IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 41, No. 6, December, 2024, pp. 3203-3211
Journal homepage: http://iieta.org/journals/ts

Design of Corner Detection System Based on FPGA

Hadj Fredj Amira[1*], Kechiche Lilia[2], Mejbri Nesrine[3], Malek Jihene[4,5]

[1] EPI Digital School, EPI - International Multidisciplinary School, University of Sousse, Sousse 4002, Tunisia
[2] Department of Science and Technology, University College of Ranyah, Taif University, Taif 21944, Saudi Arabia
[3] Department of Electrical Engineering, Higher Institute of Applied Technologies of Kairouan, Kairouan 3110, Tunisia
[4] Department of Electronics, Higher Institute of Applied Sciences and Technology of Sousse, University of Sousse, Sousse 4002, Tunisia
[5] Laboratory of Electronics and Microelectronics, LR99ES30, University of Monastir, Monastir 5000, Tunisia

Corresponding Author Email: amira.hadjfredj@gmail.com

Copyright: ©2024 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/ts.410636

Received: 10 May 2024
Revised: 30 September 2024
Accepted: 4 December 2024
Available online: 31 December 2024

Keywords:
Harris corner detector, segmentation, FPGA, Zedboard, OV7676 module

ABSTRACT

This paper explores the significance of embedded electronic systems, focusing particularly on the implementation of the Harris Corner Algorithm for computer vision applications. Embedded systems are pivotal in managing diverse machinery, vehicles, and environmental parameters as they are known for reliability and efficiency in real time processing in addition to their embedded features. The Harris Corner Algorithm stands out for its proficiency in object detection, image registration, and feature matching within the realm of computer vision. The study proposes a tailored approach of deploying the Harris Corner Algorithm on a Field Programmable Gate Array (FPGA) to enhance its efficiency for image processing tasks. The proposed algorithm is implemented and tested using VHDL language for a target Zedboard FPGA development board and an OV7676 camera module. Experimental results show an efficiency of the algorithm with minimal power consumption and high precision in detecting corners within images captured through the camera module with image resolution of 640×480. This study underscores the significance of embedded electronic systems in advancing computer vision capabilities, particularly through tailored algorithmic implementations on FPGA platforms.

1. INTRODUCTION

Recent technological advancements have had a significant impact on image and video processing over the years. From object recognition to facial recognition, these technologies have evolved with greater accuracy, precision, and speed. One of the known techniques in computer vision applications is image segmentation. Image segmentation can be defined as the problem of dividing an image into multiple parts in order to get a visual understanding of the contained objects. It allows us to obtain a compact representation of the useful parts of an image. These parts may be used for two different perspectives, a high-level perspective, which is the recognition, and a low-level perspective, which is the delineation. In general, segmentation is preceded by some image preprocessing steps that can be crucial for better results like enhancing contrast, improving quality, and reducing noise in images [1-3]. A good segmentation result allows images to be analyzed at different levels of detail, making it easy to identify particular features or regions within them. Image segmentation covers a large domain of applications such as object recognition [4, 5], facial recognition [6, 7], motion tracking [8], image editing [9], image compression [10, 11], and medical imaging [12-14]. In medical imaging, image segmentation is used to identify abnormalities in the image, such as tumors [12, 13], cysts [14], and classifications [15]. It allows doctors and physicians to locate and analyze regions of interest in the image. Image segmentation also helps with neural network processing to diagnose and treat neurological conditions [16].

As segmentation partitions an image into meaningful regions, this step helps group pixels with similar characteristics, such as color or texture into meaningful regions. Once the image has been segmented into regions, feature extraction algorithms will be used to extract features from these segments. Features could include attributes such as edges, texture patterns, color histograms, corners, or any characteristic according to a specific task. There exist many feature detection algorithms that differ from each other by the used method and the target application. We mention the Shi-Tomasi corner detector [17, 18], Feature from Accelerated segment (FAST) [19], Scale Invariant feature transform (SIFT) [20, 21] and Speed Up Robust Feature (SURF) [22, 23]. The choice of the right algorithm depends on various factors, such as available resources and the specific requirements of the application. The Shi-Tomasi algorithm computes a score for each pixel in an image based on the minimum eigenvalue of the structure tensor. It uses a scoring function to identify the most prominent corners. Although this algorithm has proved a high accuracy, it is noise-sensitive and is not a scale invariant. The FAST algorithm is considered as

a computationally efficient algorithm which is suitable for real-time applications. The algorithm compares the intensity of pixels around a candidate corner point with a threshold. It is characterized by a High computational efficiency and robustness to noise. Some of the common drawbacks are the generation of an enormous number of false positives in certain scenarios and being not scale-invariant. The SIFT is an algorithm that performs both feature detection and description. This algorithm detects corners and is robust to many image modifications like rotation, illumination, and scale. It's widely used in computer vision tasks such as object recognition and image stitching. One of the most important drawbacks of this algorithm is its computationally expensive nature, especially when used to match features for large datasets. It necessitates a large amount of memory to store the feature descriptors.

The SURF algorithm is similar to the SIFT algorithm with the feature of being more computationally efficient which makes it suitable for real-time applications with limited resources.

Harris Corner detector [24-26] is also a feature detection algorithm which works by detecting the local changes in the intensity of the image. The algorithm first calculates a gradient for each pixel in the image and then divides the image into small windows. Then, it constructs a matrix for each window, which is used to measure the intensity variation across the window. The algorithm identifies corners by calculating the variation in the matrix's eigenvalues. These eigenvalues indicate that the intensity variation is higher at the corners of the image. The Harris Corner Algorithm is commonly used in numerous applications, such as facial recognition, object detection, and image registration. In medical imaging, the algorithm can detect the features of particular regions in an image, helping doctors diagnose and treat medical conditions more accurately and precisely. Compared to other algorithms, the Harris corner is preferred over the others for its rotational invariance, noise resilience, good localization accuracy, and stable corner detection.

