Threshold-Guided Design and Optimization for Harris Corner Detector Architecture

Bhavan A. Jasani, Siew-Kei Lam, *Member, IEEE*, and Pramod Kumar Meher, *Senior Member, IEEE*, and Meiqing Wu

***Abstract*—High-speed corner detection is an essential step in many real-time computer vision applications, e.g. object recognition, motion analysis and stereo matching. Hardware implementation of corner detection algorithms such as the Harris corner detector (HCD) has become a viable solution for meeting real-time requirements of the applications. A major challenge lies in the design of power, energy and area efficient architectures that can be deployed in tightly constrained embedded systems while still meeting real-time requirements. In this paper, we proposed a bit-width optimization strategy for designing hardware-efficient HCD that exploits the thresholding step in the algorithm to determine interest points from the corner responses. The proposed strategy relies on the threshold as a guide to truncate the bit-widths of the operators at various stages of the HCD pipeline with only marginal loss of accuracy. Synthesis results based on 65-nm CMOS technology show that the proposed strategy leads to power-delay reduction of 35.2%, and area reduction of 35.4% over the baseline implementation. In addition, through careful retiming, the proposed implementation achieves over 2.2 times increase in maximum frequency while achieving an area reduction of 35.1% and power-delay reduction of 35.7% over the baseline implementation. Finally, we performed repeatability tests to show that the optimized HCD architecture achieves comparable accuracy with the baseline implementation (average decrease of repeatability is less than 0.6%).**

***Index Terms*—Corner detection, hardware acceleration, VLSI, embedded vision**

I. Introduction

REAL-time computer vision algorithms are extensively used in a wide range of applications such as vision-based navigation for image-guided medical interventions [1], navigation of unmanned vehicles [2] and robots [3], video encoding on unmanned aerial vehicles [4], video tracking [5] and visual SLAM [6]. A fundamental step in these applications is the detection of corners which represent identifiable anchor points in the image. Corners are used for visual odometry, stereo matching, optical flow computation, object tracking and as robust image representation when combined with feature descriptors for object recognition. Hence, it is important to have efficient implementation of the corner detection operations for deployment in embedded systems which have stringent performance, power, and resource constraints.

Several corner detectors have been proposed in the literature [7][8] and comparative evaluations have shown that the Harris corner detector (HCD) [9] achieves some of the best results [8][10]. The HCD is also the most widely used feature detection algorithm due to its robustness in detecting corners in noisy images [11]. However, the computationally intensive operations in the algorithm incurs a performance bottleneck and significant power consumption on general purpose microprocessors [11]. This is aggravated by fact that corner detectors contribute to a large portion of the overall runtime in many computer vision processes. For example, corner detectors are used for computing the ORB feature descriptor in visual SLAM [12]. The runtime of ORB computation contributes to over a third of the real-time tracking process. The work in [13] reported that the Harris algorithm contributes to over 36% of the execution time for stereo correspondence.

There are some recent attempts to reduce the computational complexity of corner detection algorithms such that they can be realized on embedded processors [14]-[16]. These techniques typically rely on pruning the search space for corners using simple approximations before applying a more complex corner measure step to evaluate the candidate set. While it has been shown that the method proposed in [14] achieves substantial speedup over the conventional algorithms, they are still unable to meet the power, performance, and area requirements of real-time embedded vision applications when realized on software-based platforms.

Motivated by the need to meet real-time requirements, hardware implementations of corner detection algorithms have been proposed [17]-[20]. These techniques often exploit the inherent parallelism in the corner detectors and find a reasonable trade-off between the number of line buffers and the

Manuscript submitted February 11, 2017.

Bhavan A. Jasani, was with School of Computer Science and Engineering, Nanyang Technological University, Nanyang Avenue, Singapore. He is now with Carnegie Mellon University, Pittsburgh, USA (e-mail: bhavan.jasani@gmail.com).

Siew-Kei Lam is with School of Computer Science and Engineering, Nanyang Technological University, Nanyang Avenue, Singapore (e-mail: siewkei_lam@pmail.ntu.edu.sg).

Pramod K. Meher is with School of Computer Science and Engineering, Nanyang Technological University, Nanyang Avenue, Singapore (e-mail: ASPKMeher@ntu.edu.sg).

Meiqing Wu is with School of Computer Science and Engineering, Nanyang Technological University, Nanyang Avenue, Singapore (e-mail: meiqingwu@ntu.edu.sg).

computational resources to manage the resource utilization while ensuring high throughput and acceptable result quality. However, there has been little effort undertaken to investigate data-path optimizations for achieving power-delay-area gains. Reducing power and energy consumption is particularly important given the increasing use of complex computer vision applications on battery-operated embedded devices [21]-[23].

Our study reveals that the complexity of the operators in the HCD data-path increases notably from one pipeline stage to the next due to bit-width aggregation. The HCD algorithm computes the corner response for every pixel in the image, and the pixel position is considered a corner if the corresponding corner response exceeds a given threshold. The bit-width of the Harris corner response is about 6 times larger than the input pixel bit-width. We propose a novel bit-width optimization strategy for hardware-efficient HCD implementation based on the intuition that the errors arising from bit-width truncation that are lesser than the threshold, will not result in loss of accuracy. As such, the threshold can be used to determine the bit-width truncation at various stages in order to achieve power-delay-area gains while guaranteeing the robustness of the implementation. Synthesis results based on 65-nm CMOS technology show that the proposed strategy can lead to significant power-delay-area gain over the baseline algorithm without bit-width truncation. To the best of our knowledge, the proposed strategy is the first to exploit the threshold value for reliably optimizing the bit-width in hardware realizations of computer vision algorithms.

The throughput of the HCD architecture can be further increased through pipelining. However, classical retiming methods that rely on discrete timing models of the operators do not lead to efficient pipelining. The proposed bit-width optimization strategy facilitates bit-level estimation of the propagation delays across different paths of the HCD architecture for accurate estimates of the critical path. This provides a means for applying pipelining at finer granularity to the HCD architecture, which leads to significant improvement in the throughput and power-delay efficiency. While we only focused on HCD in this paper, the proposed strategies can be applied to a wide range of computer vision algorithms (e.g. edge detection and object detection) that employs thresholding to obtain results.

The rest of the paper is organized as follows. Section II discusses previous work on hardware implementation of the HCD algorithm and existing techniques on bit-width optimization. Section III provides a brief overview of the HCD algorithm and the baseline hardware implementation. Section IV describes the proposed bit-width optimization strategy and the optimized hardware implementation of HCD. We also show that the performance of the optimized HCD implementation can be further improved through seamless pipelining using the connected timing model. In Section V, we evaluate the accuracy of the proposed hardware implementations using repeatability tests and provide the synthesis results to demonstrate the power-delay-area gains of the proposed strategy. We conclude the paper in Section VI.

II. Related work

In this section, we discuss existing work on hardware implementations of HCD and previously reported bit-width optimization strategies. At the end of this section, we list the main contributions of this paper.

A. Harris Corner Detector Architectures

Hardware implementations of HCD have been proposed on ASIC [24], FPGA [20][25][26], cell processor [27], and SIMD architecture [28]. The HCD algorithm computes an auto-correlation matrix for each pixel using the first-order derivatives of the intensity values and this matrix represents the degree of intensity variations in different directions around the corresponding pixel. A complex corner measure computation is then performed for every pixel in the image. This step is highly compute-intensive, and becomes a bottleneck for real-time vision tasks. In [29], a modified number representation for the corner response is used in custom instructions on the NIOS II processor. In [30], a hardware implementation that performs HCD on a rank transform image instead of the original image is presented.

