A Low Latency Feature Extraction Accelerator with Reduced Internal Memory

Rongdi Sun*, Peilin Liu, Jun Wang and Zunquan Zhou
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
*Email: the.sun@sjtu.edu.cn

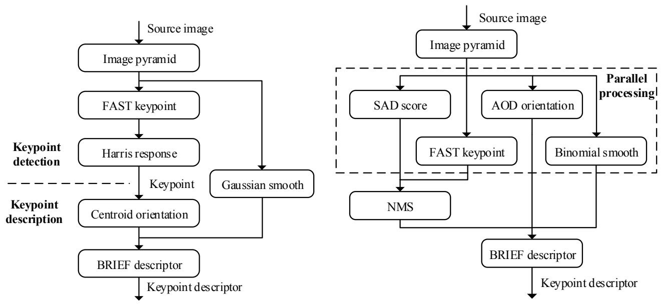
***Abstract*—ORB (Oriented FAST and Rotated BRIEF) feature extraction is popular in embedded vision applications like visual navigation due to its higher speed and robustness in many situations. However, feature description in ORB still accesses large amounts of image patches especially when an image pyramid is built. In order to reduce internal memory cost as well as maintain low latency processing, we design a hybrid pipeline architecture for ORB feature extraction. The accelerator combines different levels of computing granularity and migrates image pyramids to external memory. In addition, a data reuse scheme is adopted in descriptor generation to minimize external memory access, and achieve the ability to operate in multiple scales. The synthesis result shows 700kb internal memory cost and 24.5mW low power consumption. Experiments demonstrate 22% bandwidth reduction on average by the data reuse scheme. The system is verified on an FPGA platform and can provide 4000 features per frame, achieving up to 81fps in 1080p resolution at 100MHz frequency.**

Keywords—feature extraction; hardware accelerator; ORB

I. Introduction

Feature or keypoint extraction is one of the most fundamental steps in a set of visual applications, such as object recognition, 3D reconstruction and simultaneous localization and mapping (SLAM). Among various algorithms, Scale Invariant Feature Transform (SIFT) [1], which is invariant to scaling, rotation, illumination and noise, can provide robust features with the best matching accuracy. However, it is only suitable for high performance processing systems because of massive computations. As the demand for implementing such computing-intensive task in mobile devices and embedded platforms, many alternatives like Speeded Up Robust Features (SURF) [2] are proposed to reduce algorithm complexity while keep robustness to a certain extent. Recently, a computing efficient feature extraction method, Oriented FAST and Rotated BRIEF (ORB) [3], attracts a lot of interest and has been successfully used in visual SLAM applications [4].

On the other hand, many hardware architectures are proposed to perform these feature extraction algorithms in real time. FAST feature detection are implemented in [5] [6], but they are lack of non-max suppression (NMS). As a result, many detected keypoints are likely to huddle together and it is difficult to locate these keypoints accurately. Furthermore, some solutions combine FAST and BRIEF for feature detection and description [7] [8], yet they do not take scaling and rotation into consideration. To the best knowledge of us, the work in [9] is the first hardware solution based on the ORB method. However, storing image pyramids in the internal memory results in large resource overhead, and also limits the ability to compute descriptors in more scales.



(a) The primitive ORB algorithm (b) The modified ORB algorithm

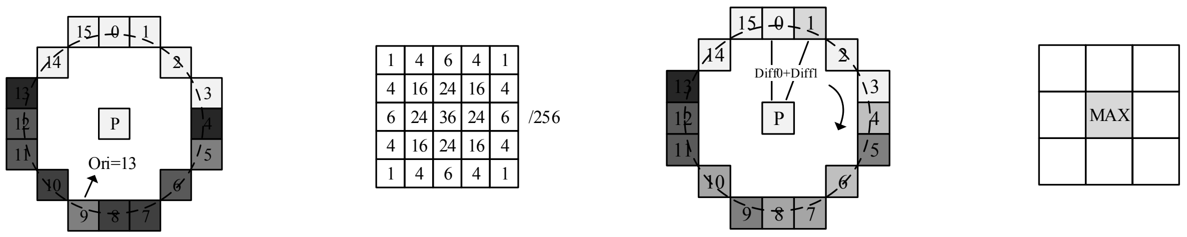
Fig. 1. The ORB algorithm flow and its optimization.

