VLSI Implementation of Iris Localization Using Circular Hough Transform

Akshinth M
Dept of Electronics and Communication Engineering, St.Xavier's Catholic College of Engineering,
Kanyakumari, Tamil Nadu.
akshithak91@gmail.com

Antony Ajith J V
Dept of Electronics and Communication Engineering, St.Xavier's Catholic College of Engineering,
Kanyakumari, Tamil Nadu.
antonyajithvj@gmail.com

Jerwin N H
Dept of Electronics and Communication Engineering, St.Xavier's Catholic College of Engineering,
Kanyakumari, Tamil Nadu.
nhjerwin123@gmail.com

Dr. S Caroline
Assistant Professor, Dept of Electronics and Communication Engineering, St.Xavier's Catholic College of Engineering,
Kanyakumari, Tamil Nadu.
caroline@sxcce.edu.in

***Abstract*—Deep learning-based models such as the interactive variant of U-Net and deep multi-task attention networks, while powerful, are computationally intensive, require large-scale annotated datasets and involve complex parameter tuning—factors that limit their practicality for fundamental tasks like edge detection. In contrast, classical methods such as Canny edge detection and the Circular Hough Transform offer efficient, robust and interpretable alternatives. Canny edge detection provides noise resilience, sub-pixel accuracy and thin edge localization, making it well-suited for object recognition and segmentation. The Circular Hough Transform is highly effective in detecting circular patterns with varying radii and orientations, finding broad applications in medical imaging, iris recognition, industrial inspection and robotics. These classical techniques continue to serve as essential tools for feature extraction, shape analysis and object tracking across diverse domains, especially where computational efficiency and reliability are paramount.**

***Keywords* - MATLAB, Xilinx Vivado, Gaussian Smoothing, Canny Edge Detection, Circular Hough Transform(CHT), Very Large Scale Integration(VLSI), Verilog HDL(Hardware Description Language).**

I. INTRODUCTION

Biometric authentication has emerged as a cornerstone of modern security infrastructures, offering reliable and accurate means of personal identification. Among the suite of biometric modalities, iris recognition stands out due to the unique and stable nature of iris patterns [1][4]. In contrast to fingerprints, which may degrade over time or facial recognition, which is influenced by expression and aging, the iris remains largely unchanged throughout a person's life. This inherent stability, combined with the high distinctiveness of iris patterns, positions iris recognition as a preferred choice for high-security applications such as national identification systems, border control, financial services and access management [7][25].

A pivotal component of iris recognition is iris segmentation [2], the process of isolating the iris region from a captured eye image. The accuracy of this step directly affects subsequent stages of feature extraction and pattern matching. However, segmentation is frequently challenged by factors including variable lighting, occlusions from eyelids or eyelashes, reflections and image noise [5][8][16]. These challenges necessitate the development of robust and efficient segmentation techniques that can maintain performance under non-ideal imaging conditions [10][15].

II. MOTIVATION

Conventional iris segmentation algorithms, typically implemented in software, often rely on iterative and computationally intensive methods [14]. These approaches are not optimal for real-time applications, particularly when deployed on resource-constrained or embedded systems. Moreover, environmental factors such as uneven illumination, shadowing and motion blur further degrade segmentation accuracy [6][17].

To address these limitations, this work proposes a hardware-accelerated iris segmentation technique combining Canny Edge Detection with the Circular Hough Transform (CHT) [18]. This method is tailored for implementation in VLSI (Very Large Scale Integration) hardware, facilitating real-time operation while maintaining segmentation accuracy. The edge detection stage enhances boundary features and suppresses noise, while the CHT robustly detects circular structures corresponding to the iris and pupil boundaries [3][19]. Together, these algorithms offer a reliable and computationally efficient solution, suitable for challenging image conditions.

III. PROPOSED SYSTEM

The proposed system introduces a hardware-accelerated architecture for iris localization, specifically designed to support biometric authentication applications requiring real-time performance and high computational efficiency. The focus is directed toward translating the iris segmentation pipeline into a VLSI-compatible architecture, with particular emphasis on implementation using Field-Programmable Gate Arrays (FPGAs) [12][14]. This approach enables parallel processing and low-latency operation, making it suitable for integration into embedded vision systems where speed and accuracy are critical [13][20].