In [17], a frame buffer based approach is described in which every individual step of the HCD algorithm is sequentially carried out over the full image frame. This requires storing intermediate data of the entire frame. Multiple image regions can be processed separately in parallel which would allow very high operating frequency at the cost of large memory requirements. In [11][18]-[20], row buffer based hardware implementation of the HCD is employed, in which image is locally processed. A small scanning window is utilized to determine whether the center image pixel is a corner or not. This approach requires only a few row buffers. Since all the sequential steps of HCD algorithm are performed locally rather than over the complete image, the intermediate data can be stored in local registers which results in low storage requirements but at the cost of increase in latency. In [19], a visual pipeline architecture is proposed based on row buffers which is a combination of parallel and pipelined architecture to achieve a higher speed. Power, speed and area trade-offs have been previously explored for the row buffer approach by varying the scanning window size and number of row buffers. The work in [18] compared the resource utilization and speed of two architectures with varying scanning window size and different row buffer configurations.

Since many computer vision applications employing corner detectors run on tightly constrained embedded systems e.g. mobile robots, mobile devices, UAVs, etc., there is a need to investigate design techniques that not only leads to real-time computation but also results in low power and energy efficient solutions. However, existing hardware implementations of corner detectors often only investigate area-delay trade-offs between the number of row buffers and computational resources, while neglecting data-path optimizations that can potentially lead to low power or energy efficient realizations.

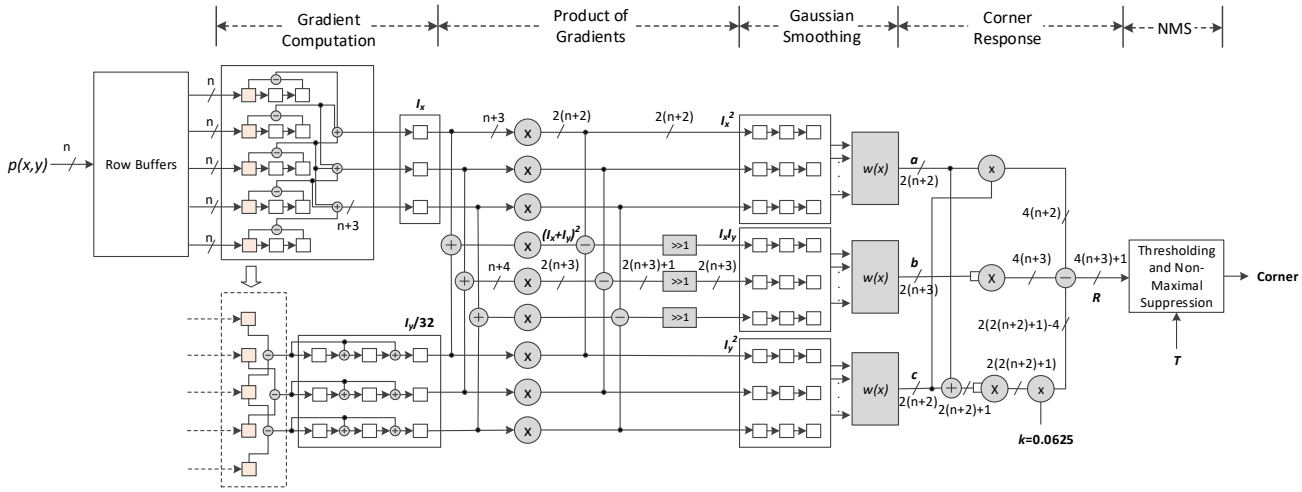


Fig. 1: Baseline HCD architecture with conservative bit-widths (each square box represents a register).

B. Bit-width Optimization

Bit-width optimization is an effective approach for reducing the hardware complexity by streamlining the data-paths using custom bit-width operators [31]-[35]. This can lead to large savings in area resources and power consumption. The challenge is to identify suitable bit-widths for each operator such that the overall implementation can still produce results with acceptable quality. The work in [11] truncates the Harris measure to 8 bits for more efficient hardware implementation, but did not provide any theoretical or analytical justification for the choice of bit-width truncation. An extra Gaussian filtering is incorporated between the Harris corner response computation and non-maximal suppression to reduce the resulting error due to saturation, which incur additional hardware resources. This work did not evaluate the impact of their bit-width truncation on the accuracy of the output. Most of the existing methods for determining optimal operator bit-widths of a given algorithm rely on methods that are based on interval arithmetic [36][37] and affine arithmetic [38]-[40]. Interval arithmetic assumes all values of arguments vary independently leading to drastic overestimation of the range. Affine arithmetic based methods overcome the limitations of interval arithmetic by providing tighter bounds on the range. The work in [40], which employs affine arithmetic, first calculates the ranges of all the intermediate signals (after each arithmetic operation) and output signals based on the range of input signals. This information is then used for determining the optimal bit-widths at every stage in the pipeline. Such an approach thus requires computations that are proportional to the number of arithmetic operations and any changes in the range of inputs requires entire re-computation. Exploring bit-width optimizations for varying ranges of the inputs (due to bit-width truncations) to meet the desired accuracy can therefore be a laborious process. In addition, non-affine invariant arithmetic operations like multiplication and division in which the resulting output is not in affine form, is approximated to an affine form. This can result in estimation error during bit-width truncation. In addition, this error can be propagated through the pipeline stages and eventually result in high inaccuracy at the output.

Our approach for finding optimal bit-width overcomes the shortcomings of interval arithmetic and affine arithmetic by providing an accurate estimation of truncation errors without relying on approximation. In addition, our approach requires only a one-time computation for deriving error equations that help determine the optimal bit-width to meet a desired accuracy. As such, the proposed method avoids the laborious effort required by affine arithmetic for determining the bit-width truncations of different input ranges. To the best of our knowledge, this work is the first to exploit the thresholding step found in many computer vision algorithms to optimize the operator bit-widths. Since the thresholding step effectively filters results that do not meet the given threshold, we can use the threshold value to guide bit-width truncation without incurring unacceptable loss of accuracy.

C. Main Contributions

The main contributions of this work are summarized as follow:

- To the best of our knowledge, this is the first work to demonstrate the power-delay-area advantages of bit-width optimization in corner detector architectures.
- We propose a novel threshold-guided bit-width optimization strategy for the HCD that enables effective accuracy and hardware-efficiency trade-offs. Based on the given threshold, the proposed strategy first rapidly identifies an initial bit-width optimized solution for the HCD pipeline that does not lead to accuracy loss. The initial solution is then further refined to obtain a final truncated bit-width solution that incurs marginal accuracy loss, while achieving large power-delay-area gains. It is worth mentioning that the proposed strategy can be easily adapted to other computer vision algorithms that employ thresholding (e.g. edge detection and object detection).
- We conducted detailed estimation of propagation delay at the critical path of the proposed HCD and performed efficient pipelining to further reduce the critical path delay and improve the throughput rate.
- Repeatability tests are undertaken to demonstrate that the optimized HCD architecture produces results with high accuracy.