Motivated by the analysis above, this paper presents an ORB feature extraction accelerator. It accesses image sequence from sensors directly to remove the latency involved by frame buffering. Besides, it adopts pixel-level and task-level pipelines for keypoint detection and descriptor generation respectively. In addition, image pyramids are offloaded to external memory and a data reuse scheme is utilized when fetching local patches from pyrimads.

The rest of this paper is organized as follows. First, we review ORB algorithm and introduce several optimizations to reduce hardware complexity. Then the accelerator architecture and its features are presented in detail. After that the performance and overhead are evaluated through experiments.

II. ORB Algorithm and Optimizations

Image feature extraction is usually composed of two steps: keypoint detection and descriptor generation. The first step is to find pixels with local geometric information, called keypoints. In the second step, keypoints are represented by distinctive vectors called descriptors. For the ORB algorithm, seen in Fig. 1a, keypoint candidates are detected by using FAST method. It computes the intensity difference of the center pixel and those in a circular ring around the center. Then remove the candidates with lower Harris response, which is called non-max suppression (NMS). After that the orientations of keypoints are computed by their intensity centroid. Finally 256 pixel-pairs selected by a learning method within a smoothed local patch



(a) Orientation estimation based on the absolute value of difference

(b) 5×5 binomial coefficient matrix

(c) Pix scoring based on the sum of the absolute difference

(d) NMS with a tolerable distance 1

Fig. 2. The simplified computing patterns.

is compared to create a binary descriptor of each keypoint. In order to reduce the computing complexity and hardware cost, we reorder the process of the ORB algorithm. As shown in Fig. 1b, keypoint score and orientation are measured on each input pixel in parallel with FAST detection, cutting down on the processing time of these intermediate stages. Besides, we simplify several computing patterns in orientation estimation, image smoothing, pixel scoring and NMS.

A. Orientation Estimation

Centroid-based orientation measurement involves many complex operations such as multiplication, division and even trigonometric function. As shown in Fig. 2a, a keypoint has more than 9 darker or brighter continuous pixels on its ring. In [9], the absolute value of difference between each pair of adjacent pixels on the ring is calculated, and the ID number with the max value is the orientation. We learn from this and define the ID number of the middle pixel on the arc as the orientation. Thus the orientation estimation can be easily integrated into FAST detection with few overhead and is more robust when the darker or brighter arc has kind of symmetrical distribution.

B. Image Smoothing

Ahead of computing keypoint descriptors, the source image is usually convolved with a Gaussian kernel. Unlike the SIFT algorithm, the Gaussian convolution in the ORB is only to filter noise. Therefore, we use a 5×5 binomial coefficient filter shown in Fig. 2b to approximate a Gaussian filter. Coincidentally, the multiplications and divisions in convolution can be replaced by addition and shift operations. For example, $M\times36=M\times32+M\times4=M\ll5+M\ll2$. Due to the symmetric distribution, the number of total operations can be further reduced by summing the symmetric pixels first.

C. Pixel Scoring

The primitive ORB algorithm computes keypoint scores with Harris response. However, Harris response requires plenty of gradient and multiplication operations when computing the auto-correlation function of a local image patch. Instead, we define the score as the sum of the absolute difference (SAD) between the circle pixels and the center pixel (Fig. 2c). This process is also part of the FAST detection where the same computing pattern is performed, thus it requires no more than an accumulator.

D. Non-max Suppression

Following the pixel scoring, a tolerant distance based NMS is adopted to remove over-dense keypoints, as seen in Fig. 2d. We make a rule that only if the score of a candidate is higher than that of its 8-connected neighbourhood, it will be regarded as a keypoint. This method can suppress keypoints with poor

