Stream-based ORB Feature Extractor with Dynamic Power Optimization

Phong Tran*†, Thinh Hung Pham*, Siew-Kei Lam*, Meiqing Wu*, and Bhavan A. Jasani‡
*Nanyang Technological University, Singapore
Email: n1702560d@e.ntu.edu.sg, pham_ht@ntu.edu.sg, siewkei_lam@pmail.ntu.edu.sg
†Ho Chi Minh city, University of Technology, Vietnam
‡Carnegie Mellon University, United States

***Abstract*—The Oriented Fast and Rotated BRIEF (ORB) feature extractor, which consists of key-point detection and descriptor computation, is a key module in many computer vision systems. Existing hardware implementations of ORB feature extractor only focus on increasing performance with power optimization as a post consideration. In this paper, we present a stream-based ORB feature extractor that incorporates mechanisms to lower the dynamic power consumption. These mechanisms exploit the fact that the number of detected key-points is typically small. The proposed solution significantly lowers the switching activity of the key-point detection and descriptor computation stages by early pruning of non-likely key-points and gating the descriptor computation stages. Further power reduction and resource minimization are achieved by employing a threshold-guided bit-width optimization strategy to truncate the redundant bits in the key-point detection stage. Finally, we propose an approximation method to achieve rotation invariance of the descriptors. FPGA implementation targeting the Altera Aria V device shows that the proposed strategies lead to over 25% reduction in dynamic power and lower resource utilization, with only marginal loss in accuracy.**

I. Introduction

Visual feature extraction is a fundamental module in many computer vision systems found in robotics, augmented reality, and autonomous vehicles. Many feature descriptors such as Scale-Invariant Feature Transform (SIFT) [1], Speeded Up Robust Features (SURF) [2], Oriented Fast and Rotated BRIEF (ORB) [3], etc. have been previously explored. Studies in [4], [5], [6], [7] have shown that the ORB descriptor achieves better performance compared to other descriptors.

ORB feature extraction consists of two steps: key-point detection and descriptor computation. Key-point detection determines the corners in each image frame using Features from an Accelerated Segment Test (FAST) [8] or Harris-Stephans Corner Detection (HCD) [9] algorithms. In the descriptor computation step, the rotation-aware Binary Robust Independent Elementary Feature (rBRIEF) [3] descriptor is extracted from an image patch that is centered at the detected key-points. Typically, a 256-bit rBRIEF descriptor is computed by performing binary tests on 256 pre-determined point-pairs in the image patch. In order to achieve rotation invariance, the point-pairs must be oriented based on the patch's moments.

In order to achieve real-time performance, a number of hardware implementations of the ORB feature descriptor has been reported. Existing hardware realizations of the ORB feature extractor typically relies on non-stream processing architectures (e.g. [10]). These architectures assume the availability of a frame buffer that stores the image frame. The work in [11] attempts to reduce the external memory bandwidth of ORB design through data reuse. This is achieved by reusing the same image patch for computing descriptors of two continuous key-points at close proximity in the same row. The work in [12] employs word length optimization by truncating the image moments to 8 bits to improve the throughput. This enables the implementation to achieve 20 frames per second (fps) on 640x480 image frame.

Stream-based processing architectures (e.g. [13]) are extremely attractive as they do not require external memories for storing input video frames and can achieve high throughput (since the memory fetch and store latencies are avoided). More recently, [14] presented a multi-level pyramid architecture for ORB feature extraction. The authors investigated the appropriate number of pyramid levels but did not elaborate on the details of the architecture. The existing works mainly focus on improving the throughput of the system while neglecting power efficiency, which is a primary concern in embedded systems.

Power optimization for stream-based ORB feature extractor is challenging. Firstly, the key-point detection algorithm (e.g. HCD) is computationally intensive due to the numerous addition and multiplication operators needed for calculating the corner response. Secondly, a large number of row buffers are typically used in stream processing to cache the input pixel stream, which contributes to significant dynamic power consumption and hardware resources. In particular, as the row buffers must be in continuous active state, existing power reduction techniques such as clock gating cannot be employed. Furthermore, the operations for computing the rBRIEF descriptors are generally kept in active state to maintain the desired throughput.

The main contributions of this work are as follows:

- We propose a new stream-based architecture for ORB feature extraction that utilizes an approximation angle discretization method to achieve rotation invariance of the descriptors that avoids costly operations. The proposed architecture significantly outperforms a recently reported implementation in terms of accuracy, resource utilization and power.

978-1-7281-0214-6/18/$31.00 ©2018 IEEE
DOI 10.1109/FPT.2018.00024

- We present strategies to lower the dynamic power by effectively inhibiting unnecessary signal activities in the key-point detection and descriptor computation pipeline. This is achieved by integrating the method in [15] that approximates the corner response at the early pipeline stages to detect the non-likely corners, and gating the descriptor computation units such that they are in idle state when no corners are detected.
- The power consumption is further lowered by introducing a bit-width optimization strategy in the key-point detection stage, which is based on the intuition that errors arising from the truncated bit-width that are less than the corner response threshold will not result in loss of accuracy.
- We will show that the proposed strategies maintain a marginal loss of accuracy while achieving notable savings in dynamic power and resource utilization on the Altera Aria V FPGA.