One of the key challenges in corner detection algorithms is their computational complexity. They require a high number of operations to analyze the image correctly. In medical imaging, corner detection algorithms can take a long time to process medical images, which can hinder patient care. As with other corner detection algorithms, the Harris corner detection algorithm represents a computationally intensive process as it requires processing gradient calculations, computing a second-moment matrix, applying a Gaussian smoothing, computing the corner response, and applying thresholding. Traditional available software-based implementations of the Harris Corner Algorithm often struggle to achieve real time processing constraints imposed by the high-resolution image and video data. In addition to its computational complexity, it is subjected to many other constraints. The algorithm may need to process large datasets containing high-resolution images or videos which can increase the computational load. In addition, tasks that must meet real-time processing constraints often face difficulties when optimized for general-purpose CPUs, as these letters are not inherently designed for parallel processing.

Therefore, there is a need for an efficient implementation of the data processing techniques. To optimize the processing time for medical applications, several optimization strategies can be employed. For example, code optimization is critical in reducing processing times. By using efficient algorithms and an optimized code, processing times can be reduced more effectively. Additionally, using faster processors or more powerful computers can also help speed up processing times [27]. Cloud computing which is a distributed computing model, can also help reduce processing time by providing faster access to resources. Techniques such as parallel processing, caching, and hardware accelerators can also reduce processing time. One of the commonly used platforms for hardware acceleration are Field programmable gate array (FPGA). These platforms are used for real-time medical image processing [28, 29]. Their highly parallel architecture allows them to be used to implement algorithms with real-time constraints and high computational demands like segmentation, registration, and reconstruction in the medical field. FPGA platforms allow the implementation of complex algorithms in real-time, which is not achievable with standard processors. FPGA platforms reduce processing time, enabling applications to be executed more efficiently. Furthermore, FPGAs have low power consumption and heat dissipation, making them ideal for portable medical devices.

Despite the advancements in hardware acceleration techniques, achieving both high performance and low power consumption remains an important gap in the literature for the FPGA-based implementation of this algorithm.

The presented paper addresses this gap by proposing a novel FPGA implementation of the Harris Corner Algorithm which, compared to previous methods, the presented approach takes advantage of the parallel processing capabilities of FPGAs to enhance processing speed while reducing power consumption, making it well-suited for real-time applications.

The contributions in this paper can be cited as:

- The proposition of a hardware architecture of the Harris corner algorithm.
- The Design of a low-power system that can be used in several applications.

The remaining sections of this paper are organized as follows: Section 2 presents background information on Harris Corner systems and architectural designs, along with an overview of related research on FPGAs. Section 3 describes the Harris Corner Detector Algorithm. Section 4 explains the hardware implementation of FPGA. Section 5 showcases the implementation results and ensuing discussions. Finally, in Section 6, we conclude from our findings and outline potential future avenues of research.

2. RELATED WORK

Nowadays, with the advances in imaging technology and the availability of high-resolution cameras and sensors, image processing tasks have become highly complex and resource-demanding tasks. These advances have led to large sizes of modern images with resolutions ranging from tens of megapixels to hundreds of gigapixels. These facts have led researchers to propose new methods to meet the computational demands of the applications. In this context, one of the commonly used solutions is hardware accelerators which are specialized computing devices that can be used to optimize and accelerate specific tasks, including image processing. The FPGA platforms are commonly used for this purpose. In the literature [30], the authors use the Zynq platform to implement the real-time application of the Canny edge detection algorithm on the ZC702 development board. The system is built using Vivado High Level Synthesis (HLS) and includes an in-house video library. The result is a system that can

achieve frame rates of 60 fps at 1080p resolution. A traffic sign image recognition and classification system are proposed utilizing 1920×1080 resolution images captured by an integrated camera on a Zedboard development board [31]. This study found that classifying traffic signs in the developed system took approximately 5 seconds. The optical flow algorithm was optimized in study [32]. Therefore, the same algorithm was tested on two different architectures, a PC with Core i7 processor and the ARM processing unit of the Zynq-7000 chip without optimization, and then with a special Optimization of the Programmable Logic Unit architecture. The code of the system was synthesized in C language using the VIVADO HLS program. The results demonstrate that the performance of this algorithm running on Zynq is close to that of a PC with a Core i7 processor [32]. In their study [33], the authors presented a project situated at the convergence of embedded systems and image processing realms. The goal is to develop an embedded vision system tailored for video acquisition and processing. This involves the integration of hardware accelerators to facilitate the extraction of specific image characteristics. The utilization of reconfigurable platforms, including the latest All Programmable System on Chip (APSoC) platforms, together with advancements in high-level Electronic Design Automation (EDA) tools for their configuration, have given rise to FPGA-SoC-based image processing. This approach has proven to be a viable and practical solution for addressing a myriad of computer vision challenges. The authors [34] proposed an FPGA acceleration for real-time classification of hyperspectral imagery (HIS) application. HSI applications are frequently used in several critical applications like disaster control, precision farming, industry and others. The support vector machine (SVM) is proposed as a classification algorithm. As this algorithm has a complexity that grows linearly with the number of support vectors, authors proposed a hardware implementation to reach real-time performances. The proposed architecture contains a software part on a host computer to receive HSI streams and a hardware application on Artix-7 35 T FPGA to perform the online classification. A prior offline classification using Matlab was performed in order to get statistics on memory usage and independent operations. The obtained results show a timing of 1.09 µs and 3.51 µ for the hardware implementation against a 31.5 µs, 15.6µ for software solutions.

Corner detection algorithms are vital in various image processing and computer vision tasks as they are used to identify points, in an image, which are the areas where pixels' intensity sharply changes in multiple directions. These corners are used in various applications like object detection, motion tracking, and 3D modeling. Multiple corner detection algorithms exist like Shi-Tomasi [17], SIFT, Harris corner and other hybrid algorithms where the authors make a mix of more than one method [35-37]. Each one of the presented algorithms is characterized by various trade-offs in terms of speed, accuracy, and computational complexity. The software implementations of these algorithms can be sufficient for less demanding applications. For the real-time image processing tasks and resource-constrained environments, there is a need for hardware implementations to meet performance levels. FPGAs are utilized for these implementations for their parallel processing and low latency. The comparison of the existing hardware implementations of the different corner detections can be done based on 3 main points, architecture, resource utilization, and efficiency. The authors [35] implement the proposed hardware architecture with reduced line buffers, using SRAM, and implement an interleaved memory access for better throughput. The obtained results achieve up to 33% line buffer savings and show support for 4K resolution at 60 frames per second with an accuracy loss of 2.29%. The proposed design uses a 7×3 convolution buffer and logic folding to enhance resource sharing and to lower multiplexing overhead [36]. They achieve a 20% reduction in critical path delay and, 24% area reduction, and maintain high accuracy with only a 0.0588 difference in the repeatability rate. The supported resolution is 1080p with 60 frames per second. The proposed architecture is based on an early-stage pruning which aims to reduce the computational load in order to obtain a few pixels requiring full corner detection calculations [37]. The results show a 75% speedup on a Nios-II processor with a retained corner detection accuracy. However, the pruning can struggle with highly textured images, potentially missing corners in complex regions. The Harris Corner Algorithm is used with FPGA implementations in order to obtain a balance between computational efficiency and robustness. Unlike other algorithms which may be more computationally expensive or which sacrifice some accuracy for speed, the Harris corner represents an efficient solution for real-time processing applications.