The architecture consists of three primary image processing stages: Gaussian Smoothing, Canny Edge Detection and Circular Hough Transform—each designed to perform specific tasks in the iris localization pipeline [21][22]. These modules are implemented in Verilog HDL and simulated using Vivado Design Suite. The Gaussian Smoothing module is responsible for noise reduction, enhancing the quality of edge detection by smoothing out fine-grain artifacts in the

image [11]. The Canny Edge Detection module includes gradient calculation, non-maximum suppression and double thresholding with hysteresis tracking, ensuring robust and precise edge detection under varying illumination and noise conditions [23]. The processed edge image is then fed into the Circular Hough Transform module, which votes in parameter space to detect circular boundaries that correspond to the iris [25].

To enable FPGA-based simulation and testing, image data is first preprocessed and converted into a text format using MATLAB, then stored in Block RAM (BRAM). Each stage of the pipeline reads from and writes to separate BRAM blocks, enabling a modular and pipelined approach. Finite State Machines (FSMs) are used for control flow within each module, ensuring that the pipeline operates in a synchronized and deterministic manner.The FPGA implementation allows parallel data processing and hardware-level optimization, significantly reducing processing latency compared to software-based approaches. Simulation results validate the correctness of each module, with visual outputs confirming accurate iris boundary detection. The system's ability to process image frames efficiently confirms its feasibility for real-time biometric systems such as access control, surveillance and identity verification.

By leveraging the parallelism inherent in FPGA architectures, the proposed system bridges the gap between software simulations and high-performance embedded systems. This design lays the foundation for future ASIC implementation, aligning with the objectives of energy-efficient, scalable and real-time iris recognition systems. This effort is part of the broader national initiative under the "Chips to Startup" (C2S) program funded by MeitY, Government of India, which supports indigenous VLSI design and innovation.

IV. BLOCK DIAGRAM

The proposed VLSI architecture for iris localization using the Circular Hough Transform (CHT), as shown in Figure 1, is designed to efficiently detect iris boundaries in digital images by leveraging optimized hardware processing [12]. The process starts with converting a 320×240 input image into a binary text file using MATLAB, generating 76,800 pixel values (each 8-bit), which are then instantiated into RAM for initial storage [13]. To enhance edge detection accuracy, a Gaussian Smoothing Module applies a filter to reduce noise [11], improving feature clarity and the smoothed image is temporarily stored in Smoothed Image BRAM. The Canny Edge Detection Module processes this data to detect significant intensity changes, marking object boundaries, with results stored in Edge Image BRAM.

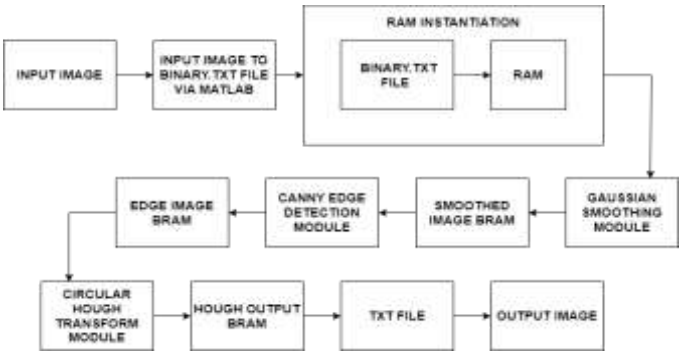


Fig. 1: VLSI architecture of Iris localization using CHT

The Circular Hough Transform Module then analyzes the edge-detected data, mapping it into a parameter space to identify circular features, particularly the iris boundary, by accumulating evidence over possible center coordinates and radii, with detected parameters stored in the Hough Output BRAM. A Txt File Module writes the processed CHT data, including detected circle parameters, to a text file for verification, further processing or visualization. Finally, the system generates an Output Image with the detected iris boundaries, making it suitable for biometric authentication and further analysis in identification systems.

A. *Gaussian Smoothing*

The Gaussian Smoothing Module, illustrated in Figure 2, plays a critical role in the preprocessing stage of the proposed architecture by reducing high-frequency noise and enhancing the overall quality of the input image prior to edge detection. It performs a Gaussian blur operation, wherein each pixel value is replaced by a weighted average of its neighboring pixels based on a Gaussian distribution. This process suppresses insignificant image details and emphasizes key features relevant to edge localization.

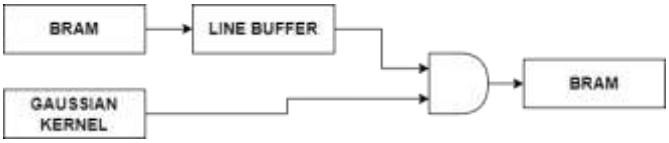


Fig. 2: Block diagram of Gaussian smoothing

The input image is initially stored in an Input Block RAM (BRAM), which provides structured access to pixel data. A line buffer is employed to hold multiple rows of the image concurrently, enabling efficient application of the Gaussian kernel across the pixel matrix [11][13]. The Gaussian kernel, implemented as a fixed-size convolution mask, assigns weights to neighboring pixels based on their Euclidean distance from the center pixel, with values following the Gaussian function.