II. ORB Feature Extraction

In this section, we will provide a background on ORB feature extraction, which consists of key-point detection and rBRIEF descriptor computation.

A. Harris Corner Detection

Key-point detection aims to determine all the corners (i.e. key-points) in an image frame and can be achieved using the FAST or HCD algorithms. As discussed in [14], even though the HCD algorithm is more computational intensive than FAST, it calculates the corner response that yield more robust corners. As such, similar to the implementation in [14], we adopt the HCD algorithm for key-point detection.

The HCD algorithm determines corners based on gradient change in intensity which is approximated by matrix *M* within a small window *W* of each pixel *p(x,y)* as shown in (1).

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

where I_x and I_y are the horizontal and vertical gradients, and *w(x)* is the Gaussian weight function. The matrix *M* yields two eigenvalues: λ_1 and λ_2 which indicate the intensity change in a window *W* centered on the pixel *p(x,y)*. *p(x,y)* is a corner if its eigenvalues are high. HCD algorithm evaluates only the determinant and trace of matrix *M* to compute the corner measure *R* as shown in (2),

$$R = (ac - b^2) - k.(a + c)^2 \quad (2)$$

where k is determined empirically (usually in the range 0.04 to 0.06). The corner response is then subjected to a threshold *T* to identify strong corners. Finally, Non-Maximal Suppression (NMS) is applied to a *nxn* image patch to select the corners with the highest corner response.

B. rBRIEF Descriptor Computation

Rotated BRIEF (rBRIEF) [3] is a binary descriptor that overcomes the limitations of BRIEF [16] which suffers from rotation variance and scale changes. This is accomplished by first calculating the intensity centroid [17] of the image patch that is centered at a detected corner as follows:

$$\begin{aligned} m_{01} &= \sum_{x,y} y \times p(x,y), \\ m_{10} &= \sum_{x,y} x \times p(x,y), \end{aligned} \quad (3)$$

where p(x,y) is the intensity of pixel at position (x,y). The orientation is then performed on the two computed values m_{01} and m_{10} as follows:

$$\theta = atan2(m_{01}, m_{10}) \quad (4)$$

Next, the pre-determined point-pairs (samples) in the patch are multiplied with the rotation matrix [3] as shown in (5).

$$\begin{bmatrix} rx \\ ry \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (5)$$

The coordinates (rx, ry) of rotated samples are rounded to the nearest pixel. The final descriptor is an n_d dimensional bit-string where each bit is the result of point pair (s_i, s_j) comparison in (6).

$$\tau(p; s_i; s_j) := \begin{cases} 1 \ if \ p(s_i) < p(s_j) \\ 0 \ otherwise \end{cases} \quad (6)$$

We choose n_d to be 256 bits and the point pairs will be computed on a 37x37 image patch. This has been shown in previous studies to be sufficient for achieving high accuracy.

III. Overview of Proposed ORB Feature Extractor

Fig. 1 shows the overview of the proposed stream-based ORB feature extractor. Although the original ORB algorithm incorporates a multilevel pyramid to eliminate the scale variance, we will limit our discussion to a single level implementation in order to highlight our proposed strategies. The proposed architecture can be easily extended to multilevel by simply replicating the design in Fig. 1 for each pyramid level. In this section, we will describe the basic computation blocks of ORB feature extraction. The mechanisms for lowering dynamic power (denoted in red blocks in Fig. 1), will be discussed in the next section.

As can be observed from Fig. 1, the ORB extractor consists of row buffers, the Key-point Detector (*KD*) and rBRIEF Descriptor Computation (*DC*) modules. Under the assumption that the image is read sequentially using a raster scan mode at a rate of one pixel per clock cycle, the incoming pixels need to be cached locally using a set of row buffers. 36 row buffers are concatenated in the form of FIFO (First-In, First-Out) delay buffers to cache the incoming pixels. This is due to the fact that the rBRIEF descriptor will be computed on a 37x37 image patch that is centered at each detected key-point. The size of each row buffer is equivalent to the horizontal resolution of the image L (for example L = 640 for 640x480 image).