3. HARRIS CORNER DETECTOR ALGORITHM

The Harris corner detector is a popular algorithm used to find corners in an image. It works by detecting the local changes in the intensity of the image. The algorithm looks for areas of the image where the intensity changes rapidly in all directions. These areas are likely to be corners. The algorithm then calculates a score for each point in the image, and this score is used to determine which points are corners. The higher the score, the more likely the point is to be a corner. The algorithm then returns the coordinates with the highest scores. The Harris detector works by first computing the derivatives of the image in two orthogonal directions. Then, it computes the second-moment matrix of the derivatives, which is a matrix of the products of the derivatives in each direction. Finally, it computes the corner response function, which is a measure of the corners of the image at each point. A picture's corners are areas where the intensity changes significantly and where the image resists distortion. To correspond to wedges, Harris points are generated based on the pixel light intensity. It is suggested to use a detector based on the autocorrelation expression of the following intensity fluctuations:

$$M(x,y) = \sum_{u,v} w(u,v).\begin{pmatrix} I_x^2(x,y) & I_x I_y(x,y) \\ I_x I_y(x,y) & I_y^2(x,y) \end{pmatrix} \quad (1)$$

where, $w(u,v)$ is a weighting on the window (u,v), and I_x and I_y are the local derivatives in x and y. One can ascertain if a point is an outline, a homogeneous region, or a corner by examining the eigenvalues of the matrix M. M is used to compute a final criterion that allows one to choose the kind of point found. The Harris detector's multiple stages are depicted on the graph in Figure 1. The value K varies between 0.04 and 0.06. the Gaussian convolution in Eq. (2).

$$\mathrm{G(x,y)} = \sum_{\mathrm{u=-k}}^{\mathrm{k}} \sum_{\mathrm{v=-k}}^{\mathrm{k}} \mathrm{H[u,v]F[x-u,y-v]} \quad (2)$$

where, x represents the horizontal distance from the origin, and y represents the vertical distance from the origin, H[u, v] is the Gaussian kernel of size (2k+1)×(2k +1) defined in Eq. (3).

$$H(u,v) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(u^2+v^2)}{2\sigma^2}} \quad (3)$$

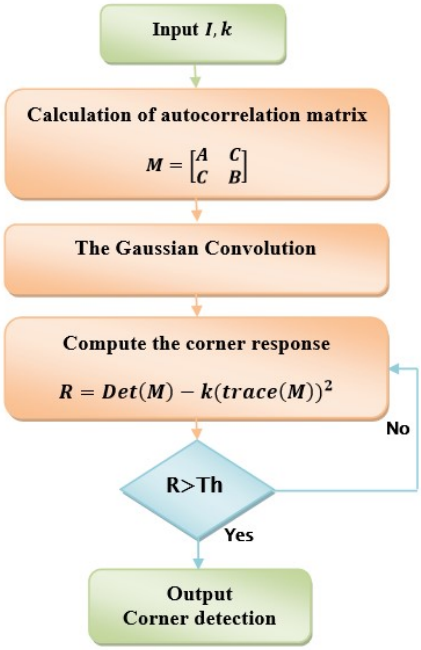

Figure 1. Stages of the Harris detector

4. HARDWARE IMPLEMENTATION

The Harris corner detection algorithm has a computationally intensive nature as it has iterative tasks and needs to compute derivatives and eigenvalues for every pixel in an image. The computation process contains a convolution of the image with derivative kernels, a computation of the structure tensor for each pixel, and a computation of the eigenvalues to determine corner responses.

The various steps of the hardware implementation methodology are outlined in Figure 2 and described as follows:

1. **Image Capture:** The image is captured using the OV7670 image sensor module. The module provides VGA resolution images, which are converted into an 8-bit grayscale format for further processing.
2. **Preprocessing:** The grayscale image is passed through a preprocessing step where a 6×6 window is created using D flip-flops and line buffers implemented with block RAM. This window is crucial for the subsequent gradient calculations.
3. **Gradient Calculation:** The next step involves computing image gradients in the X and Y directions. A 5×5 pixel window is used, and an array of 50 subtractions is used to perform these gradient calculations in parallel.
4. **Structure Tensor Computation:** From the computed gradients, components of the structure tensor are calculated in parallel using multipliers and adders. This includes computing G_x^2, G_y^2, G_x G_y values for the matrix.
5. **Harris Score Calculation:** Using the structure tensor, the Harris corner score is computed. The score is compared to a threshold value, and if it exceeds this threshold, the point is marked as a corner.
6. **Non-Maximum Suppression:** To ensure only the most prominent corners are selected, a non-maximum suppression technique is applied, where local maxima are identified, and weaker responses are suppressed.
7. **Display of Results:** The processed image along with the detected corner points are output to a VGA monitor for visualization. A frame buffer separates the camera and VGA signal domains.
8. **FPGA Implementation:** The design is synthesized using VHDL/Verilog in the Xilinx VIVADO IDE, targeting the ZedBoard's FPGA. The synthesized design includes logic for the corner detection algorithm and interfaces for the camera and display.

