mathematics

MDPI

Article

A Genetic Algorithm Approach for Parabolic Curve Detection Enhanced by FPGA-Based Hardware Acceleration

Francisco Javier Iñiguez-Lomeli [1,*], Valentin Flores-Payan [1], Lilia del Carmen Castillo-Villarruel [1] and Horacio Rostro-Gonzalez [2,3]

[1] Department of Basic Sciences, Bio-Technological Development Division, Ciénega University Center, University of Guadalajara, Ocotlán 47810, Jalisco, Mexico; valentin.flores@academicos.udg.mx (V.F.P.); lilia.castillo@academicos.udg.mx (L.d.C.C.-V.)
[2] GEPI Research Group, IQS, School of Engineering, Ramon Llull University, Via Augusta 390, 08017 Barcelona, Spain; horacio.rostro@iqs.url.edu
[3] Department of Electronics Engineering, DICIS-University of Guanajuato, Carretera Salamanca-Valle de Santiago, Salamanca 36885, Guanajuato, Mexico
* Correspondence: francisco.ilomeli@academicos.udg.mx

Abstract

Detecting rotated parabolic shapes in digital images remains a significant challenge in computer vision, especially in embedded environments constrained by computational and memory resources. This study introduces a novel field-programmable gate array (FPGA)-based genetic algorithm (GA) architecture specifically tailored for rotated parabola detection, implemented as an intellectual property (IP) core on a PYNQ-Z1 system-on-chip (SoC) platform. The architecture encodes four parabola parameters into fixed-length chromosomes, assesses their geometric consistency with a 640 × 480 binary edge image using a hardware fitness function, and executes the entire evolutionary process in programmable logic. Image pre-processing is executed on an external CPU, using Canny edge detection for synthetic images and Holistically Nested Edge Detection (HED). For real images, post-processing and result visualization are conducted on the ARM processor using the PYNQ framework. Experimental results on synthetic images demonstrate mean accuracies of 98.47% and 95.23%, with detection success rates of up to 96%. For real images, since manually annotated ground truth is not available, results are presented as qualitative observations of convergence consistency across 100 independent runs. These findings demonstrate the feasibility of detecting rotated parabolas on resource-constrained embedded platforms and indicate promising applications in domains where parabolic patterns are prevalent, such as structural inspection, biomedical imaging, and perception modules for autonomous vehicles and driver-assistance systems.

Keywords: parabola detection; genetic algorithm; System-on-a-Chip (SoC); High-Level Synthesis (HLS); FPGA; hardware implementation; IP core; image processing

MSC: 68M07; 68T45; 68T05

Check for updates

Academic Editor: Ioannis Tsoulos

Received: 29 April 2026
Revised: 10 June 2026
Accepted: 12 June 2026
Published: 1 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Detecting parabolic shapes in digital images remains a significant and open challenge within computer vision and image processing. The relevance of this problem spans several areas, including structural inspection, non-destructive testing, medical image analysis, and autonomous systems [1,2]. Parabolas, as conic sections, frequently occur in real-world

scenarios: they can be seen in crack tips in ultrasonic B-scan images, in parabolic trajectories observed by machine vision systems, and in anatomical features such as the plantar arch or vascular patterns in retinal fundus images [1]. Yet, reliably detecting these shapes in practical situations is not straightforward. Factors like image noise, partial occlusion, and complex backgrounds complicate the process. Moreover, the challenge is heightened by the need to estimate at least four independent parameters from the image data: the vertex coordinates (h, k), the aperture a, and the rotation angle θ [3,4].

Classical approaches to shape detection rely heavily on the Hough Transform (HT) and its generalizations [3,4]. The HT maps each edge point in image space to a curve in parameter space, and shape detection reduces to finding a peak in an accumulator array. While conceptually elegant, the HT suffers from severe computational and memory demands when extended to parabolas, since the parameter space becomes at least four-dimensional [2]. Probabilistic and randomized variants of the HT partially address these limitations but remain sensitive to quantization of the parameter space and the presence of outliers [5]. The Random Sample Consensus (RANSAC) algorithm offers an alternative robust fitting paradigm that iteratively samples minimal subsets of edge points to estimate shape parameters; however, its convergence rate and accuracy depend critically on the inlier ratio, and it may require a large number of iterations in heavily cluttered images [5].

Evolutionary and bio-inspired algorithms have become strong alternatives for image shape detection, reframing the task as a global optimization problem over the parameter space [6–8]. Genetic Algorithms (GAs) represent candidate shapes as chromosomes, where each chromosome encodes a randomly sampled edge point to set the parabola vertex, directly yielding the coordinates (h, k). The aperture a and rotation angle θ are then selected using Look-Up Tables (LUTs) that define a discrete set of feasible parabolic shapes. These candidate populations are evolved using selection, crossover, and mutation, guided by a fitness function that assesses geometric consistency with the edge map [6]. This approach has seen success in circle detection [6,7] and has also been extended to the detection of ellipses [9,10]. A particularly notable contribution comes from Guerrero-Turrubiates et al., who applied Estimation of Distribution Algorithms (EDAs), a probabilistic generalization of GAs [11,12] to identify parabolic shapes in synthetic and medical images, achieving significant reductions in execution time compared to both the generalized Hough transform and RANSAC [1]. Despite these advances in software, evolutionary algorithms still face a computational bottleneck due to the need for repeated fitness evaluations over large populations. This limitation makes them ideal candidates for hardware acceleration. Recently, the hardware acceleration of full metaheuristic loops on System-on-Chip (SoC) FPGA platforms has proven highly effective for solving complex, non-linear estimation problems in other domains. For instance, ref. [13] successfully proposed an FPGA-accelerated ϵ-greedy Ant Colony Optimization (ACO) framework for maritime target motion analysis. While such approaches demonstrate the immense potential of embedding bio-inspired algorithms in programmable logic, their application to the geometric optimization problem of detecting rotated parabolas remains unexplored.

Field-Programmable Gate Arrays (FPGAs) offer a reconfigurable, parallel computing substrate that is well suited to accelerate iterative, population-based algorithms [14–16]. The inherent parallelism of FPGAs enables simultaneous evaluation of multiple candidate solutions, while fixed-point arithmetic reduces resource consumption compared to floating-point software implementations. FPGA-based accelerators have been demonstrated for circle detection using GAs on Zynq SoC devices [14], and for ellipse detection using GAs synthesized with High-Level Synthesis (HLS) tools on the PYNQ-Z1 platform [15]. These implementations achieve accuracy rates exceeding 99% while operating under strict area and power constraints, highlighting the suitability of the GA-FPGA co-design paradigm

for embedded computer vision. FPGAs have also been deployed for real-time ellipse estimation using RANSAC in road sign detection [17] and for eye-tracking applications [18], and broader surveys confirm the growing role of FPGA accelerators in object detection tasks [16].

The PYNQ (Python Productivity for Zynq) framework provides a high-level software layer built on an embedded Linux operating system running on the ARM processing system (PS) of the Zynq SoC, enabling Python-based control and data transfer to hardware overlays executed on the programmable logic (PL) [14,19]. This hardware–software co-design approach enables rapid prototyping and validation of FPGA intellectual property (IP) cores without requiring expertise in low-level HDL design, making it an attractive platform for deploying evolutionary algorithms in embedded image processing systems [15].

Despite these advances, a critical limitation persists across most parabola-detection methods: most approaches restrict detection to non-rotated parabolas with a fixed or partially constrained aperture. For instance, although the Hough-based methods of Oloumi and Rangayyan parameterize the parabola through its vertex (h, k) and aperture a [20,21], they do not account for arbitrary orientation angles, thereby limiting detection to parabolas aligned with the image axes. Similarly, the Genetic-Based Inverse Voting Hough Transform (GBIVHT) proposed by Maalmi et al. for parabolic crack detection in ultrasonic B-scan images explicitly restricts the search space to horizontal parabolas with fixed aperture in order to maintain a tractable parameter space [22]. Even the EDA-based approach of Guerrero-Turrubiates et al., while capable of handling orientation as a parameter, reports experiments primarily on vertically oriented parabolic shapes and does not demonstrate detection of parabolas under arbitrary rotations [1]. This systematic restriction to axis-aligned or low-dimensional configurations represents a significant gap: in real-world and vision scenarios, parabolic signatures may appear at arbitrary orientations, as in crack propagation fronts, parabolic reflector cross-sections, or biological membrane curvatures. A complete parabolic model must therefore consider four independent parameters: the vertex coordinates (h, k), the aperture a, and the rotation angle θ, allowing the detection of any parabola in any orientation within the image.

On the hardware side, System-on-Chip (SoC) FPGAs that integrate ARM processing cores with reconfigurable programmable logic have emerged as the platform of choice for embedded computer vision systems requiring both algorithmic flexibility and real-time throughput [23,24]. These devices exploit the complementary strengths of software programmability on the ARM processor and spatial parallelism on the FPGA fabric: the ARM core handles image acquisition, pre- and post-processing, and result interpretation, while custom hardware accelerators on the programmable logic implement the computationally intensive inner loops of vision algorithms with deterministic, sub-millisecond latency [25]. Recently, researchers have combined genetic algorithms with FPGAs to make image filters automatically. By doing the genetic optimization and fitness checks on the FPGA, they can get the same results as software, but much faster [26]. Furthermore, design-space exploration workflows based on multi-objective genetic algorithms have been validated on Zynq-class FPGAs, confirming that GA-driven hardware optimization is not only feasible but highly efficient in terms of latency and resource utilization [27].

In this paper, we present, to the best of our knowledge, the first hardware architecture for parabola detection in digital images using a Genetic Algorithm implemented on a PYNQ SoC FPGA platform. The proposed system is designed as an IP core synthesized using Vitis High-Level Synthesis (HLS) and integrated with the Zynq-7000 SoC on the PYNQ-Z1 development board. A single edge point is randomly sampled from the binary edge image as the parabola vertex, directly providing the vertex coordinates (h, k); the aperture a and rotation angle θ of the candidate parabola are then selected from pre-computed

Look-Up Tables (LUTs) that enumerate a discrete range of feasible shapes and orientations. A fitness function measures the geometric consistency between the candidate parabola and the edge map of the image under test. The evolutionary process operates entirely on the programmable logic, while image pre-processing is performed on an external CPU for both synthetic and real images, and result visualization are handled by the ARM processor via the PYNQ Python framework. Experimental results on synthetic images demonstrate mean accuracies of 98.47% and 95.23%, with detection success rates up to 96% across independent runs. For real images, where manually annotated ground truth is not available, results are presented as qualitative observations of convergence consistency across 100 independent runs, demonstrating the feasibility of detecting rotated parabolas on resource-constrained embedded platforms.

In Section 2, we dive into the material and the methods that drive this research. We lay out the architecture's design and implementation in detail, highlighting its key components and showing how genetic algorithms are adapted to parabolic detection. In Section 3, we discuss our findings on both real and synthetic images, including challenging cases of corrupted salt-and-pepper noise images. Finally, we conclude by summarizing our key findings in Section 4.

2. Materials and Methods

This section presents the complete methodology for designing and implementing the proposed parabola detection system. The system combines a software layer running on an ARM processor with a hardware accelerator implemented on reconfigurable logic (FPGA), both integrated into a single System-on-Chip (SoC) device. The following subsections describe the hardware platform, the mathematical model of the parabola, the binary edge image representation, and the full design of the Genetic Algorithm (GA) used for detection.

2.1. Hardware Platform

The system is implemented on the PYNQ-Z1 development board (Digilent Inc., Pullman, WA, USA), Table 1, which integrates a Xilinx Zynq-7000 SoC XC7Z020-1CLG400C (Xilinx, Inc., San Jose, CA, USA). This device integrates two strongly interdependent domains within a single monolithic chip:

- Processing System (PS): A dual-core ARM Cortex-A9 processor running at 650 MHz, responsible for image acquisition, edge detection, data transfer, and result visualization via the PYNQ Python framework Version 2.7.
- Programmable Logic (PL): An FPGA fabric with 85,000 logic cells, 4.9 Mb of Block RAM (BRAM), and 220 DSP slices, operating at 100 MHz, where the complete GA evolutionary loop executes in hardware.

Communication between PS and PL is carried out through high-speed AXI4 interconnects and Direct Memory Access (DMA) transfers, minimizing data movement latency [14].

Table 1. Key hardware resources of the Xilinx Zynq-7000 SoC (XC7Z020) used in this work.

Resource	Specification
Processor	Dual-core ARM Cortex-A9 @ 650 MHz
Logic Cells	85,000
Block RAM	4.9 Mb (140 × 36 Kb BRAMs)
DSP Slices	220
Programmable Logic Clock	100 MHz
Interconnect	AXI4/AXI4-Stream/AXI4-Lite
OS (PS)	Embedded Linux via PYNQ framework version 2.7

2.2. Software and Development Tools

The IP core implementing the GA was designed using Xilinx Vitis High-Level Synthesis (HLS), which translates C/C++ algorithmic descriptions into synthesizable Register-Transfer-Level (RTL) hardware [15]. HLS directives (pragmas) were applied to control loop pipelining, memory binding, and AXI interface protocols. The synthesized IP was integrated into a Vivado block design using AXI4 interconnects, and the resulting bitstream was deployed on the PYNQ-Z1 via the PYNQ Python framework, which provides a high-level hardware abstraction layer running on embedded Linux [14,19].

2.3. Parabola Mathematical Model

A parabola within a digital image is completely specified by four geometric parameters: vertex coordinates (h, k), aperture a, and rotation angle θ. The parametric equations of the rotated parabola, evaluated at the parameter $t \in [-t_{\max}, t_{\max}]$, are:

$$\begin{cases} x(t) = -a\,t^2 \sin\theta + t\cos\theta + h \\ y(t) = \quad a\,t^2 \cos\theta + t\sin\theta + k \end{cases} \tag{1}$$

where the positive branch ($t > 0$) and negative branch ($t < 0$) of the parabola are evaluated simultaneously at each iteration. This parameterization allows the detection of parabolas at any orientation in the image plane. Specifically, the parameter θ controls the axis of symmetry direction, a controls the curvature (larger $|a|$ implies tighter curvature), and the vertex (h, k) fixes the parabola tip. As a result, it overcomes the limitations of earlier detectors, which were restricted to recognizing only axis-aligned parabolas [20,22].

The parametric step Δt is defined adaptively as:

$$\Delta t = \sqrt{\frac{10}{a}} \tag{2}$$

so that the 400-step sweep always covers the full visible extent of the parabola, regardless of its aperture value.

2.4. Binary Edge Image Representation

The input to the GA is a binary edge image of resolution 640×480 pixels (307,200 pixels total), produced by the ARM processor using a standard edge detector (e.g., Canny, Sobel). To allow efficient hardware access, the binary edge map is compacted into 32-bit words, resulting in an array of 9600 words stored in on-chip BRAM using a dual-port configuration. This reduces memory bandwidth and enables single-cycle pixel reads. Additionally, the indices of all active edge pixels are stored in a separate array used during population initialization, guaranteeing that every individual's vertex gene corresponds to a real edge point.