III. Baseline Architecture for HCD

The HCD algorithm performs corner detection based on a local auto-correlation function that is approximated by matrix M within a small window W of each pixel $p(x, y)$ as shown in Eq. (1). I_x and I_y are the horizontal and vertical gradients, and $w(x)$ is the Gaussian weight function. The two eigenvalues of M, i.e. λ_1 and λ_2, indicate the intensity change in the window W centered on $p(x, y)$. Specifically, $p(x, y)$ is a corner if both the eigenvalues are large.

$$M = \begin{bmatrix} \sum_w w(x){I_x}^2 & \sum_w w(x) I_x I_y \\ \sum_w w(x) I_x I_y & \sum_w w(x){I_y}^2 \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \quad (1)$$

To avoid explicit computations of the eigenvalues, the HCD combines the eigenvalues into a single corner measure R as shown in Eq. (2), where k is an empirical constant and is usually set between 0.04 and 0.06. A threshold T is applied on the corner response to discard the obvious non-corners. The pixels with the highest corner response are then selected as corners after applying non-maximal suppression.

$$\begin{aligned} R &= \lambda_1 \cdot \lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2 \\ &= \det(M) - k \cdot trace^2(M) \\ &= (ac - b^2) - k \cdot (a + c)^2 \end{aligned} \quad (2)$$

Fig. 1 shows the baseline HCD architecture. We assume a single input pixel of n-bit (in our implementation n = 8 for grayscale image) arrives at each clock cycle. Five row buffers as shown in Fig. 2 are concatenated in the form of FIFO delay buffers to cache the incoming pixels. The size of each row buffer is equivalent to the horizontal resolution of the image, and hence each row buffer effectively delays the input by one row. The pixels at the tail end of each row buffer are shifted into the pipeline stages.

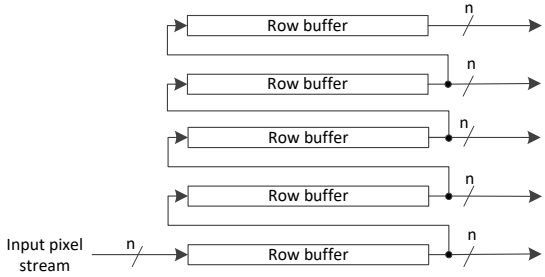


Fig. 2: Row buffers to cache incoming pixels.

The HCD architecture consists of the following five pipeline stages:

- *Gradient Computation*: The first pipeline stage computes the horizontal and vertical gradients I_x and I_y of the incoming pixels from the row buffers. The gradient computation is performed on a 5x3 neighborhood of pixels. This allows for simultaneous computations of three sets of I_x and I_y values (corresponding to the pixels in the middle column registers) at every clock cycle. In Fig. 1, the top block of the gradient computation stage computes the I_x gradients whereas the bottom block computes the I_y gradients. Note that the first column registers in the top and bottom blocks (highlighted in pink) are common and hence, the I_y computation block only requires 6 registers.
- *Product of Gradients*: The second pipeline stage computes the product of gradients to generate ${I_x}^2$, ${I_y}^2$ and $I_x I_y$. The multiplication operations for computing $I_x I_y$ can be reduced to less complex squaring, subtraction and 1-bit shift operations by utilizing the results of ${I_x}^2$ and ${I_y}^2$ as shown in Eq. (3). Three sets of ${I_x}^2$, ${I_y}^2$ and $I_x I_y$ values are generated in each clock cycle in this pipeline stage.

$$I_x I_y = \frac{(I_x + I_y)^2 - {I_x}^2 - {I_y}^2}{2} \quad (3)$$

- *Gaussian Smoothing*: Gaussian smoothing is applied to the product of gradients in this stage to produce a, b and c as shown in Eq. (1). This is achieved by caching the product of gradients using three sets of 3x3 registers and applying the Gaussian weight function $w(x)$ to compute a, b and c in parallel, where $w(x) = 0.0625 \cdot [(1\ 2\ 1)(2\ 4\ 2)(1\ 2\ 1)]^T$.
- *Corner measure*: The fourth pipeline stage computes the Harris corner measure R as shown in Eq. (2). Similar to the implementation in [17], we have approximated the value of k to be 0.0625 (1/16) so that constant shift operation can be used in place of a multiplier.
- *Non-Maximal Suppression*: In the final pipeline stage, the corner response R is first compared with a threshold T. If the corner measure is less than T, R is set to 0 (this avoids unnecessarily storing negative corner responses which are non-corners), otherwise the original value of R is retained. The corner responses are then cached using three row buffers as the non-maximal suppression (NMS) operation requires a 3x3 pixel region whereas only a single output is produced at the Corner Response stage. The NMS operation determines whether the center pixel of the 3x3 region is a corner or not by comparing its corner response to the corner responses of its 8 adjacent pixels. Fig. 3 shows the architecture of the NMS pipeline stage.

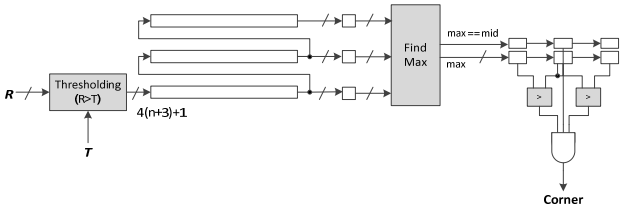


Fig. 3: Thresholding and non-maximal suppression.

IV. Proposed Implementation

It can be observed from the baseline HCD architecture in Fig. 1 that generally the operator bit-widths increase notably from one pipeline stage to the next due to the multiplication and addition operations. Specifically, multiplication leads to doubling of the operand bit-widths while addition results in an additional bit. For example, using the conservative bit-width assignment, if the bit-width of the pixel n = 8 (for gray scale images), the Harris corner measure R would be 45 bits wide. This is about six times the size of the input pixels.

TABLE 2: ERROR PROPAGATION EQUATIONS AT EACH PIPELINE STAGE DUE TOM-BIT TRUNCATION AT THE GRADIENT COMPUTATION STAGE

Pipeline Stage	Error Propagation Equations
Gradient Computation	$\Delta I_x = 2^m - 1$ $\Delta I_y = 2^m - 1$
Product of Gradients	$\Delta {I_x}^2 = 2 \cdot \lvert I_x \rvert \cdot (\Delta I_x)$ $\Delta {I_y}^2 = 2 \cdot \lvert I_y \rvert \cdot (\Delta I_y)$ $\Delta I_x I_y = \sqrt{\left(I_y \cdot \Delta I_x\right)^2 + \left(I_x \cdot \Delta I_y\right)^2}$
Gaussian Smoothing	$\Delta a \approx \Delta {I_x}^2$ $\Delta b \approx \Delta I_x I_y$ $\Delta c \approx \Delta {I_y}^2$
Corner Response	$\Delta R = \sqrt{(\Delta R_1)^2 + (\Delta R_2)^2 + (\Delta R_3)^2}$ $\Delta R_1 = \sqrt{(c \cdot \Delta a)^2 + (a \cdot \Delta c)^2}$ $\Delta R_2 = 2 \cdot \lvert b \rvert \cdot \Delta b$ $\Delta R_3 = (a + c) \cdot (\frac{\sqrt{(\Delta a)^2 + (\Delta c)^2}}{8})$

The significant bit-width increase implies that the complexity of the operators in the data-path increases notably from one pipeline stage to the next. Bit-width optimization can lead to significant reduction in computational resources, power reduction and increase in maximum operating frequency. The challenge lies in determining suitable bit-width truncation that will not compromise on the quality of results. However, none of the previously reported hardware implementation of corner detectors has addressed this challenge. In this section, we present a novel approach to determine the optimal bit width of the HCD based on the accuracy requirement, which is governed by the threshold value used in the thresholding step of the algorithm.