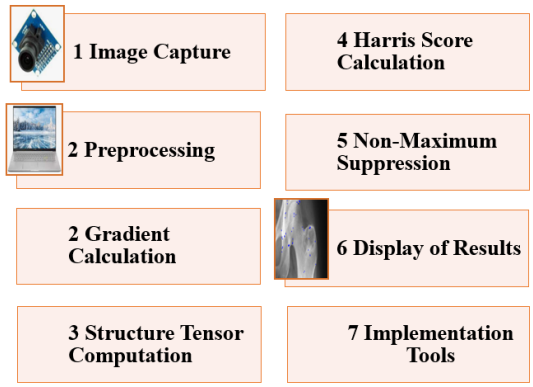

Figure 2. Step-by-step process flow of hardware implementation

With the increase in image size and the accuracy levels that are required, the algorithm has a huge computational complexity. As a result, implementing Harris corner detection on hardware accelerators is of great significance. Hardware accelerators allow to benefit from parallel processing and specialized hardware resources to execute computationally intensive operations. In this work, we have chosen the ZedBoard.

The ZedBoard is a development board designed by Digilent [38]. The ZYNQ Systems on Chip (SOC) from Xilinx incorporate:

- The Processing System (PS) is powered by a dual-core ARM Cortex-A9 processor, capable of running an operating system like LINUX.
- The Programmable Logic (PL) system features an FPGA from the XILINX-7 series, which includes essential logic elements, RAM, DSPs, and standard I/O interfaces.

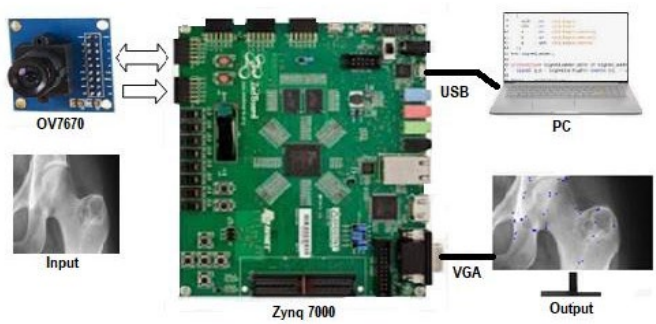

Figure 3. System block design

The system utilizes an OmniVision OV7670 image sensor and displays results on a VGA monitor. The FPGA component is based on the Xilinx Zedboard. A system block diagram is presented in Figure 3.

The OV7670 module [39] is a compact image sensor with a low operating voltage that nevertheless provides all the functions of a single-chip VGA camera. The Serial Camera Control Bus (SCCB), allows displaying the entire under-sampled image in a window with 8- bit data resolution. The VGA sampling rate is 30 frames per second. Image quality, data format, and transmission mode can be configured. However, functions related to image processing, including gamma curves, white balance, saturation, and chromaticity, cannot be programmed through the SCCB interface. The integrated OmniVision image sensor system enhances image quality by reducing or eliminating optical or electronic defects such as fixed pattern noise, color issues, and image clarity and stability. The divided data will be converted back to the 16-bit RGB565 format by the camera data capture module.

The Harris corner detector provides the corner feature output, as illustrated in Figure 4. To address the timing mismatch between the camera and VGA signals, a frame buffer is utilized to isolate the two timing domains. However, this component can be eliminated to cut costs if a display is not needed. External DRAM can also be used to avoid consuming block RAM. RGB565 pixel data is converted to 8-bit grayscale using the formula: 0.5 * G + 0.25 * R + 0.25 * B, where R, G, and B represent the red, green, and blue color components, respectively. The multiplication is performed using bit-shift operations. The pixels are then processed through D flip-flops and a line buffer to form a 6x6 window, with line buffers implemented using block RAM resources.

