Real-Time ORB Accelerator with ROS Integration for Embedded FPGA SoCs

Andre Costa, Pedro Tomás, Nuno Roma, Nuno Neves
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
E-mail: andre.mestre.costa@tecnico.ulisboa.pt, {pedro.tomas, nuno.roma, nuno.neves}@inesc-id.pt

***Abstract*—As computer vision continues to expand across various application domains - including localisation, mapping, object recognition, and 3D reconstruction - feature extraction methods such as Oriented FAST and Rotated BRIEF (ORB) gained widespread adoption due to their rotation and scale invariance. However, existing efforts to accelerate these techniques through hardware implementations faced challenges related to high resource and power consumption demands, limiting their feasibility for low-power embedded devices. Accordingly, this paper proposes a new scalable and efficient ORB accelerator, designed for low-power resource-constrained environments. It introduces a novel resource-efficient architecture that exploits quantisation of the feature orientation angle into discrete rotation sectors. A complete ROS node based on the proposed ORB accelerator is also deployed, providing seamless integration with other computer vision-enabled systems. When compared to other state-of-the-art solutions, the proposed system, implemented on an embedded System-on-Chip (SoC) with a low-cost FPGA, offers between 6.7x and 16.2x energy efficiency improvements, while requiring fewer hardware resources.**

***Index Terms*—Hardware Accelerator, Embedded SoC, ORB Feature Extraction, ROS Integration, Computer Vision**

I. Introduction

Oriented FAST and Rotated BRIEF (ORB) [1] is a widely adopted algorithm for feature extraction in computer vision due to its significant invariance to rotation and scale [2–5]. ORB extracts relevant features from images, which are then matched to an existing database, making it useful for object classification [6] and 3D reconstruction [2]. ORB is also integral to Visual Odometry (VO) [7] and Simultaneous Localisation and Mapping (SLAM) [8–11], enabling systems like autonomous Unmanned Aerial Vehicles (UAVs) [12] or robots to track their movement [13]. In augmented reality, ORB helps overlay virtual elements on real-world objects [14]. Its computational efficiency and ability to work in real-time applications make ORB particularly suited for embedded systems, mobile devices, and robotics, where high-performance and low-power consumption are essential [1–5, 14].

However, ORB is still the main computational bottleneck in many applications. Consequently, several efforts have been made to accelerate corner detection and feature extraction with dedicated hardware, often implemented in Field Programmable Gate Arrays (FPGAs) [4, 5, 15, 16], with the less resource-consuming tasks usually being implemented in Central Processing Units (CPUs). As an example, Janosch Nikolic et al. [15] implemented the FAST [17] corner detector in hardware to accelerate a multiple camera SLAM system in an embedded Z-7020 SoC. By taking a step further, Weikang Fang et al. [4] and Vibhakar Vemulapati et al. [5] both developed a complete ORB accelerator targeting the higher-end Intel Altera Stratix V FPGA and the Xilinx ZU3EG SoC, respectively. While both works encourage the possibility of attaining a real-time implementation of an ORB accelerator, they are still above the power consumption constraints that often characterise embedded systems. This is particularly critical when light, portable, battery-powered devices are the target.

Accordingly, the work herein presented takes a step further from previous efforts, by proposing a new ORB feature extraction accelerator targeted at low-power embedded systems that can be easily integrated into a diverse set of application domains. We highlight the following contributions and features:

- A new efficient ORB accelerator architecture for low-power embedded SoCs, characterised by a configurable datapath and offering accuracy and complexity trade-offs for different applications;
- A complete feature extraction system, implemented on an embedded FPGA SoC, comprising the proposed ORB accelerator deployed on the FPGA fabric and a software module running in the CPU;
- Deployment of the proposed ORB system as a complete Robot Operating System (ROS) [18] node, capable of obtaining image frames from an HDMI source input and publishing the extracted features as a ROS topic.

The proposed system was deployed on a low-power Digilent Zybo Z7-20 board and validated with standard image sequences from the TUM-VI [19] dataset. In what concerns the feature extraction, it shows a performance comparable with the original software implementation, while offering improved energy efficiency over previous solutions.

