A Hardware Architecture of Feature Extraction for Real-Time Visual SLAM

Jialin Li, Liangji Zhang, Xuewei Shen, Yifan Gong, Ying Lei, Chen Yang, Li Geng*
School of Microelectronics
Xi'an Jiaotong University
Xi'an, China
gengli@xjtu.edu.cn

IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society | 978-1-6654-8025-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/IECON49645.2022.9968745

***Abstract*—Feature extraction is one of the performance bottlenecks in embedded vision like visual navigation because of its high computational intensity. However, when applying it to simultaneous localization and mapping (SLAM), it is important to consider not only the time consumption but also the accuracy of the system. The more homogeneous the extracted features are distributed, the more accurately the feature matching can represent the geometric relationships in 3D space. In this paper, we propose a new hardware architecture of feature extraction with the addition of mask operation and a mini-grid pipeline to achieve homogeneous distribution. This work is implemented on a Xilinx Zynq SoC. Compared with the Intel i5 and ARM v8.2 CPU implementation of this work, our feature extractor achieves up to 15× and 35× acceleration, and up to 453× and 23× energy efficiency improvement. The evaluation results on the EuRoC dataset show accuracy comparable to the software implementation of the related work. Our architecture is hardware-friendly and achieves a good balance of accuracy and performance.**

Keywords—feature extraction, hardware accelerator, Visual SLAM, FPGA

I. Introduction

Feature extraction is a crucial module in visual simultaneous localization and mapping (SLAM) systems because the quality of feature point selected directly affects the accuracy of camera pose estimation and mapping. Common visual SLAM solutions for embedded system usually use feature points such as ORB [1], SIFT [2] and FAST [3] to extract, but these feature extraction methods are time and resource consuming. ORB feature extraction was applied in [4] on a quad-core ARM v8 mobile SoC that consumed more than 50% of the CPU resources. Feature extraction from a resolution 640 × 480 image takes about 50 ms, resulting in a frame rate of only about 20 FPS.

Due to the high computational intensity of feature point extraction, implementing the algorithm on a low-power platform is a challenge for image processing. There have been many previous efforts to design and accelerate feature point extraction. In 2015, Fularz et al. [5] proposed the first embedded system architecture for feature detection that facilitates the application of SLAM systems. Fang et al. [4] accelerated the SLAM pipeline to meet real-time requirement and well-balanced energy consumption and performance by optimizing the word length. [6] proposed a fully integrated visual inertial odometry accelerator that reduces the feature extraction time to 5.8ms. This approach uses tracking to match features between frames, avoiding the memory and computational overhead to compute, store, and match descriptors.

The authors thank the supports from the Key R&D Program of Shaanxi Province 2019ZDLGY03-07-01.

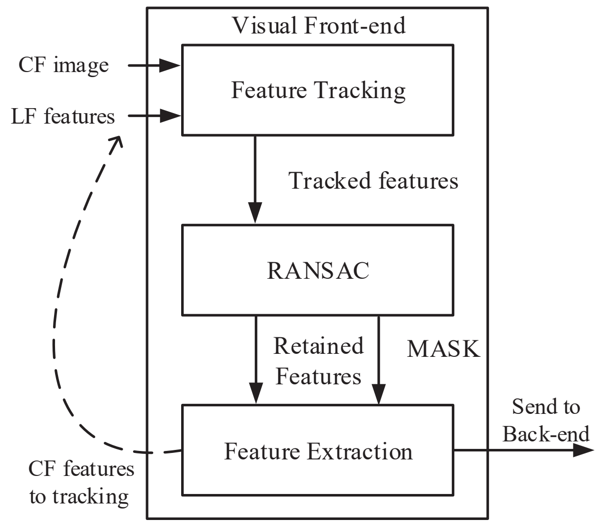


Fig. 1 The front end of visual SLAM system.

However, not only is it important to extract features quickly but accuracy is also crucial in SLAM. Features tend to be concentrated in texture-rich regions, while the number of features in regions lacking texture will be much less. Too concentrated distribution of features can affect the accuracy of the SLAM system. Therefore, feature extraction also needs to achieve homogeneous distribution [7].

To achieve this, in the software implementation, [7] added a quadtree structure to select feature points homogeneously, which solves this problem well. However, it also brings uncertain iterative computation, more complex logic branches and a large number of repeated memory reads, all of which are hardware-unfriendly factors that severely limit the application of the algorithm to small embedded platforms.