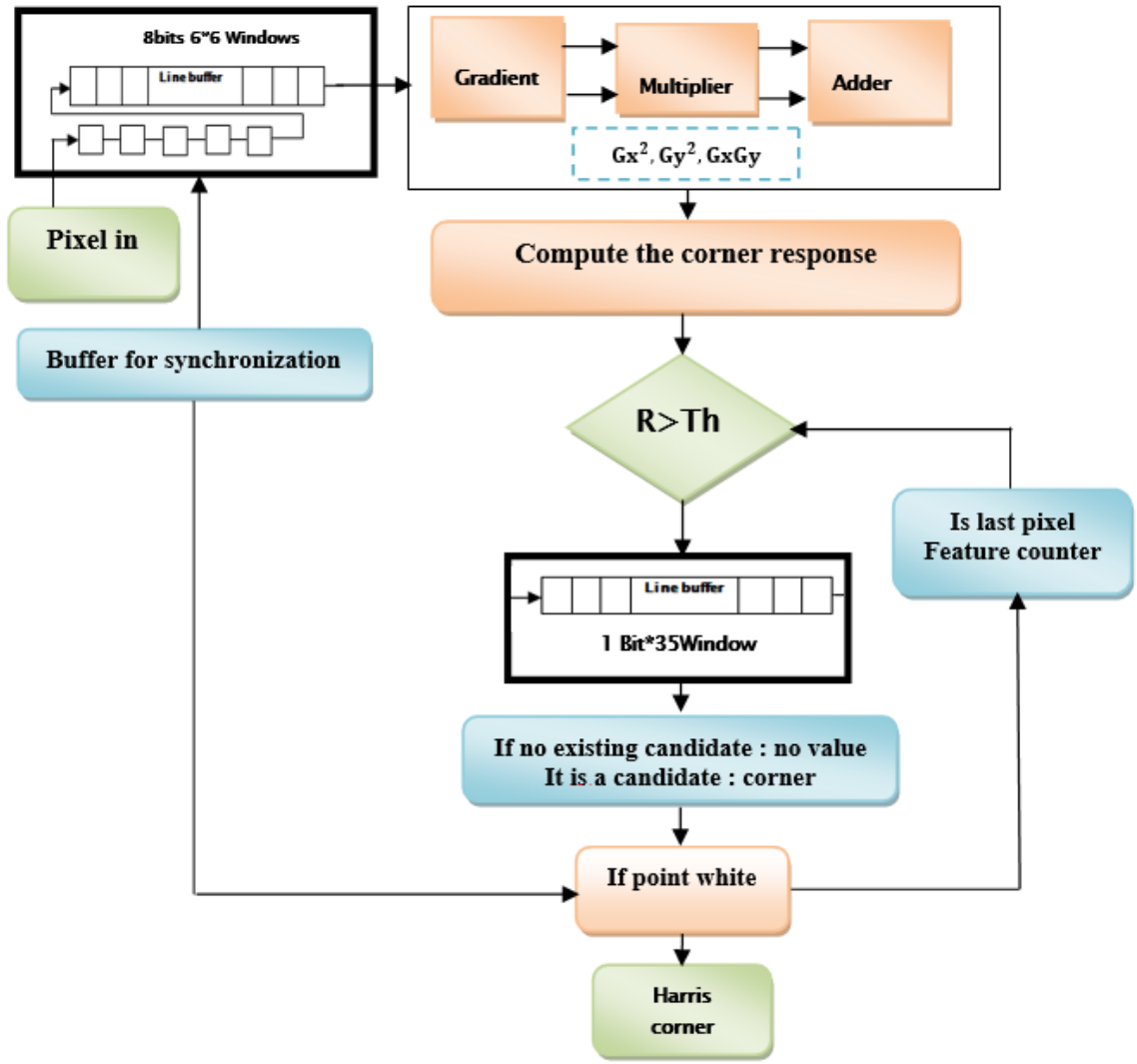

Figure 4. Hardware architecture of Harris detector

A set of 50 subtractions is used to calculate the image gradients in the X and Y directions, with a window size of 5×5 pixels. Multipliers and adders work together in parallel to compute the components of the 2×2 matrix M.

The components M1 and M2 are the same, while 3 components are required for the calculation of G_x^2, G_y^2, G_x, G_y.

The Harris score is calculated using the estimated equation and then compared against a predefined threshold. In the event that it is greater than the edge, it may be a corner including the candidate. The proposed method provides a corner feature counter at the end of the processing of every frame. Indeed, if the corner candidates at the end of the frame are out of the demanded values, the threshold (R) will be multiplied by 2 in order to fine-tune the sensitivity of corner detection to better match the specific requirements of the application or the characteristics of the image.

However, if the corner value obtained is in the expected range, the threshold stays the same. The feature counter is

considered a corner feature when the Harris score (R) is greater than the threshold (Th). The feature candidate is transmitted as a 1-bit feature window across 35 windows. The local area should contain only a corner feature. Our approach uses a binary feature window based on traditional non-maximum suppression, meaning that values outside the expected range ($R_¡ > Th$) will be delayed through feedback. The chosen multiplication is based on the obtained features which are proportional to a polynomial function. The threshold should have minimum and maximum values. Our calculation method has the objective of finding the most useful value of the Harris detector in order to obtain the clearest image.

5. RESULTS AND DISCUSSIONS

5.1 Test environment

All tests were conducted in a laboratory environment with controlled temperature and appropriate lighting conditions to ensure a stable operation of the FPGA and the image sensor (Figure 5). We utilized:

FPGA Board: The Harris Corner Detection algorithm was implemented on a Xilinx ZedBoard (Zynq-7000 SoC).

Camera Module: The OV7670 image sensor was used to capture real-time images with a resolution of 640×480 pixels.

Memory:

- Block RAM (BRAM) within the FPGA was used for temporary storage of image data during processing.
- External DRAM was utilized for storing the processed image data before output.

Display: The processed images were output to a VGA monitor connected to the ZedBoard.

Vivado Design Suite (Version 2017.2): We used the Xilinx Vivado Design Suite to write and synthesize the HDL (VHDL/Verilog) code for the FPGA implementation. This tool was also used for debugging and simulation.

Test Images: Both static and dynamic scenes were used for testing. Static scenes involved simple objects like books or boxes, while dynamic scenes included movement and variations in lighting to test the robustness of corner detection.

Real-time images have been successfully procured on a VGA screen by manually adjusting the camera in different directions. Figure 6 presents the results of the simulations. The images depict the object along with the detected key points for each object. The original grayscale image is shown, with white points marking the positions of the detected corner features. Colored points have manually been added to these images to enhance visualization.

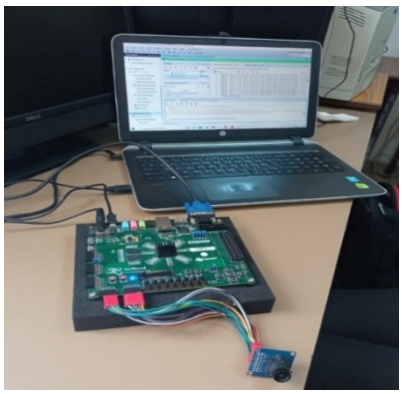

Figure 5. Sytem implementation

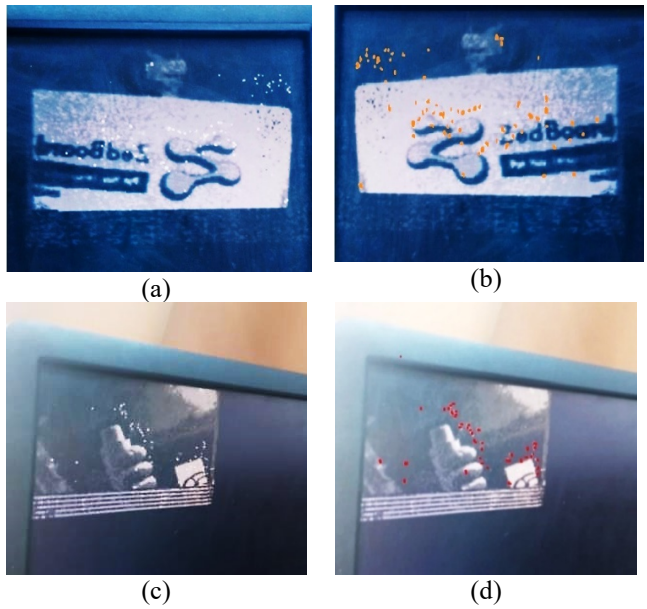

Figure 6. Harris Corner detected by our proposed FPGA implementation

5.2 Synthesis result

To evaluate our proposed system, we utilized three key performance metrics:

- **Frame Rate**: The system achieved a real-time frame rate of up to 102 frames per second (FPS) at a resolution of 640×480 pixels, demonstrating its suitability for applications that require high-speed image processing.
- **Power Consumption**: Power consumption was measured during testing using VIVADO's power analysis tools, and the system operated at approximately 0.33W under full load. This low power consumption makes it ideal for use in energy-constrained environments or portable devices.
- **Accuracy**: The accuracy of corner detection was assessed by visually inspecting the detected corners in the output images, which were displayed on a VGA monitor. The high precision of detected corners confirmed the hardware implementation reliability in identifying key points within the images.