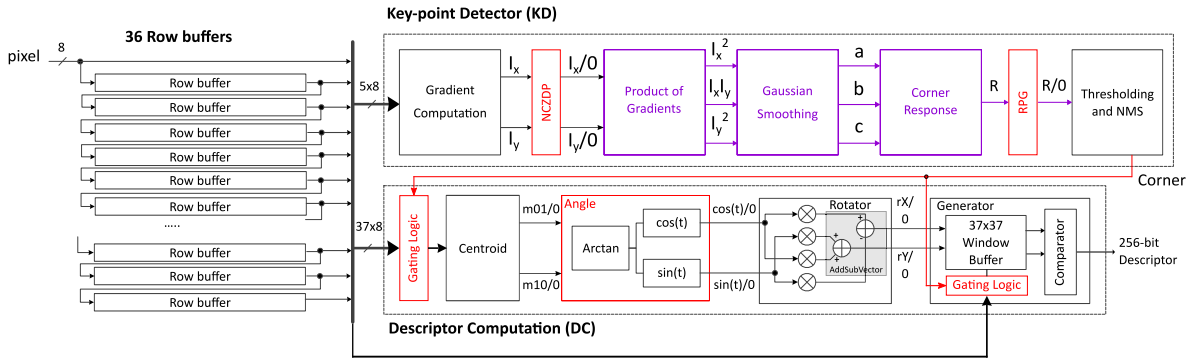

Fig. 1: Feature Extraction Architecture using ORB algorithm

A. *Key-point Detector (KD)*

The *KD* module is based on the HCD architecture in [18], and consists of five pipeline stages:

- *Gradient Computation*: The first pipeline stage computes the horizontal gradient I_x and vertical gradient I_y on a 5×3 neighbor of pixels which are streamed from the row buffers.
- *Product of Gradients*: The second pipeline stage is responsible for computing the product of gradients, i.e. I_x^2, I_y^2 and $I_x I_y$.
- *Gaussian Smoothing*: The Gaussian weight function $w(x)$ is applied to 3×3 product of gradients to produce a, b, c in parallel as shown in (1).
- *Corner Response*: This stage computes the corner response as shown in (2). The constant k is chosen to be 0.0625 similar to the implementation in [19] so that the calculation involves only shift operations.
- *Non-Maximal Suppression (NMS)*: The final pipeline stage first compares the corner response R with threshold T, and propagates the value of R if it is larger than T, otherwise it propagates 0. Next, pixels with the largest corner response in the 7x7 neighborhood of pixels are determined as corners.

B. *Descriptor Computation (DC)*

Fig. 1 shows the *DC* module which consists of four units to calculate the rBRIEF descriptor that was discussed in Subsection II.B:

- *Centroid*: This unit is responsible for calculating image moments: m_{01} and m_{10} as in (3). The details of the circuitry can be found in [14].
- *Angle*: This unit computes the $atan2$ value of m_{01} and m_{10}, and then finds $\cos\theta$ and $\sin\theta$. We have employed an approximation method based on angle discretization to achieve rotation invariance of the descriptors to avoid costly operations. Instead of calculating $atan2$ which is a costly operation, we approximate θ as θ_i that best meets the condition in (7).

$$m'_{01} \approx m'_{10} \times tan(\theta_i), \tag{7}$$

where θ_i is an angle which is nearest to θ in the set of the discretized values in $[0, 2\pi]$. To increase the accuracy of the approximation with a constraint of a small number of discretized values, the angle is transformed to the first quadrant $[0, \frac{\pi}{2}]$, therefore, the discretization is computed in $[0, \frac{\pi}{2}]$ instead of $[0, 2\pi]$. The *quadrant* which the θ belongs to is determined based on the sign of m_{01} and m_{10}. m'_{01} and m'_{10} are the absolute values of m_{01} and m_{10}, respectively. After ascertaining θ_i, the values of $\cos\theta$ and $\sin\theta$ could be determined according to the *quadrant* using Look-up Tables (LUT) instead of using the CORDIC engine [20], [21].

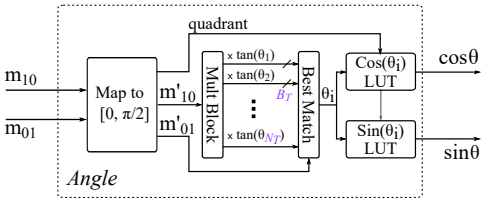

Fig. 2: *Angle* unit of *DC*

Fig. 2 shows the proposed architecture for calculating orientation. The accuracy of approximating the rotation angle in the *DC* depends on the number of discretized tangent values N_T and the number of bits B_T representing these values. By iteratively increasing the number of the discretized values and truncating bit-width to obtain an desired accuracy with a low hardware cost, N_T and B_T are found to be 25 and 11, respectively. The accuracy evaluation is presented in detail in the SubSection V-A.

- *Rotator*: The $\cos\theta$ and $\sin\theta$ values are used to perform the rotation on the samples as in (5). Two main steps are involved in this calculation. First, the X and Y coordinate vectors of 512 samples are multiplied with the $\cos\theta$ and $\sin\theta$ values. Second, the *AddSubVector* block performs addition and subtraction of the multiplication's results to generate the rotated coordinates.
- *Generator*: The last unit receives 512 rotated samples that are calculated from the previous unit to construct 256

descriptor bits. The 37×37 image patch must first be cached in a window buffer and then in each clock cycle, 2 samples are extracted for the binary test.

IV. Dynamic Power Optimization

In this section, we present the proposed mechanisms for lowering dynamic power of the ORB feature extractor described in the previous section.