2.5. Genetic Algorithm Design

The proposed GA explores the four-dimensional parameter space (h, k, a, θ) to identify the parabola that most closely aligns with the edge information present in the input image. The entire evolutionary procedure, from population initialization through to the production of final results is executed within the FPGA's programmable logic at 100 MHz. The corresponding algorithmic workflow is depicted in Figure 1.

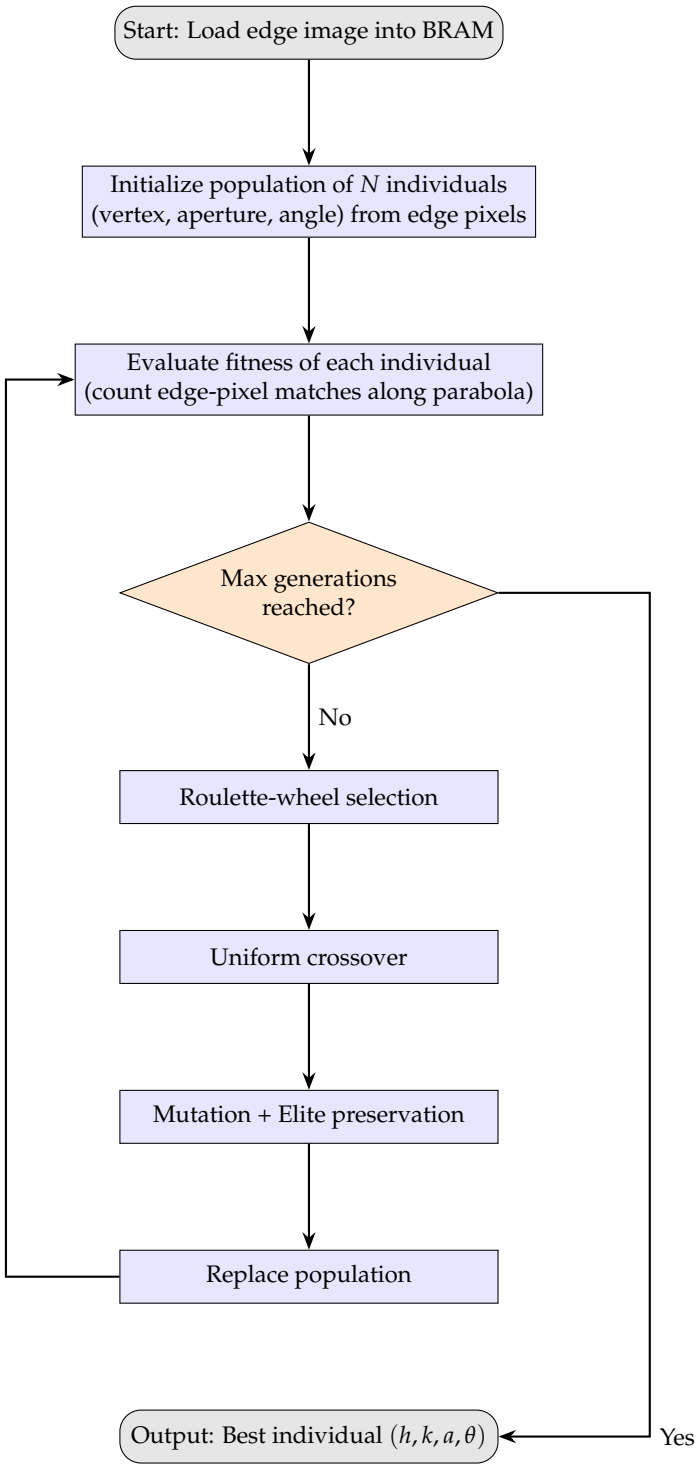


Figure 1. Flowchart of the Genetic Algorithm for parabola detection executed on the FPGA programmable logic.

2.5.1. Chromosome Representation

Each individual in the population encodes a candidate parabola as a compact 37-bit chromosome composed of three genes. The encoding is designed to be directly operable by bitwise hardware crossover and mutation without any multiplication or division. Table 2 summarizes the gene definitions.

Table 2. Chromosome structure of each individual in the Genetic Algorithm.

Gene	Parameter	Description	Bits
0	Vertex (h, k)	Serves as an index into the active edge-pixel array; it corresponds to the (x, y) location of a detected edge point, ensuring that the parabola's vertex consistently lies within the image boundaries.	19
1	Aperture a	Index into a 256-entry look-up table (LUT) covering aperture values in $[5 \times 10^{-4},\ 50 \times 10^{-3}]$ with uniform spacing. A companion LUT stores the adaptive step $\Delta t = \sqrt{10/a}$ for each index.	8
2	Rotation θ	Index into a 1024-entry LUT covering $[0, 2\pi)$. Companion LUTs store the precomputed $\cos\theta$ and $\sin\theta$ values in fixed-point format, eliminating all trigonometric operations from the PL.	10

The three genes are packed into a single 37-bit word (bits [36:18] vertex, bits [17:10] aperture, bits [9:0] angle), enabling efficient bitwise manipulation in hardware.

2.5.2. Population Initialization

A population of $N_{\text{ind}} = 128$ individuals is randomly initialized using a hardware XOR-shift pseudo-random number generator (PRNG) implemented entirely in the programmable logic. The PRNG is based on the recurrence:

$$s \leftarrow s \oplus (s \ll 13) \oplus (s \gg 17) \oplus (s \ll 5) \tag{3}$$

with a 32-bit state and initial seed $s_0 = 2,463,534,242$, providing a period of $2^{32} - 1$ with zero DSP resource usage. Three variants are derived from this generator:

- A normalized float in $[0, 1)$ for probabilistic decisions.
- A uniform integer in $[0, N)$ for array indexing.
- A raw 32-bit word for bitwise mutation masks.

The vertex gene of each individual is sampled from the active edge-pixel index array, ensuring biologically meaningful initialization (all vertices start on real edges).

2.5.3. Fitness Function

The fitness of each individual measures how well its encoded parabola aligns with the binary edge map. The evaluation procedure consists of the following steps:

1. **Fixed-point coordinate computation:** The parametric equations (Equation (1)) are evaluated using 32-bit fixed-point arithmetic (16 integer bits, 16 fractional bits), avoiding floating-point resources in the PL.
2. **Parametric sweep:** The parameter t is swept over 400 steps using the adaptive step Δt (Equation (2)). Both the right arm ($t > 0$) and left arm ($t < 0$) are evaluated at each step, yielding up to 800 candidate pixels per individual per generation.
3. **Neighborhood tolerance:** For each candidate pixel (x, y) within image bounds, a 3×3 neighborhood centered on (x, y) is checked against the BRAM edge map (checking rows $y - 1$, y, $y + 1$). This tolerance compensates for sub-pixel localization errors in the edge detector.
4. **Fitness score:** The normalized fitness is computed as:

$$f = \frac{N_{\text{hits}}}{800} \tag{4}$$

where N_{hits} denotes the number of candidate pixels that coincide with an edge pixel. At least 50 of such valid edge-pixel coincidences are required; otherwise, the fitness is assigned a value of zero to suppress noise-induced false positives. The constant denominator (800) is used to eliminate hardware division, allowing it to be implemented as a fixed multiplication instead.

Table 3 summarizes the main parameters of the Genetic Algorithm.

Table 3. Summary of the Genetic Algorithm parameters.

Parameter	Description	Value
Population size N_{ind}	Number of individuals per generation	128
Chromosome length	Total bits per individual	37 bits
Parametric steps	Number of t steps per parabola evaluation	400 (800 pixels)
Aperture range	Covered values of parameter a	$[5 \times 10^{-4},\ 50 \times 10^{-3}]$
Angle resolution	Number of discrete angles in $[0, 2\pi)$	1024
Minimum hit threshold	Minimum edge matches required for valid fitness	50 pixels
Crossover type	Uniform crossover at gene level	0.9
Mutation type	Bit-flip on aperture and angle genes	0.8
Elitism	Best individual preserved across generations	1 individual
Selection	Fitness proportionate (roulette wheel)	–
Generations	Number of generations that was executed the Genetic Algorithm	5000
Image dimension	The resolution of the images evaluated using the IP core	640×480

The genetic algorithm (GA) parameters listed in Table 3 were determined through empirical preliminary experimentation. The mutation probability, $P_{\text{MUT}} = 0.8$, was set deliberately high compared to conventional GA practices. Preliminary runs with $P_{\text{MUT}} \in \{0.1, 0.3, 0.5\}$ consistently led to premature convergence of the aperture gene a. The elevated mutation rate was therefore chosen to help maintain diversity in the (a, θ) parameters.

The crossover probability was set to $P_{\text{CROSS}} = 0.9$ to maximize recombination, while the elitism operator was included to ensure that the best individual is preserved across generations. A population size of $N = 128$ was necessary, as smaller populations ($N \in \{32, 64\}$) did not provide sufficient initial diversity, even with increased recombination and mutation rates.

A 50-hit threshold was implemented to prevent the GA from mistakenly identifying high-fitness individuals arising from noise pixels that coincidentally matched edge coordinates. This parameter effectively guided the algorithm toward geometrically consistent parabolic solutions.

Additionally, a 3×3 pixel tolerance was introduced to address a convergence problem in which the best individual at the end of the test represented only half of the parabolic arc, resulting in an aperture a approximately twice the expected value. This tolerance relaxed the pixel-matching criterion, allowing candidates whose generated parabolas passed within a 3×3 neighborhood of each target edge pixel to be accepted. This adjustment facilitated convergence to the correct full-arc solutions.

2.5.4. Genetic Operators

2.5.5. Selection

Fitness-proportionate selection (roulette wheel) is used. The cumulative probability vector $P_{\text{accu}}[i] = \sum_{j=0}^{i} f_j \,/\, \sum_j f_j$ is computed each generation. A new parent is chosen

by finding the first index i such that $P_{\text{accu}}[i] \geq r$, where $r \in [0,1)$ is drawn from the hardware PRNG.

2.5.6. Crossover

Algorithm 1 summarizes the gene-level uniform crossover operator implemented in the FPGA. The population is processed in consecutive pairs $(i, i{+}1)$. For each parent pair, the hardware pseudo-random generator `XOR_0_to_1()` produces a real value $p_{\text{cross}} \in [0,1)$, which is compared against the crossover probability P_{CROSS}; crossover is only applied when $p_{\text{cross}} < P_{\text{CROSS}}$. In that case, all genes j of the two parents are processed and, for each gene, the function `XOR_0_to_range(2)` generates an integer $s \in \{0,1\}$ that decides if the gene is exchanged: when $s = 1$, gene j is swapped between individuals i and $i{+}1$, implementing a uniform crossover in which each gene is independently exchanged with probability 0.5.

Algorithm 1 Uniform crossover

Input: Population of size $N_{\text{ind}} = 128$, chromosome length `N_Gene` $= 3$, Crossover probability $P_{\text{CROSS}} = 0.9$
Output: New Population

for $i = 0$ **to** $N_{\text{ind}} - 1$ **step** 2 **do**
  $p_{\text{cross}} \leftarrow$ XOR_0_to_1()
  if $p_{\text{cross}} < P_{\text{CROSS}}$ **then**
    for $j = 0$ **to** `N_Gene`-1 **do**
      $s \leftarrow$ XOR_0_to_range(2)
      if $s = 1$ **then**
        Swap gene j of individuals i and $i{+}1$
      end if
    end for
  end if
end for

2.5.7. Mutation

Algorithm 2 summarizes the mutation stage of the genetic algorithm implemented on the FPGA. For each individual in the population, the hardware PRNG `XOR_0_to_1()` generates a real value $p_{\text{mut}} \in [0,1)$, and mutation is applied only when $p_{\text{mut}} < P_{\text{MUT}}$. Once an individual is selected, each of its three genes is processed independently: a binary variable `do_mutate` obtained with `XOR_0_to_range(2)` decides whether the current gene is modified. Gene 0 (stored in `Gene`$[i,0]$) encodes the vertex index and is resampled by drawing a new edge index uniformly in $[0, \text{Num_Edges} - 1]$, ensuring that the vertex always corresponds to a valid edge pixel. Gene 1 (`Gene`$[i,1]$) encodes the aperture index into a LUT of size $\text{LUT_SIZE}_a = 256$ and undergoes a macro-mutation driven by a 32-bit random word `rand_val`: a 2-bit pattern `mutation_type` selects between assigning a new random index (`rand_val & 255`), halving the current index to obtain a wider parabola, or doubling it when it lies in the lower half of the LUT (with random reinitialization otherwise). Finally, Gene 2 (`Gene`$[i,2]$) encodes the orientation angle and is mutated by drawing a new index in $[0, \text{LUT_SIZE}_\theta - 1]$ using `XOR_32bits() & 1023`, which avoids integer divisions. This bit-level implementation keeps all indices within the valid LUT ranges and minimizes memory accesses and arithmetic operations, which is crucial to sustain a high clock frequency in the FPGA.

Algorithm 2 Mutation

Input: Population NewPopulation of size `Nind` × `N_Gene` (`N_Gene` = 3)
Parameters: Mutation probability P_{MUT}, Num_Edges, $\text{LUT_SIZE}_a = 256$, $\text{LUT_SIZE}_\theta = 1024$
Output: Mutated population NewPopulation
for $i = 0$ **to** `Nind` - 1 **do**
  $p_{\text{mut}} \leftarrow$ XOR_0_to_1()
  if $p_{\text{mut}} < P_{\text{MUT}}$ **then**
    for $j = 0$ **to** `N_Gene` - 1 **do**
      $do_mutate \leftarrow$ XOR_0_to_range(2)
      if $do_mutate = 1$ **then**
        if $j = 0$ **then**
          Gene$[i, 0] \leftarrow$ XOR_0_to_range(Num_Edges)
        else if $j = 1$ **then**
          $rand_val \leftarrow$ XOR_32bits()
          $mutation_type \leftarrow rand_val$ & 3
          if $mutation_type = 3$ **then**
            $mutation_type \leftarrow 2$
          end if
          $a \leftarrow$ Gene$[i, 1]$
          if $mutation_type = 0$ **then**
            $a \leftarrow rand_val$ & 255 // random in $[0, 255]$
          else if $mutation_type = 1$ **then**
            $a \leftarrow a \gg 1$ // wider parabola
          else
            if $a < (\text{LUT_SIZE}_a \gg 1)$ **then**
              $a \leftarrow a \ll 1$ // narrower parabola
            else
              $a \leftarrow rand_val$ & 255
            end if
          end if
          Gene$[i, 1] \leftarrow a$
        else if $j = 2$ **then**
          Gene$[i, 2] \leftarrow$ XOR_32bits() & 1023
        end if
      end if
    end for
  end if
end for

2.5.8. Elitism

In every generation, the best individual is directly transferred to position 0 of the next population, without any changes, before crossover and mutation are applied to the other individuals. This ensures that the best fitness value can only stay the same or improve from one generation to the next.