In the hardware implementation, filtering (or heap sorting) and non-maximum suppression (NMS) were applied by [8] and [9], which filters out high-quality features but this only ensures that the extracted feature points are not close together in a very small region. Suleiman et al. [6] detected feature points on a grid to keep the number of features in different regions constant. A total of 1824 feature points were detected per frame, up to 200 features can be added as needed. But to sort these features, this process also increases the unexpected computation time.

Therefore, this paper adopts the tracking-based SLAM system and proposes a new hardware-friendly architecture of feature extraction with the addition of mask operation and a mini-grid pipeline. This optimization could limit the number of considered features and reduce the sorting time. The aim is to solve the feature homogeneous distribution problem to improve accuracy while ensuring performance.

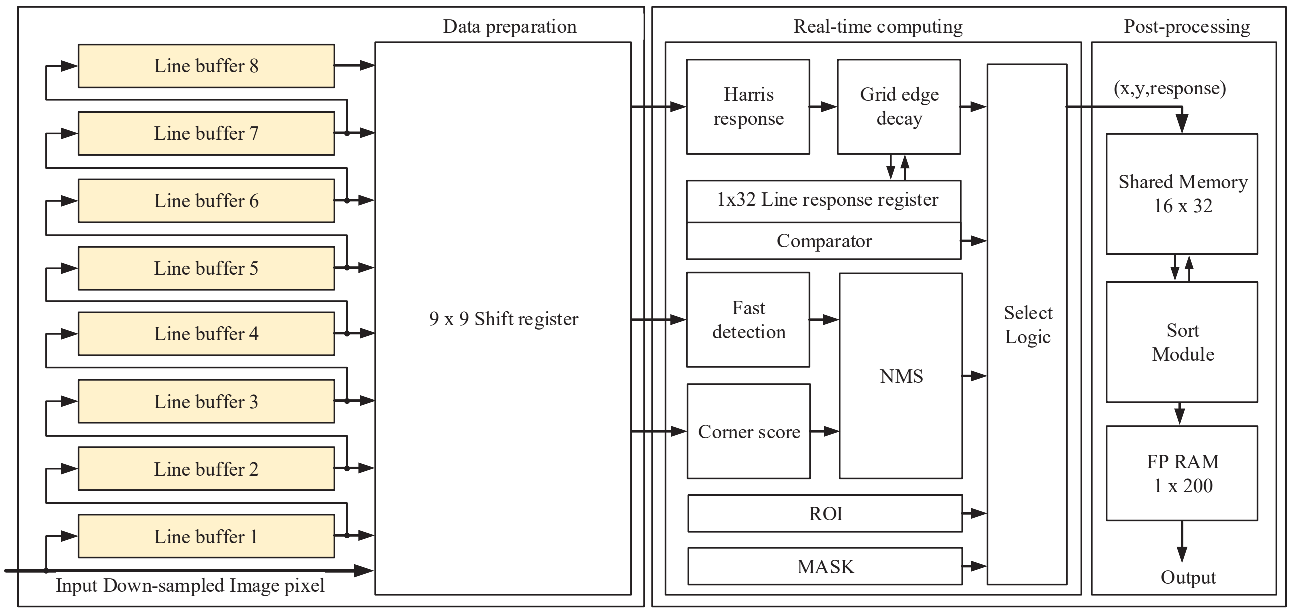


Fig. 2 Hardware architecture of feature extraction accelerator.

The rest of this paper is organized as follows. Section II gives an overview of the front end of the SLAM system. Section III describes the detailed hardware architecture of feature extraction. Section IV is the hardware implementation and the performance evaluation of the feature extractor. Finally, Section V concludes this paper.

II. System Overview

The visual front-end of a feature-based SLAM system shows in Fig. 1, which consists of feature tracking, RANSAC (Random Sample Consensus), and feature extraction. Tracking tracks the position of feature points in the last frame (LF) on the current frame (CF) by optical flow. Successfully tracked feature points are matched into point pairs, while untracked old points are discarded. RANSAC randomly selects three of these matched pairs to train the estimation model at a time. The model with the minimized error over multiple iterations is used to evaluate the pairing accuracy. It removes any incorrect outliers and also generates a mask to prohibit the appearance of new feature points near the successfully tracked ones. Feature extraction detects and selects a specific number of new feature points in the current frame based on the mask and the feature points' quality. It compensates for the feature points missed in the first two steps and sends the feature points to the back-end to optimize the camera pose.

