Approximation Trade Offs in an Image-Based Control System

***Abstract*—Image-based control (IBC) systems use camera sensor(s) to perceive the environment. The inherent compute-heavy nature of image processing causes long processing delay that negatively influences the performance of the IBC systems. Our idea is to reduce the long delay using coarse-grained approximation of the image signal processing pipeline without affecting the functionality and performance of the IBC system. The question is: how is the degree of approximation related to the closed-loop quality-of-control (QoC), memory utilization and energy consumption? We present a software-in-the-loop (SiL) evaluation framework for the above approximation-in-the-loop system. We identify the error resilient stages and the corresponding coarse-grained approximation settings for the IBC system. We perform trade off analysis between the QoC, memory utilisation and energy consumption for varying degrees of coarse-grained approximation. We demonstrate the effectiveness of our approach using a concrete case study of a lane keeping assist system (LKAS). We obtain energy and memory reduction of upto 84% and 29% respectively, for 28% QoC improvements.**

I. Introduction

Image-Based Control (IBC) systems are data-intensive feedback control systems whose feedback is provided by camera sensor(s). Emergence of low-cost CMOS cameras and efficient image-processing algorithms has contributed to the increasing popularity of IBC systems. They are widely used in automotive applications like advanced driver assistance systems (ADAS) and autonomous driving systems [1]. An IBC system consists of three main tasks: sensing (T_s), control computation (T_c) and actuation (T_a). T_s processes the frames captured by the camera, T_c executes the control algorithm and T_a applies the decisions taken by the controller as shown in Fig. 1. T_s is compute-heavy and the worst-case execution time (WCET) of T_s is usually several times higher than that of T_c and T_a (see Fig. 1), resulting in a long sensing-to-actuation delay τ (*delay* ($T_s + T_c + T_a$)).

The control performance of an IBC system can be improved by reducing the long sensing-to-actuation delay τ of the system [2]. The sensing task (T_s) is the main bottleneck in this. One way to reduce the long WCET of the sensing task (T_s) is to consider approximate computations. Approximate computing trades off quality of desired output for performance as well as energy improvements in error resilient applications. Prior literature has shown that image processing is inherently error resilient [3], [4]. So, in this work, we trade off the quality of the desired output in the sensing stage for performance, memory utilization and energy improvements of the entire IBC system while guaranteeing the system functionality.

Typical IBC systems are equipped with image signal processing (ISP) pipelines that are designed for photography [4]. The ISP stage preprocesses the RAW image data from the camera sensor and converts it to a compressed format. It produces high-quality images for visual consumption. Our key observation is that these photography-oriented ISP pipelines are an overkill for IBC systems, as the control algorithms do not need the high level of visual quality produced by these pipelines. So, *we consider different coarse-grained approximation settings on the ISP stage to evaluate its trade off on the control performance of the entire system.*

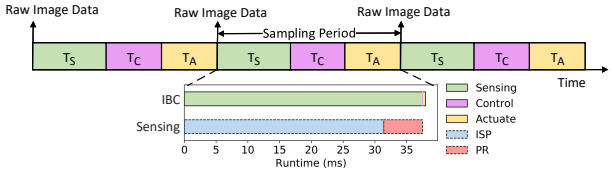

Fig. 1. Tasks in an IBC system: Runtimes for different sub-tasks are shown for a lane keeping assist system (LKAS).

The key contributions of this work are:

1) Identifying error resilient workloads/stages in the IBC system which can give maximum gains on approximation (using time and energy footprints of the entire system) (see Section IV-A).
2) Obtaining different coarse-grained approximation settings on the identified stages and analyzing its impact on the control performance of the entire IBC system (see Section IV-B).
3) Investigating the impact of approximating the ISP stage on the subsequent stages in the IBC system, in combination with approximation-aware tuning of the subsequent stages (see Section IV-C).
4) Trade off analysis between the quality of control (QoC), degree of approximation, memory utilization and energy consumption in an IBC system (see Section V).

This work focuses on the approximation of the ISP stage as it contributes to more than 80% of the entire workload of the system (see Fig. 1). We apply coarse-grained approximation by skipping different sub-stages in the ISP pipeline. Fine-grained approximations are also possible, but we limit the scope to coarse-grained ones. We demonstrate the effectiveness of our approach on the concrete case study of a lane keeping assist system (LKAS). Approximating the ISP pipeline requires approximation-aware application specific tuning of the subsequent stages. It is worth noting that our work is generic and can be applied to any IBC system with an ISP pipeline.