2.6. System Integration and Data Flow

Figure 2 illustrates the end-to-end data flow of the proposed system. The ARM processor acquires an input image, applies edge detection, packs the binary edge map into 32-bit words, and transfers it to the FPGA BRAM via DMA. It also transfers the active edge-pixel index array to BRAM. The FPGA IP core then executes the complete GA autonomously and returns the best individual (h, k, a, θ) to the ARM via the AXI interface. Finally, the ARM reconstructs the detected parabola and plots it on the original image for visualization.

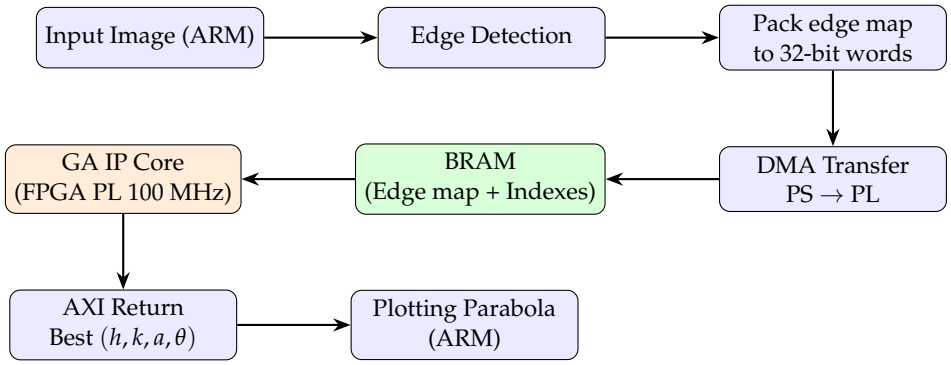


Figure 2. End-to-end data flow of the parabola detection system on the PYNQ-Z1 SoC.

3. Results

3.1. Synthetic Images

For synthetic images, edge detection was performed using an external CPU with a custom 3×3 Sobel operator. Compared to direct intensity thresholding, the Sobel operator produces a richer edge map. Gradient detection was performed by convolving the image with the 3×3 spatial kernels defined in Equation (5), which extract high transitions in the horizontal and vertical axes, respectively:

$$G_v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{5}$$

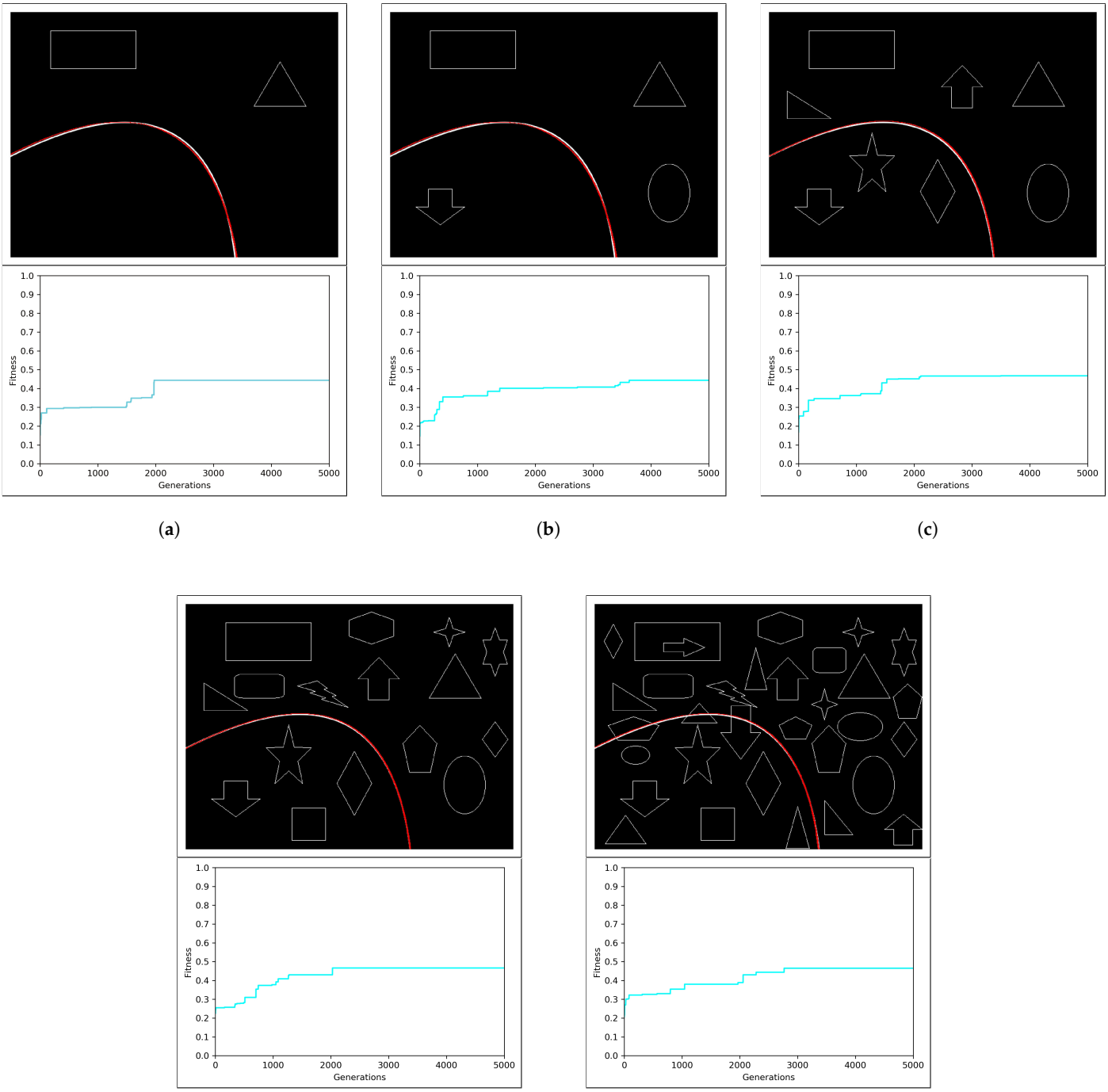
The horizontal and vertical gradient components G_h and G_v were computed and combined as $|\nabla I| = \sqrt{G_h^2 + G_v^2}$, and the resulting gradient magnitude was binarized using Otsu's automatic thresholding method [28]. Since the synthetic images consist of geometric traces rendered on a uniform black background, their intensity histograms are strongly bimodal, and the Otsu threshold converges consistently in the range $[118, 125]$ on a $[0, 255]$ scale across all tested figures, as shown in Table 4.

Table 4. Otsu threshold computed for each synthetic test image as a function of the number of additional shapes.

Figure	Shapes	Threshold
Figure 3a	2	118
Figure 3b	4	119
Figure 3c	8	122
Figure 3d	16	124
Figure 3e	32	125

The proposed GA-based parabola detection algorithm was evaluated on a set of five synthetic images, each containing a single target parabola with known parameters, progressively augmented with additional geometric shapes. Figure 3 presents the detection outcomes alongside the fitness evolution of the elite individual over 5000 generations.

In the simplest case (Figure 3a), the image comprises only two additional shapes. The fitness curve indicates that the best individual stabilizes before generation 2000, after which no further improvement is observed. This behavior suggests that the algorithm rapidly converges to the target parabola, even in a low-clutter environment. This case yields a mean accuracy of 97.54% (Table 5), with particularly strong results for the angular parameter θ ($\text{Acc}_\theta = 99.22\%$) and the vertex coordinates h and k ($\text{Acc}_h = 97.82\%$, $\text{Acc}_k = 98.80\%$).



(**a**) (**b**) (**c**)

(**d**) (**e**)

Figure 3. Synthetic images containing a single parabola with known parameters, progressively augmented with other geometric shapes, doubling in quantity from 2 up to 32 additional figures (**a**–**e**). The top image shows the detected parabola (red), marked by the best individual found by the genetic algorithm, after 5000 generations and 128 individuals. The bottom image shows the fitness evolution (cyan) of the elite individual over the 5000 generations for a representative run.

For 4 additional shapes (Figure 3b), a similar convergence behavior is observed, with the elite individual reaching a stable optimum before generation 2000. The mean accuracy improves slightly to 97.78%, with Acc_θ reaching 99.48%, confirming that the introduction of a moderate number of geometric shapes does not hinder the search significantly.

Table 5. Accuracy of parabola parameters for synthetic images.

Figure	Shapes	Acc_h (px) [%]	Acc_k (px) [%]	Acc_a^* $(-)$ [%]	Acc_θ (°) [%]	Mean Acc [%]
Figure 3a	2	97.82	98.80	94.30	99.22	97.54
Figure 3b	4	98.37	98.95	94.30	99.48	97.78
Figure 3c	8	99.73	99.53	94.30	98.89	98.11
Figure 3d	16	99.67	99.97	94.30	99.94	98.47
Figure 3e	32	98.25	97.33	92.73	94.61	95.23

A^* For Acc_a^*, a robust metric based on the median absolute error (MedAE) with respect to $a_{\mathrm{true}} = 0.003$ was used, such that $\mathrm{Acc}_a = 100(1 - \mathrm{MedAE}_a/|a_{\mathrm{true}}|)$. The accuracies for h, k and θ were computed from the mean error, following the original definitions of Equation (8).

With 8 shapes (Figure 3c), the elite individual again stabilizes within the first 2000 generations, and the detection success rate rises to 98.11%. The results show consistent improvements across all parameters compared to the simpler scenes, reflecting the ability of the GA to exploit a larger population of potential candidates in a more demanding search landscape. At 16 shapes (Figure 3d), the same early convergence pattern holds, with the best individual stabilizing before generation 2000 and achieving a mean accuracy of 98.47%, the highest among all five test cases. The vertex accuracies reach $\mathrm{Acc}_h = 99.67\%$ and $\mathrm{Acc}_k = 99.97\%$, demonstrating excellent detection of the parabola vertex despite the denser scene.

In the most challenging scenario (Figure 3e), which contains 32 additional shapes, the convergence behavior changes notably: the fitness curve exhibits a prolonged transient phase, and the elite individual does not stabilize until approximately generation 3000, reflecting the increased complexity of the search space. This delayed convergence is accompanied by a measurable drop in detection quality, with mean accuracy falling to 95.23%, a reduction of more than 3 percentage points relative to the 16-shape case. In particular, Acc_θ decreases to 94.61% and Acc_a to 92.73%, evidencing that the highly cluttered environment makes it harder for the genetic algorithm to converge to a high-quality individual within the allotted 5000 generations. The quantitative results for all five test cases are summarized in Table 5.

3.1.1. Synthetic Image Box Plots

To further characterize the statistical behavior of the IP core across repeated executions, the GA-based detection system was run 100 independent times for each synthetic test case, and the distribution of the estimated parabola parameters was through box plots (Figure 4). This analysis complements the mean accuracy values presented in Table 5 and provides insight into the consistency and robustness of the hardware implementation under varying initial conditions.

The box plots for the vertex coordinates h (Figure 4a) and k (Figure 4b) display a distinct pattern that aligns with the convergence behavior observed in Figure 3: for the simpler scenes (2 and 4 additional shapes), both parameters exhibit tight interquartile ranges and few outliers, confirming that the IP core reliably converges to accurate vertex estimates across all runs. As scene complexity increases to 8 and 16 shapes, the spread remains small, and the medians stay close to the ground-truth values, which is in agreement with the high accuracy figures $\mathrm{Acc}_h \geq 99.67\%$ and $\mathrm{Acc}_k \geq 99.97\%$ reported for Figure 3d in Table 5. For the most cluttered scene (32 shapes, Figure 3e), the interquartile range widens noticeably, and the number of outliers increases, reflecting the greater difficulty that the genetic algorithm faces when searching for the best individual in a more complex fitness landscape.

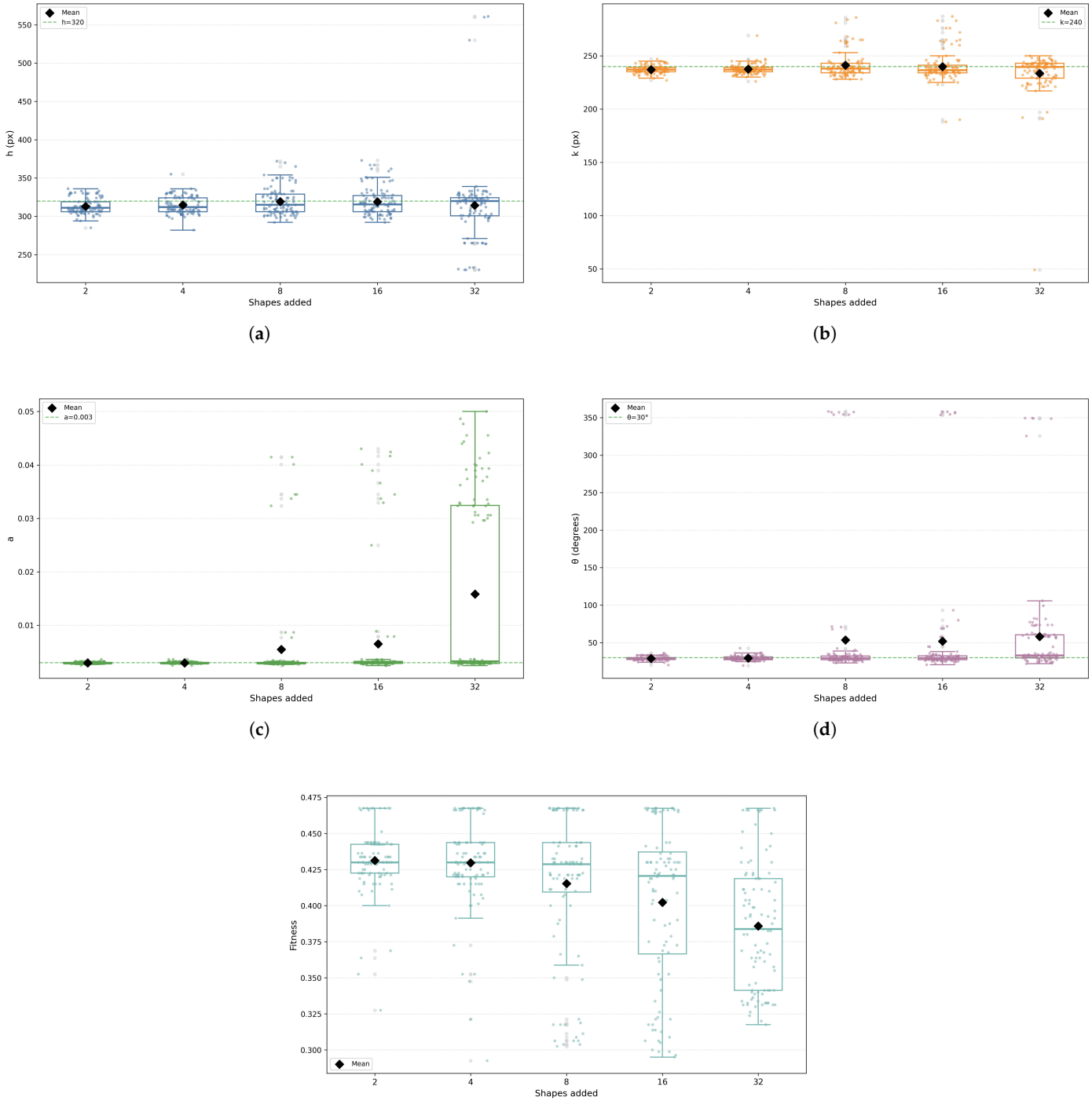


Figure 4. Box plots of the estimated parabola parameters across all synthetic images: (**a**) vertex coordinate h [px], (**b**) vertex coordinate k [px], (**c**) opening parameter a, (**d**) orientation angle θ [deg], and (**e**) best-individual fitness value. Each box shows the median, interquartile range, and outliers over the 100 independent runs performed for each synthetic image with progressively augmented shapes.

