other IBC systems [1] like automatic pedestrian detection, vision-based predictive suspension systems and so on. Compute- and data-centric approximations, efficient platform mappings and approximation-aware control design are key steps in designing any energy-efficient IBC system.

2) MODULARITY
This work demonstrates that approximating the compute-intensive sensing stage \(T_s\) in an IBC system provides significant QoC, energy and memory benefits. Computation-skipping is explored in this work, which is one of the potential solutions for approximating \(T_s\). Other techniques such as DNN models can also be explored in future for replacing these compute-intensive stages \(T_s\). They can be applied to replace either the ISP stage [11] or the PR stage [43] or both. To easily facilitate model changes, we have developed our framework in a modular fashion. To integrate DNN based approximation in our framework, we just need the execution time of the DNN model (in inference mode) on considered hardware platform/inference engine. This parameter is used for designing approximation-aware control. For QoC-energy Pareto trade-offs, we will additionally need the energy consumption of the DNN model (in inference mode) on that hardware platform/inference engine.

XI. CONCLUSION
We have shown that compute-centric (ISP stage skipping) and data-centric (variable lossy compression) approximations are promising strategies for simultaneously optimizing QoC, energy and memory of IBC systems. This further opens up the possibility of using cost-efficient platforms with lower capacity (i.e., slower frequency and lower power consumption). We have shown with extensive experiments that approximation-aware control designs are suitable when we take into account the artefacts of approximation in closed-loop systems while optimizing for quality-of-control (QoC). Energy and memory improvements of up to 92% and 88% are shown, for 44% QoC improvements compared to the accurate implementation. Our design approach is shown to be applicable to a wide range of environmental scenarios. We have also shown that approximate IBC systems designed using our approach are robust with a failure probability (FP per km) \(\leq 9.6 \times 10^{-6}\%\). The presented approximation-aware approach is shown to have high potential in a lane-keeping assist system (LKAS). Our design approach can also be integrated with other approximation techniques, which is an interesting research direction to be looked into in future.

APPENDIX

SCHEDULING OF APPROXIMATE ISP PIPELINES ON GPU
Execution time of different ISP pipeline settings (S0-S8) depends on how they are scheduled on the GPU. To generate optimized schedules, we use the Halide GPU auto-scheduler proposed in [44]. The cost function of this auto-scheduler optimizes the pipelines by splitting it into groups/segments so that each segment can be executed with accesses to the GPU shared memory only. Also, each segment corresponds to a different CUDA kernel. When moving from one segment to another, we need accesses to the global memory. It should be noted that accesses to the global memory are orders of magnitude more costly compared to the ones to shared memory since DRAM bandwidth is often much lower than the one achieved by shared memory [44]. That’s why the execution time of these pipeline settings highly depend on the number of global memory accesses.

Fig. 28 shows the GPU schedules we obtain for the different pipeline settings (S0-S8) using the auto-scheduler [44]. For splitting the pipelines into segments, the auto-scheduler starts from the output stage and continuously merges it with the previous stages as long as the data can be fit on the shared memory. If not, then it splits the pipeline into two segments and starts the same process again in the non-visited segment. The following observations are made from the schedules:

- Gamut mapping (GM) and tone mapping (TM) cannot be put into a single segment as the data cannot fit in
shared memory. So, tone mapping (TM) is put in a separate segment in S0-S2.

- The output of demosaicing (DM) has a high memory footprint. So, demosaicing (DM) cannot be put a single segment with the subsequent stages. This results in a separate segment for demosaicing (DM) in S0-S8.

- The auto-scheduler prioritizes inlining of cascaded stages (within a segment) into the consumer which maximizes the producer-consumer locality.