The rest of the paper is organized as follows: Section II discusses the related work. Section III gives a background on IBC systems with LKAS as a case study. Our proposed coarse-grained approximation methodology is described in Section IV. Section V gives the trade offs between QoC, degree of approximation, energy and memory. Section VI concludes the paper with a glimpse on possible future research directions.

II. Related Work

Approximate computing has been applied to different levels across the computing stack. However, prior research lacks an in-depth study on the impact of approximations on bigger closed-loop systems like IBC systems.

Approximations at different levels of the system stack: An algorithmic approximation approach for general-purpose programs is proposed in [5]. It trains a neural network to mimic and replace an error-resilient compute-intensive portion of a code segment. A similar learning based framework is proposed in [6] which automates the design of ISPs for new camera systems. An architectural approximation technique is reported in [7] which focuses on code acceleration using approximate multipliers in a neural accelerator. A hardware-based approximation technique is proposed in [4] that skips selected ISP stages.

Full system approximations: A method to approximate different stages of a smart-camera system like sensor, memory, compute and communication is proposed in [8]. A case study, showing benefits of approximations in biometric security systems, targeting an iris scanning application is demonstrated in [9]. A hardware-based approximation technique for a compute-intensive nonlinear model predictive control (NMPC) computation is shown in [10]. This approach considers approximation of control computation assuming that the sensing is accurate. However, approximating the sensing tasks will introduce errors into the state information of the control. This affects the control performance, which is not considered.

Prior research efforts have already focused on approximation of different individual components or subsystems [4]–[7]. However, they consider each subsystem as a stand-alone entity. There are two downsides to this approach:

1) These subsystems usually work as a part of bigger closed-loop IBC systems. So, approximating one subsystem might result in undesired behaviour in another subsystem, resulting in failure of the IBC system as a whole.

2) If a subsystem works as a part of a closed-loop system, then any approximation decision on that particular subsystem might have quality implications on the overall system at a later period of time. [15] presents tooling to evaluate closed-loop IBC systems, but it does not present a detailed analysis of approximation in closed loops. Other prior work fails to address closed-loop behaviour as it considers open-loop systems [8]–[10].

To address these limitations, this work makes an in-depth study on the impact of approximation on the control performance of the entire IBC system. To the best of the authors' knowledge, this is the first work that not only approximates different individual subsystems, but also evaluates their impact on a bigger closed-loop system.

III. Embedded Image-based Control

A typical IBC system consists of three main tasks, namely, sensing (T_s), control computation (T_c) and actuation (T_a) as shown in Fig. 2. In this work, we consider a lane keeping assist system (LKAS) as a concrete case study to analyze approximation trade offs on an IBC system. A LKAS helps to keep a vehicle centered in a detected lane autonomously. It applies mild steering torque if it identifies that the vehicle is drifting towards the side of a lane. The main stages in the LKAS system are ISP, perception (PR) and control computation. These stages are simulated and validated using a software-in-the-loop (SiL) simulator.

A. Image Signal Processing (ISP)

The ISP pipeline is responsible for converting the raw image data obtained from the camera sensor to a compressed image (JPEG in this work). ISP consists of a series of signal processing stages that are optimized to produce high-quality images suitable for human vision. There is no precise definition for the stages to be present in an ISP pipeline. Modern ISPs may comprise of hundreds of proprietary stages. We consider a set of stages common to all ISP pipelines as defined in [4]:

1) Demosaicing: The raw output of the camera sensor has a bayer layout in which each pixel contains either red, green or blue data. For a rgb image, each pixel must contain all three channels. Demosaicing fills the missing color channels by interpolating values from the neighboring pixels.

2) Denoising: The demosaiced image suffers from three sources of noise: *shot noise* due to physics of light detection, *thermal noise* of pixels and *read noise* from the readout circuitry of the camera sensor. Denoising exploits the spatial-similarity of the image to improve the signal-to-noise ratio, by averaging the neighboring pixels that resemble each other.