A. Threshold-Guided Bit-width Truncation

As discussed in the previous section, the Harris corner measure R for every pixel in the image will be compared with a threshold T in the final pipeline stage. This thresholding step is used to determine if the pixel is a candidate corner, i.e. pixels with $R > T$ are candidate corners, whereas pixels that do not meet the thresholding criteria are 'filtered' by setting their corner response R to 0. The threshold T used in HCD is typically a very large positive value (e.g. in the order of 10^{11}). Since the algorithm is concerned with large R values that exceed the threshold, the least significant bits (LSBs) of R and T can be removed if they do not affect a change in the thresholding decision. Specifically, if the maximum error in R (ΔR), that is incurred by truncating the bit-width does not exceed the threshold T (see Eq. (4)), then there will be no accuracy loss. We define ΔR as the *output error* and the inequality in Eq. (4) as the *maximum output error tolerance* for HCD.

$$\max(\Delta R) < T \quad (4)$$

Since bit-widths of the operators increase in a systematic fashion at every step of the algorithm, rather than truncating the LSBs of just the Harris corner measure, a more effective approach is to perform bit-width truncation of all the operators in the earlier pipeline stages while ensuring that the error propagation to R still satisfy Eq. (4). This can be achieved by first formulating ΔR as a function of m, which is the number of bits that is truncated at output of the first pipeline stage (i.e. Gradient Computation). Based on this formulation, we can then determine maximum number of m bits to be truncated at the first pipeline stage which will influence the bit-width truncation at the subsequent stages up to R, such that Eq. (4) is satisfied. As the error produced at the output of each pipeline stage is propagated from the error at the inputs and the error due to bit-width truncation of the operators, we need to derive the error propagation equations of each pipeline stage in order to determine ΔR, the resultant error in Harris measure.

TABLE 1: ERROR ANALYSIS OF COMMON OPERATIONS

Equation	Error
$A = X + Y - Z$	$\Delta A = \sqrt{(\Delta X)^2 + (\Delta Y)^2 + (\Delta Z)^2}$
$A = \frac{X.Y}{Z}$	$\frac{\Delta A}{\lvert A \rvert} = \sqrt{\left(\frac{\Delta X}{X}\right)^2 + \left(\frac{\Delta Y}{Y}\right)^2 + \left(\frac{\Delta Z}{Z}\right)^2}$
$A = c.X$	$\Delta A = \lvert c \rvert . \Delta X$
$A = X^n$	$\frac{\Delta A}{\lvert A \rvert} = \lvert n \rvert . \frac{\Delta X}{\lvert X \rvert}$
$A = f(X, Y, \dots)$	$\Delta A = \sqrt{\left(\frac{\Delta A}{\Delta X} . \Delta X\right)^2 + \left(\frac{\Delta A}{\Delta Y} . \Delta Y\right)^2 + \cdots}$

We use the error equations in Table 1 to derive the error propagation between the pipeline stages of the HCD architecture. Using basic differential calculus [41], one can derive equations for error in output due to error or uncertainty in the inputs of any function. Table 1 provides the error analysis for common arithmetic operations, which shows the error in output A due to uncertainty or error in the inputs for arithmetic operations on operands X, Y, Z, n, and constant c. The last equation of Table 1 shows the general equation based on differential calculus for finding error of an arbitrary function.

The error propagation equations at each pipeline stage that precede R are shown in Table 2. It can be observed that the error originating from the truncation of m bits in the first pipeline stage (Gradient Computation) propagates through all the intermediate stages to output R of the Corner Response stage to produce ΔR. For the case of Gaussian smoothing, the equations shown are approximated (actual equations are used in our experiments). The actual equations are complex as they depend on errors and values of variables of neighboring pixels. Based on these error propagation equations, we can determine the largest value of m such that Eq. (4) is satisfied. Next, we will describe our methodology for identifying suitable bit-width truncations at the various pipeline stages based on the principles discussed in this subsection.

B. Methodology

Fig. 4 illustrates the proposed methodology for bit-width optimization. Given an algorithm, the first step of the methodology derives the error propagation equations to the input of the thresholding process (e.g. Table 2) using the error analysis in Table 1. Next, the maximum output error tolerance (e.g. Eq. (4)) is defined based on the given threshold value. Based on this, the maximum number of bits (i.e. m) that can be truncated at the first stage of the algorithm (e.g. the first pipeline stage of the HCD implementation) is determined. This can be achieved in an iterative manner by evaluating incremental values of m starting from 1 on sample images until the maximum output error tolerance is violated. It is worth mentioning that this analysis does not have to be repeated in subsequent stages of the pipeline to determine the number of bits to be truncated. At the end of third step, the optimal bit-width truncation at the various stages of the algorithm can be determined while ensuring no accuracy loss. The fourth step of the methodology (dotted box in Fig. 4) is an optional step to further increase the value of m for further bit-width truncation and analyze the resulting error through empirical evaluations using sample images to perform trade-offs between the accuracy and bit-width optimization.

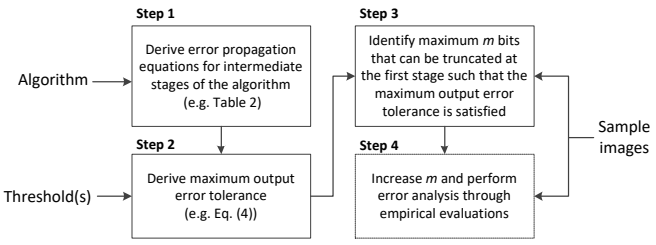


Fig. 4: Proposed methodology for bit-width optimization

The proposed methodology in Fig. 4 is applied to the HCD implementation, and the optimal value of m is found to be 3 (determined from Step 3 of Fig. 4). This means that if we truncate 3 bits at the first pipeline stage in Fig. 1 (Gradient Computation), and based on this truncate the bits proportionally in the subsequent pipeline stages leading to R, we will achieve a streamlined data-path that will not introduce accuracy loss. Note that Step 3 of the methodology can be performed rapidly without the need for comprehensive error analysis. The fourth step of the methodology is undertaken for evaluating the accuracy trade-offs when m is increased beyond 3. For the HCD implementation, we use the repeatability criteria [8] (discussed in the next section) to evaluate the accuracy trade-offs. We empirically found that if an additional 2 bits is truncated from the image gradients (i.e. $m = 3 + 2 = 5$), the average accuracy degrades by only 0.57%. This effectively truncates 20 LSBs of R, which is equivalent to removing 45% of bits in the original corner measure. Fig. 5 compares the number of bits in the baseline implementation and the proposed implementation (after bit-width optimization with $m = 5$) at the output of the HCD pipeline stages. As shown in Table 3 for $n = 8$, an average of 45% bit-width reduction is achieved at each pipeline stage.

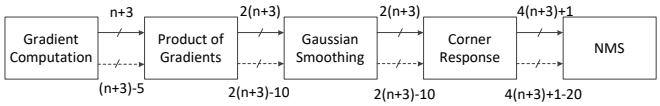


Fig. 5: Bit-width truncation at each pipeline stage (dotted lines show optimized bit-width).

Fig. 6 shows the number of bits m that can be truncated from the gradient computation block as a function of the threshold T. While the plot in Fig. 6 is obtained based on the "graf" image set in [42], it shows a general relationship between m and T. As T increases, the number of bits that can be truncated without affecting accuracy also increases. This confirms our hypotheses that a higher threshold T enables more bits to be truncated without compromising on accuracy. Fig. 6 also shows a step like behavior. This implies that there is a range of threshold T with a fixed value of m, i.e. bits that can be truncated without sacrificing accuracy. Such analysis enables us to determine a suitable threshold T to use (which impacts the number of corners generated) for a given set of images, that can provide the desired power-delay-area trade-offs.