A particularly important observation arises from the box plot for the opening parameter a (Figure 4c). Unlike h, k, and θ, whose distributions are approximately symmetric, the distribution of a exhibits a pronounced right-skewed tail to higher values of a across the last case. This asymmetric behavior reveals the existence of sporadic runs in which the GA converges to individuals with an unusually large aperture, shifting the arithmetic mean

away from the main body of the distribution and indicating that the most demanding parameter is the aperture a. For this reason, the median absolute error (MedAE) was adopted as the accuracy metric for a, rather than the mean-based definition used for the other parameters. By fixing the error measure to the median rather than the mean, the metric $\mathrm{Acc}_a = 100(1 - \mathrm{MedAE}_a/|a_{\mathrm{true}}|)$ is robust to these outlier runs and provides a more representative assessment of the typical estimation quality. As a result, Acc_a remains constant at 94.30% for all complexity levels up to 32 shapes and drops only slightly to 92.73% for Figure 3e. This small decrease, together with the skewness visible in the box plot, clearly indicates that estimating a is difficult when numerous shapes occupy the research space.

The box plot for the orientation angle θ (Figure 4d) shows compact distributions for the low complexity scenes and a moderate increase in spread for 32 shapes, consistent with the drop in Acc_θ from 99.94% (16 shapes) to 94.61% (32 shapes) observed in Table 5. Finally, the fitness box plot (Figure 4e) captures the convergence quality of the elite individual: for scenes with up to 32 shapes, the fitness values cluster tightly around a high region, confirming rapid and consistent convergence within the allotted 5000 generations. For the 32-shape case, the fitness distribution shifts downward and its spread increases, providing additional statistical evidence that the larger search space not only delays convergence (as evidenced by the fitness curve stabilizing near generation 3000 in Figure 3e) but also results in a genuinely lower-quality final solution across the 100 runs for the parameters h, k, a, and θ, we defined accuracy measures based on the corresponding ground-truth values. Let p_i denote the value of a given parameter $p \in \{h, k, \theta\}$ obtained in run i, and let p_{true} be its ground-truth value. For h, k, and θ, we used a mean-based definition consistent with the original implementation. First, we compute the sample mean

$$\bar{p} = \frac{1}{N} \sum_{i=1}^{N} p_i, \tag{6}$$

and the (signed) mean error

$$e_{p,\mathrm{mean}} = p_{\mathrm{true}} - \bar{p}. \tag{7}$$

The corresponding accuracy is then defined as

$$\mathrm{Acc}_p^{\mathrm{mean}} = 100\left(1 - \frac{|e_{p,\mathrm{mean}}|}{|p_{\mathrm{true}}|}\right), \quad p \in \{h, k, \theta\}. \tag{8}$$

The aperture parameter a occasionally exhibits a small number of large outliers, which strongly bias the mean-based accuracy and can even yield misleading values for complex scenes. To obtain a more robust measure, we use the median absolute error (MedAE) with respect to the ground-truth aperture $a_{\mathrm{true}} = 0.003$. The per-run absolute errors are

$$e_{a,i} = |a_i - a_{\mathrm{true}}|, \tag{9}$$

and the median absolute error is defined as

$$\mathrm{MedAE}_a = \mathrm{median}\{e_{a,1}, e_{a,2}, \ldots, e_{a,N}\}. \tag{10}$$

The robust accuracy for the aperture is then

$$\mathrm{Acc}_a^{\mathrm{robust}} = 100\left(1 - \frac{\mathrm{MedAE}_a}{|a_{\mathrm{true}}|}\right). \tag{11}$$

In all tables, the reported accuracies for h, k, and θ correspond to $\mathrm{Acc}_p^{\mathrm{mean}}$, while the mean accuracy for the aperture a is given by the robust $\mathrm{Acc}_a^{\mathrm{robust}}$. This convention is

motivated by the fact that only a shows pronounced outliers, for which median-based statistics yield a more faithful description of the typical performance.

3.1.2. Detection Success Rate Under Full-Scale-Range Tolerance Criteria

To assess the robustness of the genetic algorithm against the shapes added in synthetic images, a complementary success-rate analysis was performed on the same 100 independent runs described in Section 3.1.2. Since the synthetic images were computer-generated, the ground-truth parabola parameters are known exactly: $h = 320$ px, $k = 240$ px, $a = 0.003$, and $\theta = 30°$. A run was counted as successful if and only if all four estimated parameters simultaneously fell within a tolerance band defined as 3% of the full-scale range (FSR) of each variable's search space.

Figure 5 summarizes the detection success rate across the five synthetic test cases illustrated in Figure 3a–e, in which the number of additional shapes doubles progressively from 2 to 32 shapes. With only 2 additional shapes (Figure 3a), the algorithm achieves a success rate of 96%, demonstrating near-perfect reliability when the parabolic target is nearly isolated. As the scene complexity grows to 4 shapes (Figure 3b), the rate remains high at 95%, confirming that the search space augmentation is still manageable. A more pronounced degradation appears in 8 shapes (Figure 3c), where the rate declines to 81%: the higher density of competing edges begins to distort the fitness landscape, causing a fraction of runs to converge to suboptimal solutions. At 16 additional shapes (Figure 3d), the success rate drops further to 76%, a relatively small additional decrease, indicating that the strongest downward performance trends are already apparent at 8 and 16 shapes, though some runs still succeed in escaping local optima by the 5000-generation limit. Finally, with 32 shapes (Figure 3e), the success rate reaches its lowest value of 53%, indicating that a heavily cluttered scene with a large number of geometric elements poses a substantial challenge to the genetic algorithm's exploratory capability within the allotted 5000 generations and 128 individuals. The decrease in performance from 98% to 53% as the number of shapes increases from 2 to 32, corresponding to a successive decline in the detection capability of the known parameters of the parabola in the synthetic images, demonstrates a pronounced inverse relationship between scene complexity and detection performance.

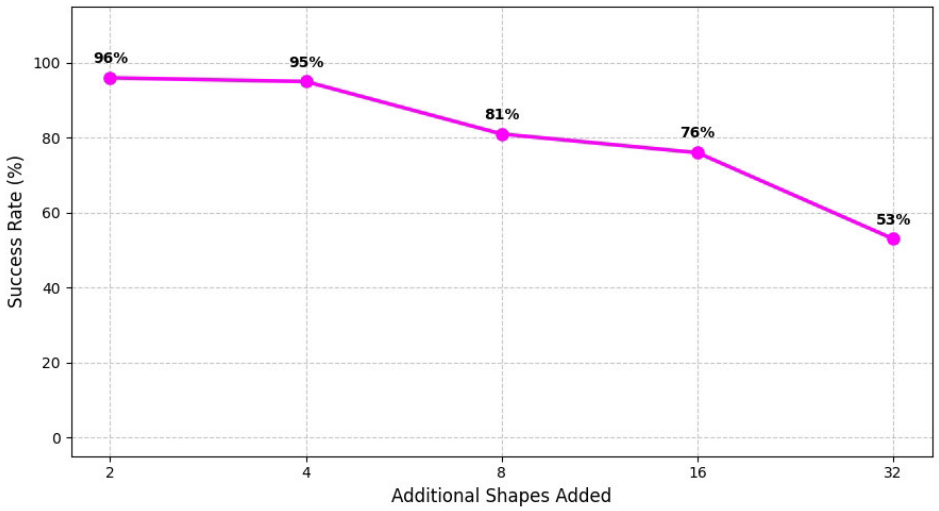


Figure 5. Detection success rate (100 independent runs) of the proposed genetic algorithm-based parabola detector across the five synthetic images (Figure 3a–e), each progressively augmented with additional geometric shapes. A run is classified as successful only if all four estimated parameters (h, k, a, θ) simultaneously satisfy the 3% full-scale-range tolerance criteria. The horizontal axis identifies each test scene by the number of shapes added, and the vertical axis reports the percentage of successful detections for that number of shapes.

3.2. Rotated Parabolas

The GA-based IP core's ability to detect parabolas across varying orientations was evaluated using five synthetic test images. Each image contains a parabola with a distinct rotation angle θ, vertex position (h, k), and aperture a. These configurations span angles of 0°, 30°, 45°, 90°, and 135°. These five cases constitute a limited proof-of-concept for rotated parabola detection under controlled conditions. Figure 6 presents qualitative detection results for these cases. In each subfigure, the upper image displays the detected parabola overlaid on the input image, as estimated by the best individual of the final generation, while the lower image shows the corresponding fitness convergence curve over 5000 generations.

In Figure 6a, the reference case corresponds to a non-rotated parabola ($\theta = 0°$) with vertex at $(h, k) = (320, 240)$ and aperture $a = 0.004$. The algorithm reliably converges to the correct solution, as evidenced by the smooth fitness curve that stabilizes well before the maximum generation count is reached.

For the 30°-rotated case shown in Figure 6b, with vertex $(200, 150)$ and $a = 0.008$, the IP core successfully adapts to the parabola's orientation. The fitness converges quickly, confirming that the rotation does not destabilize the evolutionary search.

The 45° configuration in Figure 6c, defined by vertex $(420, 150)$ and $a = 0.012$, represents the case of maximum diagonal inclination. Despite the steeper aperture and oblique orientation, the algorithm correctly identifies the target parabola, and the elite individual's fitness converges to a high value within the allotted generations.

Figure 6d presents the 90° rotated parabola, which is a horizontally oriented curve, with vertex at $(200, 330)$ and $a = 0.006$. This orientation is particularly challenging because the parabola axis is perpendicular to the canonical upright form; nonetheless, the GA-based hardware accelerator detects the target accurately, as shown by the overlaid curve and the convergent fitness trajectory. Finally, Figure 6e shows a 135° rotation with vertex $(420, 330)$ and $a = 0.010$. The detection result demonstrates that the IP core generalizes well beyond the four cardinal orientations, successfully estimating all four parameters simultaneously.

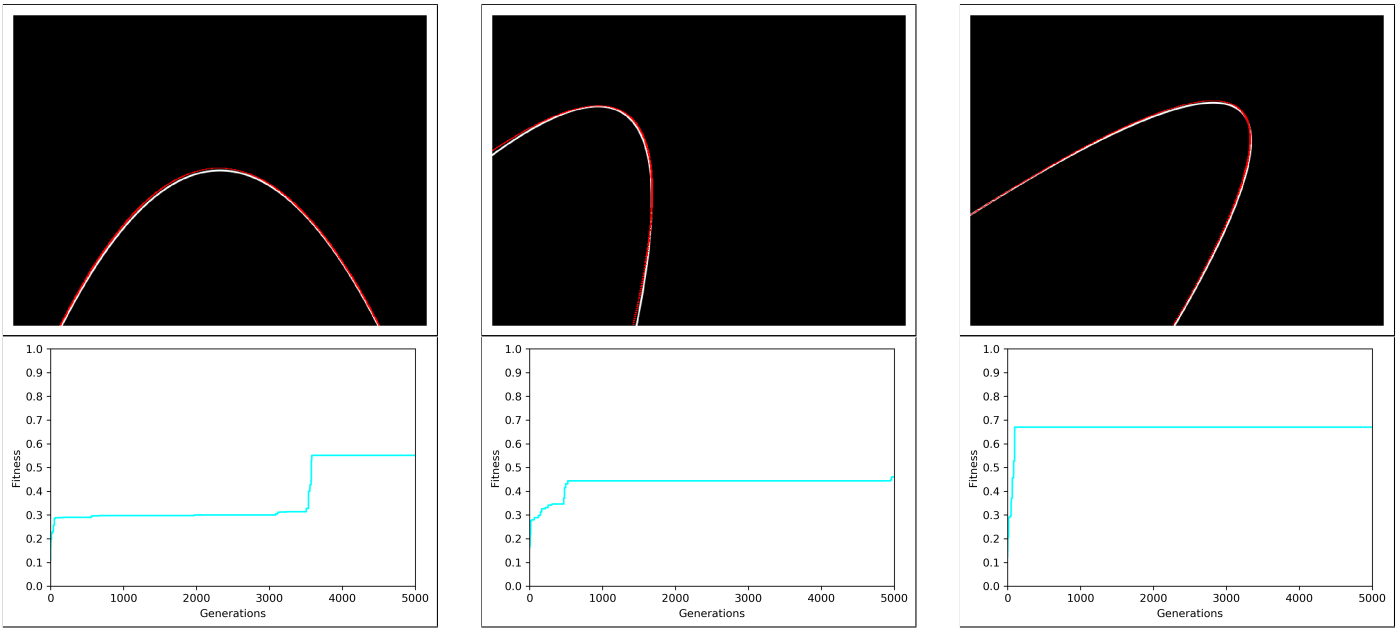
A quantitative assessment of the detection performance across all rotated configurations is provided in Table 6, which reports the Mean Absolute Error (MAE), Median Absolute Error (MedAE), Standard Deviation (Std. Dev.), Interquartile Range (IQR), and the detection Success Rate computed over 100 independent runs per image with a population of 128 individuals evolved for 5000 generations, evaluated under a 3% full-scale-range (FSR) tolerance, as mentioned in previous sections.

For the reference case of Figure 6a ($\theta = 0°$), Table 6 reports a success rate of 100%, with MAE values of 5.31 px for h, 1.43 px for k, 8.8×10^{-5} for a, and 1.38° for θ. The relatively larger MAE in h is attributable to the loose aperture of this parabola, which makes lateral displacements less penalized by the fitness function; nevertheless, the MedAE of 4 px and an IQR of 5 px confirm a tight central distribution of estimates, and the 100% success rate demonstrates perfect reliability across all 100 runs.