We observe from Fig. 28, that settings S0-S2 have three segments/groups accessing the global memory while the rest S3-S8 have two segments/groups accessing the global memory. This explains the higher execution time of S0-S2 compared to S3-S8 in Fig. 9. This also explains why S3 and S4 (each with 4 stages) take the same time as S5-S8 (each with 2 stages). Also, the execution time does not scale linearly with the number of stages in the pipeline, which explains why all S5-S8 combined seems to take less time than S0 (or S1 and S2) despite having more stages (as demosaic is common to all of them). It should be noted that the slight runtime variations in S0-S2 are due to extra thread synchronization overheads introduced by intra-segment communication through shared memories. The same reasoning applies to S3-S8.

REFERENCES

[1] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, “Three decades of driver assistance systems: Review and future perspectives,” IEEE Intell. Transp. Syst. Mag., vol. 6, no. 4, pp. 6–22, Winter 2014.

[2] S. Chakraborty, M. Lukasiwycz, C. Buckl, S. Fahmy, N. Chang, S. Park, Y. Kim, P. Leitentüer, and H. Adlkofer, “Embedded systems and software challenges in electric vehicles,” in Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE), Mar. 2012, pp. 424–429.

[3] S. De, S. Mohamed, K. Binmpisids, D. Goswami, T. Basten, and H. Corporaal, “Approximation trade-offs in an image-based control system,” in Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE), Mar. 2020, pp. 1680–1685.

[4] NVIDIA AGX Xavier, Scalable AI Platform for Autonomus Driving. [Online]. Available: https://developer.nvidia.com/embedded/jetson-agx-xavier

[5] S. Mohamed, D. Goswami, V. Nathan, R. Rajappa, and T. Basten, “A scenario-and-platform-aware design flow for image-based control systems,” Microprocessors Microsyst., vol. 75, Jun. 2020, Art. no. 103037.

[6] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, “Learning lightweight lane detection CNNs by self attention distillation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 1013–1021.

[7] P. Bhuvasar, P. Das, M. Paugh, K. Dey, and M. Chowdhury, “Risk analysis of autonomous vehicles in mixed traffic streams,” Transp. Res. Rec., J. Transp. Res. Board, vol. 2625, no. 1, pp. 51–61, Jan. 2017, doi: 10.3141/2625-06.

[8] A. Mercat, J. Bonnot, M. Pelcat, W. Hamidouche, and D. Menard, “Exploiting computation skip to reduce energy consumption by approximate computing, an HEVC encoder case study,” in Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE), Mar. 2017, pp. 494–499.

[9] Y. Tian, Q. Zhang, T. Wang, F. Yuan, and Q. Xu, “Approxma: Approximate memory access for dynamic precision scaling,” in Proc. 25th Ed. Great Lakes Symp. (VLSI GLSVLSI), New York, NY, USA: Association for Computing Machinery, 2015, p. 337–342, doi: 10.1145/2742060.2743759.

[10] H. Esmaeilzadeh, A. Sampson, L. Cze, and D. Burger, “Neural acceleration for general-purpose approximate programs,” in Proc. 45th Annua. IEEE/ACM Int. Symp. Microarchitecture, Dec. 2012, pp. 449–460.

[11] H. Jiang, Q. Tian, J. Farrell, and B. A. Wandell, “Learning the image processing pipeline,” IEEE Trans. Image Process., vol. 26, no. 10, pp. 5032–5042, Oct. 2017.

[12] A. Chandrasekharan, D. Große, and R. Drechsler, “ProACI: A processor for high performance on-demand approximate computing,” in Proc. Great Lakes Symp. (VLSI GLSVLSI), New York, NY, USA: Association for Computing Machinery, 2017, pp. 463–466, doi: 10.1145/3060403.3060415.

[13] S. De, J. Huiskens, and H. Corporaal, “Designing energy efficient approximate multipliers for neural acceleration,” in Proc. 21st Euromicro Conf. Digit. Syst. Design (DSD), Aug. 2018, pp. 288–295.