A. Inhibiting Redundant Signal Activities in KD

The original HCD algorithm requires numerous addition and multiplication operators to calculate the corner response of all pixels. However, typically only a small number of pixels are identified as corners in an image frame. For example, our studies reveal that in a 640x480 image, the number of pixels that are identified as corners is usually only in the range of 300 to 700, which is approximately 0.2% of the total pixels. As such, we adopt the scheme that was introduced in [15] to detect the likelihood of a pixel being a non-corner in the early stages of the *KD* pipeline, and put the subsequent pipeline stages of *KD* in quiescent state to reduce the dynamic power. This is achieved by incorporating the Non-Corner Detection and Zero Propagation (*NCDZP*) and Reset Product of Gradients (*RPG*) units in Fig. 1.

Fig. 3 shows the detailed architecture of *KD*, where the two red dotted boxes indicate the *NCDZP* and *RPG* units. The *NCDZP* is accountable for computing $a'c'$ based on (8) and comparing this value with threshold t. A pixel is considered as a potential corner if its corresponding $a'c'$ exceeds a threshold which is empirically found to be $t = 0.05 \times max(a'c')$, where $max(a'c')$ is the maximum $a'c'$ value in the previous image frame [22]. If the $a'c'$ term of a pixel is smaller than the threshold, the pixel is considered an unlikely key-point and *NCDZP* will propagate zeros to the subsequent pipeline stages (Product of Gradients, Gaussian Smoothing and Corner Response) to place them in idle state. Otherwise, the gradient I_x and I_y will be passed to the subsequent stages (via multiplexer) for computing the corner response of the corresponding pixel. There may be instances where the gradients of an unlikely key-point are propagated to the subsequent pipeline stages (if their neighboring pixels are likely key-points), and hence the *RPG* unit is required to reset the corresponding values of I_x^2, I_y^2, and $I_x I_y$. Details of this scheme can be found in [15].

$$a' = \sum |I_x|, \qquad c' = \sum |I_y|, \tag{8}$$

B. Inhibiting Redundant Signal Activities in DC

We can extend a similar concept to inhibit redundant signals in the *DC* module for reducing dynamic power. In particular, the *DC* module shown in Fig. 1 only needs to compute the image moments (i.e. m_{01}, m_{10}) and orientation (i.e. $\cos\theta$, $\sin\theta$) on image patches that are centered at key-points. Since the number of corners (key-points) in a typical image frame is small, the pipeline stages in the *DC* module can also be kept idle most of the time to reduce the switching activity. To achieve this, when corners are not detected in the *KD* module, the *Gating Logic* units in the *DC* module of Fig. 1 are used to inhibit the 37 pixel column and 37x37 image patch inputs to the Centroid and the window buffer of the Generator units respectively. As soon as a key-point is detected, the *Gating Logic* units enable the pixel values from the row buffers to propagate to the Centroid and Generator units so that the rBRIEF descriptor can be computed.

TABLE I: Bit-width comparison before (Pre-Opt) and after optimization (Post-Opt)

Pipeline Stage	Bit-width	Pre-Opt	Post-Opt
Gradient Computation	G	11	6
Product of Gradients	$2G$	22	12
Gaussian Smoothing	$2G+5$	27	17
Corner Response	$2(2G+5)$	54	34

C. Threshold-Guided Bit-width Optimization in KD

The scheme described in Section IV.A to inhibit the redundant signals of the *KD* module can only achieve around 10% reduction in dynamic power [15]. In order to further reduce the dynamic power of the *KD* module, we propose a threshold-guided bit-width optimization strategy that does not lead to notable degradation in accuracy.

The operator bit-width in the *KD* architecture in Fig. 3 increase notably from one pipeline stage to the next due to the multiplication and addition operations. For example, if the bit-width of the input pixels is 8 (for gray scale images), the bit-width of the gradient values will be conservatively set to 11 (i.e. G = 11). Extending this, the Harris corner response R will be 54 bits, which is over six times larger than the bit-width of the input pixels. The work in [23] has demonstrated that truncating the bit-width of the data-path can lead to significant reduction in resources usage and power consumption without sacrificing much accuracy.

As such, we devise an approach to determine the optimal bit-width of the operations in *KD* with regard to the threshold value T used in the final pipeline stage of *KD*. The optimization aims to remove the least significant bits (LSBs) of both R and T such that the truncation does not affect a change in the thresholding decision. However, the truncation should be performed at the earlier stages, i.e. *Gradient Computation* to achieve the maximum efficiency because the bit-width of operations increase in a systematic fashion at each stage as shown in Table I.