Convolution between the kernel and the buffered pixel data results in a smoothed image, where each output pixel reflects a noise-reduced, context-aware intensity. The filtered image is subsequently stored in an Output BRAM, ready for input to the edge detection module. This hardware-based implementation facilitates real-time image processing with minimal latency, making it suitable for high-performance

embedded vision applications [12][20].

B. *Canny Edge Detection*

The Canny Edge Detection Module, as represented in Figure 3, is a crucial component in the proposed architecture for iris localization using the Circular Hough Transform, responsible for detecting edges in the preprocessed image to accurately identify the iris boundaries [23]. It operates through a structured sequence of steps that progressively refine the edge detection process, ensuring high precision and reliability. By analyzing intensity variations, the module effectively identifies significant edges while minimizing noise and false detections. This systematic approach enhances the accuracy of iris localization, forming a fundamental step in the overall biometric recognition process.

Fig. 3: Block diagram of canny edge detection

Gradient computation:

The Canny Edge Detection process begins with gradient computation, where Sobel kernels are applied to calculate intensity changes in the image, as depicted in Figure 4. The gradient computation in the proposed architecture involves convolving the input image with two distinct Sobel kernels to obtain gradient values in the x and y directions, respectively.

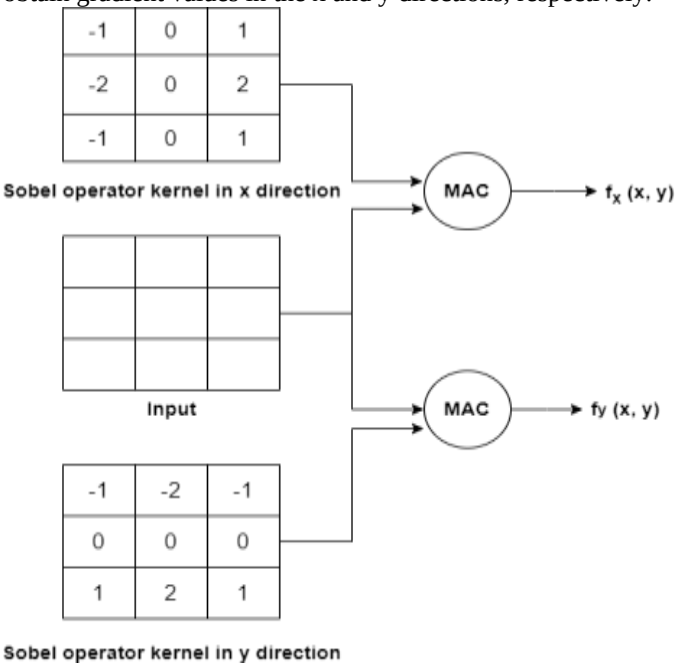


Fig. 4: Gradient computation

This module is implemented using Multiply-Accumulate (MAC) units, which perform convolution operations by calculating the dot product of the Sobel kernels and the corresponding pixel neighborhood for each pixel in the image. The resulting gradient values represent the rate of intensity change along both axes. To optimize convolution efficiency, a sliding window approach is employed, where a 3×3 window moves across the image, allowing MAC units to process pixel values with the Sobel kernels. A line buffer temporarily holds the necessary image data, ensuring that required pixel values are readily available for processing. These computed gradients form the foundation for subsequent calculations in the Canny Edge Detection process, playing a crucial role in determining gradient magnitude and direction, which are essential for edge refinement and thresholding. Accurate gradient computation directly impacts the quality of edge detection and, consequently, the precision of iris localization. The computed gradient values are stored in Block RAM (BRAM), which provides fast and efficient storage, ensuring quick data access for subsequent edge detection and transformation stages [11][23].

Let I(x,y) be the intensity of the image at pixel coordinates (x,y). Calculating the horizontal gradient Gx and vertical gradient Gy

$$Gx = I * Kx \quad (1)$$

$$Gy = I * Ky \quad (2)$$

Where,

Gx and Gy are horizontal and vertical gradients.

Kx and Ky are Kernels.

Equations (1) and (2) compute the horizontal and vertical gradient by convolving with a pair of filters, as represented in Figure 4.