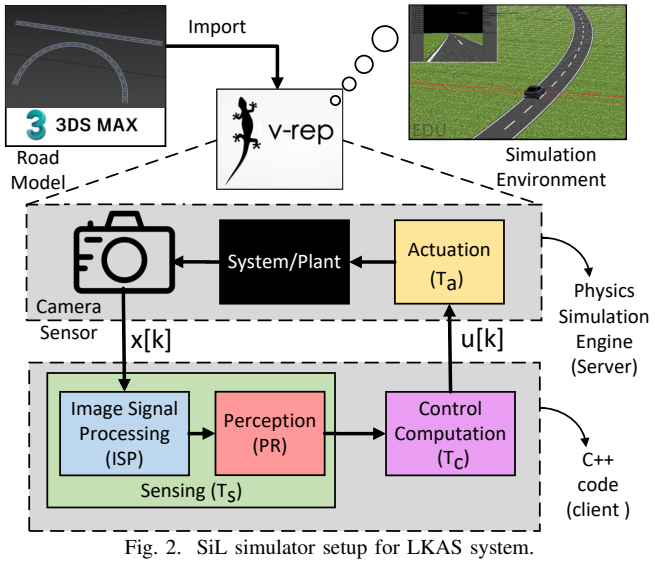

Fig. 2. SiL simulator setup for LKAS system.

3) Color Mapping & White Balancing: Color mapping reduces the intensity of the green channel to match that of the blue and the red. White balancing converts the color temperature of the image to match the lighting in the scene. These are per-pixel operations.

4) Gamut Mapping: Gamut mapping is a non-linear per-pixel transform that corrects the pixel values to be within a display's acceptable color range.

5) Tone Mapping: Tone mapping is another non-linear per-pixel transform which compresses the dynamic range of the

image and improves the contrast by making dark area brighter while not overexposing bright areas.

6) Compression: Image compression helps to reduce the storage as well as the data communication between different processing stages like the ISP, PR etc. We consider standard JPEG compression which uses DCT quantization to exploit signal sparsity in high-frequency space.

Different ISP stages along with their corresponding outputs are shown in Fig. 3(a). We consider only these stages as they represent the common functionalities that may impact the LKAS. In this paper, the CRIP [4] pipeline is used. It is optimized for performance using Halide [11].

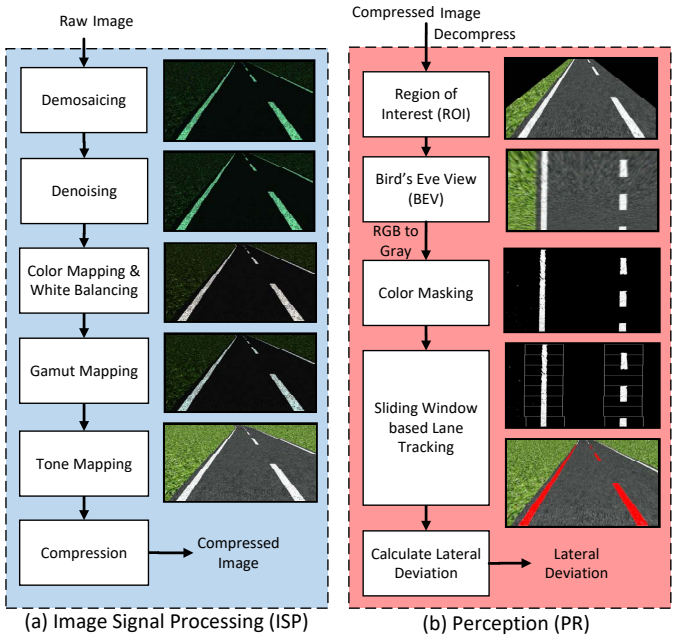

Fig. 3. Overview of ISP and PR stages.

B. Perception (PR)

The perception (PR) stage performs a set of application-specific preprocessing, feature extraction and inference steps on the image obtained from the ISP. For our LKAS setup, the PR stage extracts the lateral deviation of the vehicle from the center of the lane.

In the **preprocessing** step, first the ISP output is decompressed and then the region of interest (ROI) is extracted. A perspective transformation is performed on the ROI to get a bird's eye view of the look-ahead lane (see Fig. 3(b) row1,2).