II. Background

The ORB [1] feature extractor receives a grayscale image and outputs the detected features and their descriptors. It has two main components: the FAST [17] corner detector identifies 7×7-pixel regions of interest (features) within an input image, and the rotated BRIEF (rBRIEF) [1] constructs

Work supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under project 2022.06780.PTDC (DOI: 10.54499/2022.06780.PTDC). We also acknowledge the contributions from projects 2022.04020.PTDC (DOI: 10.54499/2022.04020.PTDC) and UIDB/50021/2020 (DOI: 10.54499/UIDB/50021/2020).

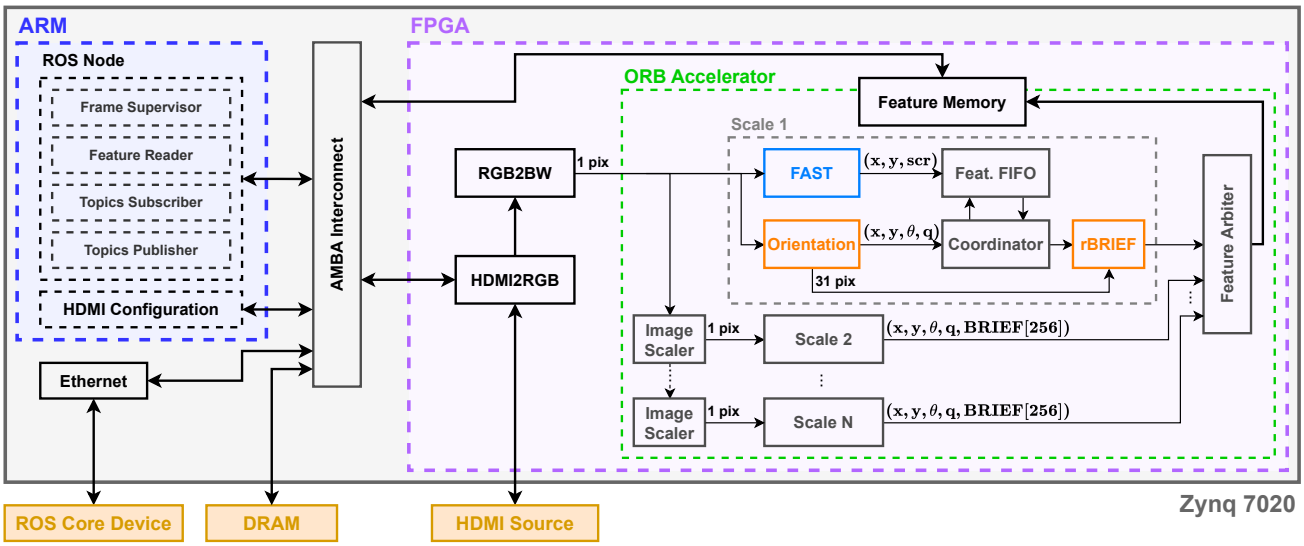

Fig. 1. Complete ORB accelerator system with all the major functionality blocks and interfaces between the SoC CPU and FPGA.

256-bit descriptors. These descriptors are subsequently used to match features between different images.

The FAST algorithm detects features by comparing the luminances of the 16 pixels on a circumference of diameter 7, with the luminance of the centre pixel. If more than 9 contiguous pixels on the circumference are either darker or brighter than the centre (difference is lower/higher than the negative/positive threshold), the region is considered to be a corner/feature. To construct a BRIEF descriptor, 256 coordinate pairs are used. Each of the 256 bits of the descriptor, $f(\mathbf{p})$, is the result of the comparison between the pixels of each pair, $\tau(\mathbf{p}, \mathbf{a}, \mathbf{b})$, where $\mathbf{a}$ and $\mathbf{b}$ are the coordinates of the two elements of the pair, and $\mathbf{p}$ is the 31×31 pixels smoothed test region centred on the detected feature. The value of τ is 1 if luminance in $\mathbf{a}$ is lower than $\mathbf{b}$, and 0 otherwise. These coordinate pairs compose the BRIEF pattern and are generated for the mentioned 31×31 pixel region. Hence, each bit (i) of the descriptor is given by:

$$f(\mathbf{p}) = \sum_{1 \le i \le 256} 2^{i-1} \tau(\mathbf{p}, \mathbf{a_i}, \mathbf{b_i}) \qquad (1)$$

