A High-Performance ORB Feature Extraction Accelerator for SLAM

Qingyu Chen, Yunfei Wang

***Abstract*— The detection and description of feature points are the foundation of algorithms in autonomous driving. In this paper, we propose a hardware ORB feature extraction system, accelerator, which has good acceleration effects for FAST (acceleration segment testing features) and rotation brief (binary robust independent basic features), and can achieve processing speeds of thousands of frames per second with low power consumption We achieved a processing speed of up to 1014.1 Mpix/s with only 4.462 watts based on KR260 evaluation. To make feature point extraction more uniform, we made minimal modifications to achieve block based feature extraction, allowing the CPU and FPGA to work together to implement octree filtering**

***Index Terms*—Accelerator, ORB, FPGA.**

I. Introduction

Simultaneous Localization and Mapping (SLAM)[1] is a pivotal algorithm for autonomous navigation and positional estimation, enabling devices to simultaneously determine their own location and reconstruct environmental maps without relying on external beacons like satellites. This capability is critical in environments where GPS signals are unavailable or unreliable, such as indoor spaces and urban canyons. Widely applied in robotics, autonomous vehicles, drones, and augmented reality (AR), SLAM continues to drive innovation across industries. As research advances, its role is expanding in smart devices, automated systems, and urban infrastructure development, promising broader societal impacts through enhanced automation and intelligent solutions.

SLAM systems are broadly categorized by sensor type, with laser-based and vision-based approaches being the most prominent. Laser SLAM[2], utilizing LiDAR, directly measures spatial relationships to generate high-precision maps and is considered a mature technology. However, its high cost and limited semantic data output hinder scalability[3]. In contrast, visual SLAM (vSLAM), which processes camera or multi-sensor inputs (e.g., inertial measurement units), offers cost efficiency, compact hardware, and adaptability. Compared with laser SLAM, visual SLAM requires lower sensor costs and simpler equipment, so it is favored by many scholars[4].

Visual odometry, a core component of vSLAM, estimates sensor motion through sequential image analysis. Feature-based methods, such as ORB-SLAM, extract and track distinctive keypoints (e.g., ORB[5] features) across frames, balancing computational efficiency with robustness against illumination changes and motion blur. Direct methods, which optimize pixel intensity gradients without feature extraction, are computationally lighter but sensitive to lighting variations[6].Given the computational intensity of SLAM algorithms, which predominantly lies in feature extraction, this study focuses on deploying ORB-SLAM on embedded FPGA platforms and optimizing its ORB feature extraction accelerator. Key innovations include:

(1) Dynamic Threshold Partitioning for Feature Extraction: A novel hardware architecture leveraging adaptive threshold segmentation to enhance feature detection efficiency while maintaining robustness.

(2) Resource-Efficient Optimization: Implementation of extensive quantization and pruning strategies, significantly reducing hardware resource consumption without compromising accuracy.

(3) High-Performance Deployment: A streamlined FPGA-based solution achieving real-time processing capabilities, balancing computational speed and power efficiency for embedded applications.

II. Related Work

Mateusz Wasala[7] employed a vectorized data format (4 pixels per clock) to efficiently reduce hardware resource usage, accelerating ORB feature extraction on an AMD Xilinx ZCU104 FPGA. This implementation achieved real-time processing of 4K UHD video streams (60 fps) with a system power consumption of only 5W, demonstrating superior energy efficiency and throughput compared to conventional embedded solutions. Weikang Fang et al.[8] proposed an FPGA-based ORB accelerator comprising an oFAST feature detector and a Steered BRIEF descriptor module. They designed a synchronous two-level pipelined buffer architecture to minimize latency and memory demands, while further reducing hardware resource consumption via bit-width optimization. Implemented on an Altera Stratix V FPGA, their design reduced latency by 51% and 41% compared to ARM Krait and Intel Core i5 CPUs, respectively, while increasing throughput by 103% and 68% and lowering power consumption by 9% and 83%.

Huang B C et al.[9] introduced a signature-based method for features, where only descriptors with identical signatures undergo Hamming distance calculations to accelerate matching. Their ORB extraction and matching hardware architecture, implemented on a Xilinx ZCU102 FPGA, achieved 193 fps for 1280×720 images and 314 fps for 640 ×480 images. Feature matching volume decreased by 69.63% to 85.7%, while maintaining over 85% accuracy. Qixing Zhang et al.[10] developed a flow-based ORB accelerator utilizing a column-caching strategy to enable

Manuscript received March 13, 2025

Qingyu Chen, School of computer science and technology, Tiangong University, Tianjin, China.