To achieve this, we derived the error propagation equations (in Table II) due to bit-width truncation from the *Gradient Computation* stage to *Corner Response* stage using differential calculus [24]. Let G be the bit-width of the gradient values. The bit-width truncation is performed by iteratively decreasing G from 11 (i.e. original bit-width) until (9) is violated. This is because the error incurred by bit-width truncation does not

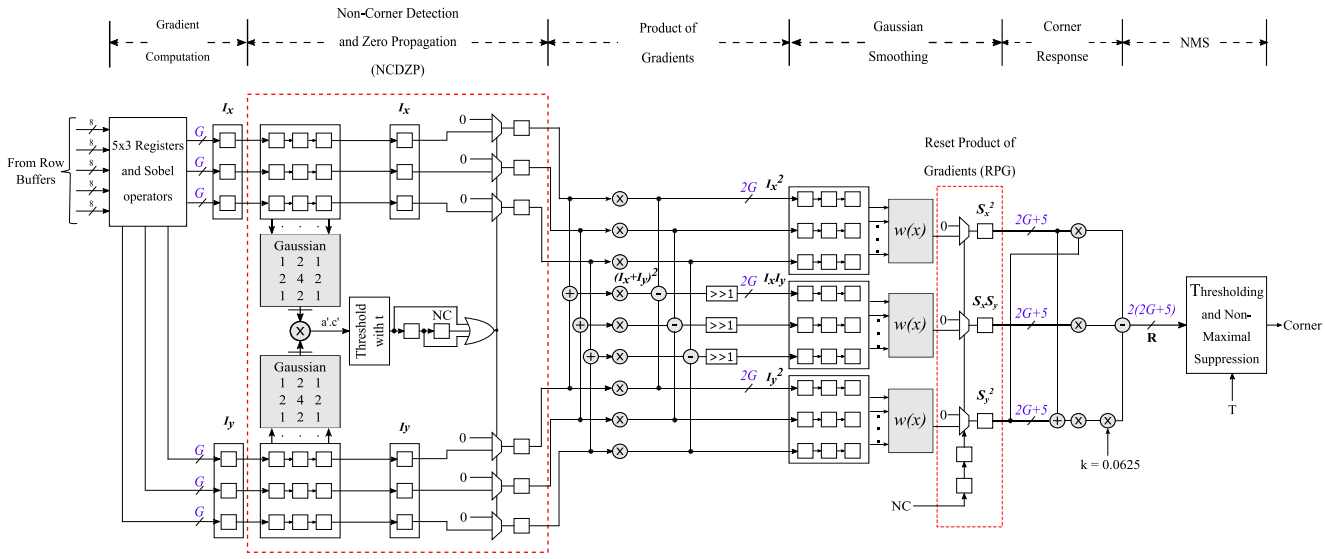

Fig. 3: Proposed *KD* architecture (each square box represents a register)

TABLE II: Error propagation at each pipeline stage of *KD* due to bit-width truncation

Pipeline Stage	**Error Propagation Equations**
Gradient Computation	$\Delta I_x = 2^{11-G} - 1$ $\Delta I_y = 2^{11-G} - 1$
Product of Gradients	$\Delta I_x^2 = 2.\|I_x\|.(\Delta I_x)$ $\Delta I_y^2 = 2.\|I_y\|.(\Delta I_y)$ $\Delta I_x I_y = \sqrt{(I_y.\Delta I_x)^2 + (I_x.\Delta I_y)^2}$
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	$\Delta R = \sqrt{(\Delta R_1)^2 + (\Delta R_2)^2 + (\Delta R_3)^2}$ $\Delta R_1 = \sqrt{(c.\Delta a)^2 + (a.\Delta c)^2}$ $\Delta R_2 = 2.\|b\|.\Delta b$ $\Delta R_3 = (a+c).(\frac{\sqrt{(\Delta a)^2+(\Delta c)^2}}{8})$

affect the thresholding decision if the *maximum output error* ΔR with respect to G does not exceed the threshold T. At the end of the iterations, G is determined as the minimal bit-width at the *Gradient Computation* stage which results in marginal error. The proposed methodology is applied to the *KD* architecture for a set of image datasets, and the optimal value of G is found to be 6. The evaluation of accuracy degradation of the proposed bit-width optimization is presented in Section V-A.

$$max(\Delta R(G)) < T \tag{9}$$

By combining our proposed bit-width truncation method with the method described in Section IV.A to inhibit redundant signal activities for the *KD* module, we achieved a significant dynamic power reduction of over 40% for the *KD* module.

V. Results and Discussion

In this section, we first discuss the impact of the proposed strategies for lowering power on accuracy, and then provide the hardware implementation gains in terms of power and resource utilization. We will also compare the results of our ORB feature extractor with a recently reported implementation (i.e. [14]). The implementation in [14] is denoted as *Existing*. In order to perform fair comparisons, we have ensured that the proposed designs and *Existing* employs similar design specifications (i.e. 37x37 image patch, clock operating frequency, single-level implementation, etc.).

A. Accuracy Evaluation

Since the descriptors are generated from image patches centered at the detected key-points, the accuracy evaluation needs to be performed at two levels: firstly the accuracy of the detected key-points (i.e. KD) and secondly, the accuracy of the descriptors.

We will first evaluate the impact of the proposed power optimization strategies on the accuracy of KD (i.e. strategies discussed in Section IV.A and Section IV.C). Our original KD implementation without power optimizations (discussed in Section III) is denoted as *Prop1-KD*. To compare the accuracy of *Prop1-KD* and the proposed KD architecture with power optimization (*Prop2-KD*), we use the repeatability criterion [25]. The principle behind the repeatability criterion is that key-point detection should be robust under variety of changes in image conditions, i.e. rotation, scaling, and illumination. Hence, an accurate feature detector should be able to detect

Fig. 4: Image set. From Top left: Bikes, Leuven, Trees, UBC.