Gradient magnitude:

To compute the gradient magnitude, the absolute values of the horizontal and vertical gradients are first determined, ensuring a non-negative representation of edge strength [23]. The maximum and minimum of these absolute values help approximate the magnitude while indicating the dominant gradient direction, as illustrated in Figure 5.

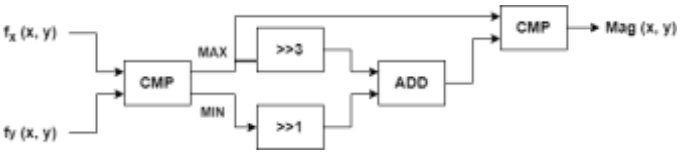


Fig. 5: Gradient Magnitude

For hardware efficiency, bitwise shift operations replace division: the maximum value is right-shifted by 3 bits (divided by 8) and the minimum by 1 bit (divided by 2). This reduces computational complexity and conserves resources. The shifted values are then summed to produce the approximate gradient magnitude, which is clamped to 255 to remain within the 8-bit range. The result is stored in the gradient buffer, a critical memory component used for further stages like non-maximum suppression and edge tracking, ensuring accurate and high-performance edge detection.

$$M(x,y) = \sqrt{Gx2 + G\,2} \quad (3)$$

where,

M(x,y) is magnitude

Equation (3) computes the gradient magnitude at each pixel, as visualized in Figure 5.

Gradient direction:

The gradient direction provides critical information about edge orientation, which is essential for the accurate detection and localization of iris boundaries [23], as shown in Figure 6. This process involves computing the angle of the gradient vector at each pixel using the previously determined horizontal (Gx) and vertical (Gy) gradient components.

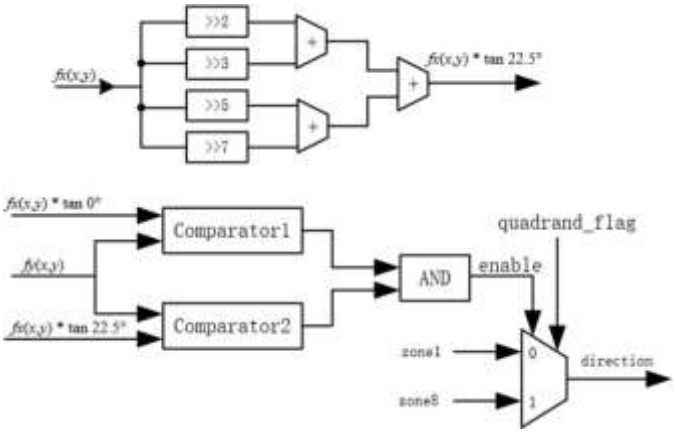


Fig. 6: Gradient direction

Initially, the algorithm checks if both $G_x = 0$ and $G_y = 0$. If true, the pixel is deemed to have no significant edge and a default direction value of zero is assigned in the direction buffer. For non-zero gradient values, the direction is quantized into one of four principal angles—0°, 45°, 90° or 135°—to balance accuracy with hardware simplicity.

- If $G_x > 0$ and $G_y \geq 0$, the direction is assigned as 0°, indicating a horizontal edge.
- If $G_x \leq 0$ and $G_y > 0$, the direction is approximated as 45°, corresponding to a diagonal edge from the lower-left to the upper-right.
- If $G_x < 0$ and $G_y \leq 0$, the direction is set to 90°, representing a vertical edge.
- If $G_x < 0$ and $G_y > 0$, the direction is classified as 135°, denoting a diagonal edge from the upper-left to the lower-right.

These discrete direction values are encoded and stored in the direction buffer for efficient access in subsequent processing stages such as non-maximum suppression and edge tracking by hysteresis.

$$\theta(x,y) = \tan{-1}(Gy/Gx) \quad (4)$$

where,

θ(x,y) is direction

Equation (4) computes the gradient direction at each pixel, as visualized in Figure 6.

Non maximum suppression:

The proposed architecture for the non-maximum suppression stage in the Canny edge detection algorithm is designed to enhance edge thinning, ensuring that only the most significant edges are preserved in the final image, as shown in Figure 7.