[14] H. Jiang, F. J. H. Santiago, M. S. Ansari, L. Liu, B. F. Cockburn, F. Lombardi, and J. Han, “Characterizing approximate adders and multipliers optimized under different design constraints,” in Proc. Great Lakes Symp. (VLSI GLSVLSI), New York, NY, USA: Association for Computing Machinery, 2019, pp. 393–398, doi: 10.1145/3299887.3319454.

[15] S. De, J. Huiskens, and H. Corporaal, “An automated approximation methodology for arithmetic circuits,” in Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED), Jul. 2019, pp. 1–6.

[16] A. Raha, S. Sutar, H. Jayakumar, and V. Raghunathan, “Quality configurable approximate DRAM,” IEEE Trans. Comput., vol. 66, no. 7, pp. 1172–1187, Jul. 2017.

[17] J. S. Miguel, M. Badr, and N. E. Jerger, “Load value approximation,” in Proc. 47th Annua. IEEE/ACM Int. Symp. Microarchitecture, Dec. 2014, pp. 127–139.

[18] Y. Kim, S. Venkataramani, N. Chandrachoodan, and A. Raghunathan, “Data subsetting: A data-centric approach to approximate computing,” in Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE), Mar. 2019, pp. 576–581.

[19] A. Ranjan, A. Raha, V. Raghunathan, and A. Raghunathan, “Approximate memory compression,” IEEE Trans. Very Large Scale Integ. (VLSI) Syst., vol. 28, no. 4, pp. 980–991, Apr. 2020.

[20] S. Hashemi, H. Tann, F. Buttaufooco, and S. Reda, “Approximate computing for biometric security systems: A case study on iris scanning,” in Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE), Mar. 2018, pp. 319–324.

[21] A. Raha and V. Raghunathan, “Approximating beyond the processor: Exploring full-system energy-accuracy tradeoffs in a smart camera system,” IEEE Trans. Very Large Scale Integ. (VLSI) Syst., vol. 26, no. 12, pp. 2884–2897, Dec. 2018.

[22] M. Buckler, S. Jayasuriya, and A. Sampson, “Reconfiguring the imaging pipeline for computer vision,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 975–984.
[23] Y. Zhu, A. Samajdar, M. Mattina, and P. Whatmough, “Euphrates: Algorithm-SoC co-design for low-power mobile continuous vision,” in Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA), Jun. 2018, pp. 547–560.

[24] A. Raha, S. Venkataramani, V. Raghunathan, and A. Raghunathan, “Energy-efficient reduce-and-rank using input-adaptive approximations,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 2, pp. 462–475, Feb. 2017.

[25] 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 Proc. 8th Medit. Conf. Embedded Comput. (MECO), Jun. 2019, pp. 1–4.

[26] O. Michel, “Cyberbotics Ltd. Webots: Professional mobile robot simulation,” Int. J. Adv. Robotic Syst., vol. 1, no. 1, p. 5, Mar. 2004, doi: 10.5772/5618.

[27] AR1335 CMOS Digital Image Sensor for Automotive Applications. [Online]. Available: https://www.onsemi.com/products/sensors/image-sensors-processors/image-sensors/ar1335

[28] NVIDIA LaneNet. High-Precision Lane Detection. [Online]. Available: https://blogs.nvidia.com/blog/2019/07/10/drive-labs-neural-nets-prec.

[29] R. Frank. (Jul. 1, 2006). Steering in the Right Direction. Accessed: Aug. 17, 2020. [Online]. Available: https://www.electronicsdesign.com/markets/automotive/article21801204/steering-in-the-right-direction

[30] J. Kosecka, R. Blasi, C. J. Taylor, and J. Malik, “Vision-based lateral control of vehicles,” in Proc. Conf. Intel. Transp. Syst., 1997, pp. 900–905.

[31] G. F. Franklin, J. D. Powell, and M. L. Workman, Digital Control of Dynamic Systems, vol. 3. Menlo Park, CA, USA: Addison-Wesley, 1998.