The SLAM system in this study tracks 200 feature points per frame. Considering the generally low image resolution (752 × 480 or 640 × 480) used in SLAM systems at this stage and the characteristics of the input image data stream, the system also requires the feature extraction pipeline should preferably be able to process one pixel per clock cycle. Accordingly, a novel and practical feature point extractor proposed in this paper.

III. Hardware Architecture

The feature extraction circuit is generated by high-level synthesis (HLS). There is a large amount of parallelism in feature extraction, and proper optimization can significantly reduce the computation time. In this work, the algorithm is manually rewritten in a nested-loop programming style suitable for easy implementation by HLS. And the circuit is optimized by adding directives during the circuit synthesis process.

Fig. 2 shows the hardware architecture of the feature extractor. Generally, the feature extractor processes the input pixel stream of the down-sampled image and finally outputs no more than 200 filtered high-quality feature points. The architecture consists of the data preparation module, real-time computing module, and post-processing module , where both the data preparation module and the real-time computation module are pipelined for the input signal to meet the real-time requirements of the system. The buffers, registers, and shared memory are used to store intermediate results.

A. Feature detection

Before extracting feature points, it is necessary to preprocess the raw data since only 200 feature points need to be output eventually. Direct processing of the raw image captured by the sensor requires more on-chip resources and is inefficient. Therefore, in this paper, the 24-bit RGB pixels of the raw image are converted to 8-bit grayscale pixels. And a down-sampling operation with a scale of 0.5 is carried out. The amount of image data to be processed is reduced to 1/12.

Then, the data preparation module reads a pixel from the down-sampled data stream in sequence, combined with the pixels in the first eight rows of the buffer cache. They form a sliding window of images in a 9 × 9 neighborhood near the center point stored in a shift register matrix composed of 8-bit registers. The sliding window is shown in Fig. 3.

To maximize the performance of the FPGA, the Harris response, fast detection and corner score calculations are executed simultaneously in the real-time computing module. Fig. 3 illustrates the use of pixel data in the sliding window by each part of the algorithm. The Harris response computation requires convolution of a 7 × 7 window using the Sobel operator, which requires a total of 9 × 9 neighborhoods. According to the FAST-9 algorithm, only the pixels on the 7 × 7 circular patch need to be processed. However, the non-

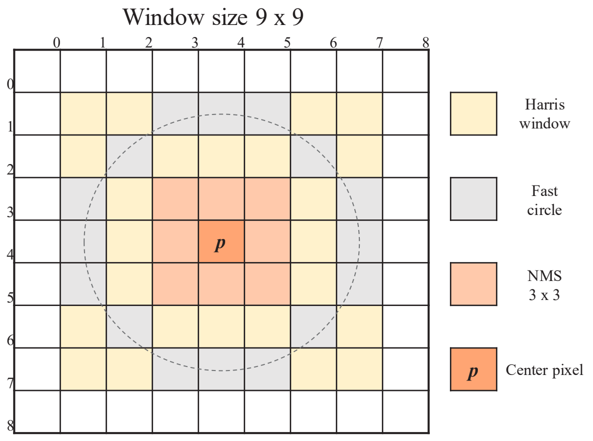


Fig. 3 9 x 9 Shift-register bank and data usage.

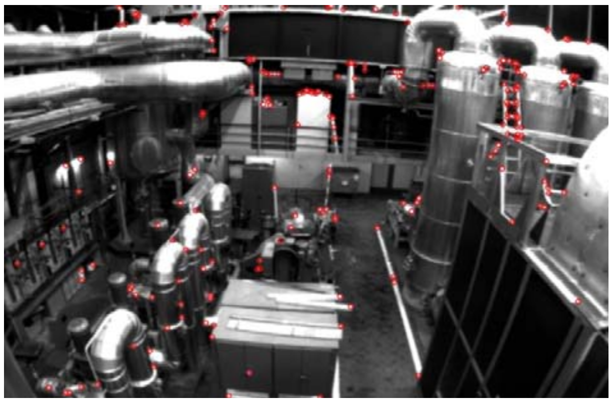


Fig. 4 Example of feature detection results.

maximum suppression (NMS) operation needs to compare the scores of 3 × 3 neighbors, so it also needs a window of 9 × 9. Considering the above factors, this paper decides to determine the size of the sliding window as 9 × 9 to store the necessary data needed for one calculation.