TABLE 3: BIT-WIDTH COMPARISON BETWEEN BASELINE AND PROPOSED IMPLEMENTATION FOR $N = 8$

Pipeline Stage	Baseline	Proposed
Gradient Computation	11	6
Product of Gradients	22	12
Gaussian Smoothing	22	12
Corner Response	45	25

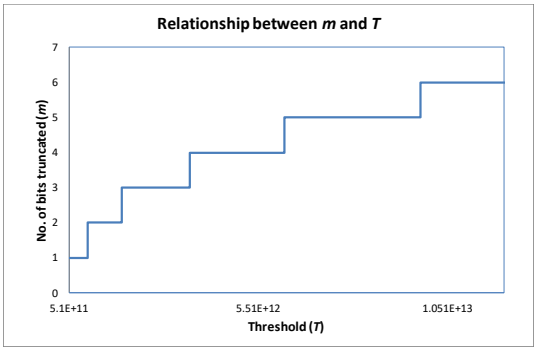


Fig. 6: Number of bits m that can be truncated in the first pipeline stage as a function of threshold T.

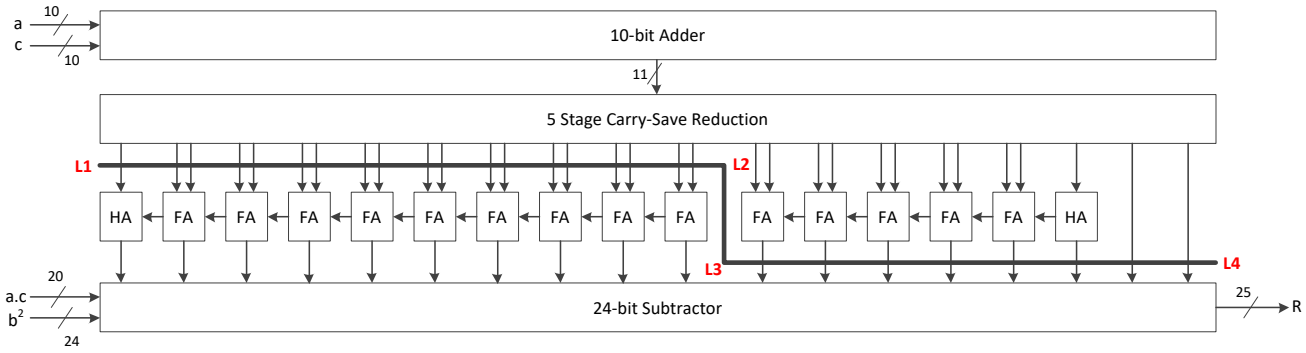


Fig. 8: Two-stage pipelining of Corner Response

C. *Seamless pipelining*

The bit-width optimization strategy discussed in the earlier sub-sections has led to significant reduction in the area and power consumption. In order to further increase the throughput of the architecture, we perform bit-level estimation of the propagation delays across different paths for accurate estimate of the critical path. Besides, we identify efficient feed-forward cut-sets to reduce the critical path to a minimum [43].

As shown in dotted red line in Fig. 7, the critical path of the baseline architecture is $T_{ADD} + T_{MULT} + T_{AA}$ where T_{ADD} is the delay of a 10-bit adder, T_{MULT} is the delay of a 11-bit multiplier and T_{AA} is the delay of a 3-input 24-bit subtractor. T_{ADD} can be approximated to be $9 \cdot T_{FAC} + T_{FA}$, where T_{FA} is the delay of a full adder and T_{FAC} is the delay required by a full adder to generate the output carry. T_{AA} can be approximated to be $23 \cdot T_{FAC} + 2 \cdot T_{FA}$ based on the connected timing model discussed in [43]. As such, we need to perform pipelining within the 11-bit multiplier in order to achieve effective register balancing. As explained in [44], it is not possible to achieve effective pipelining using array multipliers based on carry propagate adders. Hence, we have utilized the Wallace tree multiplier in our design.

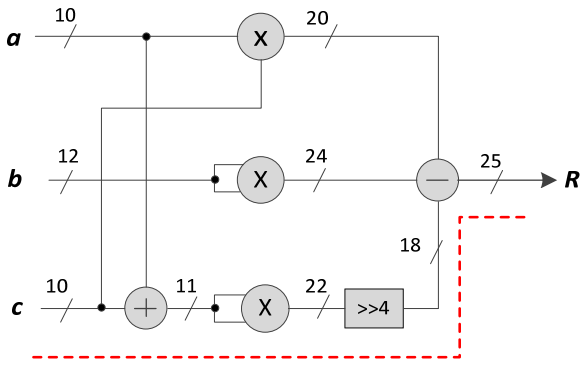


Fig 7: Critical path of HCD architecture (after bit-width optimization)

The proposed two-stage pipelining on the Wallace multiplier is shown in Fig. 8. It can be observed that pipeline registers are inserted along the line L1→L2→L3→L4 to partition the critical path of the Corner Response stage into two pipeline stages such that the critical path delay is divided into almost two equal halves. The first pipeline stage consists of the 10-bit adder followed by a carry save reduction of the Wallace multiplier and addition of 5 lower carry and sum bits. The critical path of the first pipeline stage is $(9 \cdot T_{FAC} + T_{FA}) + (4 \cdot T_{FA} + T_{HA}) + (T_{HA} + 5 \cdot T_{FAC})$, where T_{HA} is the delay of a half adder. The second pipeline stage consists of addition of 10 upper carry and sum bits and the 3-input 24-bit subtractor. The critical path delay of the second pipeline stage is $(9 \cdot T_{FAC} + T_{FA}) + (23 \cdot T_{FAC} + 2 \cdot T_{FA})$. The other multipliers along the non-critical paths of the Corner Response stage have also been pipelined in a similar manner.

V. Experimental Results

In this section, we will provide experimental results for the proposed implementation in terms of accuracy and hardware synthesis results.

A. *Accuracy Evaluation*

It is paramount to perform accuracy evaluation of the proposed architecture as bit-width optimization may affect the accuracy of corner detection. This may render the algorithm unsuitable for certain applications e.g. simultaneous localization and mapping for autonomous vehicles which involve tracking of features over consecutive frames. It is noteworthy that most of the previous works on hardware implementation of corner detectors do not provide quantitative accuracy evaluation of their implementations. In this paper, we have adopted the repeatability criteria [8] to compare the accuracy of the baseline and proposed implementations. The repeatability criterion is based on the notion that detection of corners should be invariant to imaging conditions e.g. blurring, zooming, and rotation of the scene. An accurate feature detector should be robust to the changes in imaging conditions and hence, should be able to detect features at close proximity between images with changes in viewpoint. The repeatability rate is defined as the ratio of the number of repeated features between two images within certain pixel allowance, to the minimum number of features that are in common region of the two images of the same scene but with changes in imaging condition(s).

We have used the image dataset from [42] for the accuracy evaluation. These challenging datasets contains three sequences (*Boat*, *Graf* and *UBC*) of images with various image transformations such as changes in viewpoint, zoom, rotation and illumination. Each set contains a base image and 5 images, where image transformation are progressively applied. Some samples of the images are shown in Fig. 9. The repeatability

criteria measure how well the detector match corresponding corners between the base image and images with transformations. It is expected that the repeatability progressively decreases with higher degree of image transformations.