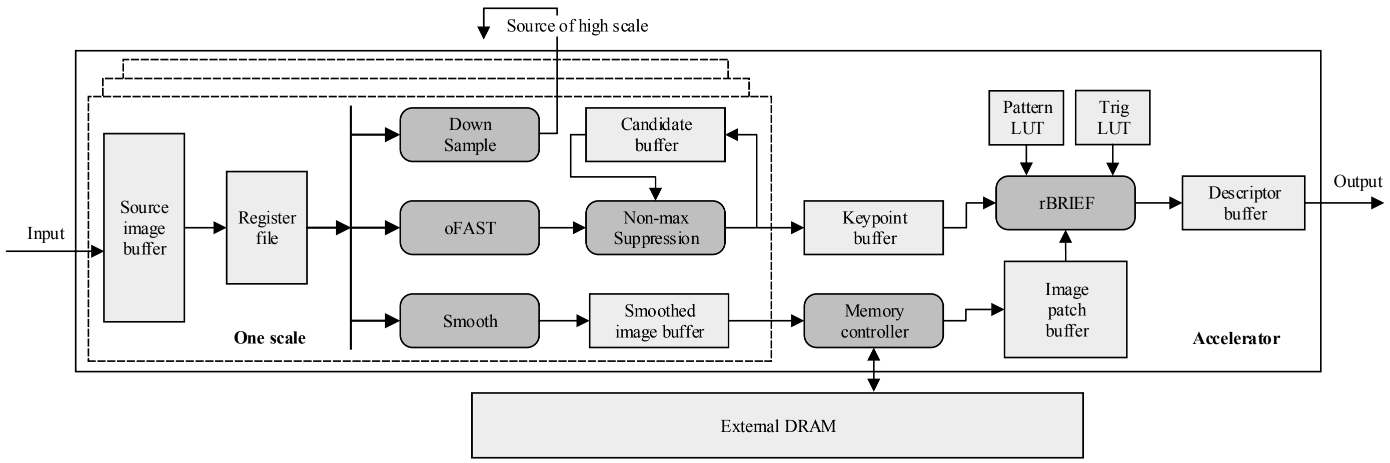


Fig. 3. The ORB accelerator architecture

quality, which actually have the same geometric imformation as their neighbours.

III. Hardware Architecture

The proposed accelerator is shown in Fig. 3. We set the same scale number (3) and scale factor (2/3) as [9]. The part in the dot line is duplicated to detect keypoints in multiple scales. The rest part computes descriptors of keypoints from all the scales. The source image is first written into a multi-bank buffer, where each bank stores a row of pixels. Then the pixels of each bank are written into a 2-D register file in parallel. When enough data are available in the register file, the following modules fetch data from registers with their own patterns. Image is down sampled by bilinear interpolation, and the result will be the source of the next scale. The source image is also smoothed by a binomial coefficient matrix and stored to the external DRAM. The oFAST module not only detects keypoint candidates but also computes their orientations and scores. Then the keypoint candidates are filtered by NMS, and written into the keypoint buffer. Based on the keypoints information, the rBRIEF module reads orientation values, trigonometric function values and pair-patterns from the Look-Up Tables. At the same time a local image patch is fetched from the DRAM. After that, it reads pixl-pairs from the local patch buffer according to the rotated patterns, and creates a binary descriptor. In the following we will present implementation details of keypoint detection, NMS and feature descripton.

A. String Searching based FAST Detection

As shown in Fig.4, the architecture of FAST-9 test consists of two unified data paths, one for bright test and the other for dark test. Take the dark test as an example, it contains 16 parallel comparators, and each of them compares the intensity of one circle pixel with that of the center pixel. If the center pixel is darker than the circle pixel by a threshold, then the compare result is ”1”, otherwise ”0”. The 16 results arranged in order form a test string, where a pointer *sptr* is attached. The segment test is performed within the higher 9 bits of this string, called ”test domain”. If ”0” occurs in the test domain, *sptr* will point to the rightmost ”0”, and the string is left cyclic shifted by *sptr*. Only if all bits in the test domain are ”1”, the center pixel is denoted as an interest point. It has been demonstrated in [7] that 4 cycles are required for non-keypoint judgement in the worst case. Thus, we design a 4-cycle pipeline with the same logic because image pixels are input one-by-one without