In the hardware implementation of the FAST-9 algorithm, a three-loop parallel unrolling and three-stage pipeline design is used. Three pixels in a row are processed simultaneously, and all nine pixels are processed in three times. Firstly, the constant matrix of the fast detector is constructed based on the threshold value t and stored in ROM. All cases are classified into three categories. Then, the 16 pixels on the circular patch are divided diagonally into eight pairs to quickly recognize the three patterns of brighter, darker and similar [10]. The classification criteria are as follows.

$$tab_x = \begin{cases} 2'b01, V_x \le V_p - t \\ 2'b00, V_p - t \le V_x \le V_p + t \\ 2'b10, V_p + t \le V_x \end{cases} \tag{1}$$

where V_p is the value of center pixel, V_x is the value of the pixel on the circular patch.

The remaining loops of FAST are unrolled to judge whether there are 9 continuous pixels brighter or darker than the center pixel. The qualified fast corners take the maximum absolute of the difference between the central pixel value and the 16-pixel values as the corner score. The mathematical expression is as follows.

$$Score = \max_x \left| V_p - V_x \right| \tag{2}$$

NMS is used to avoid detecting feature points adjacent to each other. Only when the score of a corner is higher than that of its eight neighbors, the corner is regarded as a candidate of feature point. For filtering out the high-quality feature points. The next step is to calculate the Harris response values of these candidates.

The Harris response calculation uses a hardware implementation similar to that of FAST. It first calculates the gradients I_x and I_y in the x and y directions for all points on the Harris window W, and then accumulates them to obtain the matrix A used to describe the grayscale variation near the center point. The matrix A can be defined as the following formula:

$$A = \sum_W w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{3}$$

where $w(u,v)$ is a weighting function for point (u,v) in the Harris window.

The scale coefficients of the Harris window are converted into fixed-point numbers in advance, and the shift operation is used instead of the division method. The reconstructed response value R is calculated as follows,

$$R = \left(\sum_W I_x^2 \times \sum_W I_y^2 - \left(\sum_W I_x I_y \right)^2 - k \left(\sum_W \left(I_x^2 + I_y^2 \right) \right)^2 \right) \times scale >> 52 \tag{4}$$

where $scale = 1.732873552846874$ and $k = 0.04$.

Fig. 4 shows an example of the intermediate results of feature point detection. As can be seen from the figure, the feature points tend to be concentrated in the texture-rich areas marked by the yellow dashed boxes. Too concentrated feature distribution can affect the accuracy of the SLAM system in estimating the camera pose. Therefore, it is important to ensure a homogeneous distribution for feature extraction.

B. Homogeneous distribution

To ensure homogeneous distribution, we divide the image into a grid. Unlike [7], our method no longer performs iterative judgments, but directly divides the grid into pre-designed minimum cells. If the tracking of the last frame is successful, we use the remaining points to generate a mask that prohibits the extraction of new feature points in these grid cells to avoid repeated extraction. This mask is stored by 16 32-bit registers corresponding to each grid cell, where *1'b1* is selective pass and *1'b0* is blocking.

The feature points located at the edges around the image often cannot find an exact match in the next frame due to the camera motion. Therefore, our proposed method sets up an image region of interest (ROI) that can avoid errors caused by mismatches and omit the edge padding during feature detection.