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). The difference in the repeatability of *Prop2-KD* and *Prop1-KD* is computed using (10).

$$\Delta r = \frac{|repeat_{Prop2-KD} - repeat_{Prop1-KD}|}{repeat_{Prop1-KD}}.100\% \quad (10)$$

We use four 640x480 image sets (Bikes, Leuven, Trees, UBC) from the image dataset in [26], which has various transformations such as blur, zoom, and illumination for accuracy evaluation. Fig. 4 shows the original images of each image set. Fig. 5 illustrates the difference in repeatability rate of *Prop1-KD* and *Prop2-KD*, where the x-axis is the truncated bit-width of gradient values (G = 11 to 5). The results show that the repeatability difference increases with larger bit-width truncation which implies higher accuracy degradation. When 6 bits of the gradients are truncated, the loss of accuracy of the proposed architecture is marginal (less than 6%) for all datasets, but the gains in terms of power and resource utilization reduction are significant (as shown in the next sub-section).

Next, we compare the overall accuracy of the proposed ORB feature extractor (with power optimizations on *KD* and *DC* as discussed in Section IV) with an existing design [14] using the Hamming distance [27] of the descriptor bits. We denote the proposed ORB feature extractor as *Prop2*. The Hamming distance metric defines the number of different symbols in each position between two equal-length strings. The descriptor accuracy is determined by computing the Hamming distance of the 256-bit descriptor vectors of each implementation (*Existing* and *Prop2*) with the software ORB implementation that uses double precision floating points. Fig. 6 shows the average Hamming distance of *Prop2* and *Existing*. It is evident that our design results in fewer error bits than *Existing*. Particularly, the

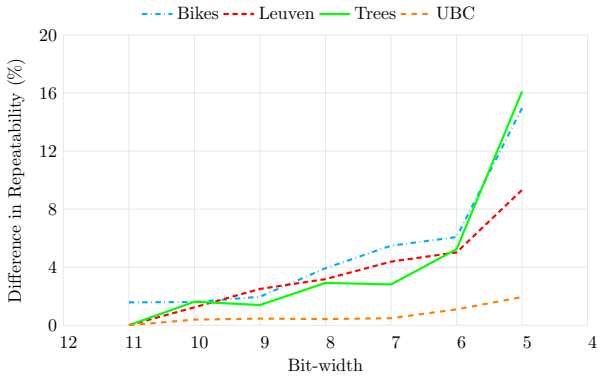

Fig. 5: The difference in repeatability rate of *Prop1-KD* and *Prop2-KD*.

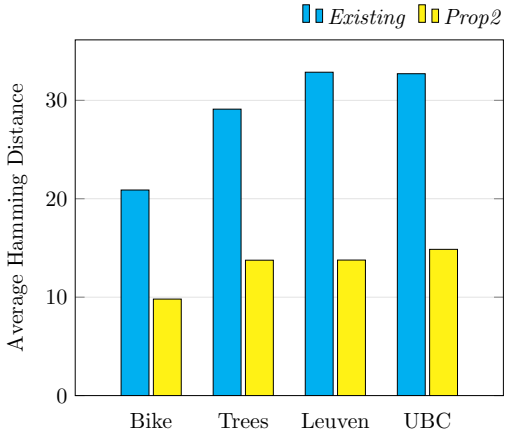

Fig. 6: Hamming distances with respect to the implementation using double precision.

proposed ORB feature descriptor achieves about 50% reduced Hamming distance compared to *Existing* for all four datasets. One of the main reasons that our design resulted in higher accuracy is the rounding scheme that we have employed in the Angle unit for approximating the nearest index of the $\cos\theta$ and $\sin\theta$ LUT, and for determining the point-pairs for binary test. Unlike *Existing*, we employ rounding to the nearest integer, which produces more accurate results.

B. Power and Area Evaluations

In the previous sub-section, we have shown that the proposed power optimization strategies incur marginal loss in accuracy. Next, we will evaluate the hardware implementation gains in terms of power and resource utilization. We use Verilog to implement all the designs. Quartus II Version 15.1.0 is used to synthesize the design which target the Altera Aria V (5AGXFB3H4F35C4) FPGA. We denote the base hardware implementation of our ORB feature extractor without power optimization (described in Section III) as *Prop1*, and the implementation with power optimizations (described in Section IV) as *Prop2*.

TABLE III: Number of detected non-likely corners in *NCDZP* stage and final detected corners

Dataset	Unlikely Keypoints	Final Keypoints	
		Prop1-KD.	Prop2-KD.
Bikes	290181	672	668
Leuven	282655	477	478
Trees	210064	623	624
UBC	270981	798	798

Table III reports the number of non-likely corners detected at the *NCDZP* stage, and the final corners detected by *Prop2*. It is evident that a large number of pixels (average 78.27% of image frame) are typically detected as non-corners in the *NCDZP* stage. This shows that most of the time, the *Product of Gradients*, *Gaussian Smoothing*, and *Corner Response* pipeline stages in *KD* are kept in 'idle' state, leading to reduce dynamic power consumption. In addition, the final number of corners detected in the *KD* module is also usually very small (average 0.21% of image frame). This implies that the *DC* module in *Prop2* is also kept 'idle' most of the time.