Table 1 outlines the synthesis and implementation outcomes for the proposed design. According to the VIVADO synthesis tool, the total power consumption was 0.33 W, consisting of 0.203 W for dynamic power and 0.126 W for static power. Although high-level synthesis tools provide optimization techniques such as resource sharing and pipelining to improve performance, these optimizations are outside the scope of this work.

Table 1. Implementation results of the proposed Harris corner detector

Resource	Utilization	Available	%
LUT	8132	53200	15.92
LUT RAM	7	17400	0.04
Flip-Flops	4054	106400	3.81
BRAM	46	140	32.86
DSP	5	220	2.27
I/O	35	200	17.50
BUFG	14	32	12.5

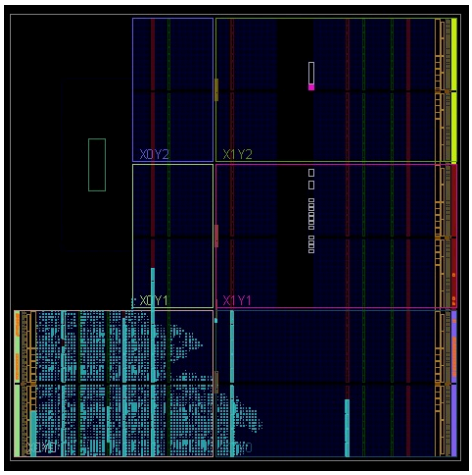

Figure 7. Floorplan of proposed Harris design implemented in an FPGA

Figure 7 illustrates the floor planning design implementation for the Xilinx Zedboard, performed using the VIVADO Design Suite.

5.3 Comparison results

The synthesis results of our proposed algorithm were compared with those of other algorithms. As shown in Table 2, our implementation demonstrates superior area efficiency for the same target FPGA compared to other solutions. Specifically, our design utilized 46 BRAM, while Chao and Wong [30] used 64 BRAM. Additionally, in terms of input/output performance, our implementation outperforms others due to reduced design complexity. This is attributed to shorter critical paths, resulting in a higher operational frequency on the FPGA.

Other authors have not shared power dissipation data; although, various implementations of the Harris corner detector have been developed on FPGA platforms, and in this study, we compare the results of our synthesis of the proposed algorithm with those of other reported algorithms, as shown in Table 2. The design presented in study [40] requires 69 BRAMs, 4131 flip-flops, and 110 DSPs. The design in study [41] requires 37 BRAMs, 506 LUTRAMs, 10981 flipflops, 21 DSPs, 102 I/Os, and 1 BuFG.

Table 2. Proposed Harris corner detector implementation results

Resource	Proposed	[41]	[40]	[42]	[30]	[33]
LUT	8132	5917	9485	17555	NA	8425
LUT RAM	7	506	NA	5443	1400	NA
Flip-Flops	4054	10981	4131	2337	NA	4252
BRAM	46	37	69	75	64	11
DSP	5	21	110	55	110	44
I/O	35	102	NA	48	NA	NA
BUFG	4	1	NA	7	NA	4

LUT: Look Up Table

Finally, the design in study [33] requires 11 BRAMs, 8425 LUTs, 4252 flip-flops, 44 DSPs, and 4 BuFGs. In contrast, our proposed IP requires approximately 46 BRAMs, 8132 LUTs, 7 LUTRAMs, 35 INPUT/OUTPUT, 5 DSPs, and 4252 flip-flops, with a power consumption of 0.329W. Based on this comparison, our hardware architecture achieves high throughput performance, albeit with relatively large hardware resources compared to other implementations.

The comparison between our FPGA-based Harris corner detection system and the approaches in the cited works highlights different trade-offs in terms of performance, accuracy, and resource utilization. In study [35], the FAST-C corner detection algorithm focuses on achieving high throughput, supporting 4K resolution at 60 frames per second through the use of SRAM and interleaved memory access. This approach sacrifices some accuracy for speed, whereas our Harris implementation emphasizes precision in corner detection at a lower resolution (640x480), with a focus on maintaining high accuracy while still achieving real-time processing. In study [36], the authors propose an FPGA-based FAST detector that optimizes area-time efficiency by using data-path unrolling and logic folding to reduce critical path delay and resource usage. While this design excels in minimizing resource consumption, our Harris implementation leverages the Zedboard's parallel processing capabilities to strike a balance between computational efficiency and detection accuracy, albeit with higher computational complexity. Finally, the method in study [37] uses low-complexity pruning to speed up corner detection but risks missing corners in highly textured images, contrary to , our Harris corner detector which maintains robustness across a wide range of image textures, ensuring accurate corner detection, though at the cost of slightly increased computational load. Overall, our approach distinguishes itself by achieving low power consumption (0.33W) and real-time performance while maintaining high corner detection accuracy, making it ideal for embedded systems with stringent performance and power requirements.

6. CONCLUSION AND FUTURE WORK

Our study successfully implemented the Harris Corner Detection algorithm on an FPGA, achieving a real-time processing speed of 102 frames per second while maintaining low power consumption at 0.33W. This highlights the efficiency of using FPGA-based hardware for corner detection in real-time image processing applications, especially where power efficiency and speed are critical. Moreover, the design demonstrated resource efficiency, utilizing fewer hardware resources such as BRAMs and DSPs compared to other implementations in the literature, while still achieving high detection accuracy. These findings underscore the potential of our FPGA-based system for deployment in embedded vision applications, such as robotics, medical imaging, and autonomous systems, where real-time performance and low power consumption are essential.