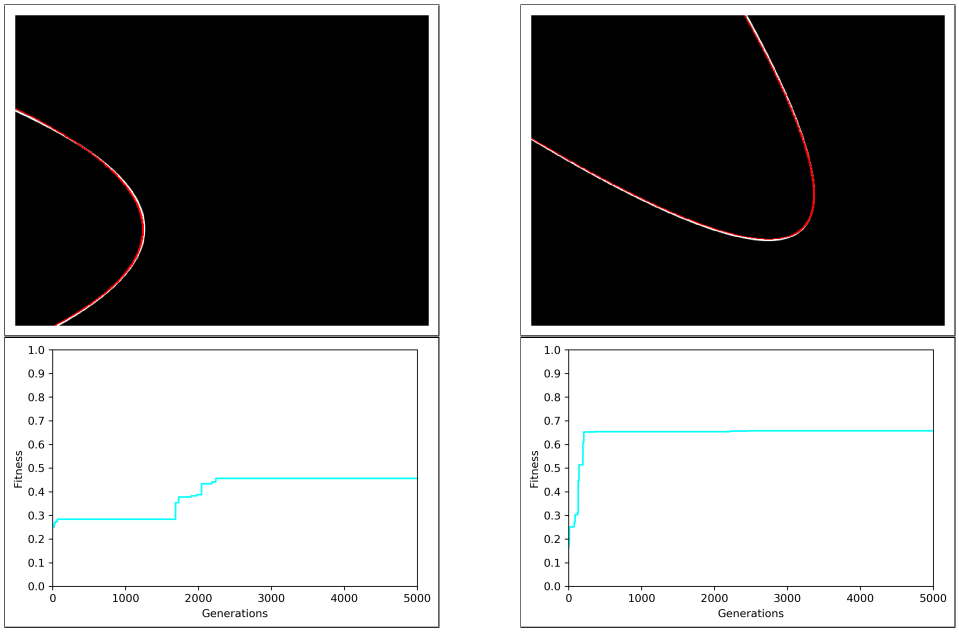
In the 30° configuration of Figure 6b, Table 6 shows that the algorithm also achieves a 100% success rate. The MAE values are 2.18 px (h), 2.29 px (k), 2.3×10^{-4} (a), and 0.84° (θ), all well within the 3% FSR tolerance band. The MedAE and IQR values are consistently low, indicating that the distribution of parameter estimates is concentrated around the ground-truth values with minimal outliers.

For the 45°-rotated parabola of Figure 6c, Table 6 records the only case in which the success rate falls below 100%, reaching 98%. This marginal reduction can be attributed to the combination of the highest aperture coefficient ($a = 0.012$) and oblique inclination, which together increase the complexity of the fitness landscape. Even so, the MAE values remain small (1.17 px for h, 1.16 px for k, 2.1×10^{-4} for a, and 0.33° for θ), and the close

success rate to 100% confirms that the hardware-accelerated GA handles this challenging case with high robustness.



(**a**) (**b**) (**c**)



(**d**) (**e**)

Figure 6. Experimental results for rotated parabola images (**a**–**e**). The top row of each subfigure displays the detected target marked by the best individual (red) found by the genetic algorithm. The bottom row illustrates the fitness evolution of the elite individual over 5000 generations (cyan) for a representative run.

The 90° case of Figure 6d achieves a 100% success rate with notably low MAE values: 0.76 px for h, 2.93 px for k, 9.4×10^{-5} for a, and 0.56° for θ (Table 6). The sub-pixel MAE for h and the sub-degree angular error highlight the precision of the IP core when the

parabola axis is aligned horizontally. The low IQR values further confirm the consistency of the estimations across all 100 runs.

Table 6. Statistics and detection success rate across 100 independent runs for the five synthetic images with different parabolic parameters.

Figure	Parameter	Ground Truth	MAE	MedAE	Std. Dev.	IQR	Success Rate
Figure 6a	h (px)	320.0	5.3100	4.0000	6.8455	5.0000	100.00%
	k (px)	240.0	1.4300	1.0000	1.0037	1.0000	
	a (-)	0.004	0.000088	0.000006	0.000177	0.000000	
	θ (°)	0.0	1.3887	0.7031	2.0456	1.4941	
Figure 6b	h (px)	200.0	2.1800	2.0000	2.4430	1.0000	100.00%
	k (px)	150.0	2.2900	3.0000	2.1471	2.0000	
	a (-)	0.008	0.000229	0.000265	0.000230	0.000141	
	θ (°)	30.0	0.8496	0.8203	0.9944	0.3516	
Figure 6c	h (px)	420.0	1.1700	1.0000	1.4635	2.0000	98.00%
	k (px)	150.0	1.1600	1.0000	1.0515	2.0000	
	a (-)	0.012	0.000214	0.000147	0.000532	0.000000	
	θ (°)	45.0	0.3305	0.3516	0.5238	0.3516	
Figure 6d	h (px)	200.0	0.7600	1.0000	0.8124	1.0000	100.00%
	k (px)	330.0	2.9300	3.0000	2.9403	2.0000	
	a (-)	0.006	0.000094	0.000065	0.000128	0.000065	
	θ (°)	90.0	0.5695	0.5273	0.9224	0.7031	
Figure 6e	h (px)	420.0	0.9800	1.0000	1.2134	0.2500	100.00%
	k (px)	330.0	0.7800	1.0000	1.0190	1.0000	
	a (-)	0.010	0.000194	0.000206	0.000180	0.000024	
	θ (°)	135.0	0.2637	0.3516	0.3386	0.3516	

For the 135° configuration of Figure 6e, Table 6 reports a 100% success rate, with MAE values of 0.98 px (h), 0.78 px (k), 1.9×10^{-4} (a), and 0.26° (θ). Notably, the angular MAE of 0.26° is the lowest across all five configurations, and the IQR for θ of 0.35° reflects exceptional rotational accuracy. These findings show that the genetic algorithm reliably finds the correct orientation, even for non-standard axis-oriented parabolas.

Overall, the results compiled in Table 6 demonstrate that the GA-based IP core running on the FPGA achieves a mean success rate of 99.6% across the five rotated configurations, with four out of five cases attaining a perfect 100% detection rate. The MAE and MedAE values for all parameters remain small relative to the image dimensions and the parameter ranges, and the low Std. Dev. and IQR values confirm that the estimations are both accurate and highly repeatable. These metrics collectively validate that the proposed hardware-accelerated genetic algorithm is robust to parabola rotation in non-challenging background scenarios, vertex displacement, and aperture variation, demonstrating its suitability for real-world detection tasks where parabolic shapes may appear at arbitrary orientations. Additionally, these results should be interpreted as indicative of prototype behavior rather than as a general performance bound.

3.3. Real Images

To assess the IP core behavior under uncontrolled conditions, five real images were selected to represent a broad range of visual scenarios in which parabolic shapes appear naturally: a Mars photograph (Figure 7d), a curved road (Figure 7c), an Earth photograph (Figure 7e), a rainbow arc photograph (Figure 7b), and a Retinal Vessel image [29], in which the main vessel arcade delineates a complex parabolic arc (Figure 7a). Unlike the synthetic images, which were binarized using Otsu's automatic thresholding applied to the Sobel gradient magnitude (Section 3.1), each real image was pre-processed on an external CPU using the Holistically Nested Edge Detection (HED) [30–32], a deep learning-based edge detector that employs a fully convolutional network derived from VGGNet with deep supervision at multiple side-output layers, enabling multi-scale and multi-level edge prediction us-

ing the OpenCV DNN module with the pre-trained Caffe model and its corresponding network definition. A custom `CropLayer` was registered to align intermediate feature maps. The input blob was constructed with a scale factor of 0.7 and mean subtraction of $(105, 117, 123)$ (BGR). The network output was rescaled to $[0, 255]$ and saved as a grayscale edge map, which was subsequently binarized using a pixel intensity threshold of $\tau > 70$ (on a $[0, 255]$ scale) to produce the binary edge image. This binary image was then packed into 32-bit words and transferred to the FPGA BRAM via DMA as the input to the IP core, and as with synthetic images, the IP core was executed 100 times independently for each image under identical conditions: a population size of 128 individuals and a maximum of 5000 generations (see Table 3). The qualitative results for each test case are summarized in Figure 7, which shows, from top to bottom: the original image with the detected parabola plotted, the corresponding edge images with the same parabola detected, and the fitness evolution of the elite individual across generations. Statistical results of the 100 runs per image are provided in Table 7 and the corresponding box plots in Figure 8.

Table 7. Descriptive statistics of the detected parabola parameters (h, k, a, θ) for real images, based on 100 runs with 5000 generations each test.

Exp.	Param	Mean	Median	Std	Q1	Q3	IQR	MAD	CV%
Figure 7a	h (px)	73.21	74	3.6273	71	76	5	2	4.955
	k (px)	207.3	206	4.8231	204	211	7	3	2.327
	a (-)	0.0122	0.011	0.0036	0.0106	0.012	0.0014	0.0004	29.471
	θ (°)	280.1391	279.8438	3.8162	278.0859	281.25	3.1641	1.4062	1.362
Figure 7b	h (px)	307.91	307.5	5.238	304	311	7	3.5	1.701
	k (px)	131.34	131	2.3278	130	132	2	1	1.772
	a (-)	0.0059	0.0059	0.0001	0.0059	0.0059	0.0001	0	2.186
	θ (°)	283.3594	358.2422	146.5109	357.5391	358.9453	1.4062	0.7031	51.705
Figure 7c	h (px)	432.66	417	22.8072	414	455	41	5	5.271
	k (px)	138.39	143	16.4678	118.75	148	29.25	6	11.900
	a (-)	0.0078	0.0075	0.0006	0.0075	0.0083	0.0008	0.0002	7.584
	θ (°)	78.1594	78.3984	1.9275	76.6406	79.1016	2.4609	1.0547	2.466
Figure 7d	h (px)	155.96	162.5	26.8937	142.5	173.25	30.75	14.5	17.244
	k (px)	212.08	211	40.744	196.75	219	22.25	11	19.212
	a (-)	0.005	0.0034	0.0074	0.0032	0.0036	0.0004	0.0002	148.508
	θ (°)	41.8254	35.5078	48.826	29.3555	39.4629	10.1074	4.7461	116.738
Figure 7e	h (px)	393.29	405	49.3739	378.75	424	45.25	21	12.554
	k (px)	191.79	201	19.545	191.75	204	12.25	4	10.191
	a (-)	0.0104	0.0021	0.0172	0.0021	0.0024	0.0004	0	166.063
	θ (°)	200.9531	189.8438	30.9284	186.6797	194.4141	7.7344	3.5156	15.391

3.3.1. Analysis of Results on Real Images

The five test images span a wide range of real-world image scenarios, enabling a thorough assessment of the IP core genetic algorithm's ability to detect parabolic shapes under natural image conditions. The following analysis integrates the qualitative inspection of Figure 7, the statistical distributions in Table 7, and the box plots in Figure 8.

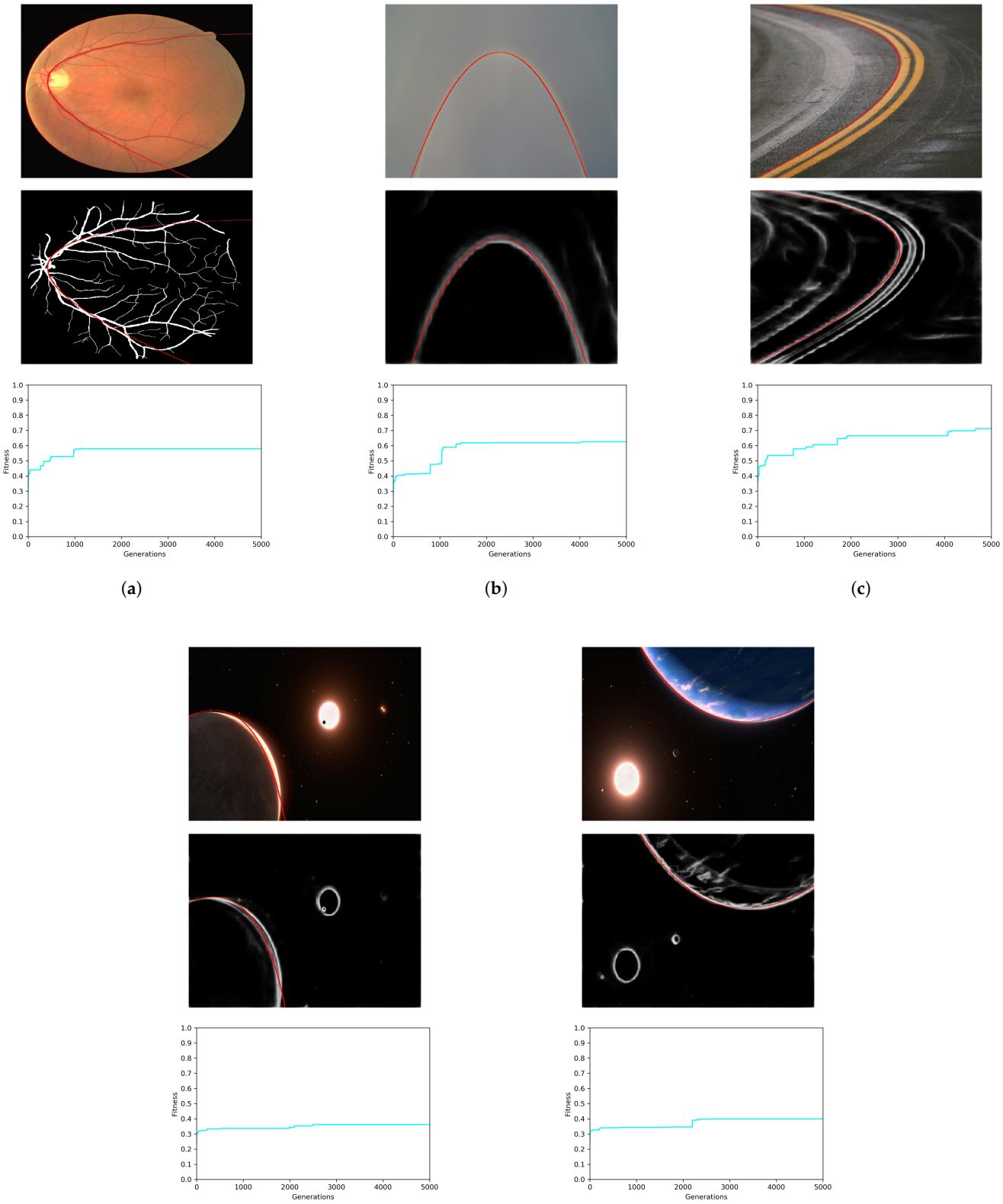


Figure 7. Real images used to evaluate the proposed parabola detection algorithm (**a**–**e**). For each test case, the top image shows the original input, the middle image shows the detected parabola overlaid by the best individual (red) found by the genetic algorithm, and the bottom image shows the fitness evolution of the elite individual over 5000 generations (cyan) for a representative run.