Fig 9: Image set "Graf"; the top left image is the original and subsequent images have increasing viewpoint changes.

The results of the repeatability evaluation for the three image sets are shown in Fig. 10 (a pixel allowance of 1.5 pixels is used in the evaluations). The threshold T used for all the images considered is in the order of 10^{11}. Note that the values in the y-axis denote the percentage repeatability difference between the proposed architecture and that of the original software implementation of HCD. The negative difference in Fig. 10 means that the proposed architecture has accuracy degradation. For most images considered, the proposed architecture resulted in slightly better repeatability rate i.e. an increase in accuracy compared to the original HCD algorithm (as indicated by a positive value of the difference in repeatability) whereas for others there is a slight decrease in the repeatability rate. It is evident that the overall difference in repeatability is marginal for the image set considered, i.e. only 0.57% average reduction in repeatability. Hence, the proposed architecture has a high degree of robustness. It is noteworthy that although there's about 10% decrease in repeatability between proposed and original implementation for image 5 of *Boat*, there is still a high degree of matched pixels in the original and proposed implementation. In particular, only 4 out of 69 corners fail to match between the proposed and original implementation for this image.

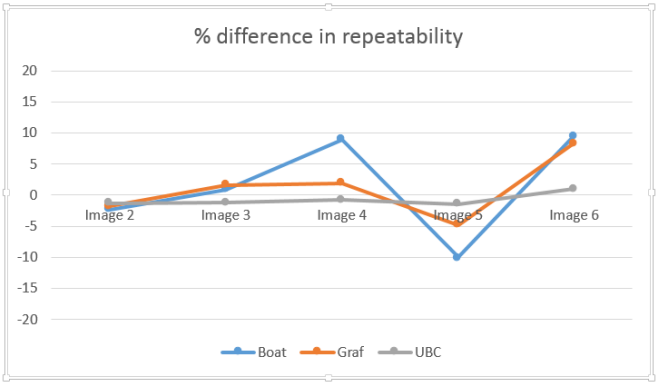


Fig 10: Difference in repeatability rate between proposed implementation and original HCD implementation.

We also observed that images with highly uniform texture lead to largest accuracy degradation for the proposed implementation since bit-width truncation removes the least significant bits of Harris measure, while only the least significant bits of the corner measure differ in uniform image textures. Since corner detectors are typically employed in non-uniform textured images where corners can be easily determined, this does not pose as a limitation to the proposed architecture. In the following sub-section, we discuss the power-delay-area gains of the proposed architecture.

B. ASIC Synthesis Results

The baseline architecture (*Baseline*), proposed architecture with bit-width optimization (*Proposed A*), and proposed architecture with bit-width optimization and seamless pipelining (*Proposed B*) were implemented using Verilog and synthesized with Synopsys DC using the TSMC 65-nm CMOS technology library.

Fig. 11 shows the percentage area, delay and power reduction of *Proposed A* over the *Baseline* when m, the number of bits truncated at the first stage of the algorithm, is varied from 1 to 5. The power is measured based on the maximum achievable frequency of each design. It can be observed that the percentage area, delay and power reduction increases monotonically with m. The percentage reduction in area and delay is expected since the bit-width truncation will lead to lesser computational resources and lower critical path delay. Even though *Proposed A* has higher operating frequency, it can still achieve lower power than *Baseline*.

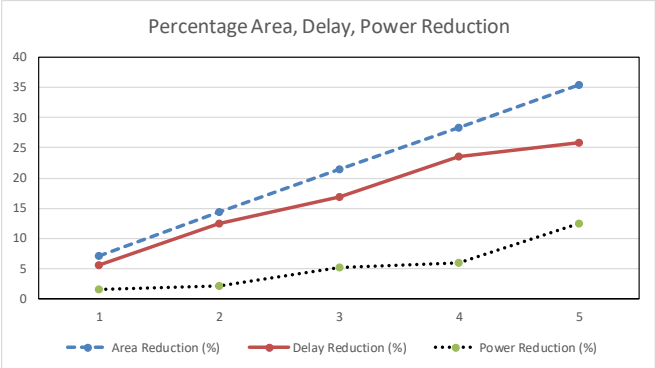


Fig 11: Percentage area, delay and power reduction of *Proposed A* compared with *Baseline* with varying m.

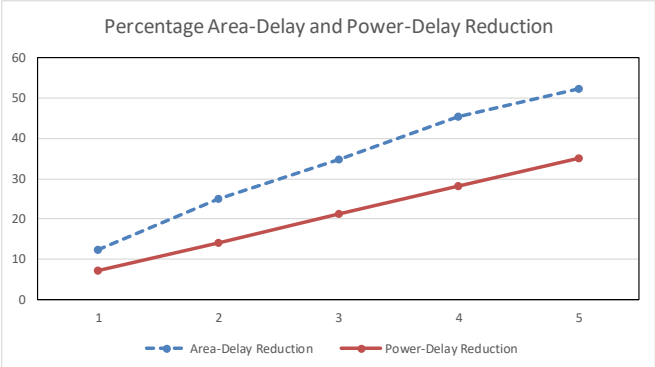


Fig 12: Percentage area-delay product and power-delay product reduction of *Proposed A* compared with *Baseline* with varying m.

TABLE 3: SYNTHESIS RESULTS FROM SYNOPSIS DESIGN COMPILER USING TSMC 65-NM LIBRARY

Design	Area (Sq um)	Maximum Frequency (MHz)	Power (mW)	Area-Delay Product	Power-Delay Product	Frames per second for 640 x 480 images	Frames per second for HD images (1280 x 720)
Baseline	1101088.40	224	174	4899843.38	773.23	731	243
Proposed A	710953.53	327	164	2168408.27	500.87	1067	355
Proposed B	714146.01	500	249	1428292.02	497.12	1627	542

TABLE 4: FPGA RESULTS

Design	Platform	Resolution	Resource Utilization			Maximum Frequency (MHz)	Performance (FPS)
			LUT-FF/Logic Elements	Memory Blocks	DSP Blocks/ Embedded Multipliers		
[19]	Altera Cyclone II	640x480	8050	360	29	-	50
[11]	Altera Aria V	1024x1024	8624	43	76	232	219
Baseline	Altera Cyclone IV	1024x1024	1990	20	33	115	109
Proposed A	Altera Cyclone IV	1024x1024	1191	14	15	176	167
Proposed B	Altera Cyclone IV	1024x1024	1964	13	9	211	201

Fig. 12 extends the analysis to compare the percentage reduction of the area-delay and power-delay product of *Proposed A* over *Baseline*. It is evident that this reduction is significant, where over 50% area-delay reduction and over 35% energy-delay reduction is observed when $m = 5$. These results clearly show the power-delay-area benefits of the proposed bit-width truncation method.

It can be observed from Table 3 that significant area and delay reduction can be achieved through the proposed bit-width optimization strategy. When compared to *Baseline*, *Proposed A* achieves 35.4% reduction in area and about 5.5% reduction in power. There is also a significant increase of 45.9% in the maximum achievable frequency. The percentage reduction in the area-delay product and power-delay product of *Proposed A* over *Baseline* is 55.8% and 35.2% respectively. When bit-width optimization is combined with seamless pipelining (*Proposed B*), there is a significant increase in maximum frequency over *Baseline*. In particular, *Proposed B* achieves over 2.2X increase in maximum frequency over *Baseline* while still achieving about 35.1% area reduction. The slight increase in area utilization of *Proposed B* over *Proposed A* is due to the additional registers that are included for seamless pipelining. Due to the incorporation of these registers and higher operating frequency, the power consumption of *Proposed B* is higher than *Baseline*. However, the significant reduction in the critical path delay of *Proposed B* has resulted in a high area-delay product reduction of about 70.9% over *Baseline*. The power-delay product reduction of *Proposed B* over *Baseline* is about 35.7%.