Future work, one immediate direction will be to optimize the design for higher-resolution images, such as 1080p or 4K, by refining memory management and parallel processing to maintain real-time performance. Additionally, while our implementation has proven accurate, we observed some sensitivity to noise, prompting us to explore the integration of noise-reduction techniques, such as adaptive thresholding, to improve robustness in noisy environments. Further, we plan to extend our work to include other corner detection algorithms, like FAST, SIFT, and SURF, to benchmark their performance on the same FPGA platform, enabling broader comparisons and applications. We also intend to explore the deployment of our system on edge devices with constrained resources, potentially targeting low-cost FPGA platforms or FPGA-SoC

architectures for even more power- and cost-sensitive applications. Finally, applying this system to real-world scenarios such as autonomous vehicle navigation and medical image analysis will allow us to validate its performance in dynamic environments and assess its practical benefits for end users.

We have also highlighted the limitations of our current implementation. In terms of scalability, while our design performs efficiently with images of 640×480 resolution, processing higher-resolution images such as 1080p or 4K may necessitate further optimization of the memory architecture and processing pipeline to ensure real-time performance. Additionally, like other corner detection algorithms, our implementation is somewhat sensitive to image noise, leading to potential false positives in certain scenarios. To address this, future work will focus on integrating noise-reduction techniques to enhance the algorithm's robustness.

ACKNOWLEDGMENT

The authors would like to acknowledge Deanship of Graduate Studies and Scientific Research, Taif University for funding this work.

REFERENCES

[1] Chi, B., Yu, M., Jiang, G., He, Z., Peng, Z., Chen, F. (2020). Blind tone mapped image quality assessment with image segmentation and visual perception. Journal of Visual Communication and Image Representation, 67: 102752. https://doi.org/10.1016/j.jvcir.2020.102752

[2] Li, K., Yu, L., Heng, P.A. (2022). Towards reliable cardiac image segmentation: Assessing image-level and pixel-level segmentation quality via self-reflective references. Medical Image Analysis, 78: 102426. https://doi.org/10.1016/j.media.2022.102426

[3] Sagheer, S.V.M., George, S.N. (2020). A review on medical image denoising algorithms. Biomedical Signal Processing and Control, 61: 102036. https://doi.org/10.1016/j.bspc.2020.102036

[4] Ben Abdallah, M., Malek, J., Azar, A.T., Montesinos, P., Belmabrouk, H., Esclarín Monreal, J., Krissian, K. (2015). Automatic extraction of blood vessels in the retinal vascular tree using multiscale medialness. International Journal of Biomedical Imaging, 2015(1): 519024. https://doi.org/10.1155/2015/519024

[5] Abdallah, M.B., Malek, J., Tourki, R., Monreal, J.E., Krissian, K. (2013). Automatic estimation of the noise model in fundus images. In 10th International Multi-Conferences on Systems, Signals & Devices 2013 (SSD13), Hammamet, Tunisia, pp. 1-5. https://doi.org/10.1109/SSD.2013.6564014

[6] Meenpal, T., Balakrishnan, A., Verma, A. (2019). Facial mask detection using semantic segmentation. In 2019 4th International Conference on Computing, Communications and Security (ICCCS), Rome, Italy, pp. 1-5. https://doi.org/10.1109/CCCS.2019.8888092

[7] Benini, S., Khan, K., Leonardi, R., Mauro, M., Migliorati, P. (2019). Face analysis through semantic face segmentation. Signal Processing: Image Communication, 74: 21-31. https://doi.org/10.1016/j.image.2019.01.005

[8] Jiang, S., Cui, R., Wei, R., Fu, Z., Hong, Z., Feng, G. (2023). Tracking by segmentation with future motion estimation applied to person-following robots. Frontiers in Neurorobotics, 17: 1255085. https://doi.org/10.3389/fnbot.2023.1255085

[9] Zhang, J., Yang, P., Wang, W., Hong, Y., Zhang, L. (2020). Image editing via segmentation guided self-attention network. IEEE Signal Processing Letters, 27: 1605-1609. https://doi.org/10.1109/LSP.2020.3022289

[10] Akbari, M., Liang, J., Han, J. (2019). DSSLIC: Deep semantic segmentation-based layered image compression. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, pp. 2042-2046. https://doi.org/10.1109/ICASSP.2019.8683541

[11] Liu, Z., Meng, L., Tan, Y., Zhang, J., Zhang, H. (2021). Image compression based on octave convolution and semantic segmentation. Knowledge-Based Systems, 228: 107254. https://doi.org/10.1016/j.knosys.2021.107254

[12] Wadhwa, A., Bhardwaj, A., Verma, V.S. (2019). A review on brain tumor segmentation of MRI images. Magnetic Resonance Imaging, 61: 247-259. https://doi.org/10.1016/j.mri.2019.05.043

[13] Jiang, H., Diao, Z., Yao, Y.D. (2022). Deep learning techniques for tumor segmentation: A review. The Journal of Supercomputing, 78(2): 1807-1851. https://doi.org/10.1007/s11227-021-03901-6

[14] Abdolali, F., Zoroofi, R.A., Otake, Y., Sato, Y. (2016). Automatic segmentation of maxillofacial cysts in cone beam CT images. Computers in Biology and Medicine, 72: 108-119. https://doi.org/10.1016/j.compbiomed.2016.03.014

[15] Malek, J., Tourki, R. (2013). Inertia-based vessel centerline extraction in retinal image. In 2013 International Conference on Control, Decision and Information Technologies (CoDIT), Hammamet, Tunisia, pp. 378-381. https://doi.org/10.1109/CoDIT.2013.6689574

[16] Alaeddine, H., Jihene, M. (2023). Wide deep residual networks in networks. Multimedia Tools and Applications, 82(5): 7889-7899. https://doi.org/10.1007/s11042-022-13696-0

[17] Kaur, A., Kumar, M., Jindal, M.K. (2022). Shi-Tomasi corner detector for cattle identification from muzzle print image pattern. Ecological Informatics, 68: 101549. https://doi.org/10.1016/j.ecoinf.2021.101549

[18] Bansal, M., Kumar, M., Kumar, M., Kumar, K. (2021). An efficient technique for object recognition using Shi-Tomasi corner detection algorithm. Soft Computing, 25(6): 4423-4432. https://doi.org/10.1007/s00500-020-05453-y

[19] Jian, C., Xiang, X., Zhang, M. (2019). Mobile terminal gesture recognition based on improved FAST corner detection. IET Image Processing, 13(6): 991-997. https://doi.org/10.1049/ietipr.2018.5959