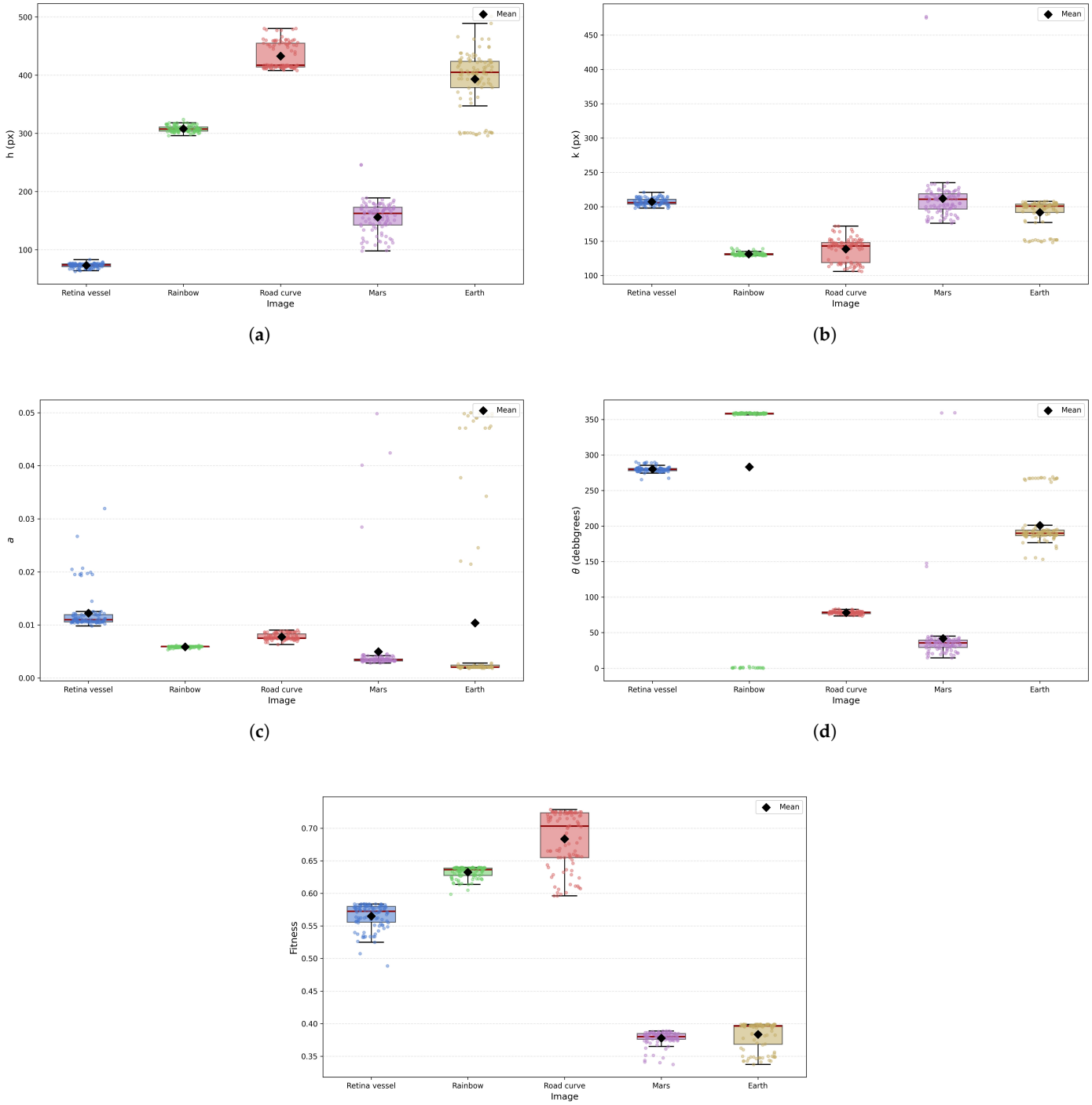


Figure 8. Box plots of the estimated parabola parameters across all real images tested: (**a**) vertex coordinate h [px], (**b**) vertex coordinate k [px], (**c**) opening parameter a, (**d**) orientation angle θ [deg], and (**e**) best-individual fitness value. Each box shows the median, interquartile range, and outliers over the 100 independent runs performed for each real image.

Retinal Vessel Image (Figure 7a)

The retinal vessel image is the most favorable among the five real tests. The superior arcade of the optic fundus forms a disrupted parabolic shape, a high-contrast parabolic arc after edge detection, providing the genetic algorithm with a dense and unambiguous set of boundary points. Table 7 confirms the consistency of the 100 independent runs: the vertex coordinates exhibit very low dispersion (h: mean = 73.21 px, median = 74 px, std = 3.63 px,

CV = 4.96%; k: mean = 207.3 px, median = 206 px, std = 4.82 px, CV = 2.33%), indicating that the algorithm reliably detects the vertex across all repetitions. The orientation angle θ is equally stable (mean = 280.1°, std = 3.82°, CV = 1.36%), and the box plot in Figure 8d shows a very compact whisker range for this image. The opening parameter a has a moderately higher coefficient of variation (CV = 29.47%), which is consistent with the observation that vessel curvature generates a relatively shallow arc, the exact angular extent of which is difficult to determine with precision; nonetheless, the interquartile range remains narrow (IQR = 0.0014). The fitness evolution curves in Figure 7a confirm a fast convergence before 1000 generations, with the elite individual typically reaching a stable fitness value for the rest of the generation limit.

Rainbow arc Photograph (Figure 7b)

The rainbow arc image exhibits the most consistent convergence behavior among all real images tested, with the positional and curvature parameters showing the lowest variability across 100 independent runs, while also exposing a specific, identifiable limitation of the angle representation. The vertex coordinates are the most tightly constrained of all real images (h: CV = 1.70%; k: CV = 1.77%), and the opening parameter a achieves the lowest coefficient of variation of all five images (CV = 2.19%, std = 0.0001), demonstrating that the shape of the arc is detected with outstanding repeatability. This is consistent with the visual result in Figure 7b, where the detected parabola closely follows the curvature of the rainbow band.

However, the orientation angle θ presents a markedly anomalous statistical behavior: while the mean is 283.36°, the median is 358.24°, and the standard deviation reaches 146.51° (CV = 51.70%), with $Q_1 = 357.54°$ and $Q_3 = 358.95°$. This apparent inconsistency is not a failure of the algorithm, but rather a direct consequence of the angular periodicity inherent in the parabola model. The rainbow arc is oriented with θ very close to the 0°/360° boundary, so the genetic algorithm correctly identifies equivalent solutions on both sides of this discontinuity: a substantial fraction of runs converges to $\theta \approx 358°$–$359°$ (compact cluster at high values, as confirmed by the tight IQR of 1.41° around the median), while the remaining runs wrap around to $\theta \approx 0°$–$2°$. Both families of solutions are geometrically identical when the periodicity of the angle is taken into account; the bimodal distribution is clearly visible in the box plot of Figure 8d, where the whiskers for the rainbow image span the full range $[0°, 360°)$ while the box itself is narrow, an unmistakable signature of bimodality near a circular boundary. This effect is not captured by the arithmetic mean or the standard deviation reported in Table 7, which are not appropriate summary statistics for circular data near a circular point; the median, Q_1, and Q_3 together with the box plot provide the correct interpretation. Regarding the fitness convergence, the evolution curve in Figure 7b shows that the elite individual reaches a stable value at approximately generation 1800, after which no significant improvement is observed; this early and decisive convergence is consistent with the algorithm having correctly identified a well-supported solution, notwithstanding the angular ambiguity discussed above.

Curved Road Photograph (Figure 7c)

The road curve image represents an intermediate complexity scenario. The parabolic structure is geometrically present, but the perspective projection of the lane markings introduces slight distortions, and the edge map contains competing contours from road boundaries, lane dividers, and background elements, increasing the difficulty of the optimization problem. The vertex coordinates show a moderate spread (h: std = 22.81 px, CV = 5.27%; k: std = 16.47 px, CV = 11.90%), and the interquartile ranges ($IQR_h = 41$ px, $IQR_k = 29.25$ px) indicate that while the majority of runs converge to a similar region,

a non-negligible fraction of runs still identifies a shifted vertex. This behavior is clearly captured in the h and k box plots (Figure 8a,b), which show a wider box for this image compared to the retinal and rainbow cases. The opening parameter a and the orientation angle θ are, by contrast, well-determined (CV = 7.58% and CV = 2.47% respectively), confirming that the algorithm robustly identifies the correct curvature direction even when the exact vertex location varies between runs. The fitness evolution in Figure 7c reflects a slower convergence compared to the retinal vessel, with the elite individual still exhibiting incremental improvements through approximately generation 4000, after which the curve flattens, and no further significant gains are observed within the rest of the generations.

Mars Photograph (Figure 7d)

The Mars image constitutes one of the two most challenging cases in this set. The visible parabolic shape is embedded in a complex, noisy background at the edge of the image's vertex. The resulting statistical profile shows substantially higher variability across all parameters: vertex h (std = 26.89 px, CV = 17.24%, IQR = 30.75 px), vertex k (std = 40.74 px, CV = 19.21%, IQR = 22.25 px), and in particular the opening parameter a (mean = 0.0050, median = 0.0034, std = 0.0074, CV = 148.51%). The large discrepancy between the mean and median of a (mean is 47% above the median) indicates a positively skewed distribution dominated by outlier runs in which the algorithm settled on a wide, nearly flat parabola that covers a large portion of the noisy edge map. The orientation angle is similarly dispersed (mean = 41.83°, std = 48.83°, CV = 116.74%), suggesting that in many independent runs the algorithm converged to local optima with qualitatively different orientations. The box plots for a and θ in Figure 8c,d show extended whiskers and numerous outliers for this image, consistent with a fitness landscape having several competing local optima of comparable quality. The fitness evolution curves in Figure 7d corroborate this, displaying a less steep convergence and a lower final fitness value relative to the simpler cases, indicating that the 5000 generations are sufficient to improve but not to reliably identify a unique global solution. Inspection of the fitness evolution curve in Figure 7d reveals that the elite individual undergoes a slow, irregular ascent throughout the run, reaching an approximate value only around generation 2400–2500; beyond this point, the curve remains nearly flat, confirming that the remaining generations do not produce further meaningful refinement and that the algorithm has exhausted its exploratory capacity.

Earth Photograph (Figure 7e)

The Earth image is the most problematic scenario for the proposed detector. Although the planet's horizon forms an intrinsically parabolic arc, the edge map is distorted by cloud boundaries, atmospheric gradients, and the smooth transition between the Earth's surface and space, resulting in a broad, diffuse edge region rather than a sharp parabolic boundary. Table 7 reflects this difficulty: the vertex coordinates exhibit the highest absolute dispersion of all images (h: std = 49.37 px, CV = 12.55%, IQR = 45.25 px; k: std = 19.55 px, CV = 10.19%). The opening parameter a shows the most extreme distributional anomaly of the entire experiment: mean = 0.0104, median = 0.0021, std = 0.0172, CV = 166.06%, with $Q_1 = Q_3 = 0.0021$ and IQR = 0.0004. This combination, where the mean is nearly five times the median and the IQR is extremely narrow, indicates that the vast majority of runs correctly converge to a tight cluster of solutions with low a (a very open parabola, appropriate for a planetary arc), but a minority of runs produce outlier solutions with very large a values that dramatically inflate the mean and standard deviation. This bipartite pattern is clearly observable in the box plot of a (Figure 8c), where the body of the box is compact at low values while isolated outlier points are scattered toward much higher values. The orientation angle θ (mean = 200.95°, std = 30.93°, CV = 15.39%) also shows

moderate spread, consistent with a fitness landscape that has a shallow and broad global optimum. The qualitative result in Figure 7c confirms that the detected parabola captures the general curvature of the horizon, although the fitness convergence curve is the slowest and noisiest of all five test cases, indicating that the genetic algorithm encountered significant difficulty in finding best solutions. The fitness evolution curve in Figure 7e corroborates this behavior: the elite individual improves slowly and irregularly, with an approximate value emerging around generation 2300, after which the remaining generations yield no substantial gain,the lowest and most erratic convergence profile observed among all five real images.

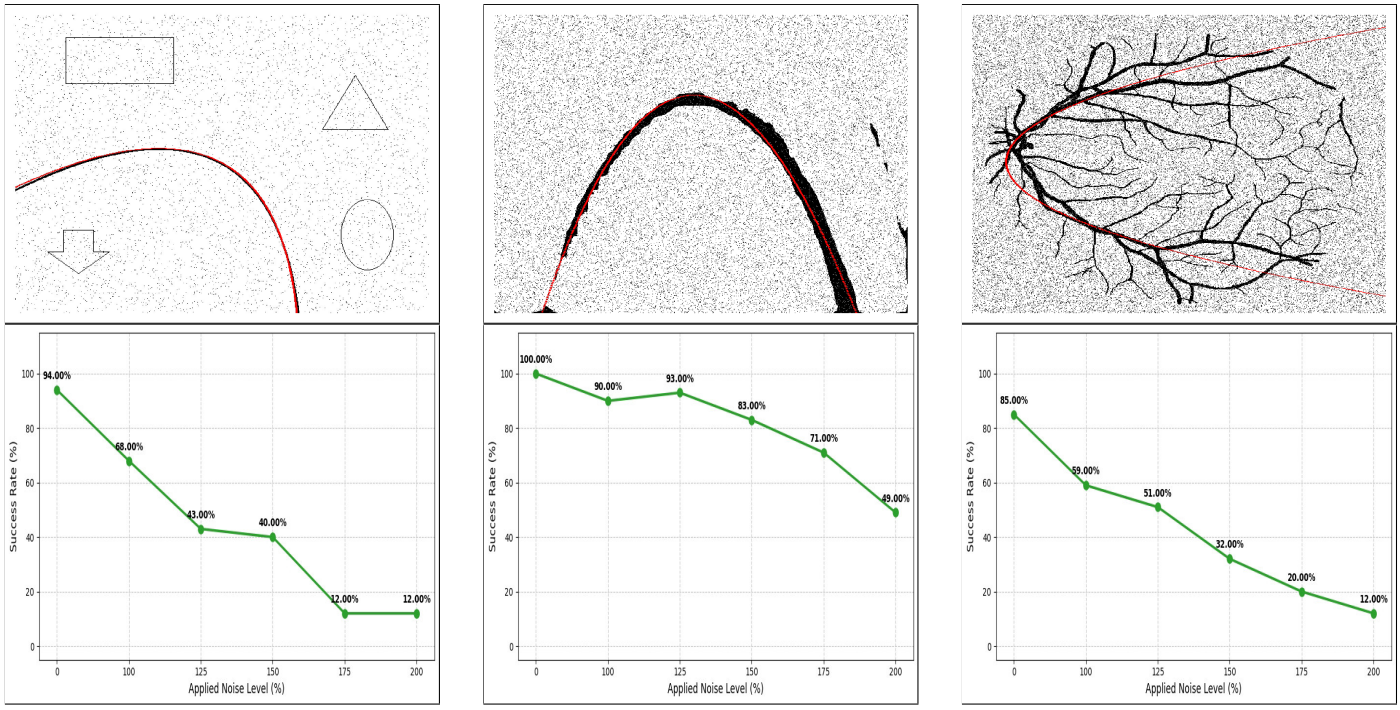
Overall Comparative Assessment

Considering all five real-image tests jointly, the results follow a clear hierarchy of detection difficulty that can be traced directly to the quality and specificity of the edge map produced by the binary preprocessing stage. Images in which the parabolic shape generates a dense, isolated, and high-contrast edge (Retinal vessel, Rainbow arc) allow the genetic algorithm to reliably converge on consistent solutions across all 100 runs, with CV values below 5% for most geometric parameters. As the edge density decreases, competing edge sources increase, and the local contrast of the parabolic arc diminishes (Road curve), the variance in vertex localization grows, but the curvature shape remains robustly detected. At the most difficult extreme (Mars, Earth), the fitness value shows that the best individual dominated the population in early generations, which was insufficient to consistently reach the global optimum, leading to large between-run variability across all parameters and the emergence of outlier solutions. The box plots in Figure 8 offer a quick overview of this hierarchy. The interquartile ranges for all five parameters expand systematically from the Retinal vessel case to the Earth case, and the extent and frequency of outliers increase accordingly. These observations demonstrate both the strength of the genetic algorithm approach, its ability to identify parabolic structures in diverse and complex real scenes, and its primary limitation: dependence on the quality of the upstream edge extraction, which directly determines the sharpness of the fitness value and the reliability of convergence within a fixed computational budget. Notably, the fitness convergence generation also follows this difficulty hierarchy: the Rainbow arc stabilizes around generation 1800, the Earth and Mars images require until approximately generation 2300 and generation 2400–2500 respectively, and the Road curve continues to improve until near generation 4000, all of which is visually evident in the fitness curves of Figure 7 and further supports the interpretation that the quality of the edge map is the primary determinant of both the success rate and the convergence speed of the genetic algorithm.