[32] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe, “Halide: A language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines,” in Proc. 34th ACM SIGPLAN Conf. Program. Lang. Design Implement., 2013, pp. 519–530.

[33] A. Raha, A. Chakrabarty, V. Raghunathan, and G. T. Buzzard, “Embedding approximate nonlinear model predictive control at ultrahigh speed and extremely low power,” IEEE Trans. Control Syst. Technol., vol. 28, no. 3, pp. 1092–1099, May 2020.

[34] S. Mohamed, A. U. Awan, D. Goswami, and T. Basten, “Designing image-based control systems considering workload variations,” in Proc. 58th IEEE Conf. Decis. Control (CDC), Dec. 2019, pp. 3997–4004.

[35] The TuSimple Lane Detection Challenge. Accessed: Aug. 17, 2020. [Online]. Available: https://github.com/TuSimple/tusimple-benchmark

[36] K. He, S. E. Li, and D. Kun, “Synthesis of robust lane keeping systems: Impact of controller and design parameters on system performance,” IEEE Trans. Intell. Transp. Syst., vol. 20, no. 8, pp. 3129–3141, Oct. 2019.

[37] NVIDIA Edge Devices: Embedded Systems for Next-Generation Autonomous Machines. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems

[38] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[39] NVIDIA Deep Learning Inference Benchmarks. Accessed: Aug. 17, 2020. [Online]. Available: https://developer.nvidia.com/embedded/jetson-agx-xavier-dl-inference-benchmarks

[40] Energy Consumption of Full Electric Vehicles. Accessed: Aug. 17, 2020. [Online]. Available: https://jev-database.org/cheatsheet/energy-consumption-electric-car

[41] D. Neven, B. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool, “Towards end-to-end lane detection: An instance segmentation approach,” in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2018, pp. 286–291.

[42] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhao, and K. Zieba, “End to end learning for self-driving cars,” 2016, arXiv:1604.07316. [Online]. Available: http://arxiv.org/abs/1604.07316

[43] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, “Explaining how a deep neural network trained with End-to-End learning steers a car,” 2017, arXiv:1704.07911. [Online]. Available: http://arxiv.org/abs/1704.07911

[44] D. Neven, B. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool, “Towards end-to-end lane detection: An instance segmentation approach,” in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2018, pp. 286–291.

[45] S. Sioutas, S. Stuijk, T. Basten, H. Corporaal, and L. Somers, “Schedule synthesis for halide pipelines on GPUs,” ACM Trans. Archit. Code Optim., vol. 17, no. 3, pp. 1–25, Aug. 2020, doi: 10.1145/3406117.

**SAJID MOHAMED** (Member, IEEE) received the Bachelors of Technology (B.Tech.) degree in electrical and electronics engineering from the National Institute of Technology (NIT), Calicut, India, and the Masters of Technology (M.Tech.) degree in embedded controls and software from the Indian Institute of Technology (IIT) Kharagpur. He is currently a Researcher with the Electronic Systems Group, Department of Electrical Engineering, Eindhoven University of Technology (TU/e). His research interests include model-based design of embedded and cyber-physical systems.

**HENK CORPORAAL** received the M.Sc. degree in theoretical physics from the University of Groningen, and the Ph.D. degree in electrical engineering (computer architecture) from the Delft University of Technology.

He is currently a Professor of embedded system architectures with the Eindhoven University of Technology (TU/e), The Netherlands. He has coauthored over 500 journal and conference papers. He invented a new class of VLIW architectures, the transport triggered architectures, which is used in several commercial products, and by many research groups. His research interests include low power multi-processor, heterogenous processing architectures, their programmability, the predictable design of soft and hard real-time systems, research and design of embedded system architectures, including CGRAs, SIMD, VLIW, and GPUs, on accelerators, the exploitation of all kinds of parallelism, fault-tolerance, approximate computing, architectures for machine and deep learning, optimizations and mapping of deep learning networks, and the (semi-)automated mapping of applications to these architectures.