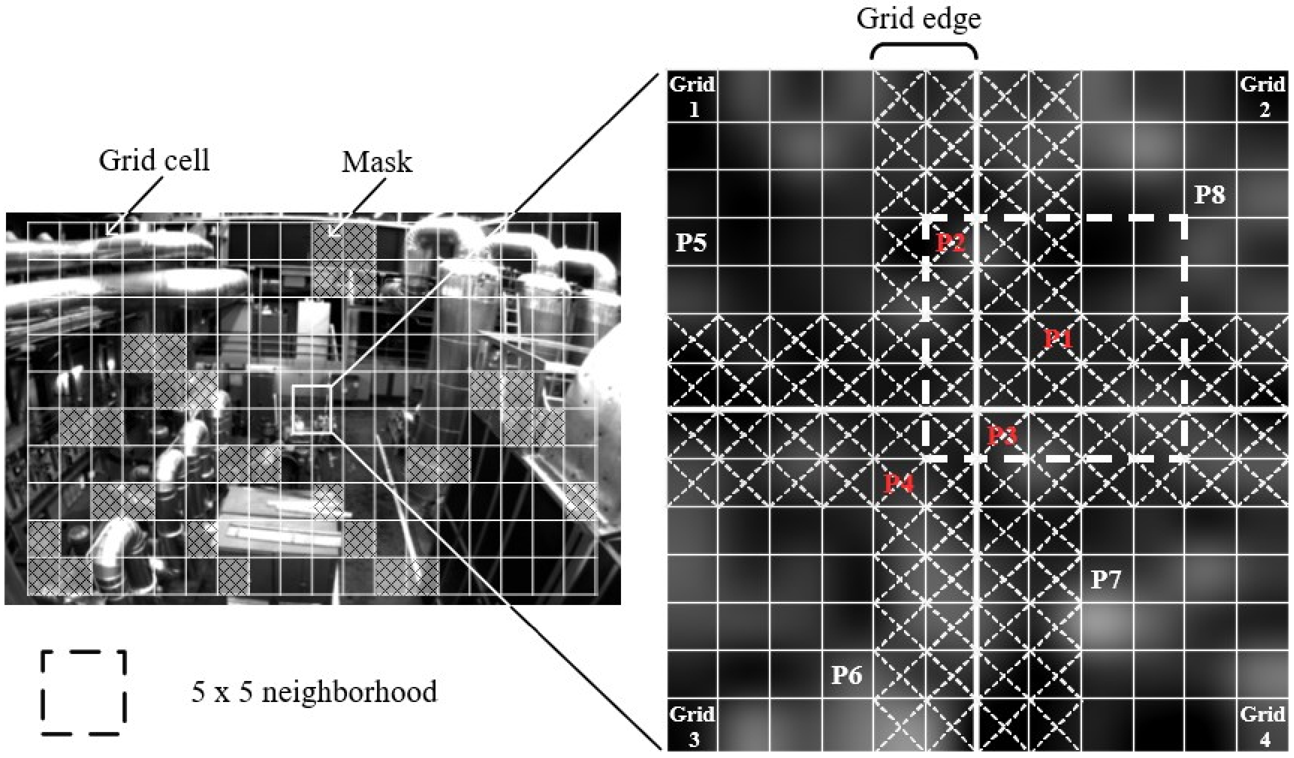


Fig. 5 Homogeneous distribution of features (left) and the decay of the Harris response at the edge of the grid (right).

As shown in Fig. 5 (left), to achieve a homogeneous distribution of feature points, the ROI of the image is divided into 16 × 32 grids according to the size of 14 × 11 pixels, and each grid cell can finally output only one feature point with the highest response value. It is output to the shared memory by the real-time module.

The images are processed in the order of scan lines from the top-left pixel to the bottom-right pixel. All candidate feature points within a grid cell can be detected only when all pixels within the grid have been input. Add a response register to store the maximum response value in the current grid cell. When the response value of the current candidate feature point is greater than the corresponding response value in the register, the candidate point is stored as a feature point in the shared memory and update the register. After every 14 rows of pixels are detected, the feature point register is refreshed, and the detection of the next grid row is continued.

To further avoid over-density of feature points at the grid cell boundaries, for example, the case where *P1*, *P2* and *P3* are selected simultaneously as shown in Fig. 5 (right). Therefore, for feature points within 2 pixels from the edge of the grid cell, their response values are multiplied by a decay factor. Under the condition that the response values are approximate, the algorithm preferentially selects *P5* to *P8* as the main feature points under the guidance of the decay factor.

Finally, the post-processing module reads out the feature points in the shared memory. The zeros in it are removed, the number of feature points to be added is selected by sorting, and the result is output as a BRAM interface. Considering the characteristics of the input image data stream, there is a gap between two camera exposures where no data is output. The feature extractor can take advantage of this time for post-processing of the output of the real-time module. And since there is no strict throughput limit and the computation is small, the post-processing module does not require much optimization as long as it is done in the gap between two frames. Fig. 6 shows an example of the intermediate results of homogeneous distribution, and also this is the final result of feature extraction.