We first investigate the impact of the proposed strategies on the resource utilization and dynamic power of the *KD* module. Table IV shows the maximum frequency, area utilization and percentage difference in resource utilization between the *KD* modules of *Prop1-KD* and seven *Prop2-KD* implementations. The number indicated after *Prop2-KD* indicates the truncated bit-width of the gradient *G*. As can be observed, the maximum frequency of both architectures varies between 173 MHz and 179 Mhz. In terms of resource utilization, *Prop2-KD-11* has 21% higher Adaptive Logic Module (ALM) than *Prop1-KD* due to the additional resources introduced (i.e. *NCZDP* and *RPG*). However, when 6 bits of the gradient are truncated (*Prop2-KD-06*), the resource utilization reduces to only 80% of *Prop1-KD*. As discussed in the previous sub-section, the accuracy loss due to this bit-width trunctation is marginal (less than 6% as shown in Fig. 5).

To obtain accurate power analysis, we perform timing simulation using ModelSim-Altera 10.3d to obtain the actual switching activity statistics of the architecture based on the original images in the four image sets. The switching activity statistics are then used for power analysis in Altera PowerPlay. For a fair comparison, we employed the same clock operating frequency of 170 MHz for both *Prop1-KD* and *Prop2-KD*.

TABLE IV: Maximum clock frequency and resource utilization of *KD*

Method	Maximum Frequency (MHz)	Area (ALM)	Relative Utilization
Prop1-KD	173.55	1257	100%
Prop2-KD-11	179.02	1524	121%
Prop2-KD-10	174.64	1405	111%
Prop2-KD-09	174.86	1332	106%
Prop2-KD-08	177.30	1206	96%
Prop2-KD-07	173.76	1146	91%
Prop2-KD-06	178.35	1000	80%
Prop2-KD-05	176.71	891	71%

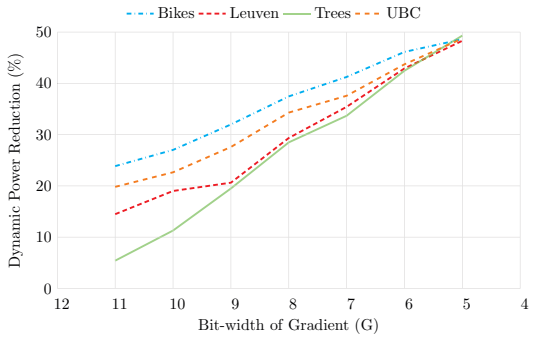

Fig. 7: Dynamic power reduction of *Prop2-KD* with varying bit-width of *G*

Fig. 7 shows the percentage of dynamic power reduction of *Prop2-KD* with respect to *Prop1-KD* when the bit-width of *G* varies from 5 to 11 in the four datasets. In can be observed that as soon as bit-width truncation is applied, the power reduction is significant, from 6% reduction in Trees to 25% reduction in Bikes. The percentage power reduction increases significantly when larger number of bits are truncated. The percentage power reduction converges to approximately 48% when *G* is 5 bits for all images. This shows the effectiveness of the proposed strategies for inhibiting redundant signal activities and bit-width truncation. As 6 bit truncation of *G* leads to over 40% dynamic power and 20% hardware resource reduction with marginal accuracy loss, we choose this parameter in the final implementation of *Prop2*. Table V reports the dynamic power of *DC* for *Prop1* (*Prop1-DC*) and *Prop2* (*Prop2-DC*) for the four image datasets. It can be observed that *Prop2-DC* achieves an average power reduction of approximately 20% due to the Gating Logic inserted to keep the *DC* pipeline stages in idle state when no corners are detected.

Finally, we compare the hardware resources and dynamic power of three designs (combining *KD* and *DC*): *Existing*, *Prop1*, and *Prop2* with 6 bit truncation of the gradient values. All three designs have the same clock operating frequency of 150 MHz (based on [14]). Table VI reports the results of the

TABLE V: Dynamic power consumption (mW) of *DC*

Method	Prop1-DC	Prop2-DC	Reduction (%)
UBC	261.26	208.43	20.22
Bikes	254.58	200.94	21.07
Trees	269.86	219.42	18.69
Leuven	259.07	207.25	20.00

TABLE VI: Resource utilization and dynamic power comparison

Design	Memory bits	Logic Cells	DSPs	Dynamic Power
Existing	953328	34650	92	593.33 mW
Prop1	488740	15868	12	354.46 mW
Prop2	411931	15320	13	265.04 mW

three designs. The reported dynamic power is based on the average power obtained for the 4 datasets. It is evident that *Prop2* not only achieves about 25% power reduction but also consumes less hardware resources than *Prop1*. As discussed earlier, the accuracy degradation of *Prop2* is less than 6% compared to *Prop1*. In addition, *Prop2* exhibits significantly lower resource utilization and dynamic power consumption compared to [14] [1].