YunFei Wang, School of computer science and technology, Tiangong University, Tianjin, China.

non-blocking rBRIEF descriptor computation, significantly boosting throughput. Deployed on a Zynq UltraScale SoC, their design achieved an average latency of 1.4 ms (44% faster than state-of-the-art solutions) and a power consumption of 1.5W, making it ideal for low-power scenarios.

III. ARCHITECTURE

A. Overall Algorithm Overview

When processing images using the ORB algorithm on a CPU, the workflow begins by detecting candidate corners through the FAST method, which rapidly identifies keypoints by comparing pixel intensity variations. These raw corners are then assessed for quality using the Harris scoring metric to evaluate their stability and distinctiveness, ensuring only robust features proceed. To avoid redundancy, a non-maximum suppression (NMS) step removes overlapping or densely clustered corners, prioritizing spatially distributed keypoints. Further refinement discards those lacking sufficient surrounding pixels to support a complete 31×31 pixel contextual analysis window, ensuring reliable descriptor generation.

For surviving keypoints, rotation invariance is achieved by calculating their orientation via the intensity centroid method, which determines directional vectors based on local gradient distributions. Before generating descriptors, a Gaussian filter smooths the 31×31 pixel neighborhood around each keypoint, reducing noise sensitivity and enhancing feature consistency. Finally, rotation-aware rBRIEF descriptors are created by systematically sampling and binarizing intensity comparisons within the smoothed region, producing compact binary strings optimized for efficient matching.

In vSLAM systems, the distribution quality of feature points directly impacts the robustness and accuracy of pose estimation. Traditional ORB feature extraction algorithms select the top N feature points with the highest response values as keypoints. However, as illustrated in Figure 1(a), this global-ranking-based selection strategy exhibits a critical flaw: high-texture regions (e.g., building edges, intricate decorations) accumulate excessive feature points, while low-texture areas (e.g., plain walls, skies) suffer from sparse or missing features. Such non-uniform distribution not only increases feature redundancy but also risks feature matching failures in low-texture scenarios, potentially causing SLAM tracking loss. To address this limitation, ORB-SLAM incorporates a hierarchical filtering mechanism that integrates image pyramids, spatial uniformity constraints, and dynamic threshold adjustments, achieving adaptive optimization of feature point distribution. To solve this problem, ORB-SLAM uses block dynamic threshold to extract feature points and octree to filter feature points. The feature point extraction effect is shown in Figure 2.

Fig.1 TOP-k filtering

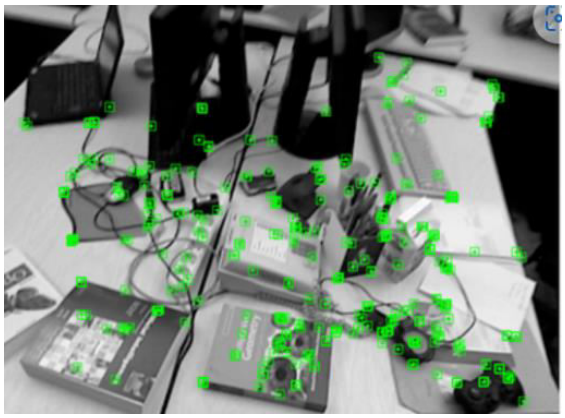

Fig.2 Octree filtering

B. Overall architecture

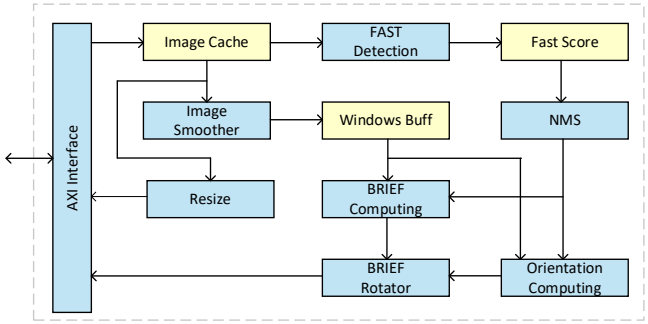

Fig.3 Overall hardware architecture

As shown in Figure 3, the hardware architecture implemented by our accelerator is also a common architecture for ORB feature extraction accelerators. Based on this, we have adjusted this architecture to support block based feature point extraction. Firstly, image preprocessing and rough keypoint extraction are completed: the original image is loaded into blocks through the AXI interface into the Image Cache, and then noise is suppressed by a Gaussian filter. The output smoothed image is temporarily stored in a dedicated cache area (Smoothed Image Cache). Based on this, the FAST-9 detection module traverses pixels in a circle (radius of 3 pixels), quickly identifies candidate keypoints through parallel differential comparison circuits, and combines the 3×3 sliding window filtering of the Non Maximum Suppression (NMS) module to preliminarily obtain a candidate point set with local maximum response.