[20] Chen, S., Zhong, S., Xue, B., Li, X., Zhao, L., Chang, C.I. (2020). Iterative scale-invariant feature transform for remote sensing image registration. IEEE Transactions on Geoscience and Remote Sensing, 59(4): 3244-3265. https://doi.org/10.1109/TGRS.2020.3008609

[21] Kasiselvanathan, M., Sangeetha, V., Kalaiselvi, A. (2020). Palm pattern recognition using scale invariant feature transform. International Journal of Intelligence

and Sustainable Computing, 1(1): 44-52. https://doi.org/10.1504/IJISC.2020.104826

[22] Jagadeeswari, M., Manikandababu, C.S., Aiswarya, M. (2022). Integral images: Efficient algorithms for their computation systems of speeded-up robust features (Surf). In Pervasive Computing and Social Networking: Proceedings of ICPCSN 2021, pp. 663-672. https://doi.org/10.1007/978-981-16-5640-8_50

[23] Arora, P., Mehta, R., Ahuja, R. (2024). An adaptive medical image registration using hybridization of teaching learning-based optimization with affine and speeded up robust features with projective transformation. Cluster Computing, 27(1): 607-627. https://doi.org/10.1007/s10586-023-03974-3

[24] Vasco, V., Glover, A., Bartolozzi, C. (2016). Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea (South), pp. 4144-4149. https://doi.org/10.1109/IROS.2016.7759610

[25] Glover, A., Dinale, A., Rosa, L.D.S., Bamford, S., Bartolozzi, C. (2021). Luvharris: A practical corner detector for event-cameras. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(12): 10087-10098. https://doi.org/10.1109/TPAMI.2021.3135635

[26] Haggui, O., Tadonki, C., Lacassagne, L., Sayadi, F., Ouni, B. (2018). Harris corner detection on a NUMA manycore. Future Generation Computer Systems, 88: 442-452. https://doi.org/10.1016/j.future.2018.01.048

[27] Fredj, A.H., Malek, J. (2017). GPU-based anisotropic diffusion algorithm for video image denoising. Microprocessors and Microsystems, 53: 190-201. https://doi.org/10.1016/j.micpro.2017.08.003

[28] Hadj Fredj, A., Malek, J. (2021). A fast and robust osrad filter for telemedicine applications. International Journal of Computers and Applications, 43(1): 70-79. https://doi.org/10.1080/1206212X.2018.1512235

[29] Hadj Fredj, A., Malek, J. (2021). FPGA-accelerated anisotropic diffusion filter based on SW/HW-codesign for medical images. Journal of Real-Time Image Processing, 18(6): 2429-2440. https://doi.org/10.1007/s11554-021-01100-3

[30] Chao, T.L., Wong, K.H. (2015). An efficient FPGA implementation of the Harris corner feature detector. In 2015 14th IAPR International Conference on Machine Vision Applications (MVA), Tokyo, Japan, pp. 89-93. https://doi.org/10.1109/MVA.2015.7153140

[31] Mestiri, H., Barraj, I., Machhout, M. (2021). AES high-level SystemC modeling using aspect oriented programming approach. Engineering, Technology & Applied Science Research, 11(1): 6719-6723. https://doi.org/10.48084/etasr.3971

[32] Rafiammal, S.S., Jamal, D.N., Mohideen, S.K. (2020). Reconfigurable hardware design for automatic epilepsy seizure detection using EEG signals. Engineering, Technology & Applied Science Research, 10(3): 5803-5807. https://doi.org/10.48084/etasr.3419

[33] Alsheikhy, A., Yahia, F.S. (2019). Design of embedded vision system based on FPGA-SoC. International Journal of Advanced Computer Science and Applications, 10(10): 91-98. https://doi.org/10.14569/IJACSA.2019.0101013

[34] Gyaneshwar, D., Nidamanuri, R.R. (2022). A real-time FPGA accelerated stream processing for hyperspectral image classification. Geocarto International, 37(1): 52-69. https://doi.org/10.1080/10106049.2020.1713231

[35] Lee, Y.H., Chen, T.C., Liang, H.C., Liao, J.X. (2020). Algorithm and architecture design of FAST-C image corner detection engine. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 29(4): 788-799. https://doi.org/10.1109/TVLSI.2020.3031294

[36] Lam, S.K., Lim, T.C., Wu, M., Cao, B., Jasani, B.A. (2019). Data-path unrolling with logic folding for area-time-efficient FPGA-based FAST corner detector. Journal of Real-Time Image Processing, 16: 2147-2158. https://doi.org/10.1007/s11554-017-0725-0

[37] Ramakrishnan, N., Wu, M., Lam, S.K., Srikanthan, T. (2016). Enhanced low-complexity pruning for corner detection. Journal of Real-Time Image Processing, 12: 197-213. https://doi.org/10.1007/s11554-014-0396-z

[38] Zedboard. Avnet. http://www.zedboard.org/overview-zedboardkit.

[39] OmniVision. version 1.4. OmniVision Technologies, Inc, 2006. https://www.ovt.com/press-releases/omnivision-launches-seventh-generation-vga-camerachip-for-mobile-applications/.

[40] Komorkiewicz, M., Kryjak, T., Chuchacz-Kowalczyk, K., Skruch, P., Gorgoń, M. (2015). FPGA based system for real-time structure from motion computation. In 2015 Conference on Design and Architectures for Signal and Image Processing (DASIP), Krakow, Poland, pp. 1-7. https://doi.org/10.1109/DASIP.2015.7367241

[41] Sikka, P., Asati, A.R., Shekhar, C. (2021). Real time FPGA implementation of a high speed and area optimized Harris corner detection algorithm. Microprocessors and Microsystems, 80: 103514. https://doi.org/10.1016/j.micpro.2020.103514

[42] Liu, S., Lyu, C., Liu, Y., Zhou, W., Jiang, X., Li, P., Li, Y. (2017). Real-time implementation of Harris corner detection system based on FPGA. In 2017 IEEE International Conference on Real-time Computing and Robotics (RCAR), Okinawa, Japan, pp. 339-343. https://doi.org/10.1109/RCAR.2017.8311884