Based on these results, it is evident that the proposed optimization strategies have led to significant power-delay-area gains of the HCD architecture without compromising heavily on the accuracy (in most cases the proposed implementation has higher repeatability rate than *Baseline*). Both *Proposed A* and *Proposed B* have lower area utilization, lower critical path delay and higher energy efficiency than *Baseline*. Columns 7 and 8 of Table 3 show that the proposed implementations can lead to very high frame processing rates.

C. FPGA Synthesis Results

Table 4 compares the FPGA results of the *Baseline*, *Proposed A* and *Proposed B* implementations with the work in [11][19]. We would like to highlight that the existing work in Table 4 were implemented on different Altera devices. In general, *Baseline*, *Proposed A* and *Proposed B* requires significant lesser number of resources than [11] and [19]. The work in [11] truncates the Harris measure to 8 bits, which leads to higher operating frequency. However, the authors did not provide any theoretical or analytical justification for their choice of bit-width truncation. They also did not evaluate the impact of their bit-width truncation on the accuracy of corner detection. On the other hand, we proposed a methodology that leads to effective bit-width truncation without incurring notable degradation in accuracy.

VI. CONCLUSION

This paper presents a novel method for achieving high power-delay-area gains in the HCD architecture by exploiting the thresholding step in the algorithm to enable effective accuracy and hardware-efficiency trade-offs. The proposed method formulates the output error of the algorithm as a function of the number of bits that can be truncated at the preceding pipeline stages. Using this formulation and the maximum error tolerance that is defined by a threshold, the entire data-path of the HCD architecture can be streamlined with marginal loss in accuracy. We also applied seamless pipelining to further increase the throughput of the architecture. Repeatability tests are undertaken to demonstrate that the optimized HCD architecture produces results with high accuracy, and a FPGA prototype was developed to demonstrate the real-time capability of the proposed implementation. The methods proposed in this paper can be applied to numerous computer vision algorithms that rely on thresholding.

REFERENCES

[1] D.J. Mirota, M. Ishii, and G.D. Hager, “Vision-based navigation in image-guided interventions”, *Annual Review of Biomedical Engineering*, Vol. 13, pp. 297–319, 2011

[2] S. Ehsan, and K.D. McDonald-Maier, “On-board vision processing for small UAVs: time to rethink strategy”, *Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems*, 2009

[3] A. Schmidt, M., Kraft, and A. Kasinski, “An evaluation of image feature detectors and descriptors for robot navigation”, *Computer Vision and Graphics*, Vol. 6375, pp. 251–259, 2010

[4] M. Bhaskaranand, and J.D. Gibson, “Low-complexity video encoding for UAV reconnaissance and surveillance”, *Military Communications Conference*, pp. 1633–1638, 2011

[5] S. Gauglitz, T. Hollerer, and M. Turk, “Evaluation of interest point detectors and feature descriptors for visual tracking”, *International Journal of Computer Vision*, Vol. 94, pp. 335–360, 2011

[6] A. Gil, O.Mozos, M.Ballesta, and O.Reinoso, “A comparativeevaluation of interest point detectors and local descriptors forvisual SLAM”, *Machine Vision and Applications*, Vol. 21, pp. 905–920, 2010

[7] T. Tuytelaars, and K. Mikolajczyk, “Local invariant feature detectors: a survey”, *Foundations and Trends in Computer Graphics and Vision*, Vol. 3 No. 3, pp. 177-280, January 2008

[8] C. Schmid, R. Mohr, and C. Bauckhage, “Evaluation of interest point detectors”, *International Journal of Computer Vision*. Vol. 37, No. 2, pp. 151–172, 2000

[9] C. Harris, M. Stephens, “A combined corner and edge detection”, *Proceedings of The Fourth Alvey Vision Conference*, pp. 147-151, 1988

[10] H.Aanæs, A.L. Dahl, and K.S. Pedersen, “Interesting interest points: A comparative study of interest point performance on a unique data set”, *International Journal of Computer Vision*, Vol. 97, No. 1, pp 18-35, March 2012

[11] P.R. Possa, S.A. Mahmoudi, N. Harb, C. Valderrama, and P. Manneback, “A Multi-Resolution FPGA-based Architecture for Real-Time Edge and Corner Detection”, *IEEE Transactions on Computers*, Vol. 63, No. 10, pp. 2376-2388, October 2014

[12] R. Mur-Artal, J.M.M. Montiel, and J.D. Tardos, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”, *IEEE Transactions on Robotics*, Vol. 31, No. 5, October 2015, pp. 1147-1163

[13] V. H. Schulz, F. G. Bombardelli, and E. Todt, “A Harris Corner Detector Implementation in SoC-FPGA for Visual SLAM”, *Latin American Robotics Symposium*, pp. 57-71, September 2016

[14] M. Wu, N.Ramakrishnan, S.-K. Lam and T. Srikanthan, "Low-Complexity Pruning for Accelerating Corner Detection", *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2012, pp. 1684-1687

[15] N. Ramakrishnan, M. Wu, S.-K. Lam, and T. Srikanthan, “Enhanced Low-Complexity Pruning for Corner Detection”, *Journal of Real-Time Image Processing*, 2014

[16] N. Ramakrishnan, M. Wu, S.-K. Lam, T. Srikanthan, “Automated Thresholding for Low Complexity Corner Detection”, *NASA/ESA Conference on Adaptive Hardware and Systems*, 2014, pp. 97-103

[17] M.F. Aydogdu, M. F. Demirci, C. Kasnakoglu, “Pipelining Harris corner detection with a tiny FPGA for a mobile robot”,*Proc. IEEE International Conference on Robotics and Biomimetics*, ROBIO, 2013

[18] A. Amaricai, C.-E. Gavriliu and O. Boncalo, "An FPGA sliding window-based architecture Harris corner detector", *Field Programmable Logic and Applications (FPL), 2014 24th International Conference*, pp. 1-4

[19] P. Y. Hsiao, C.L. Lu, L.C. Fu “Multilayered Image Processing for Multiscale Harris Corner Detection in Digital Realization”, *IEEE Trans. On Industrial Electronics*, Vol. 57, Issue 5, 2010.

[20] T.L. Chao and K. H. Wong, "An efficient FPGA implementation of the Harris corner feature detector",14th IAPR *International Conference on Machine Vision Applications (MVA)*, 2015, pp. 89-93.