The **feature extraction** step detects the candidate lane pixels from the bird's eye view of the image. First the RGB image is converted to a grayscale image to compute only on a single channel. Then a color masking step is applied which performs per-pixel static thresholding on the image. The outcome is an image with the white lane markers clearly differentiable from the rest of the image (see Fig. 3(b) row3). This is followed by a sliding window based lane detection technique. At first, a column wise histogram of the bottom half of the image is obtained which gives the base for the left and right lane markers. These base points are used to center the bottom pair of windows within which candidate lane pixels searched. Once all candidate pixels are identified, their mean position is used to center the next pair of windows and new candidate pixels are identified. This process is repeated until all windows within the height of the image have been covered (see Fig. 3(b) row4, 5).

In the **inference** step, the lateral deviation of the vehicle from the center of the lane is identified. The previously identified positions of left and right line markers are fit to a second degree polynomial. For a given look-ahead distance, the center of the lane is identified using these polynomials while the center of the image gives the vehicle's current position. Using these, the lateral deviation is calculated at the look-ahead distance.

C. Discrete-time control implementation (T_c)

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

$$\dot{x}_c(t) = A_c x_c(t) + B_c u(t),\ y_c(t) = C_c x_c(t), \qquad (1)$$

where $x_c(t)$, $u(t)$, $y_c(t)$ represents the *state*, *input* and *output* of the system, respectively. A_c, B_c, C_c represent the state, input and output matrices of the system, respectively. For our case study, we consider the model in [12], where

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

$C_c = [0\ 0\ 1\ 0\ 0]$. The control input $u(t)$ is the front-wheel steering angle δ_f and the output $y_c(t)$ is the lateral deviation from the desired centerline point at look-ahead distance y_L.

We consider a time-triggered implementation for actuation task T_a to guarantee constant sensing-to-actuation delay τ. Each setting of the coarse-grained approximations Si is annotated with a pair (h_i, τ_i) that models the sampling period and delay associated with it. The interval between two consecutive executions of the sensing task is defined as the sampling period. A zero-order sample-and-hold approach can then be used to discretize the system based on the coarse-grained approximation setting Si as follows:

$$x[k+1] = A_i x[k] + B_i^0 u[k] + B_i^1 u[k-1],\ y[k] = C_c x[k].$$

$$A_i = e^{A_c h_i},\ B_i^0 = \int_0^{h_i - \tau_i} e^{A_c s} ds\ B_c,\ B_i^1 = \int_{h_i - \tau_i}^{h_i} e^{A_c s} ds\ B_c.$$

We define new system states $z[k] = [x[k]\ \ u[k-1]]^T$ with $z[0] = [x[0]\ \ 0]^T$ to obtain a higher-order augmented system as follows:

$$z[k+1] = A_i^d z[k] + B_i^d u[k],\ A_i^d = \begin{bmatrix} A_i & B_i^1 \\ 0 & 0 \end{bmatrix},\ B_i^d = \begin{bmatrix} B_i^0 \\ I \end{bmatrix}$$

0 and I represent the zero and identity matrices. A check for controllability [13] is done for this augmented system. If the system is not controllable, controllability decomposition is done to obtain a controllable subsystem.

Control law: The control input $u[k]$ is a *state feedback* controller of the form given below, where F_i is the state feedback gain and $F_{f,i}$ is the integral gain both designed for the coarse-grained approximation setting Si.

$$u[k] = F_i z[k] + F_{f,i} \int (C_c z[k] - reference) \qquad (2)$$

To design F_i and $F_{f,i}$, we use the standard linear quadratic regulator control with integral action (LQI) [14].

D. Software-in-the-loop (SiL) simulator

A SiL simulator for LKAS is set up using the IMACS framework [15]. This SiL simulator simulates the plant or system using a physics simulation engine. The plant in this work is a vehicle with a top look-ahead camera. The simulator takes the raw image containing state information $x[k]$ from the camera sensor as input. This raw image goes through the ISP and PR stages, at the end of which the state information $x[k]$ is extracted. $x[k]$ is fed to the controller which computes the input $u[k]$ and communicates it back to the physics simulation engine for actuation. The frame rate for the camera can be set either in the simulation engine or can be triggered synchronously using the simulator.