VI. Conclusions

This paper presents a power-efficient architecture for ORB feature extractor which is capable of processing a pixel stream at high throughput without the need of any image frame buffers. To reduce hardware complexity, we proposed approximate computation strategies to determine the orientation angle of the patch. In order to reduce dynamic power, we devised methods to lower the switching activity of the key-point detection and descriptor computation stages by early pruning of non-likely key-points and gating the descriptor computation stages. A threshold-guided bit-width optimization strategy is employed to truncate the redundant bits in the key-point detection stage. FPGA implementation results show that the proposed strategies can achieve over 25% dynamic power reduction and fewer hardware resources with minimal accuracy loss (less than 6%).

Acknowledgment

This research project is partially funded by the National Research Foundation Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

References

[1] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov 2004.

[2] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *European Conference on Computer Vision (ECCV)*. Springer Berlin Heidelberg, 2006, pp. 404–417.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *International Conference on Computer Vision (ICCV)*, Nov 2011, pp. 2564–2571.

[4] J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative Evaluation of Binary Features," in *European Conference on Computer Vision (ECCV)*. Springer Berlin Heidelberg, 2012, pp. 759–773.

[1]The reported results in [14] are based on a nine-level ORB feature extractor (same single level design is replicated nine times), and hence we have approximated their results for a single level implementation by dividing the resource utilization and dynamic power accordingly.

[5] J. Hartmann, J. H. Klssendorff, and E. Maehle, "A comparison of feature descriptors for visual SLAM," in *European Conference on Mobile Robots*, Sept 2013, pp. 56–61.

[6] D. Mukherjee, Q. M. JonathanWu, and G. Wang, "A comparative experimental study of image feature detectors and descriptors," *Machine Vision and Applications*, vol. 26, no. 4, pp. 443–466, May 2015.

[7] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[8] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *International Conference on Computer Vision (ICCV)*, vol. 2, Oct 2005, pp. 1508–1515.

[9] C. Harris and M. Stephens, "A combined corner and edge detector," in *Fourth Alvey Vision Conference*, 01 1988, pp. 147–151.

[10] K.-y. Lee, "A Design of an Optimized ORB Accelerator for Real-Time Feature Detection," *International Journal of Control and Automation*, vol. 7, pp. 213–218, 03 2014.

[11] R. Sun, P. Liu, J. Wang, and Z. Zhou, "A low latency feature extraction accelerator with reduced internal memory," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[12] W. Fang, Y. Zhang, B. Yu, and S. Liu, "FPGA-based ORB feature extraction for real-time visual SLAM," in *International Conference on Field Programmable Technology (ICFPT)*, Dec 2017, pp. 275–278.

[13] O. Ulusel, C. Picardo, C. B. Harris, S. Reda, and R. I. Bahar, "Hardware acceleration of feature detection and description algorithms on low-power embedded platforms," in *International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2016, pp. 1–9.

[14] J. Weberruss, L. Kleeman, D. Boland, and T. Drummond, "FPGA acceleration of multilevel ORB feature extraction for computer vision," in *International Conference on Field Programmable Logic and Applications (FPL)*, Sept 2017, pp. 1–8.

[15] S. K. Lam, R. K. Bijarniya, and M. Wu, "Lowering dynamic power in stream-based Harris corner detection architecture," in *International Conference on Field Programmable Technology (ICFPT)*, Dec 2017, pp. 176–182.

[16] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *European Conference on Computer Vision (ECCV)*. Springer Berlin Heidelberg, 2010, pp. 778–792.

[17] P. L. Rosin, "Measuring corner properties," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 291 – 307, 1999.

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

[19] M. F. Aydogdu, M. F. Demirci, and C. Kasnakoglu, "Pipelining Harris corner detection with a tiny FPGA for a mobile robot," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2013, pp. 2177–2184.

[20] S. Suchitra, S. K. Lam, and T. Srikanthan, "Novel schemes for high-throughput image rotation," in *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, vol. 2, Nov 2004, pp. 1884–1888 Vol.2.

[21] S. Suchitra, S. k. Lam, C. T. Clarke, and T. Srikanthan, "Accelerating rotation of high-resolution images," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 153, no. 6, pp. 815–824, Dec 2006.

[22] M. Wu, N. Ramakrishnan, S. K. Lam, and T. Srikanthan, "Low-complexity pruning for accelerating corner detection," in *2012 IEEE International Symposium on Circuits and Systems*, May 2012, pp. 1684–1687.

[23] T. H. Pham, S. A. Fahmy, and I. V. McLoughlin, "Low-Power Correlation for IEEE 802.16 OFDM Synchronization on FPGA," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 8, pp. 1549–1553, Aug 2013.

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

[25] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of Interest Point Detectors," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 151–172, Jun 2000.

[26] Affine covariant features. [Online]. Available: http://www.robots.ox.ac.uk/˜vgg/data/data-aff.html

[27] R. W. Hamming, "Error Detecting and Error Correcting Codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, April 1950.