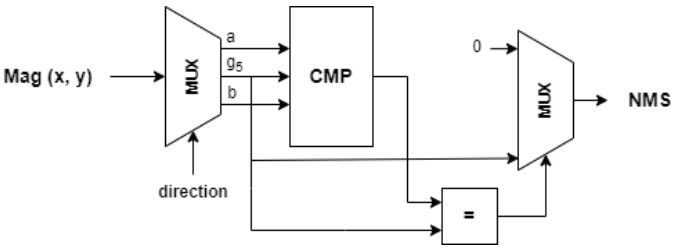


Fig. 7: Non maximum suppression

The process begins with an initial multiplexer that directs the input gradient magnitude data from the edge detection stage into the appropriate channels, selectively routing data to the comparator based on gradient direction to ensure correct neighboring pixel comparisons. The core component, the comparator, evaluates the gradient magnitudes of the current pixel against its neighbors along the gradient direction, determining whether the current pixel is a local maximum. Following this, a second multiplexer channels the data based on the comparison results, allowing only potential local maxima to proceed while suppressing non-maximum pixels [23]. An equality checker then performs the final validation step, ensuring that the current pixel's gradient magnitude equals the maximum value determined by the comparator, thereby retaining only true edge pixels. This architecture, utilizing multiplexers, comparators and an equality checker, effectively thins edges by accurately comparing gradient magnitudes and selectively preserving maximum values, enhancing the precision of edge detection and ensuring that only the most significant edges are maintained in the final output image.

High and low threshold computation:

Thresholds are computed in hardware (Figure 8) by first combining the gradient magnitudes with an XOR gate and then scaling them down via >>2 and >>1 bit-shifts. A selector picks the most significant scaled values, and a sorter arranges them so the high threshold can be set at a chosen percentile of the maximum gradient. Finally, a simple multiplier derives the low threshold as a fixed fraction of that high threshold. The entire datapath is fully pipelined—new gradient values can enter every clock cycle, with only a few cycles of fixed latency—ensuring continuous, high-throughput operation. Because all operations are fixed-point combinational logic, the design occupies minimal FPGA resources (LUTs and registers) and avoids the power and area overhead of iterative,

memory-based methods. Control registers allow the high-threshold percentile and low-threshold fraction to be updated at runtime, enabling adaptive threshold tuning under varying image conditions. This efficient, low-latency pipeline delivers precise thresholds without any software intervention, making it ideal for real-time embedded edge detection [14].

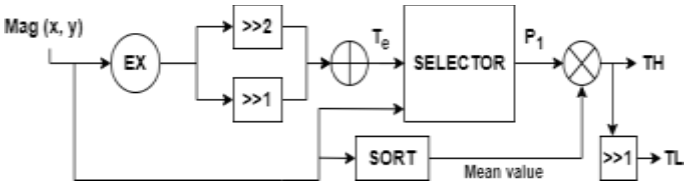


Fig. 8: High and low threshold computation

In the edge detection process, thresholding is applied to the gradient magnitude to classify pixels into three categories:

- Strong Edges: Pixels for which are considered strong edge pixels.
- Weak Edges: Pixels with gradient magnitudes such that are classified as weak edges.
- Non-Edges: Pixels where are treated as non-edge pixels and are discarded.

Hysteresis thresholding:

Hysteresis thresholding is an essential step for refining edge detection results, ensuring the retention of meaningful edges while eliminating noise, as shown in Figure 9.

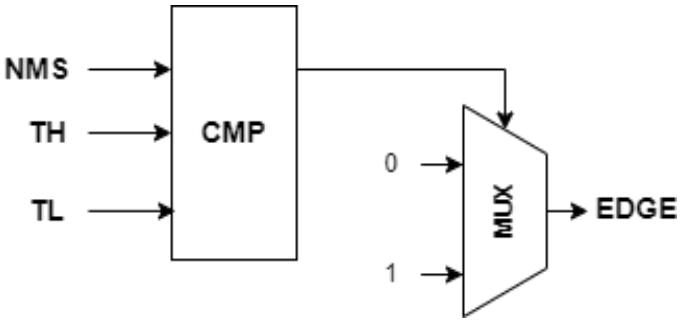


Fig. 9: Hysteresis thresholding

The architecture comprises two primary components: the Comparator (CMP) block and the Multiplexer (MUX) block. The CMP block classifies pixels based on their gradient magnitudes by comparing each pixel against two predefined thresholds: a high threshold (TH) and a low threshold (TL). Pixels with gradient magnitudes above TH are classified as strong edges, while those below TL are discarded as non-edges and those between TH and TL are marked as potential weak edges. The MUX block then performs edge tracking by hysteresis, evaluating the connectivity of weak edge pixels to strong edge pixels [18][24]. A weak edge pixel is retained only if connected to a strong edge pixel; otherwise, it is discarded. This process ensures the continuity of edge contours, which is crucial for accurate iris localization by preserving significant edges necessary for detecting the circular boundaries of the iris. By implementing this architecture, the system ensures precise and reliable edge detection, leading to improved iris localization using the Circular Hough Transform.