IMACS includes the virtual robot experimentation platform (V-REP) [16] as the physics simulation engine. A 3D road environment is modeled using 3ds Max [17]. V-REP is operated using a remote API in client-server architecture, in which the server is the V-REP and the client is the software written in C++ (see Fig. 2). Each V-REP simulation step progresses in full synchronization with the software. The client receives the image captured by the camera sensor in V-REP, runs the software and sets the control input $u[k]$.

IV. Proposed Methodology

This section outlines the various design strategies for evaluating the trade off between QoC and approximation for an IBC system. Although we present our results for LKAS, similar strategies can be formulated for other IBC systems.

A. Where to spend the effort?

A LKAS consists of four main sub-stages (see Fig. 2): ISP, PR, control computation and actuation. Actuation depends on the vehicle dynamics and cannot be approximated. However, prior literature has shown that all ISP [4], [6], PR [5] and control computation [10] stages can be approximated. To figure out the best approximation opportunities in LKAS, which can give maximum gains in terms of performance and energy efficiency, we performed a full system profiling of LKAS. Both time and energy aware profiling is performed to find the 'hot' approximable regions in LKAS.

Fig. 4 shows the time and energy profiling results for LKAS. The profiling results are shown for a general purpose 8-core Intel i9 CPU[1] running ubuntu 18.04, with Kaby Lake architecture, 256KB L2 cache, 16MB L3 cache and 64 GB RAM. For **time profiling**, to eliminate the impact of cache misses, each stage in LKAS is executed 10000 times per input image. To consider image workload variations, each stage is tested for 200 different images. Refer to Algorithm. 1 for explanation of time profiling steps. For actuation, we consider a WCET of 0.5 ms [18]. For **energy profiling**, we need the power consumption of each stage along with its execution time (obtained from time profiling). We calculate

[1]Using an embedded automotive platform will also have similar profiling trends as the absolute profile values will change, not the relative values.

Algorithm 1: Time Profiling of LKAS

```
lkas ← {ISP, PR, Control}
database ← get 200 different images from V-REP
for each stage in lkas do
    for all i in 0 ≤ i < 200 do
        for all j in 0 ≤ j < 10000 do
            execute stage for input database[i]
            T_j ← store execution time
        end
        Min_i ← min{T_j | 0 ≤ j < 10000}^2
    end
    F_stage ← median{Min_i | 0 ≤ i < 200}^2
end
return F_ISP, F_PR, F_Control
```

the instantaneous power using the Running Average Power Limit (RAPL) interface [19] which estimates power usage by using hardware performance counters and I/O models.

It is evident from Fig. 4 that ISP takes the most (82%) of the total runtime of the system. Also, ISP consumes the most (93%) of the total energy of the system. So, we focus on approximation of the ISP as it will give maximum gains.

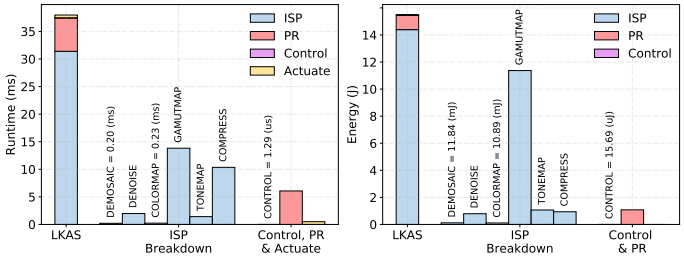

Fig. 4. Time and energy profiling of LKAS.

B. Coarse-grained Approximation

In this work, we apply coarse-grained approximations to the ISP by skipping one or more sub-stages within the ISP (see Fig. 3(a)). We believe that coarse-grained approximations can give highest performance and energy gains. However, if not done judiciously, they will result in high quality implications on the QoC of LKAS. To analyze this, we examine the sensitivity of QoC to each ISP stage of LKAS. Testing all possible stage combinations is intractable as it results in an exponential search space with high computational overheads. So, we start by skipping the entire ISP at once, which failed due to lack of any detailed features in the output image. Then we performed experiments by **skipping one stage at a time**. In these experiments, we noticed that skipping the demosaic stage causes LKAS to fail entirely, resulting in a vehicle crash. This is due to the fact that PR algorithms do not take the Bayer pattern into account. So, we conclude that QoC of LKAS is highly sensitive to the demosaic stage. Skipping other stages also resulted in the vehicle to crash intitally. However, minor modifications to the PR stage made it working (explained in Section IV-C). It is also noticed that skipping compression negatively impacts the energy and QoC of LKAS. The reason is that skipping compression increases the amount of data

[2]A typical target platform for LKAS implementation will have a real-time OS. We consider the *minimum* per image to reduce the impact of cache misses due to the non-real-time OS (ubuntu 18.04). Considering the *median* per stage compensates the impact of image workload variation.