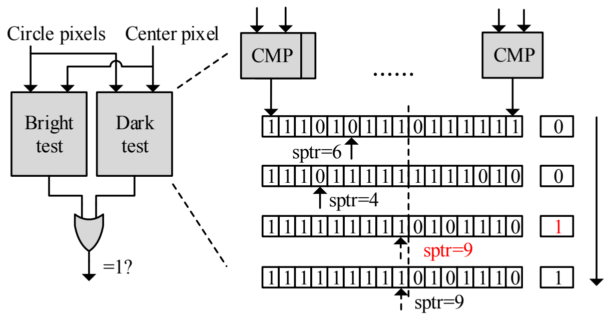


Fig. 4. The architecture of string searching based FAST-9 detection

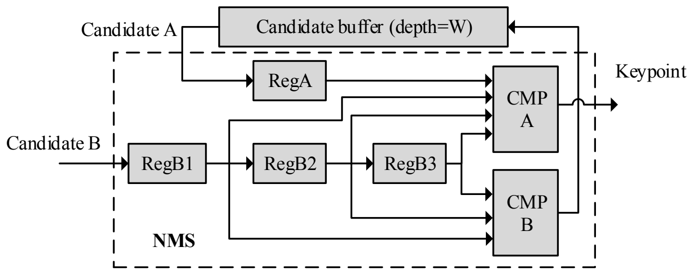


Fig. 5. The architecture of 3×3 NMS

extra buffers. During these 4 cycles, once the condition of keypoint judgement is met, the keypoint flag is set ”1”. As a result, we also remove the early-rejection scheme proposed by [7] due to the pipeline architecture.

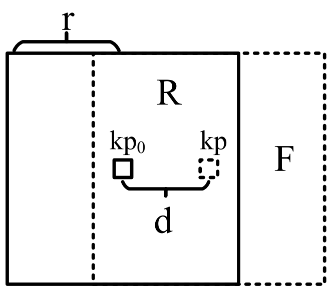
B. Optimized 3×3 NMS

The output of the oFAST module including both candidates and non-keypoints, they are collectively referred as ”Candidate B” in Fig .5. In the beginning, Candidate B is buffered into 3 registers in serial so that the data in RegB1, RegB2 and RegB3 are 3 adjacent pixels. Comparator B (CMPB) tests whether the pixel in RegB2 is a candidate point and has the highest score among these three pixels. If so, the pixel in RegB2 is stored with its keypoint flag in the candidate buffer, prepared for the second NMS. Otherwise, it is also stored in the buffer but the keypoint flag is set ”0”. This is the first NMS (NMS I). Then a pixel in the candidate buffer is fetched into RegA, at the same time the pixel in RegB2 is exactly the one under the pixel in RegA. CMPA performs the similar task among the pixels in all the 4 registers. Candidate A will be written into the keypoint buffer only if it passes the second NMS. In general, the first NMS selects the candidate with the max score from its adjacent pixels in the same row, while the second NMS further filters the winner of the first NMS by comparing it with the neighbouring pixels in the next row.

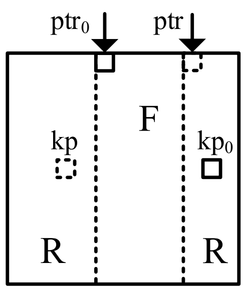
C. Descriptor Generation with Data Reuse

As the image pyramid is stored in external memory, descriptors are created in two steps. First, the boundary of the required local patch is calculated, and compared with the previous boundary to compute the region of fetched data from external memory. Second, when the required patch are written into the patch buffer, the rBRIEF algorithm is performed to generate the descriptor. Note that the patch buffer is organized as multi-bank SRAMs in order to match the transmission rate of the DRAM and increase the reading bandwidth for pixel-pairs.

Since keypoints are detected in row-wise, local patches required by two continuous keypoints in the same row may be overlapped. The key of data reuse is to compute the fetched region and the physical address when stored in the patch buffer. Fig .6a shows the relative positions of two keypoints when data reuse occurs. The distance d, between the last keypoint kp_0 and the current one kp, is less than the width of patch with a radius of r. Thus part of the local patch of kp_0, labeled by R can be reused for kp when computing its descriptor, and the rest part F with a size of $d \times (2r + 1)$ will be fetched from



(a) Patch region in a frame



(b) Region mapping in the patch buffer

Fig. 6. Data structure for local patch reuse.

the external memory. As shown in Fig.6b, two pointers ptr_0 and ptr are attached to the patch buffer, representing the start column address of kp_0 and kp respectively. Thus, the fetched data F are always stored on the right of ptr_0 and back to left if met with the boundary. As a result, ptr is computed by $ptr_0 + d$ or $ptr_0 + d - (2r + 1)$, and the physical address of pixel-pairs can also be figured out by taking ptr as a reference.