c. *Circular Hough Transform*

The VLSI architecture for the Circular Hough Transform (CHT) is designed to efficiently detect circular features in digital images using specialized hardware modules, as shown in Figure 10. The algorithm transforms edge points in the image space into a parameter space, wherein circular patterns emerge as distinct peaks. For each edge point in the edge-detected image, the architecture iteratively explores combinations of potential circle parameters—center coordinates and radius . Each combination is evaluated using the circle equation:

$$(x - a)2 + (y - b)2 = r2 \quad (5)$$

Equation (5) represents a circle centered at a radius.

The CHT pipeline comprises several key stages: edge detection preprocessing, parameter space generation, accumulator voting, peak detection and circle validation.

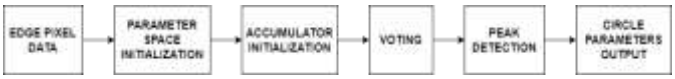


Fig. 10: Block diagram of Circular Hough Transform(CHT)

In the proposed hardware architecture, edge pixels are first extracted from the input image and forwarded to the radius and center calculation modules. The radius calculation module generates a range of possible radii based on application-specific constraints. The center calculation module computes potential circle centers through trigonometric operations, leveraging arithmetic components such as adders, multipliers and CORDIC (Coordinate Rotation Digital Computer) units for efficient sine and cosine evaluations [19].The voting module utilizes these parameters to update a three-dimensional accumulator array that represents the space. Each cell in this array stores the vote count corresponding to the likelihood of a circle passing through the given parameters [3]. After voting, the peak detection module identifies local maxima using comparators and thresholding, pointing to probable circle candidates [25]. Subsequently, the circle validation module refines these results by applying constraints such as minimum vote thresholds and geometric consistency to eliminate false positives. This architecture achieves parallel and high-speed processing suitable for real-time applications, making it ideal for integration into FPGA and ASIC-based embedded vision systems.

V. EXPERIMENT RESULTS

The proposed architecture was modeled in Verilog and simulated using Xilinx Vivado Design Suite to validate the functionality and performance of the Canny Edge Detection module [13][21]. The input images were preprocessed using MATLAB and stored in Block RAM (BRAM) for use during simulation. The image used for testing was of resolution

320×240 and represented as an 8-bit grayscale format.

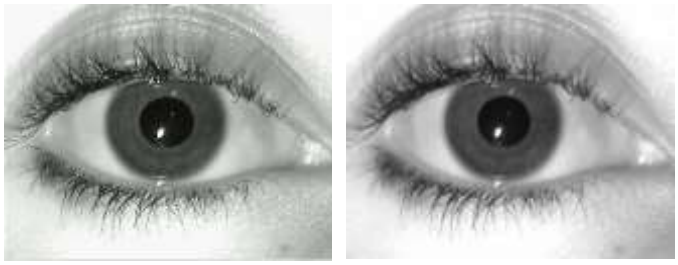
A. Simulation Setup:

The simulation environment was configured using:

- Vivado 2023.1, targeting a Xilinx Artix-7 FPGA
- Verilog HDL for hardware module implementation
- MATLAB for image-to-text conversion and result visualization

B. Gaussian Smoothing Results

The Gaussian Smoothing module successfully applied a 3×3 kernel across the image to reduce high-frequency noise [11]. This improved the clarity of edge details for the subsequent processing stage.



(a)Grayscale image (b)Gaussian Smoothing

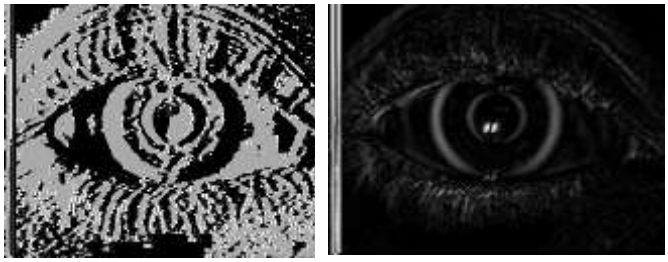
Fig. 11: Outputs of Grayscale and Gaussian smoothing.