TABLE I
CONSIDERED COARSE-GRAINED APPROXIMATION SETTINGS

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

DM: Demosaic, DN: Denoise, CM: Color Mapping,
GM: Gamut Mapping, TM: Tone Mapping, C: Compression

transmitted between DRAM and processor, and accessing DRAM is both slow and expensive in terms of energy. A second set of experiments is performed by **keeping one stage and disabling the rest of the pipeline**. For these experiments, we keep the demosaic as well as the compression stage for reasons previously mentioned. The quality implications of these experiments are reported in Section V. Based on the above discussions, we choose nine different approximation settings (see Table I), which are used for further trade off evaluation between QoC, degree of approximation, memory and energy.

C. Impact of ISP Approximation on Subsequent Stages

Approximating the ISP stage of LKAS has quality implications on other subsequent stages as well, especially, the PR stage. It is observed during experimentation that certain approximated image streams (obtained from S1-S8) are not properly handled by the PR stage, resulting in the failure of LKAS operation. A close investigation shows that the color masking step, which performs a per-pixel static thresholding on the image, fails to identify the lane markers. To counter this, we use Otsu's binarization algorithm which dynamically identifies an optimal threshold by operating on bimodal histograms. The downside to this approach is the higher computational complexity of dynamic thresholding over static thresholding. So, we added a quality check which first performs static thresholding, and switches to dynamic thresholding if and only if static thresholding results in corrupted data. This results in proper LKAS operation across all approximation settings.

V. EXPERIMENTAL RESULTS

I. Experimental Setup: We evaluate the nine different approximation settings mentioned in Table I using the IMACS SiL simulator for LKAS. We consider two different scenarios: a *straight road* with an initial positional bias for the vehicle from lane center, a *curved road* with no initial positional bias from lane center. The V-REP simulation step is set to 5 ms and the vehicle speed is set to 80 km/hr.

II. Quality Metrics: We evaluate the approximation quality of settings S0 to S8 using the *Structural Similarity (SSIM)* index. The SSIM index for two images m, n is defined as:

$$SSIM(m,n) = \frac{(2\mu_m\mu_n + C_1)(2\sigma_{mn} + C_2)}{(\mu_m^2 + \mu_n^2 + C_1)(\sigma_m^2 + \sigma_n^2 + C_2)} \quad (3)$$

where μ_m, μ_n, σ_m, σ_n and σ_{mn} are the local means, standard deviations, and cross-covariance for images m, n. C_1, C_2 are constants. High SSIM loss signifies images with higher visual difference. For evaluating the QoC of the proposed IBC system, we consider the following four metrics:

1) *Mean Square Error (MSE)*: mean cumulative sum of the squared errors, i.e.
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y[k] - reference)^2 \quad (4)$$
where n is the no. of samples and $y[k]$ is the value of the k^{th} sample. A lower MSE implies a better QoC.
2) *Settling Time (ST)*: time required for $y[k]$ to reach and stay within a certain range (5% in this work) of the final reference value. A lower ST implies a better QoC.
3) *Power Spectral Density (PSD)*: power present in the signal as a function of frequency, per unit 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 $max_k||u[k]||$. A lower MCE implies better QoC.

III. QoC-Approximation Trade offs: There are two main aspects to the QoC-Approximation trade offs that needs to be emphasized. First, the loss in image quality due to approximation may degrade the QoC of LKAS. Second, the reduced sensing time (T_s) due to approximation may improve the QoC due to faster sampling of the controller. The interplay between these two aspects determines if we gain or lose in the final QoC. We discuss these aspects below.