For the constructed descriptors to be invariant to the orientation at which the feature was detected, ORB introduced the rotated BRIEF (rBRIEF) algorithm. It computes the main orientation of a feature region (31×31 pixels), and rotates the coordinates of the pixels to be compared accordingly, using the latter for the construction of the descriptor. The orientation, Θ, of the image patch is determined through its luminance momentum on the x (m_{10}) and y (m_{01}) directions as:

$$\Theta = arctan\left(\frac{m_{01}}{m_{10}}\right), \; m_{pq} = \sum_{x,y} x^p y^q \mathbf{p}(x, y), \qquad (2)$$

where x and y are the pixel horizontal and vertical coordinates from the 31×31 pixel region ($x, y \in [-15, 15]$). Lastly, the scale invariance is achieved by applying the aforementioned steps to consecutively down-scaled images.

III. ORB Accelerator Architecture

The proposed ORB accelerator (see Figure 1) includes a dedicated data orchestration infrastructure (not directly shown in the figure) and an ORB feature extraction module.

A. Data Orchestration

The feature detection and descriptors construction require the following stages: *i*) input image buffering; and *ii*) a Feature Memory module to store the extracted features.

*1) **Input Image Buffering**:* In real-time applications, images are often transmitted one pixel per clock cycle [20], thus requiring dedicated image buffering to hold the values of each transmitted image line. Accordingly, several line and window buffers (LB and WB), implemented with 8-bit shift registers, are placed inside several modules (see also Figure 2). The WBs are populated with the output of each LB.

*2) **Feature Memory**:* Once a feature is extracted from the input image, its position, score, orientation and descriptor are stored in a memory module that is made accessible to the outside of the accelerator. To accommodate the features from multiple scales, a dedicated Feature Arbiter is used to manage the access to the feature memory (see Figure 1).

B. ORB feature extraction module

The ORB feature extraction module is divided in three main modules: *i*) the image scalers; *ii*) the FAST corner detector; and *iii*) the rotated BRIEF encoder.

*1) **Image Scaler**:* Input buffers are also used to construct a sequence of successively down-scaled images. This is done using a simple average down-sampler with a $2:1$ scaling within a 2×2 pixels region. The scaled images are then fed to multiple scale instances (see Figure 1).

*2) **FAST Corner Detection**:* The FAST [17] module (see Figure 2.a)) processes each 7×7 pixels block and computes the luminance comparisons for the 16 tested pixels (using the configured thresholds). These comparisons are stored in

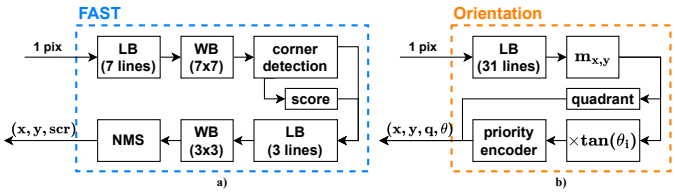

Fig. 2. ORB components representation. a) FAST. b) Orientation.

two 16-bit-wide vectors (for brighter and darker results) and checked (with AND logical operations) against bitmasks of the 16 different possible locations of 9 consecutive brighter/darker pixels. If at least one bitmask is matched, a corner is found.

To ensure that only the strongest features are chosen, a Non-Maximum Suppression (NMS) module operates on a 3×3 pixels neighbourhood. The selection score is computed as the sum of the absolute values of the circumference differences.

*3) **Rotated BRIEF Encoder**:* The construction of an rBRIEF descriptor for each detected feature is divided into three main stages: *i*) image smoothing, *ii*) orientation detection, and *iii*) descriptor construct.

- *Image Smoothing:* A Gaussian filter is used to smooth the image before computing the orientation [1]. This is implemented by convolving an integer Gaussian matrix ($\sigma = 2$) with 7×7 WBs using constant multiplicand operations [5].