3.4. Noise Corrupted Images

To quantify the robustness of the genetic-algorithm-based parabola detector under additive noise, added directly to the binary edge map image, a salt-and-pepper corruption protocol was applied to three representative images: the synthetic scene with 4 geometric shapes (Figure 9a), the rainbow arc (Figure 9b), and the retinal fundus vessel image (Figure 9c). In contrast to the synthetic evaluation of Section 3.1.2, where exact ground-truth parameters are available, these experiments rely on a surrogate reference: the median parameter obtained from 100 noise-free runs of the algorithm was taken as the pseudo-ground-truth baseline. Noise was then injected as salt-and-pepper pixels whose count was set to a fixed percentage of the total number of detected edge pixels in each image, ranging from 100% to 200% in steps of 25%. For each noise level, 100 independent runs were executed and the detection success rate computed as in the Section 3.1.2 with respect to the surrogate ground truth was recorded. The upper frames of Figure 9 show the best-detected

parabola at the maximum noise level of 200%, while the lower frames plot the detection success rate as a function of the applied noise percentage.



(**a**) (**b**) (**c**)

Figure 9. Representative results of the salt-and-pepper noise robustness test applied to three selected images: a synthetic scene with 4 geometric additional shapes (**a**), the rainbow arc (**b**), and the retinal fundus vessel image (**c**). For each case, the upper frame shows the best parabola detected (painted red) by the genetic algorithm over 100 independent runs at the maximum noise level of 200%, while the lower frame reports the detection success rate as a function of the applied noise level, ranging from 100% to 200%.

3.4.1. Synthetic Image with 4 Additional Shapes (Figure 9a)

The synthetic scene exhibits the steepest degradation curve among the three test cases. At 100% noise (i.e., a number of corrupted pixels equal to the total edge count), the algorithm still achieves a detection success rate of approximately 94%, confirming a reasonable tolerance to moderate corruption. However, the success rate drops sharply to ≈68% at 125% noise and further declines to ≈43% at 150%, marking the inflection point beyond which the genetic algorithm fails to converge reliably. At 175% and 200% noise levels, the success rate continues to decline to ≈40% and ≈12%, respectively, indicating that the false edges introduced by the salt-and-pepper pattern overwhelm the genuine parabolic contour and severely distort the fitness. The best individual detected at 200% (upper frame) visually confirms this: although the parabola shape is preserved, the estimated position and curvature deviate noticeably from the noise-free reference. The synthetic image is the most sensitive scenario because its edge map is sparse and precisely defined by the known geometric shapes; any random noise pixel carries proportionally higher misleading weight than in a real image.

3.4.2. Rainbow arc (Figure 9b)

The rainbow image demonstrates the highest robustness to salt-and-pepper noise of the three scenarios evaluated. At 100% noise, the detection success rate reaches ≈100%, and remarkably remains at ≈93% even at 150%, suggesting that the genetic algorithm

successfully exploits the strong, spatially extended chromatic edge of the rainbow arc even in the presence of heavy corruption. A meaningful degradation begins only at 175% (≈71%), and at the maximum level of 200% the success rate settles at ≈49%. The superior noise resilience of this image can be attributed to the rainbow arc, which produces a dense, high-contrast edge signal that retains significant discriminative power even after substantial noise injection. The best detection at 200% (upper frame) confirms that the algorithm still recovers the correct parabolic arc even under extreme noise conditions.

3.4.3. Retinal Fundus Vessel Image (Figure 9c)

The retinal image occupies an intermediate position in terms of noise robustness, but exhibits the first occurrence of degradation among the three test scenarios. Starting from ≈85% at 100% noise, already lower than the other two scenarios at the same level, the success rate declines steadily to ≈69% at 125%, ≈51% at 150%, and ≈32% at 175%, reaching its minimum of ≈20% at 200%. The consistently lower baseline success rate reflects the intrinsic difficulty of fitting a single parabola to a complex vascular network where multiple arc-shaped structures compete for detection. As noise increases, the genetic algorithm progressively loses its ability to discriminate the dominant vessel arc from the corrupted background, and the detected parabola drifts away from the target structure, as clearly visible in the upper frame of Figure 9c at 200% noise, where the red parabola is completely misaligned with any meaningful vascular contour.

The three test cases were deliberately selected to cover three representative scenarios: a controlled synthetic image (known geometry, precise edge map), a real image whose dominant structure is genuinely parabolic (rainbow arc), and a real image where the parabolic model is only an approximation of the underlying anatomy (retinal fundus vessel). This design allows the noise-sensitivity behavior of the genetic algorithm to be interpreted in terms of the physical properties of each edge map rather than the image category alone.

The primary factor governing robustness to salt-and-pepper noise is the thickness and density of the edge structure that defines the target parabola. The rainbow arc produces a broad chromatic band whose edge map contains many pixels distributed across the full width of the arc; even after a large fraction of those pixels are corrupted or replaced by noise, a sufficient number of coherent edge points survive to guide the genetic algorithm toward a valid solution, sustaining a detection success rate above 90% up to 150% noise. By contrast, the synthetic image contains thin geometric contours typically one to two pixels wide, so each edge pixel carries disproportionate information weight; as salt-and-pepper noise is added at the same proportion of the total edge count, these sparse contours are rapidly overwhelmed, causing the success rate to collapse sharply beyond 125% noise. The retinal image presents an additional challenge: the retinal vessel network does not constitute a true parabolic structure, and the genetic algorithm is inherently fitting an approximation to a complex, branching vascular geometry. Even in the noise-free case this approximation introduces ambiguity, which explains the lower baseline success rate relative to the other two scenarios; the progressive addition of noise further degrades the edge cues that the algorithm relies upon to resolve that ambiguity, leading to the most consistent decline across all noise levels.

3.5. Time Performance

This work centers on the timing characterization of an FPGA IP core, specifically designed to process a fixed input interface. The core accepts a binary edge image with a resolution of 640 by 480 pixels, which is organized into 9600 32-bit words, totaling 307,200 bits. This image is transferred from the memory of the ARM processor to the

FPGA Block RAM (BRAM) via Direct Memory Access (DMA), with the transfer occurring once per image. This time taken to transfer the image is reported in Table 8 as (SW) time, and hardware (HW) time, corresponding to the actual genetic-algorithm execution on the FPGA fabric. Across all configurations, the SW time remains virtually constant at approximately 0.36 ms, while the HW time dominates the total execution time at around 13,087 ms on average, yielding a mean total execution time of 13,088.01 ms. The near-constant HW time across images of different complexity confirms that the execution budget is fixed by the algorithm parameters (5000 generations, 128 individuals) and is independent of scene content.

Table 8. Execution Time of the Genetic-Algorithm IP Core for All Test Images: Software (SW), Hardware (HW) FPGA Execution, and Total Time for 5000 Generations.

Test Image	SW Time (ms)	HW Time (ms)	Total Time (ms)
Synthetic images (Figure 3)			
Figure 3a—2 shapes	0.3488	13,087.9109	13,088.2597
Figure 3b—4 shapes	0.3380	13,087.1054	13,087.4434
Figure 3c—8 shapes	0.3432	13,084.5656	13,084.9088
Figure 3d—16 shapes	0.3461	13,083.6099	13,083.9561
Figure 3e—32 shapes	0.3524	13,089.8964	13,090.2488
Real images (Figure 7)			
Figure 7a—Retinal fundus vessel	0.3786	13,104.0596	13,104.4382
Figure 7b—Rainbow arc	0.3662	13,077.7125	13,078.0788
Figure 7c—Road curve	0.4012	13,083.0545	13,083.4557
Figure 7d—Mars	0.3563	13,093.6815	13,094.0378
Figure 7e—Earth	0.3551	13,084.9513	13,085.3064
Mean	0.3586	13,087.6548	13,088.0134

Table 9 reports the per-generation execution time of the IP core, obtained by dividing the HW execution time of each run by the fixed number of generations (5000). The SW overhead remains unchanged, while the HW time per generation is remarkably consistent across all images at approximately 2.617 ms/generation, confirming that the hardware throughput is fully determined by the fixed algorithmic parameters and is independent of image content.

Beyond the constant per-generation latency reported in Table 9, the fitness-evolution curves in Figures 3 and 7 provide a more detailed view of how image complexity translates into effective computation time until convergence. In the synthetic experiments, the cases with 2, 4, 8, and 16 additional shapes (Figure 3a–c), and Figure 3d all exhibit early stabilization of the elite individual before generation 2000, corresponding to an estimated convergence time of approximately $2000 \times 2.617 \approx 5.23$ s of hardware computation. In contrast, the most cluttered synthetic scene (32 shapes, Figure 3e) shows a prolonged transient phase in which the elite fitness does not plateau until around generation 3000, increasing the effective convergence time to $3000 \times 2.617 \approx 7.85$ s a 50% increase in runtime with respect to the simpler configurations.

A similar complexity-driven pattern emerges in the real image experiments shown in Figure 7. Geometrically well-defined parabolic structures, such as the rainbow arc (Figure 7b) and retinal fundus vessel (Figure 7a), exhibit early elite fitness stabilization on the order of 1000–1500 generations, translating to estimated convergence times of approximately $1000 \times 2.617 \approx 2.62$ s and $1500 \times 2.617 \approx 3.93$ s of hardware execution, respectively. In contrast, more cluttered or ambiguous scenes, the road curve (Figure 7c), Mars (Figure 7d), and Earth (Figure 7e) display prolonged transients with plateaus near

2000–3000 generations (roughly 5.23–7.85 s), mirroring the synthetic clutter effect and highlighting scene complexity as the primary determinant of effective runtime. Overall, these fitness curves demonstrate that, despite the fixed 5000-generation budget, practical convergence occurs 40–60% earlier in simpler cases, enabling potential early-termination optimizations in future IP-core implementations.

Table 9. Per-Generation Execution Time of the Genetic-Algorithm IP Core: Hardware (HW) Time and Total Time per Generation.

Test Image	SW Time (ms)	HW Time/gen (ms)	Total Time/gen (ms)
Synthetic images (Figure 3)			
Figure 3a—2 shapes	0.3488	2.6176	2.9664
Figure 3b—4 shapes	0.3380	2.6174	2.9554
Figure 3c—8 shapes	0.3432	2.6169	2.9601
Figure 3d—16 shapes	0.3461	2.6167	2.9628
Figure 3e—32 shapes	0.3524	2.6180	2.9704
Real images (Figure 7)			
Figure 7a—Retinal fundus vessel	0.3786	2.6208	2.9994
Figure 7b—Rainbow arc	0.3662	2.6155	2.9817
Figure 7c—Road curve	0.4012	2.6166	3.0178
Figure 7d—Mars	0.3563	2.6187	2.9750
Figure 7e—Earth	0.3551	2.6170	2.9721
Mean	0.3586	2.6175	2.9761

3.6. IP Core Resource Consumption

Table 10 reports the hardware resource utilization of the Genetic IP core synthesized on the Zynq 7000 XC7Z020-CLG400C device. The IP core implements six functional stages of the genetic algorithm, each mapped to an independent hardware module whose resource footprint reflects its computational role.

The ini_population module initializes the population of 128 candidate individuals, each encoded as a set of four parabola parameters, consuming 907 LUTs (1.70%) and 689 flip-flops (0.65%). No DSPs are required since initialization relies exclusively on pseudo-random integer arithmetic implemented in LUT fabric. The Sum_fitnesses, Prob_by_indi, and Prob_Accu_by_indi modules implement the roulette-wheel selection preparation stage: the first accumulates the raw fitness values of all individuals (97 LUTs, 113 FFs), the second computes the normalized selection probability for each candidate (178 LUTs, 102 FFs), and the third builds the cumulative probability vector used to draw individuals stochastically (97 LUTs, 120 FFs). Together, these three modules consume only 372 LUTs (0.70%) and 335 flip-flops (0.31%), confirming that the probabilistic selection logic is lightweight in hardware.

The Elite_Search module locates the best individual in the current population by scanning all fitness scores, requiring 214 LUTs (0.40%) and 181 flip-flops (0.17%). The Selection module then draws the parent pool via the roulette mechanism, consuming 446 LUTs (0.84%) and 306 flip-flops (0.29%). The Crossover_and_Mutation module performs gene-level uniform crossover over the selected parent pairs, independently swapping each gene between paired parents with probability $P_{\mathrm{CROSS}} = 0.9$, and applies gene-level mutation with configurable probability $P_{\mathrm{MUT}} = 0.8$, accounting for 1852 LUTs (3.48%) and 1084 flip-flops (1.02%). All three modules operate entirely on integer and fixed-point logic, requiring zero DSPs.

Table 10. Resources consumption by Genetic IP core implemented in Zynq 7000 (XC7Z020-CLG400C).