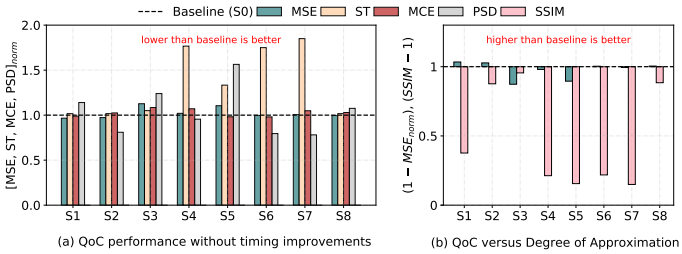

Fig. 5. Analysis of QoC degradation due to image quality loss. Results are shown for a straight road scenario.

1) QoC degradation due to loss in image quality

For this evaluation, all the approximation settings from S1 to S8 are simulated using the sensing-to-actuation delay of the accurate version S0 (τ_0). No timing improvements due to approximation are considered. Fig. 5(a) shows the performance of the four QoC metrics, MSE, ST, MCE and PSD, for different approximate settings S1-S8, each corresponding to different levels of quality loss in output images. All results are normalized to the accurate setting S0 (baseline). We notice that ST for S4-S7 degrade compared to the baseline due to higher oscillations in output. But all the approximation settings perform relatively similar to baseline in terms of MSE & PSD, sometimes even improving on it. *This proves that ISP pipelines optimized for human vision are an overkill for LKAS.*

Fig. 5(b) shows the trade offs between QoC and degree of image approximation. We consider MSE as the QoC metric for this study. We notice that even though some settings produce images with high SSIM loss (more than 80% for S6, S7),

they have minor impact on QoC. For instance, S7 and S8 have the same QoC as the baseline, even though S7 has 85% SSIM loss, while S8 has only 11% SSIM loss. High SSIM loss signifies images with higher visual difference in terms of contrast, white balance, tint etc. However, the lane marking features are not affected by these visual tunings, which results in proper LKAS operation. This explains the minor impact of high SSIM loss on QoC. This also shows that approximating a subsystem has different quality implications when considered as part of a bigger closed-loop system compared to considering it as a standalone component.

2) QoC improvement due to reduced sensing delay

In this experiment, we evaluate the impact of reduced sensing delay on QoC of LKAS by considering the timing improvements obtained from approximation. Fig. 6(b) shows the reduced sensing-to-actuation delay for each approximate setting S1-S8 (normalized to S0). These reduced delays are considered for the sampling period of the controller.

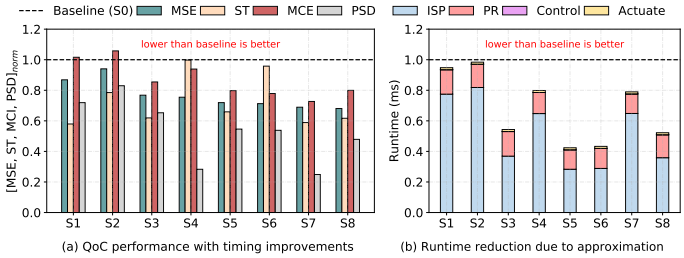

Fig. 6. Analysis of QoC improvement due to reduced sensing delay. Results are shown for a straight road scenario.

Fig. 6(a) shows that reduced sensing delay has a major impact on the improvement of QoC. We observe upto 32% (S8), 42% (S1), 75% (S7) and 27% (S7) improvements in MSE, ST, PSD and MCE respectively. *The negative impact on QoC due to image quality loss is not as significant as the positive impact of reduced sensing delay. This results in a better final QoC for LKAS.*

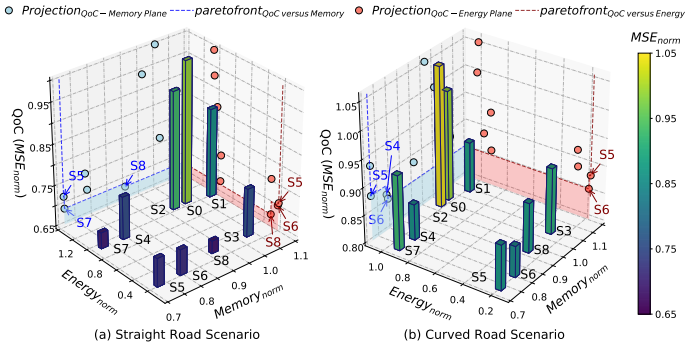

Fig. 7. Pareto fronts showing trade offs between QoC, Energy, Memory.