- *Orientation Detection:* While the orientation quadrant is easily checked with the signal of the x and y momentums (see Equation 2), finding the θ angle of the feature involves computing an arc tangent operation. To avoid such a resource-consuming task, we discretise the orientations into N adjacent sectors. Since $tan(\theta) = y/x$, we design a priority encoder using the $x \times tan(\theta_i) > y$ condition for all N possible angles. Furthermore, to avoid storing an entire 31×31 pixels region centred on the feature, we employ the equations suggested by Vibhakar Vemulapati et al. [5] which only require tracking the sums of the incoming and outgoing columns.

- *rBRIEF Descriptor Encoder:* The rBRIEF constructor module operates over 31×31 elements, performing 256 comparisons according to the rotated BRIEF patterns. Rotating a square 31×31 BRIEF pattern would imply the need for a 37×37-pixel WB. To make the module more resource-efficient, we limit the coordinates of the considered pairs to a circle with 31 pixels in diameter (instead of a 31×31 square region) so that the rotated coordinates are never greater than 31. Furthermore, to avoid the need for a large multiplexing structure, we implement this window buffer using structured memory elements addressable in 32-bit lines. Hence, by using 4 dual-port memories with 32-bit wide words we can write/read the 31 elements (8 bits each) of a region's column per clock cycle. We replicate this window 3 times to access 6 different pixels per clock cycle. As such, we read 6 pairs of pixels per clock cycle, resulting in 86 cycles to construct the descriptor. This is done through a rotating bitmask that allows incremental composition of the descriptor. This strategy increases latency but significantly reduces resource usage.

Finally, since rotating 6 coordinates would require a considerable amount of hardware resources, we follow the simplification proposed in [1] to pre-compute the rotated patterns.

IV. ORB Accelerator Deployment in a ROS System

The proposed ORB accelerator was deployed as part of an embedded feature extraction device targeting minimal latency, low-power consumption ($<5W$), and easy integration in larger systems. It was prototyped and deployed on a Digilent Zybo Z7-20 board carrying an XC7Z020 (xc7z020clg400-1) SoC (CPU+FPGA). The FPGA Programmable Logic (PL) accommodates the ORB accelerator and the modules required to obtain the frame sequence from a HDMI video stream, while the ARM CPU is used to parameterise the HDMI interface and to launch a Robot Operating System (ROS) [18] node that can be used to report extracted features to a ROS capable system.

A. ORB Accelerator Implementation

Given the resource limitations of the implementation platform, the ORB accelerator was carefully dimensioned to fit in the FPGA PL fabric. This is done by parameterizing both the number of scales and orientation sectors, also allowing for a flexible resource management adapted to each application. Moreover, the contrast threshold of FAST can be dynamically changed at run-time according to the processed scene. The HDMI driver configures the HDMI physical layer parameters through its memory-mapped registers. The communication between the accelerator and the SoC's CPU is ensured through the native 32-bit AXI ports of the device (see Figure 1). This way, the Feature Memory, the ORB module control interface, and HDMI physical layer control registers can be accessed by the CPU through a memory-mapped interface.

B. ROS Node Infrastructure

The integration of the proposed ORB accelerator within a full-stack system is done over Ethernet, UART, or other available interfaces. To do so, the CPU runs a Linux image deploying a full ROS node. The node (see Figure 1) is responsible for identifying the frame being streamed to the ORB accelerator in the FPGA (Frame Supervisor), reading the contents of the Feature Memory (Feature Reader), publishing the detected features as a ROS topic (Topics Publisher), and listening to request for a change of parameters (Topics Subscriber). This hardware-software stack allows the ORB accelerator to be easily tuned and integrated into larger modular systems.

V. Experimental Evaluation

The proposed accelerator was evaluated on three main perspectives: *i*) accuracy and scalability; *ii*) hardware resources usage; and *iii*) energy efficiency. For this purpose, the device was tested using several image sequences from the well-known TUM-VI [19] dataset, with a 640×480-pixel resolution.

A. Accuracy and Scalability

As previously stated, one of the premises of the proposed architecture lies in the adaptability of the computation structure used to easily balance the compromise between attained accuracy and hardware resources. Such scalability was achieved by quantizing the feature orientation angle into a discrete

TABLE I
ORB ACCELERATORS PERFORMANCE COMPARISON TABLE IN TERMS OF RESOURCE USAGE, THROUGHPUT, AND ENERGY EFFICIENCY.