Then focus on the rotation invariance compensation and flow calculation of descriptors. For each candidate keypoint, the direction calculation module first solves the main

direction angle θ based on the grayscale centroid method within a neighborhood of 15 pixels in radius; Subsequently, the BRIEF rotator utilizes a pre stored rotation lookup table (LUT) to perform θ angle rotation transformation on 256 pairs of sampling point coordinates, avoiding the overhead of real-time triangulation operations. Within the 31 × 31 pixel window after rotation correction, the BRIEF calculation module generates a 256 bit binary descriptor by comparing the results of point pairs, effectively suppressing noise interference and calculating the results. The final filtered result is returned after feature point filtering.

C. Block based dual threshold extraction

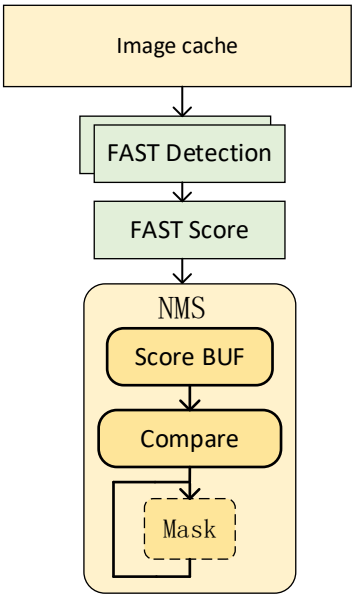

Fig.3 Implementation of the FAST module

The algorithm employs an image pyramid, where each pyramid layer is divided into 32×32 grid cells, and feature extraction is performed independently within each cell. If feature detection fails under the initial threshold, a lower threshold is applied. However, this poses a challenge in hardware implementation, as it typically requires two parallel FAST feature extraction blocks to simultaneously detect keypoints under both high and low thresholds. Notably, while FAST detection is threshold-dependent, the computed FAST score remains threshold-independent. Leveraging this property, the Non-Maximum Suppression (NMS) stage inherently incorporates results from the higher threshold when processing the lower-threshold-detected features, as the latter encompasses all candidates from the former. This design avoids the need for redundant computational units to handle dual thresholds, significantly minimizing hardware resource consumption without compromising detection integrity.

The design requires two FAST Detection units: one computes corners under the lower threshold, while the other operates at the default threshold, collectively generating mask flags. The FAST score is calculated for low-threshold corners, which are then fed into the NMS module. The NMS independently processes both thresholds to produce filtered results for high and low thresholds. Corners rejected by both thresholds are immediately discarded. For other cases, retention or rejection is deferred until the entire block is processed to determine the final outcome. As depicted in the FAST block architecture in Figure 3, additional FAST detection units are integrated. Since FAST Detection consumes minimal hardware resources, this modification incurs negligible overhead.

The system temporarily stores detected feature points in FIFO buffers to manage data flow during processing. Each feature point's coordinates are translated into a corresponding block identifier, which dynamically updates status records stored in RAM to track processing progress. A dedicated module monitors whether the entire block associated with a feature point has been fully analyzed. Once block processing concludes, the system queries the RAM to classify the block as either a default- or low-threshold region, determining whether the point is retained or discarded based on this classification. Configurable components, indicated by dashed lines in the architecture, enable adaptive customization of the pipeline to suit specific operational requirements. Notably, the original merge-sort-based prioritization mechanism is replaced with an ARM-optimized approach, eliminating resource-intensive sorting logic while preserving hardware efficiency. This streamlined design ensures spatially balanced feature distribution across varying texture environments, maintaining real-time performance without compromising accuracy or increasing hardware overhead.

D. Octree filtering

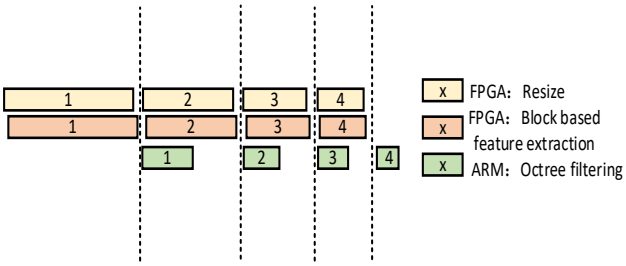

Fig.4 Octree filtering sequence diagram

This design establishes a hybrid ARM-FPGA co-processing architecture: 1) A scaling unit constructs Gaussian pyramids for multi-scale analysis; 2) The FPGA logic layer executes parallel block-wise feature extraction; 3) The ARM processor performs octree-based feature optimization. By pipelining stages 2 and 3, the computational latency of the octree optimization is effectively masked.