[21] A. Petrovai, A. Costea, F. Oniga, and S. Nedevschi, "Obstacle detection using stereovision for Android based mobile devices", *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2014

[22] J.M. S´aez, F. Escolano, and M.A. Lozano, "Aerial obstacle detection with 3D mobile devices", *IEEE Journal of Biomedical Health Information*, Vol. 19, No. 1, 74-80, 2015

[23] T. Schops, J. Engel, and D. Cremers, "Semi-dense visual odometry for AR on a smartphone*", IEEE International Symposium on Mixed and Augmented Reality*, 2014

[24] C. Chih-Chi, L. Chia-Hua, L. Chung-Te, S.C. Chang, C. Liang-Gee, "iVisual: an intelligent visual sensor SoC with 2790fps CMOS image sensor and 205GOPS/W vision processor", *Design Automation Conference*, pp. 90-95, 2008

[25] H. Orabi, N. Shaikh-Husin and U. Ullah Sheikh, "Low cost pipelined FPGA architecture of Harris Corner Detector for real-time applications", *International Conference on Digital Information Management*, 2015

[26] A. Hernandez-Lopez, C. Torres-Huitzil and J.J. Garcia-Hernandez, "FPGA-based flexible hardware architecture for image interest point detection", *International Journal of Advanced Robotic Systems*, Vol. 12, No. 93, 2015

[27] T. Saidani, L. Lacassagne, S. Bouaziz, T. Khan, "Parallelization strategies for the points of interests algorithm on the cell processor", *Parallel and Distributed Processing and Applications*, Vol. 4742, pp. 104-112, 2007

[28] F. Hosseini, A Fijany, J.-G Fontaine, "Highly parallel implementation of Harris Corner detector on CSX SIMD architecture", *Conference on Parallel Processing*, 2011

[29] S. Piskorski, L. Lacassagne, S. Bouaziz and D. Etiemble, "Customizing CPU instructions for embedded vision systems", *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, 2007

[30] B. Tippetts, D.-J. Lee and J. Archibald, "An on-board vision sensor system for small unmanned vehicle applications", *Machine Vision Applications*, 23(2), pp. 1-13, 2012

[31] Y. F. Tong, R. A. Rutenbar and D. F. Nagle, “Minimizing Floating-Point PowerDissipation Via Bit-Width Reduction”, *25th International Symposium on Computer Architecture*, 1998

[32] S. Gupta, A. Agrawal, K. Gopalakrishnan and P. Narayanan,“Deep Learning with Limited Numerical Precision”, *CoRR*, abs/1502.02551, 2015

[33] P. Gysel, M. Motamedi, S. Ghiasi, “Hardware-oriented Approximation of Convolutional Neural Networks”, *CoRR*, abs/1604.03168, 2016

[34] A. A. Gaffar, O. Mencer, W. Luk and P. Y. K. Cheung, "Unifying Bit-width Optimisation for Fixed-point and Floating-point Designs",*12th IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 79-88, April 2004

[35] A. Gaffar, J. Clarke and G. Constantinides, "PowerBit- Power aware arithmetic bitwidth optimization", *IEEE International Conference on Field Programmable Technology*, pp. 289-292, 2006

[36] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde and I. Bolsens,“A methodology and design environment for DSP ASIC fixed pointrefinement,” *Proceedings of ACM/IEEE Design Automation and Test in Europe Conference*, pp. 271–276, 1999

[37] M. Willems, V. Burgens, H. Keding, T. Grotker and H. Meyr, “Systemlevel fixed-point design based on an interpolative approach,”*Proceedings of ACM/IEEE Design Automation Conference*, pp. 293–298, 1997

[38] D-U. Lee, A. A. Gaffar, O. Mencer and W. Luk, "MiniBit: bitwidth optimization via affine arithmetic", *Proceedings of 42nd Design Automation Conference*, 2005

[39] J. Cong, K. Gururaj, B. Liu, C. Liu, Z. Zhang, S. Zhou and Y. Zou, “Evaluation of Static Analysis Techniques for Fixed-Point Precision Optimization”, *17th IEEE Symposium on Field Programmable Custom Computing Machines*, pp. 231-234, April 2009

[40] D.-U. Lee, A. Gaffar, R. Cheung, O. Mencer, W. Luk and G. Constantinides, "Accuracy-Guaranteed Bit-Width Optimization", *IEEE*

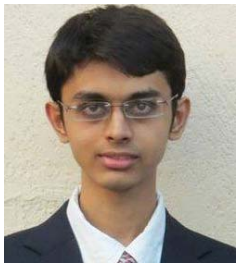
Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 10, pp. 1990-2000, Oct 2006

[41] Harvard University, "A Summary of Error Propagation", 2007. Available: http://ipl.physics.harvard.edu/wp-uploads/2013/03/PS3_Error_Propagation_sp13.pdf

[42] Affine Covariant Features, Available: http://www.robots.ox.ac.uk/~vgg/research/affine/

[43] P.K. Meher, "On Efficient Retiming of Fixed-Point Circuits", *IEEE Transactions on VLSI Systems*, 2015

[44] P.K. Meher, "Seamless Pipelining of DSP Circuits", *Journal of Circuits, Systems, and Signal Processing,* Vol. 35 Issue 4, Pages 1147-1162, April 2016



Bhavan A. Jasani received a dual degree consisting of B.E (Hons.) Electrical & Electronics Engineering and MSc. (Hons.) Physics from Birla Institute of Technology & Science, Pilani, India in 2016. Since then he has been working as a research staff at Hardware & Embedded Systems lab at School of Computer Science & Engineering, Nanyang Technological University, Singapore. He has been working on developing real-time, low-power and hardware efficient pedestrian detection systems based on FPGA's and embedded GPU's. He's research interest lies in computer vision, machine learning and embedded systems.

Siew-Kei Lam (M'03) received his BASc, MEng and PhD from School of Computer Science and Engineering (SCSE), Nanyang Technological University, Singapore. He is currently an Assistant Professor in SCSE and his research investigates methods for realizing custom computing solutions in embedded systems. His current projects include developing architecture-aware algorithms for vision-enabled sensing, and design methodologies for secure and reliable embedded systems.

Pramod Kumar Meher (SM'03) received the B.Sc. (Hons.) and M.Sc. degrees in physics, and the Ph.D. degree in science from Sambalpur University, Sambalpur, India, in 1976, 1978, and 1996, respectively. He was a Senior Research Scientist with the School of Computer Science & Engineering in Nanyang Technological University, Singapore during 2013-2016. Previously he was a Senior Scientist with the Institute for Infocom Research, Singapore during 2009-2013 and Senior Fellow with the School of Computer Engineering in Nanyang Technological University, Singapore during 2005-2009. He was a Reader in Electronics with Berhampur University, Berhampur, India, from 1993 to 1997, and a Professor of Computer Applications with Utkal University, Bhubaneswar, India, from 1997 to 2002. He has contributed nearly 250 technical papers to various reputed journals and conference proceedings including nearly 80 papers in IEEE Transactions. He is co-editor of the book "Arithmetic Circuits for DSP Applications" published by Wiley-IEEE Press. His current research interests include signal processing, cyber security, and intelligent computing for smart systems, IoT, and analytics. Dr. Meher is a fellow of the Institution of Electronics and Telecommunication Engineers, India and a Senior Member of IEEE. He was a Speaker for the Distinguished Lecturer Program of the IEEE Circuits Systems Society from 2011 to 2012. He has served as an Associate Editor of the IEEE Transactions on Circuits and Systems—II: Express Briefs from 2008 to 2011, the IEEE Transactions on Circuits and Systems—I: Regular Papers from 2012 to 2013, and the IEEE Transactions on Very Large Scale Integration Systems from 2009 to 2014. Currently he serves as an Associate Editor of the IEEE Transactions on Circuits and Systems for Video Technology and the Journal of Circuits, Systems, and Signal Processing. He was a recipient of the Samanta Chandrasekhar Award for excellence in research on engineering and technology in 1999.

Meiqing Wu received M.S. degree in Computer Engineering from Peking University, China in 2009, and her Ph.D. degree from the School of Computer Science and Engineering (SCSE), Nanyang Technological University, Singapore in 2017. Her current research interests include stereo vision, motion analysis, object detection and tracking for urban traffic scene understanding.