C. Canny Edge Detection Results

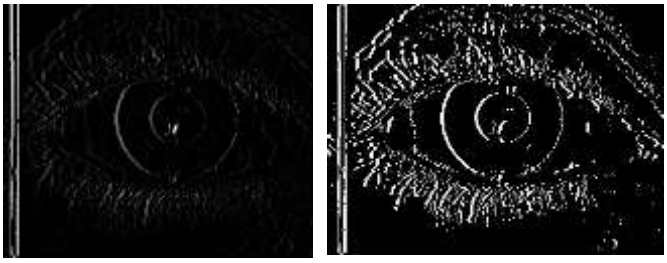
The Canny module processed the smoothed image through multiple stages including:

- Gradient Computation using Sobel kernels
- Gradient Magnitude & Direction estimation
- Non-Maximum Suppression
- Hysteresis Thresholding

Each intermediate result was validated via output files and the final edge-detected image showed clear boundary formation suitable for further processing with Circular Hough Transform.



(a)Gradient Direction (b)Gradient Magnitude



(c)Non-Maximum Suppression (d)Edge Detection

Fig. 12: Outputs of Canny edge detection

D. Circular Hough Transform (CHT) Results

The edge-detected image from the Canny module was then processed by the Circular Hough Transform module, which was implemented using a voting-based algorithm tailored for hardware efficiency [23].

(a)Circular Hough Transform (b)Localized Iris

Fig. 13: Output of Circular Hough Transform

E. Summary

Experimental validation shows our hardware pipeline (Gaussian smoothing, Canny edge detection and Circular Hough Transform) accurately and efficiently detects iris boundaries, paving the way for full biometric integration [18].

VI. CONCLUSION

This project demonstrates the successful design and implementation of a robust and efficient iris segmentation and localization system, combining advanced image processing techniques with VLSI-based hardware acceleration to meet the demands of real-time biometric authentication. The integration of Canny Edge Detection and Circular Hough Transform in the MATLAB environment [18][23] enabled precise extraction of iris boundaries, even under challenging imaging conditions. Transitioning the

algorithm to a VLSI-compatible architecture targeting FPGAs and ASICs significantly improved processing speed, power efficiency and scalability [9][22]. The proposed hardware design not only supports accurate and high-speed iris recognition but also ensures resilience to noise, occlusion and illumination variations, making it highly suitable for embedded security applications [6][24]. By bridging software-level algorithm development with hardware-level optimization, this work provides a scalable solution for next-generation biometric systems, paving the way for real-time deployment in critical security domains such as access control, identity verification and surveillance . Future work will explore full-system integration, including on-chip feature extraction and classification, to further enhance the autonomy and applicability of biometric recognition systems in real-world environments.

VII. REFERENCES

[1] Caiyong Wang, Muhammad. J, Wang. Y and Zhenan Sun (2020), 'Towards Complete and Accurate Iris Segmentation Using Deep Multi- Task Attention Network for Non-Cooperative Iris Recognition', in IEEE Transactions on Information Forensics and Security, Vol.15, pp. 2944- 2959.

[2] Caiyong Wang, Wang. Y, Boqiang Xu, Yong He, Zhiwei Dong and Zhenan Sun (2020), 'A Lightweight Multi-Label Segmentation Network for Mobile Iris Biometrics', ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, pp. 1006-1010.

[3] Chia T. Chou, Sheng-Wen Shih, Wen-Shiung Chen, Victor W. Cheng and Duan-Yu Chen, (2010), 'Non-Orthogonal View Iris Recognition System', in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 20, No. 3, pp. 417-430.

[4] Chun-Wei Tan and Ajay Kumar, (2012), 'Unified Framework for Automated Iris Segmentation Using Distantly Acquired Face Images', in IEEE Transactions on Image Processing, Vol. 21, No. 9, pp. 4068-4079.

[5] Ganeeva.Y and Myasnikov. E, (2020), 'Using Convolutional Neural Networks for Segmentation of Iris Images', 2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon), Vladivostok, Russia, pp. 1-4..

[6] Haibin Cai, Bangli Liu, Jianhua Zhang, Shengyong Chen and Honghai Liu, (2017), 'Visual Focus of Attention Estimation Using Eye Center Localization' in IEEE System Journal, VOL. 11, NO. 3, pp. 1320-1325.

[7] Hugo Proença, (2010), 'Iris Recognition: On the Segmentation of Degraded Images Acquired in the Visible Wavelength', in IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 8, pp. 1502-1516.

[8] Jasem Rahman Malgheet, Noridayu BT Manshor, Lilly Suriani Affendey and Alfian Bin Abdul Halin, (2023), 'MS-Net: Multi-Segmentation Network for the Iris Region Using Deep Learning in an Unconstrained Environment', in IEEE Access, Vol. 11, pp. 59368-59385.