As illustrated in Figure 4, octree filtering for layer x and feature extraction for layer x+1 are executed concurrently, fully utilizing the ARM′ s idle cycles. The trade-off involves a final octree filtering step for the last layer. However, due to reduced image dimensions and fewer detected features in deeper pyramid layers, this step incurs minimal computational overhead. The proposed approach retains the spatial partitioning advantages of the octree algorithm while significantly reducing FPGA resource utilization and development complexity, leveraging ARM-FPGA parallelism for efficient implementation.

E. Resource Optimization

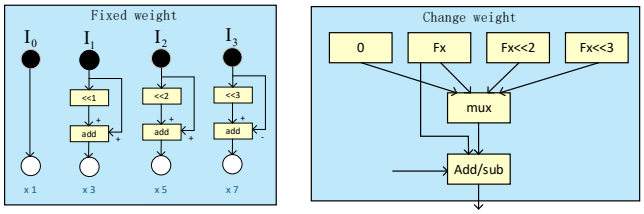

Fig.5 Multiplication Implementation

To minimize hardware resource consumption, we first apply quantization to specific computational steps, reducing numerical precision. The quantized values, typically smaller in magnitude, enable the replacement of multiplication operations with shift-add or shift-subtract logic. For fixed weights, direct shift-based arithmetic is employed, while dynamic scenarios involving floating-point weights utilize multiplexers to select preconfigured shift-add/subtract patterns. This approach simplifies computation logic and significantly reduces resource overhead without compromising functional integrity. The implementation is shown in Figure 5.

IV. Analysis Of Experimental Results

A. Experiment Settings

This study conducts a systematic evaluation of the ORB feature extraction accelerator design, focusing on computational efficiency and hardware resource utilization. The architecture was synthesized and implemented using Vivado 2023.2, and benchmarked on the AMD-Xilinx KR260 platform at a 200 MHz operational frequency. Performance validation utilized the TUM[11] public dataset to ensure reproducibility and comparative analysis.

B. Resource Utilization

Table 1 Comparison of Resource Utilization

Method	LUT	FF	DSP	BRAM
Ours	35,059	28,214	1	2.1 Mb
[12]	71,423	49,649	285	3.13 Mb
[13]	54,435	30,281	44	1.83 Mb
[14]	28,168	9,528	33	1.47 Mb
[15]	100,606	140,291	683	6.7 Mb
[16]	56,954	67,809	11	2.73 Mb

As shown in Table 1, the resources we used are compared with those used in other works We adopted quantization and used addition/subtraction instead of multiplication, so our DSP uses very little And our use of other resources is also minimal, with significant advantages in LUT and FF.

C. Performance Evaluation

Table 2 Acceleration effect

Method	Platform	power consumption	delay time
Ours	Xilinx KR260	4.462 w	0.6 ms
ORB	Intel i5	45 w	16 ms

Table 3 Accelerator performance comparison

Method	Platform	MPix/s	nLevels	result	FPS
Ours	Xilinx KR260	1014.1	4	640×480	1428
[12]	Xilinx Virtex-7	355.9	4	1920×1080	68.8
[13]	Xilinx Kirtex-7	138.2	1	1280×720	150
[14]	Xilinx Ultrascale+	485.1	108	1920×1080	108
[15]	Xilinx ZCU104	497.7	1	3840×2160	60
[16]	Xilinx XCZ7045	58.2	4	640×480	76

Table 2 shows our acceleration performance. Compared to desktop CPUs, our power consumption is only one tenth, and the latency time has been reduced from 16ms to 0.7ms, an increase of 22.8 times Compared to other ORB feature extraction accelerators, our FPS is the highest, partly due to the smaller size of the images we process Therefore, this study calculated the number of pixels processed per second by each accelerator, and we achieved first place with a performance of up to 1014.1Mpix/s