Resources and Accuracy											Peform. and Efficiency	
Work	**Device**	**Resolution**	**#Scales**	**Orientation**	**LUT**	**DSP**	**BRAM [Mb]**	**Latency [ms]**	**Max Freq. [MHz]**	**Power [mW]**	**Perf. [fps]**	**Energy Eff. [mJ/frame]**
[20]	XCZU7EV	3840x2160	1	RS-BRIEF*	62,223	668	1.62	#NP	150	5042	60	84
[5]	XCZU3EG	640x480	4	64 sectors	76,424	80	4.32	2.5	150	+4600	62	74
[4]	Altera Stratix V	640x480	2	256 sectors	25,648	8	1.18	14.8	203	4556	67	68
[21]	XCZ7045	640x480	1	RS-BRIEF*	56,954	111	2.81	9.1	100	1936	55.87	35
This work	XC7Z020	640x480	2	16 sectors	28,248	84	1.15	3.2	100	290	60	4.8
				32 sectors	28,521	84	1.44	3.2	100	312	60	5.2
				64 sectors	29,080	84	2	3.2	100	328	60	5.5

*Uses the RS-BRIEF descriptor [21], i.e., it does not rely on the rotation of the BRIEF pattern but instead the rotation of the descriptor. +Energy consumption is not provided for the isolated ORB accelerator, we deduce the value based on the provided comparison against [21]. # Not provided.

number of possible sectors. However, it is foreseeable that the division of the feature orientation into sectors negatively impacts the robustness to rotation. To evaluate this issue, we assess the value of positive matches between the detected features in the original and each rotated image (i.e., Inliers [1]). Figure 3 presents this metric considering 3 different levels of discretization: 16, 32, and 64 sectors. The chosen baseline performance for this analysis was the ORB implementation from the OpenCV [22] software library. As expected, a clear accuracy improvement is obtained when increasing the number of orientation sectors, especially from 16 to 32. Accuracy peeks occur at 90°, 180°, 270° and 360° since at those orientations the rotation of the BRIEF pattern is not approximated.

Despite the accuracy compromise from discretization, when considering popular SLAM data sets [19, 23], the robustness to rotation is enough to maintain feature tracking, since the orientation of the frame does not commonly surpass an absolute value of 90°. Nonetheless, for applications especially sensitive to in-plane rotation, the accelerator can either be configured with a single scale and additional orientation sectors or simply deployed with more sectors (and scales) on a larger FPGA.

B. Hardware resources

The proposed accelerator was implemented and deployed on the adapted FPGA SoC with the AMD Xilinx Vivado 2020.2 toolchain. Hardware resource usage and power estimation were also obtained with the available tools. Table I compares our results with related works. In most previous efforts [4, 5, 20], higher-end FPGA systems had to be used, since their architectures require substantially more hardware resources. The most similar solution is from [21], which uses a SoC of the same family, albeit with more available and used resources. The proposed ORB accelerator architecture also performs well

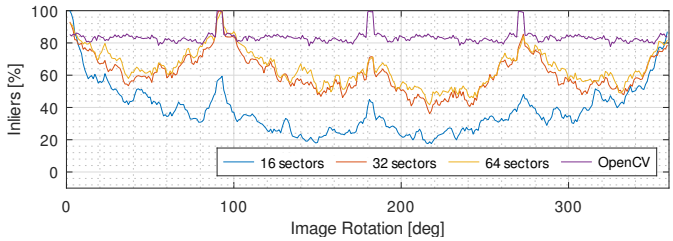

Fig. 3. Percentage of computed Inliers considering a complete plane rotation.

in terms of output latency, which refers to the time taken from the start of the transmission of an image to the output of a feature located at the lower right corner of that same image.

In Table I it is also possible to observe the resource usage of the proposed architecture when considering several different discretisation levels. These results clearly evidence the provided facility to adjust the aimed balance between the implementation cost and the resulting accuracy depending on the application requiremnts and available resources.

C. Energy efficiency studies

Being targeted for embedded and low-power devices, energy efficiency was considered a key design constraint of this accelerator. Under this premise, we used the consumed energy per processed frame as an energy efficiency indicator. The obtained results (see Table I) clearly show the advantages and efficiency of the proposed architecture. In particular, its resource efficiency not only allowed it to be deployed on a much more resource-constrained FPGA when compared to the higher-end devices used in [4, 5, 20], but it also showed to require 50% less resources when implemented on a similar grade device as [21]. This is further highlighted by energy efficiency gains between $6.7\times$ and $16.2\times$ over the other works.