TABLE I. THE FPGA RESOURCES UTILIZATION

	LUT	***FF***	***DSP***	***BRAM***
Utilization	34585 (12.5%)	30582 (5.5%)	37 (1.8%)	6.5 (0.9%)

IV. Implementation and Results

A. Hardware Implementation

The proposed feature extractor is implemented on Xilinx Zynq XC7Z100 SoC, which integrates two ARM Cortex-A9 processors, FPGA and 2GB DDR3 resources. The feature extractor is implemented in the FPGA, which is clocked at 100 MHz. The Zynq XC7Z100 was chosen most of all because it has a better ARM processor, allowing the rest of the algorithms to be flexibly deployed in the processor system, thus building the data path of the whole SLAM system. TABLE I. shows the resources utilization of the feature extractor and its peripheral data processing circuits. It includes the AMBA AXI4 interface, VDMA video input and BRAM interface result output.

B. Performance Evaluation

The performance of the proposed feature extractor is compared with the software implementations on the Intel i5-10400 processor and 2-core Arm v8.2 processor of Jetson AGX Xavier. The input image resolution is 480 × 752 and the performance comparison is shown in TABLE II. Where the power consumption of ARM is viewed in the NVIDIA jtop

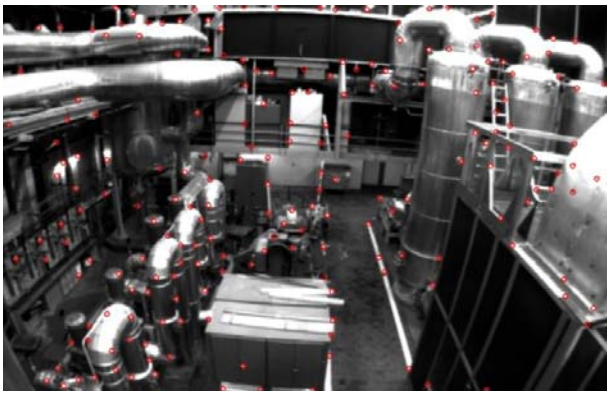


Fig. 6 Example of homogeneous distribution results.

TABLE II. PERFORMANCE COMPARSION WITH RELATED WORKS

	This work		
	Intel i5	***ARM***	***Zynq***
Runtime	55.0 ms	254.6 ms	3.6 ms
Frequency	2.9 GHz	1.2 GHz	100 MHz
Average power	65 W	714 mW	2.2 W
Energy per Frame	3575 mJ	181.8 mJ	7.9 mJ

TABLE III. EUROC DATASET. COMPARSION OF AVERAGE RMSE

	RMSE (m)					
	MH01	***MH02***	***MH03***	***MH04***	***MH05***	***Avg.***
ICE-BA[12]	0.07	0.09	0.17	0.31	0.27	0.178
NAVION [6][b]	0.09	0.09	0.17	0.20	0.27	0.164
This work	0.10	0.10	0.19	0.26	0.19	0.168

[a] Results measured by reproducing open source code on Intel i5-10400

[b] Software implementation results on a desktop Intel Xeon E5-2667

tool and the power consumption of Intel is the official published thermal design power (TDP). Accelerated by the proposed architecture, the latency of feature extraction is reduced to 3.6ms. Compared with Intel CPU and ARM, feature extraction accelerator could achieve 15.3× and 35.2× speedup in feature extraction.

In terms of energy consumption, the proposed feature extractor shows a huge advantage compared to ARM and Intel CPU. Although the power consumption of the feature extractor is increased by about 3 times compared to the ARM processor, the energy consumption per frame is still reduced by 23× based on the time consumption. Compared to the Intel i5 processor, the energy consumption is reduced by 453×.

C. Accuracy Analysis

The distribution of the extracted features will affect whether feature matching can accurately estimate the camera motion pose in space based on the before and after two frames. The number of points or the matching of points alone is not sufficient to illustrate the applicability of feature extraction to SLAM systems. Therefore, we built an evaluation platform to evaluate feature extraction with reference to a lightweight embedded SLAM system ICE-BA [12]. As shown in Fig. 7, the hardware acceleration of feature extraction is performed on an FPGA, and the rest of the front-end and back-end are handled in an embedded system Ubuntu running on ARM processor.