D. Accuracy Evaluation

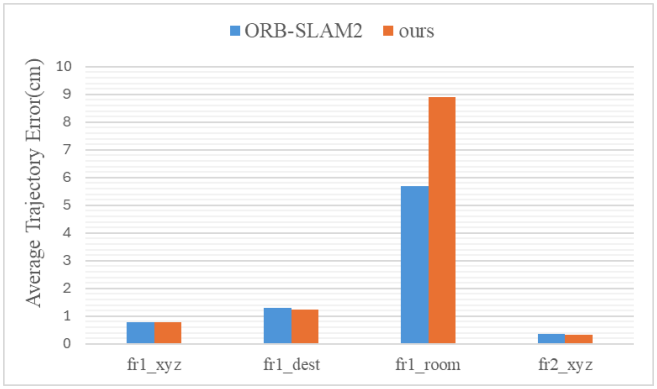

Fig.6 ATE comparison

We integrated the accelerator into the ORB-SLAM2 algorithm and evaluated it using the TUM dataset on fr1_xyz, fr1_dest, fr2.xyz, and fr1_room. The accuracy of visual SLAM systems is measured by trajectory error, which calculates the absolute trajectory error between the ground truth trajectory and the estimated trajectory. Compared with the original ORB SLAM2 algorithm, as shown in Figure 6, our work compares the average trajectory error on the four sequences of the TUM dataset with the original ORB based SLAM implementation. For the fr1/dest and fr2/xyz sequences, our ORB version has better accuracy than the original ORB implementation. However, the evaluation was slightly worse on the fr1/desk and fr2/room sequences.

V. Conclusion

This article designs an ORB feature extraction accelerator and innovatively implements block based feature extraction. The accuracy obtained in this article can be higher in some sequences By using quantitative resource conservation, the performance of 1014.1 Mpix/s was achieved with fewer resources, which is the best performance in this work, but it did not rely on stacking resources efficiency, and real-time performance.

REFERENCES

[1] Durrant-Whyte H, Bailey T. Simultaneous localization and mapping: part I[J]. IEEE robotics & automation magazine, 2006, 13(2): 99-110.
[2] Zeng F, Wang C, Ge S S. A survey on visual navigation for artificial agents with deep reinforcement learning[J]. IEEE Access, 2020, 8: 135426-135442.
[3] Choi S, Chae H W, Jeung Y, et al. Fast and versatile feature-based lidar odometry via efficient local quadratic surface approximation[J]. IEEE Robotics and Automation Letters, 2022, 8(2): 640-647.
[4] Huang L. Review on LiDAR-based SLAM techniques[C]//2021 International conference on signal processing and machine learning (CONF-SPML). IEEE, 2021: 163-168.
[5] Rublee E, Rabaud V, Konolige K, et al. ORB: An efficient alternative to SIFT or SURF[C]//2011 International conference on computer vision. Ieee, 2011: 2564-2571.
[6] Gomez-Ojeda R, Moreno F A, Zuniga-Noël D, et al. PL-SLAM: A stereo SLAM system through the combination of points and line segments[J]. IEEE Transactions on Robotics, 2019, 35(3): 734-746.
[7] Wasala M, Szolc H, Kryjak T. An efficient real-time FPGA-based ORB feature extraction for an UHD video stream for embedded visual SLAM[J]. Electronics, 2022, 11(14): 2259.
[8] Fang W, Zhang Y, Yu B, et al. FPGA-based ORB feature extraction for real-time visual SLAM[C]//2017 International Conference on Field Programmable Technology (ICFPT). IEEE, 2017: 275-278.
[9] Huang B C, Zhang Y J. A High-Efficiency FPGA-Based ORB Feature Matching System[J]. Journal of Circuits, Systems & Computers, 2024, 33(2).
[10] Zhang Q, Sun H, Deng Q, et al. NORB: A Stream-Based and Non-Blocking FPGA Accelerator for ORB Feature Extraction[C]//2023 30th IEEE International Conference on Electronics, Circuits and Systems (ICECS). IEEE, 2023: 1-4.
[11] Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems[C] IEEE, 2012: 573-580.
[12] Zhang Z, Chen H, Zhou L, et al. Bucket-FEM: A bucket-based architecture of real-time ORB feature extraction and matching for embedded SLAM applications[C]//2021 6th International Conference on Communication, Image and Signal Processing (CCISP). IEEE, 2021: 183-187.
[13] Xie Z, Wang Y, Yan Z, et al. A real-time FPGA-based architecture of improved ORB[C]//MIPPR 2019: Parallel Processing of Images and Optimization Techniques; and Medical Imaging. SPIE, 2020, 11431: 1143102.
[14] Sun R, Qian J, Jose R H, et al. A flexible and efficient real-time orb-based full-hd image feature extraction accelerator[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019, 28(2): 565-5
[15] Wasala M, Szolc H, Kryjak T. An efficient real-time FPGA-based ORB feature extraction for an UHD video stream for embedded visual SLAM[J]. Electronics, 2022, 11(14): 2259.
[16] Liu R, Yang J, Chen Y, et al. eslam: An energy-efficient accelerator for real-time orb-slam on fpga platform[C]//Proceedings of the 56th Annual Design Automation Conference 2019. 2019: 1-6.