VI. CONCLUSION

This manuscript presents a novel and efficient ORB feature extraction accelerator tailored for low-latency, low-cost, and low-power embedded systems. By employing a scalable architecture that quantizes feature orientation into discrete sectors, the proposed design successfully optimizes the balance between hardware resource usage and feature detection accuracy. Experimental evaluations of the accelerator deployed as a ROS node on an embedded SoC with a low-cost FPGA demonstrated the accelerator's significant advantages over existing state-of-the-art solutions, showing average energy efficiency gains of between $6.7\times$ and $16.2\times$. The accelerator's efficiency, and versatility provided by the ROS interface, show its suitability for real-time computer vision applications in resource-constrained environments, making it a viable and promising solution for a wide range of practical use cases.

REFERENCES

[1] 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.

[2] J. Huo, C. Zhou, B. Yuan, Q. Yang, and L. Wang, "Real-time dense reconstruction with binocular endoscopy based on stereonet and orb-slam," *Sensors*, vol. 23, no. 4, p. 2074, 2023.

[3] A. P. Tafti, A. Baghaie, A. B. Kirkpatrick, J. D. Holz, H. A. Owen, R. M. D'Souza, and Z. Yu, "A comparative study on the application of sift, surf, brief and orb for 3d surface reconstruction of electron microscopy images," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 6, no. 1, pp. 17–30, 2018.

[4] W. Fang, Y. Zhang, B. Yu, and S. Liu, "Fpga-based orb feature extraction for real-time visual slam," in *2017 International Conference on Field Programmable Technology (ICFPT)*. IEEE, 2017, pp. 275–278.

[5] V. Vemulapati and D. Chen, "Fslam: an efficient and accurate slam accelerator on soc fpgas," in *2022 International Conference on Field-Programmable Technology (ICFPT)*, 2022, pp. 1–9.

[6] A. Kaur, M. Kumar, and M. K. Jindal, "Cattle identification system: a comparative analysis of sift, surf and orb feature descriptors," *Multimedia Tools and Applications*, vol. 82, no. 18, pp. 27 391–27 413, 2023.

[7] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1. Ieee, 2004, pp. I–I.

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[9] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[10] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 international conference on computer vision*. IEEE, 2011, pp. 2320–2327.

[11] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[12] S. J. Haddadi and E. B. Castelan, "Visual-inertial fusion for indoor autonomous navigation of a quadrotor using orb-slam," in *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*. IEEE, 2018, pp. 106–111.

[13] C. Sun, X. Wu, J. Sun, N. Qiao, and C. Sun, "Multi-stage refinement feature matching using adaptive orb features for robotic vision navigation," *IEEE Sensors Journal*, vol. 22, no. 3, pp. 2603–2617, 2021.

[14] P. Chen, Z. Peng, D. Li, and L. Yang, "An improved augmented reality system based on andar," *Journal of Visual Communication and Image Representation*, vol. 37, pp. 63–69, 2016.

[15] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 431–437.

[16] R. Eyvazpour, M. Shoaran, and G. Karimian, "Hardware implementation of slam algorithms: a survey on implementation approaches and platforms," *Artificial Intelligence Review*, vol. 56, no. 7, pp. 6187–6239, 2023.

[17] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, 2005, pp. 1508–1515 Vol. 2.

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[19] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stueckler, and D. Cremers, "The tum vi benchmark for evaluating visual-inertial odometry," in *International Conference on Intelligent Robots and Systems (IROS)*, October 2018.

[20] M. Wasala, H. Szolc, and T. Kryjak, "An efficient real-time fpga-based orb feature extraction for an uhd video stream for embedded visual slam," *Electronics*, vol. 11, no. 14, p. 2259, 2022.

[21] R. Liu, J. Yang, Y. Chen, and W. Zhao, "eslam: An energy-efficient accelerator for real-time orb-slam on fpga platform," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3316781.3317820

[22] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[23] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033