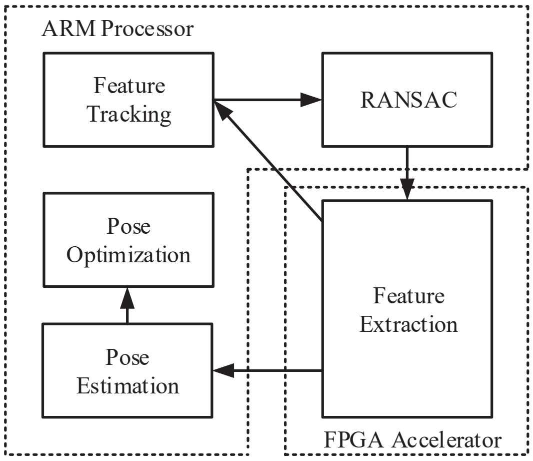


Fig. 7 Evaluation Platform Framework.

The proposed feature extractor is evaluated on EuRoC MAV dataset [11] which is a challenging drone dataset contains RGB images along with IMU information and is widely used in visual SLAM studies. The image resolution is 752 × 480. Five sequences in the dataset, MH01, MH02, MH03, MH04 and MH05 are used for the evaluation. Each sequence contains a ground truth trajectory obtained by a high precision motion capture system. Due to the randomness of RANSAC, all results are the average of five runs.

TABLE III. shows the average absolute translation root mean square error (RMSE) of this work comparing ICE-BA to software implementation on Intel i5-10400 and NAVION to software implementation on Intel Xeon E5-2667 CPU. Fig. 8 displays a comparison of the estimated trajectory implemented by the feature extractor-based SLAM with the ground truth trajectory. For MH01, MH02 and MH03 the easy and medium sequences, the software implementation has a better accuracy than the hardware implementation. However, the hardware implementation could have a good accuracy in the hard sequence MH04 and MH05. The reason could be that the high frame rate of the hardware implementation allows the system to exhibit better robustness and accuracy in the face of motion blur and drastic light and dark changes in the difficult sequences.

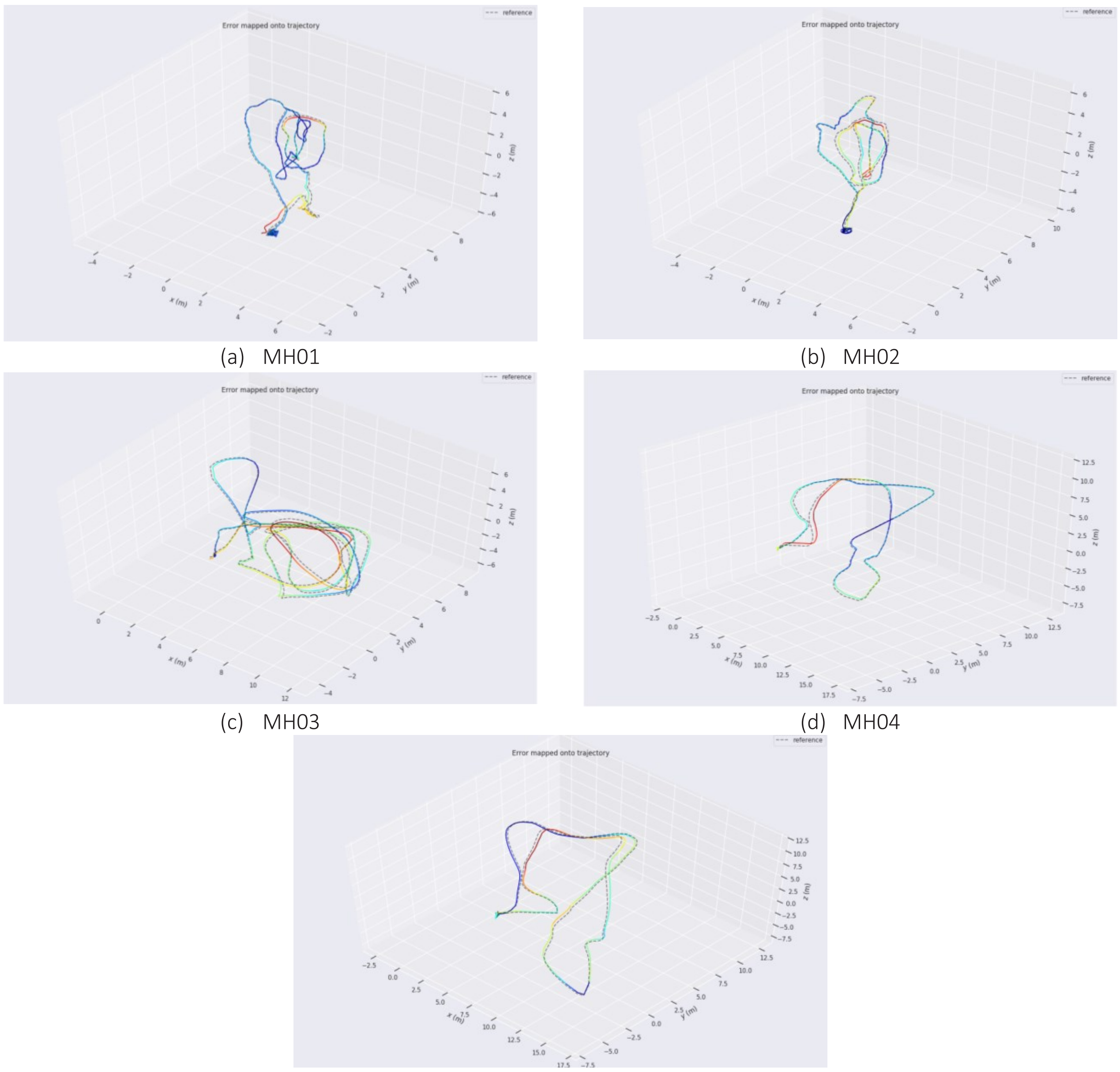
Among the five sequences, the total average error of our proposed work is about 0.168 m, which indicates that the accuracy of hardware implementation is comparable to the original software implementation.