[9] Jawad Muhammad, Caiyong Wang, Yunlong Wang, Kunbo Zhang and Zhenan Sun, (2023), 'IrisGuideNet: Guided Localization and Segmentation Network for Unconstrained Iris Biometrics', in IEEE Transactions on Information Forensics and Security, vol. 18, pp. 2723- 2736.

[10] Jinyu Zuo and Natalia A. Schmid, (2010), 'On a Methodology for Robust Segmentation of Nonideal Iris Images', in IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), Vol. 40, No. 3, pp. 703-718.

[11] Kai Wang and Yuntao Qian, (2011), 'Fast and accurate iris segmentation based on linear basis function and RANSAC', 18th IEEE International Conference on Image Processing, Brussels, Belgium, 2011, pp. 3205- 3208.

[12] Kamil Grabowski and A. Napieralski (2011), "Hardware Architecture Optimized for Iris Recognition," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 21, no. 9, pp. 1293-1303.

[13] Lihong Dai, Jinguo Liu, Zhaojie Ju and Yang Gao, (2020) "Iris Center Localization Using Energy Map With Image Inpaint Technology and Post-Processing Correction" in IEEE Access,Volume 8, pp. 16965-16978.

[14] Meng-ru, Lin, Shi-zhen , Huang, Fu-shan, Li, Rui-qi and Chen, Ruiqi. (2021). "Low-power Iris Recognition System Implementation on FPGA with Approximate Multiplier". 32. pp. 115-127.

[15] Mohammed A. M. Abdullah, Satnam S. Dlay, W. L. Woo and J. A. Chambers, (2017), 'Robust Iris Segmentation Method Based on a New Active Contour Force With a Noncircular Normalization', in IEEE Transactions on Systems, Man and Cybernetics: Systems, vol. 47, no. 12,pp. 3128-3141.

[16] Mousumi Sardar, Subhashis Banerjee and Sushmita Mitra, (2020), 'Iris Segmentation Using Interactive Deep Learning' in IEEE Access, Vol. 8,pp. 219322-219330.

[17] Rongnian Tang (2007), 'An effective iris location method with high robustness', Xi'an Jiaotong University, School of Electronics and Information Engineering, 28 Xianning West Road, Xi'an 710049, P.R. China Optica Applicata, Vol.XXXVII, No. 3, pp. 295-303.

[18] Samir Shah and Arun Ross, (2009), 'Iris Segmentation Using Geodesic Active Contours' in IEEE Transactions on Information Forensics and Security, Vol. 4, No. 4, pp. 824-836.

[19] Seung-Jin Baek, Kang-A Choi, Chunfei Ma, Young-Hyun Kim and Sung-Jea Ko, (2013) 'Eyeball Model-based Iris Center Localization for Visible Image-based Eye-Gaze Tracking Systems' in IEEE Transactions on Consumer Electronics, Vol. 59, No. 2, pp. 415-421.

[20] Van Thong Hu, Hyung-Jeong Yang, Guee-Sang Lee and Soo-Hyung Kim, (2020), 'Semantic Segmentation of the Eye With a Lightweight Deep Network and Shape Correction' in IEEE Access, Vol. 8, pp. 131967-131974.

[21] Wei Zhang, Xiaoqi Lu, Yu Gu, Yang Liu, Xianjing Meng and Jing Li, (2019), 'A Robust Iris Segmentation Scheme Based on Improved U-Net' in IEEE Access, Vol. 7, pp. 85082-85089.

[22] Xiaoqiang Wu and Long Zhao, (2019), 'Study on Iris Segmentation Algorithm Based on Dense U-Net' in IEEE Access, Vol. 7, pp. 123959- 123968.

[23] Ying Chen, Wang, Zhuang and Yeron Wang, (2019), 'An Adaptive CNNs Technology for Robust Iris Segmentation' IEEE Access. Vol. 7, pp. 64517-64532.

[24] Yingzi Du, Emrah Arslanturk, Zhi Zhou and Craig Belcher, (2011), 'Video-Based Noncooperative Iris Image Segmentation' in IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), Vol. 41, No. 1, pp. 64-74.

[25] Zhaofeng He, Tieniu Tan, Zhenan Sun and Xianchao Qiu, (2009), 'Toward Accurate and Fast Iris Segmentation for Iris Biometrics' in IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 31, No. 9, pp. 1670-1684.