IV. QoC-Energy-Memory Trade offs: Fig. 6(a) shows that there are multiple approximation settings (S3, S4, S5, S6, S7, S8), which give similar improvements in the overall QoC of LKAS. But this shows a partial picture. We are interested in solutions which not only improve QoC, but also the energy efficiency and memory footprint of the system.

A Pareto analysis between QoC, energy and memory is shown in Fig. 7 for both straight & curved road scenarios. S4-S8 turn out to be Pareto-optimal. We obtain energy benefits of upto 84% for significant QoC gains (28% for straight, 11% for curve in S5). For the same QoC gains, memory footprint of the system is reduced by 29% (see S5 in Fig. 7). Overall QoC degrades for S2 in case of a curved road (see Fig. 7(b)). This is due to marginal timing improvements of S2, resulting in a higher impact of image quality loss on QoC. The QoC-energy Pareto solutions (S5, S6, S8) keeps fewer stages in the ISP. This gives the most reduction in the sensing delay, thus, not only improving QoC, but also consuming least energy.

VI. Conclusion and Future Work

Image-based control (IBC) systems use image signal processing to pre-process the camera sensor data and obtain feedback information. Image signal processing pipelines are compute-heavy and their long processing delays negatively influence the performance of IBC systems. In this work, we apply coarse-grained approximation to reduce this delay while guaranteeing proper functionality. We show an in-depth study on how the degree of approximation influences the closed-loop quality-of-control (QoC), memory utilization and energy consumption. We evaluate our technique using a software-in-the-loop framework for a lane keeping assist system (LKAS). We show energy and memory reduction of upto 84% and 29% respectively, for 28% QoC improvements. Future work aims to validate the technique using a hardware-in-the-loop framework.

References

[1] K. Bengler et al., "Three decades of driver assistance systems: Review and future perspectives," *IEEE ITSM*, 2014.

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

[3] V. Chippa et al., "Analysis and characterization of inherent application resilience for approximate computing," in *DAC*, 2013.

[4] M. Buckler et al., "Reconfiguring the imaging pipeline for computer vision," in *ICCV*, 2017.

[5] H. Esmaeilzadeh et al., "Neural acceleration for general-purpose approximate programs," in *IEEE/ACM MICRO*, 2012.

[6] H. Jiang et al., "Learning the image processing pipeline," *IEEE Transactions on Image Processing*, 2017.

[7] S. De et al., "Designing energy efficient approximate multipliers for neural acceleration," in *DSD*, 2018.

[8] A. Raha et al., "Approximating beyond the processor: Exploring full-system energy-accuracy tradeoffs in a smart camera system," *IEEE TVLSI*, 2018.

[9] S. Hashemi et al., "Approximate computing for biometric security systems: A case study on iris scanning," in *DATE*, 2018.

[10] A. Raha et al., "Embedding approximate nonlinear model predictive control at ultrahigh speed and extremely low power," *IEEE Transactions on Control Systems Technology*, 2019.

[11] J. Ragan-Kelley et al., "Halide: A language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines," in *ACM SIGPLAN PLDI*, 2013.

[12] J. Kosecka et al., "Vision-based lateral control of vehicles," in *Proceedings of Conference on Intelligent Transportation Systems*. IEEE, 1997.

[13] R. C. Dorf et al., *Modern control systems*. Pearson, 2011.

[14] P. C. Young et al., "An approach to the linear multivariable servomechanism problem," *International journal of control*, 1972.

[15] S. Mohamed et al., "IMACS: a framework for performance evaluation of image approximation in a closed-loop system," in *MECO*, 2019.

[16] E. Rohmer et al., "V-REP: a versatile and scalable robot simulation framework," in *IROS*, 2013.

[17] K. L. Murdock, *3ds Max 2012 bible*. John Wiley & Sons, 2011.

[18] Randy Frank, "Steering in the Right Direction," *Electronic Design*, 2016.

[19] E. Rotem et al., "Power-management architecture of the intel microarchitecture code-named sandy bridge," *IEEE Micro*, 2012.