V. CONCLUSION

In this paper, a new hardware architecture of feature extraction is proposed for real-time visual SLAM and evaluated on Zynq platforms. This architecture not only meets the system's real-time requirements, but also ensures homogeneous distribution of features. The evaluation results show that this work achieves significant speedup and power efficiency improvement in performance compared to Intel i5 and ARM v8.2, and has accuracy comparable to software implementations. Our architecture proved to be hardware-friendly and achieves a good balance of accuracy and performance.

ACKNOWLEDGMENT

The authors thank the supports from the Key R&D Program of Shaanxi Province 2019ZDLGY03-07-01.



(a) MH01 (b) MH02 (c) MH03 (d) MH04 (e) MH05

Fig. 8 Estimated trajectory implemented by the feature extractor-based SLAM with the ground truth trajectory.

References

[1] Fularz, Michał, et al. "A high-performance FPGA-based image feature detector and matcher based on the FAST and BRIEF algorithms." International Journal of Advanced Robotic Systems 12.10 (2015): 141.

[2] Kwon, Yong Hyeon, and Jae Wook Jeon. "Key Point Extraction Hardware Using SIFT Algorithm in FPGA." 2021 IEEE Region 10 Symposium (TENSYMP). IEEE, 2021.

[3] Sun, Rongdi, et al. "A low latency feature extraction accelerator with reduced internal memory." 2017 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2017.

[4] Weikang Fang, Yanjun Zhang, Bo Yu, and Shaoshan Liu. 2017. FPGA-based ORB Feature Extraction for Real-Time Visual SLAM. In Proceedings of International Conference on Field Programmable Technology (FPT). 275–278.

[5] M. Fularz, M. Kraft, A. Schmidt, A. Kasiński. "A high-performance FPGA-based image feature detector and matcher based on the FAST and BRIEF algorithms." International Journal of Advanced Robotic Systems 12.10 (2015): 141.

[6] Amr Suleiman, Zhengdong Zhang, Luca Carlone, Sertac Karaman, and Vivienne Sze. 2019. Navion: A 2mW Fully Integrated Real-Time Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones. IEEE Journal of Solid-State Circuits (JSSC) (2019), 1–4.

[7] Mur-Artal, Raul, and Juan D. Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras." IEEE transactions on robotics 33.5 (2017): 1255-1262.

[8] Liu, Runze, et al. "eslam: An energy-efficient accelerator for real-time orb-slam on fpga platform." Proceedings of the 56th Annual Design Automation Conference 2019. 2019.

[9] Sun, Rongdi, et al. "A flexible and efficient real-time orb-based full-hd image feature extraction accelerator." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 28.2 (2019): 565-575.

[10] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In European Conference on Computer Vision, volume 1, pages 430–443, May 2006.

[11] Burri, Michael, et al. "The EuRoC micro aerial vehicle datasets." The International Journal of Robotics Research 35.10 (2016): 1157-1163.

[12] Liu, Haomin, et al. "Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.