IV. Experimental Results

Our design is verified on an FPGA platform. Table I presents the effect of NMS and data reuse on performance where 5 images from [10] are used for test. The fourth column shows more than half candidate points are filtered by NMS with a 3×3 window, which indicates the important role of NMS. The last column shows the external bandwidth reduction by local patch reuse ranges between 17.90% and 27.65%, because the distribution of keypoints varies with different images.

The proposed accelerator is synthesized in SMIC 40nm CMOS technology with the area of 1.4×1.4mm^2. The internal memory is 700kb including 322.56kb for image source buffers, 92.16kb for Gaussian image buffers, 149.76kb for candidate point buffers, 52.224kb for keypoint buffers, 64.8kb for the local patch buffer and 18.432kb for the descriptor buffer. Table II shows hardware cost and performance of different works. Compared with previous designs [7] [9], the core size and power consumption of this work significantly decrease due to the use of external memory and lower operating frequency. It achieves 81fps in 1080p resolution at 100MHz operating frequency. Actually, the processing ability depends more on the frame rate of sensors and the depth of the keypoint buffer since the hybrid pipeline is employed. With the current configuration, the proposed accelerator is able to extract 4000 features in real-time without loss of keypoints.

TABLE I. The impact of NMS and local patch reuse schemes

Image	Size	Number of keypoints	keypoints decrease with NMS(%)	BW reduction with local patch reuse(%)
bikes	1000×700	2864	50.30	17.90
boat	850×680	4022	56.87	27.65
cars	921×614	2015	53.95	19.65
graf	800×640	3410	61.41	23.71
ubc	800×640	3885	54.10	24.10

TABLE II. Cost and performance comparison with related works

Work	Technology (nm)	Memory (kb)	Area (mm^2)	Power (mW)	Performance (1080p)
[7]	130	1024	3.2×3.2	182	94.3fps@200MHz
[9]	65	1640	2.4×2.4	87.5	135fps@200MHz
This work	40	700	1.4×1.4	24.5	81fps@100MHz

V. Conclusion

In this paper, we proposed an hardware-oriented ORB image feature extraction method that simplifies the operations in orientation estimation, image smoothing, keypoint scoring and non-max suppression. And we also design an accelerator to perform the proposed ORB algorithm. The pipeline architecture combines different levels of computational granularity to achieve low latency processing. Furthermore, external memory is used for image pyramid storage to improve power efficiency, and the internal memory size is less than 43% compared to the previous work [9]. A data reuse scheme is proposed for descriptor generation by utilizing the locality of keypoint distribution so that the average external memory access is reduced by 22%. The accelerator can operate up to 81fps in FHD resolution at 100MHz while consumes only 24.5mW power. And it is suitable to be integrated into SoC systems with a size less than 2mm^2;

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. IEEE, 2011, pp. 2564–2571.

[4] R. Mur-Artal, J. Montiel, and J. D. Tardós, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[5] M. Kraft, A. Schmidt, and A. J. Kasinski, "High-speed image feature detection using fpga implementation of fast algorithm." *VISAPP (1)*, vol. 8, pp. 174–9, 2008.

[6] K. Dohi, Y. Yorita, Y. Shibata, and K. Oguri, "Pattern compression of fast corner detection for efficient hardware implementation," in *2011 21st International Conference on Field Programmable Logic and Applications*. IEEE, 2011, pp. 478–481.

[7] J.-S. Park, H.-E. Kim, and L.-S. Kim, "A 182 mw 94.3 f/s in full hd pattern-matching based image recognition accelerator for an embedded vision system in 0.13-cmos technology," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 5, pp. 832–845, 2013.

[8] M. Fularz, M. Kraft, A. Schmidt, and A. Kasinski, "A high-performance fpga-based image feature detector and matcher based on the fast and brief algorithms," *International Journal of Advanced Robotic Systems*, vol. 12, 2015.

[9] W. Zhu, L. Liu, G. Jiang, S. Yin, and S. Wei, "A 135 fps 1080p 87.5 mw binary descriptor based image feature extraction accelerator," 2015.

[10] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A comparison of affine region detectors," *International journal of computer vision*, vol. 65, no. 1-2, pp. 43–72, 2005.