Module	LUT (53,200)	%	Registers (106,400)	%	BRAM (280)	%	DSPs (220)	%
ini_population	907	1.70	689	0.65	0	0.00	0	0.00
Elite_Search	214	0.40	181	0.17	0	0.00	0	0.00
Sum_fitnesses	97	0.18	113	0.11	0	0.00	0	0.00
Prob_by_indi	178	0.33	102	0.10	0	0.00	0	0.00
Prob_Accu_by_indi	97	0.18	120	0.11	0	0.00	0	0.00
Selection	446	0.84	306	0.29	0	0.00	0	0.00
Crossover_and_Mutation	1852	3.48	1084	1.02	0	0.00	0	0.00
EvalNormfitness_parabola_param	7327	13.77	4410	4.14	0	0.00	20	9.09
Σ Genetic	19,216	36.12	11,895	11.18	187	66.00	31	14.09

The most resource-demanding stage is the fitness evaluation performed by EvalNormfitness_parabola_param, which assesses each candidate individual by computing the geometric fitness of its four parabola parameters (h, k, a, θ) against the set of detected edge pixels. This module alone consumes 7327 LUTs (13.77%), 4410 flip-flops (4.14%), and all 20 DSPs (9.09%) allocated to the design, reflecting the arithmetic intensity of its pipelined inner loop that evaluates up to 800 pixels per individual per generation. The remaining 11 DSPs at the top level of Genetic are used for address and index computations in the population management logic.

Overall, the Genetic IP core totals 19,216 LUTs (36.12%), 11,895 flip-flops (11.18%), 31 DSPs (14.09%), and 187 BRAMs (66.00%). The primary contributor to the high BRAM usage is the edge index buffer (128 BRAMs), which caches up to 65,536 pixel binary values for single-cycle random access during fitness evaluation, and by the image buffer (32 BRAMs), which stores the edge-pixel coordinate array. The remaining LUT and DSP headroom confirms that the IP core can be integrated alongside the AXI memory interconnect and processing system within a single Zynq 7000 device without resource contention.

The Vitis HLS synthesis report, Table 11, provides a worst-case latency estimate of 3.054 ms per generation, computed under the assumption that all 128 individuals are evaluated against the maximum of 800 edge pixels each. However, this represents the most pessimistic scenario: during the fitness evaluation, the projected parabola points are computed from the four parameters (h, k, a, θ) and tested against the edge-pixel map; if a projected point falls outside the image boundaries, it is discarded and does not contribute to the inlier count. Consequently, individuals whose parabola parameters place a significant portion of the curve outside the image dimensions result in fewer valid pixel evaluations, reducing the computing time.

This behavior is consistent with the measured hardware execution time of 2.979 ms, which is 2.44% below the HLS estimate, corresponding to an effective average of approximately 385 valid pixels per individual across the test dataset. The difference of 0.075 ms (7444 cycles) between the theoretical and measured values thus reflects the average number of out-of-bounds projections discarded per generation, confirming that the HLS worst-case bound is tight and that boundary clipping is a minor but measurable factor in the runtime of the fitness evaluation stage.

Table 11. Latency per module of the Genetic IP core (clock period = 10 ns, 100 MHz).

Module	Latency (Cycles)	Latency (µs)	Calls/ Generation	Total Cycles
ini_population	4737	47.37	1	4737
Elite_Search	641	6.41	1	641
Sum_fitnesses	1025	10.25	1	1025
Prob_by_indi	2435	24.35	1	2435
Prob_Accu_by_indi	1025	10.25	1	1025
Selection	68,609	686.09	1	68,609
Crossover_and_Mutation	17,730	177.30	1	17,730
EvalNormfitness_parabola_param	1634	16.34	128	209,152
Σ Genetic		3053.54		305,354

3.7. Comparison with Similar Works

It is worth noting that several other approaches have been proposed in the literature to address the parabola detection problem. For instance, methods based on matched filtering [33], genetic-based inverse voting Hough transforms [22], edge-gradient dimensionality reduction [34], and specialized discretization schemes for open conics [35] have been extensively explored. However, these works primarily focus on theoretical algorithmic optimization—such as reducing memory storage, avoiding the traditional $O(N^4)$ complexity of the Hough transform, or analyzing parameter discretization—and do not report quantitative, replicable execution times. Consequently, the quantitative performance comparison in this study is strictly based on the EDA-based method [1], Randomized Sampled Consensus (RANSAC), Hough transform, and MIPAV, which provide the most direct, comprehensive, and measurable baseline for execution time across both synthetic and real images.

Table 12 presents a performance and methodological comparison between the proposed hardware architecture and the software-based methods reported by Guerrero-Turrubiates et al. [1], namely the Hough transform, MIPAV, RANSAC, and the Estimation of Distribution Algorithm (EDA/UMDA). It is important to note that these results should be regarded as indicative rather than as a strict head-to-head benchmark. Beyond execution time, the table also highlights a fundamental difference in problem dimensionality: all baseline methods operate in a 3-dimensional axis-aligned parameter space (h, k, a), whereas the proposed FPGA architecture searches the full 4-dimensional space (h, k, a, θ), enabling the detection of arbitrarily rotated parabolas that the software baselines cannot represent.

When compared against the non-evolutionary baselines on synthetic images, the proposed FPGA implementation (13.08 s for 5000 generations) is faster than MIPAV (37.68 s) by a factor of 2.88×, and faster than RANSAC (16.35 s) by a factor of 1.25×, while solving a strictly harder, higher-dimensional problem on a resource-constrained embedded platform. The Hough transform reports a per-pixel time of 8.82 s/px, a metric that is not directly comparable to total execution time; however, for any image of non-trivial size, the accumulated cost renders it computationally prohibitive. Although the software approach via UMDA reports lower total execution times on synthetic images (1.41 s to 3.59 s), this advantage does not reflect an equivalent computational problem: it stems from the combined effect of the high sequential clock frequencies of CPUs and the evaluation of a geometrically simpler, lower-dimensional model on a platform with more resources than a System-on-Chip (SoC) FPGA.

The EDA method relies on a point-based sampling strategy in which three random edge pixels are selected to analytically solve for the parabola parameters. From a mathematical standpoint, three coplanar points uniquely determine an axis-aligned parabola but are insufficient to recover the rotation angle θ of a generic rotated conic, effectively constraining

the search to a 3-dimensional parameter space (h, k, a). A direct consequence of this formulation is that a significant fraction of the generated candidates are geometrically invalid, because the parameters are derived from random triplets of pixels rather than constrained within the image domain, the resulting vertex coordinates or curve trajectory frequently fall entirely outside the image boundaries. These invalid candidates consume evaluation resources without contributing useful information to the search, and their frequency varies with image content, a factor that likely contributes to the large execution time variability observed across images (with a time range of 4.4 times, from 1.41 s to 6.18 s).

Table 12. Performance and Methodological Comparison with State-of-the-Art Parabola Detection.

Work/Method	Platform	Candidate Gen.	DoF	Orientation	Test Images	Time (s)
Guerrero et al. [1]/Hough Transform	PC (Intel i5)	Accumulator	3	Axis-aligned	Synthetic	8.82/px †
Guerrero et al. [1]/MIPAV	PC (Intel i5)	Hough-based	3	Axis-aligned	Synthetic	37.68
Guerrero et al. [1]/RANSAC	PC (Intel i5)	Random subset	3	Axis-aligned	Synthetic	16.35 ‡
Guerrero et al. [1]/UMDA	PC (Intel i5)	3 Points (UMDA)	3	Axis-aligned	Synthetic	1.41–3.59
Guerrero et al. [1]/UMDA	PC (Intel i5)	3 Points (UMDA)	3	Axis-aligned	Real (Retina)	4.5838
Proposed Work	**FPGA (HW/SW)**	(h, k, a, θ)	**4**	**Rotated**	**Synthetic**	**13.08** °
	FPGA (HW/SW)	(h, k, a, θ)	**4**	**Rotated**	**Real Images**	**13.09** °
	FPGA (HW/SW)	(h, k, a, θ)	**4**	**Rotated**	**Real (Retina)**	**2.6175** *

* Proposed time is an approximation (2.6175 ms per generation and 1000 generations). ° Total time (DMA image transfer + FPGA GA execution) for 5000 generations; strictly deterministic. † Execution time is reported per pixel. ‡ RANSAC execution time corresponds to 5000 iterations.

Notably, for the retinal fundus image dataset, the proposed FPGA architecture achieves a total processing time of approximately 2.6175 s (using 1000 generations), which is 1.75× faster than the UMDA baseline of 4.5838 s on the same image type. This result is particularly relevant because retinal images represent the only common real-image benchmark reported by Guerrero-Turrubiates et al., providing a direct and equivalent comparison point. The proposed system thus not only matches the application domain of the software EDA on real medical images, but surpasses it in total execution time while simultaneously solving a harder detection problem with an additional degree of freedom and guaranteeing that every evaluated individual encodes a valid parabolic curve in the image.

In contrast, the proposed Genetic Algorithm IP core operates in the full 4-dimensional parameter space (h, k, a, θ), enabling the detection of arbitrarily rotated parabolas. Crucially, the chromosome parameters are bounded and defined directly within the image coordinate system, so every generated individual encodes a valid parabolic curve that exists within the image boundaries. This geometric guarantee eliminates wasted evaluations and ensures that the full computational budget is devoted to meaningful candidates. Instead of fragile point-to-point analytical solutions, fitness is assessed globally by projecting the encoded curve onto the binary edge image and counting inlier pixels, providing inherent robustness against noise, partial occlusion, and edge discontinuities.

From a timing perspective, the key distinction between the two approaches is not raw speed, but temporal determinism and platform suitability. The FPGA implementation runs on a Zynq 7000 SoC, a resource-constrained embedded device whose programmable logic operates at 100 MHz, far below the multi-GHz clock rates of the CPUs used in the software baseline. Despite this clock frequency disadvantage, the hardware architecture delivers a strictly constant per-generation latency of 2.979 ms regardless of image content, yielding a fixed total processing time of approximately 13.08 s for a workload of 5000 generations. The software EDA, running on a far more powerful platform, exhibits execution times that vary by a factor of 4.4 times across images, a variability that is inconsistent with embedded and real-time applications requiring bounded, predictable response times. The proposed

architecture thus solves a harder geometric problem, on a more constrained platform, with strictly deterministic timing, a combination that the software baseline cannot provide.

4. Conclusions

This work, to the best of our knowledge, presents the first deterministic embedded FPGA prototype for an FPGA IP core that implements a genetic algorithm to detect rotated parabolas in images on an embedded SoC platform. The proposed architecture was validated on 640 × 480 binary edge images, achieving mean accuracies of 98.47 and 95.23% and a success rate of up to 96% on synthetic images, as well as qualitatively consistent convergence on real images, where the absence of annotated ground truth precludes quantitative accuracy claims. Tests with salt-and-pepper noise and a comparative analysis against related works confirmed the robustness and competitiveness of the implementation, while the measured resource consumption still leaves room to extend the design to the target FPGA. Regarding the genetic parameters, we found that crossover and mutation values of 0.9 and 0.8, respectively, were crucial for the GA to converge on the best solution. The most challenging parameter in our experience was the aperture parameter a, as the best individuals tended to fit only half of the parabolic shape. With these values, the proposed architecture demonstrated consistent convergence toward valid parabolic solutions across all tested images, confirming the feasibility of the design as a deterministic embedded FPGA IP core prototype for rotated parabola detection in the full four-dimensional parameter space (h, k, a, θ). The FPGA IP-core delivers a strictly deterministic per generation latency of 2.979 ms; for 5000 generations, the total system execution time comprising DMA image transfer and hardware GA execution on the programmable logic at 100 MHz, with image pre-processing performed offline on an external CPU and not included in this boundary, is approximately 13.08 s. While the current resource consumption still leaves room to extend the design to the target FPGA, this reflects the constraints of the SoC platform rather than a fundamental algorithmic limitation. Future work will focus on developing a native software-based implementation to enable a fair hardware-acceleration comparison, on timing optimization toward real-time operation, and on extending the model to multi-parabola detection and other conic sections such as hyperbolas.

Author Contributions: Conceptualization, F.J.I.-L., L.d.C.C.-V. and H.R.-G.; Methodology, F.J.I.-L., V.F.-P. and L.d.C.C.-V.; Software, F.J.I.-L.; Validation, F.J.I.-L. and L.d.C.C.-V.; Formal analysis, F.J.I.-L., V.F.-P., L.d.C.C.-V. and H.R.-G.; Investigation, F.J.I.-L., V.F.-P. and H.R.-G.; Resources, F.J.I.-L.; Data curation, F.J.I.-L., V.F.-P. and H.R.-G.; Writing—original draft, H.R.-G.; Writing—review and editing, F.J.I.-L.; Visualization, F.J.I.-L.; Supervision, F.J.I.-L.; Project administration, F.J.I.-L.; Funding acquisition, F.J.I.-L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Guerrero-Turrubiates, J.d.J.; Cruz-Aceves, I.; Ledesma, S.; Sierra-Hernandez, J.M.; Velasco, J.; Avina-Cervantes, J.G.; Avila-Garcia, M.S.; Rostro-Gonzalez, H.; Rojas-Laguna, R. Fast Parabola Detection Using Estimation of Distribution Algorithms. *Comput. Math. Methods Med.* **2017**, *2017*, 6494390. [CrossRef] [PubMed]
2. Mukhopadhyay, P.; Chaudhuri, B.B. A survey of Hough Transform. *Pattern Recognit.* **2015**, *48*, 993–1010. [CrossRef]
3. Duda, R.O.; Hart, P.E. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM* **1972**, *15*, 11–15. [CrossRef]
4. Ballard, D. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognit.* **1981**, *13*, 111–122. [CrossRef]

5. Fischler, M.A.; Bolles, R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In *Readings in Computer Vision*; Fischler, M.A., Firschein, O., Eds.; Morgan Kaufmann: San Francisco, CA, USA, 1987; pp. 726–740. [CrossRef]
6. Ayala-Ramirez, V.; Garcia-Capulin, C.H.; Perez-Garcia, A.; Sanchez-Yanez, R.E. Circle detection on images using genetic algorithms. *Pattern Recognit. Lett.* **2006**, *27*, 652–657. [CrossRef]
7. Cuevas, E.; Ortega-Sánchez, N.; Zaldivar, D.; Pérez-Cisneros, M. Circle detection by Harmony search optimization. *J. Intell. Robot. Syst.* **2012**, *66*, 359–376.
8. Cruz-Aceves, I.; Hernandez-Aguirre, A.; Valdez, S.I. On the performance of nature inspired algorithms for the automatic segmentation of coronary arteries using Gaussian matched filters. *Appl. Soft Comput.* **2016**, *46*, 665–676. [CrossRef]
9. Scitovski, R.; Sabo, K.; Nikić, P.; Majstorović, S. A new efficient method for solving the multiple ellipse detection problem. *Expert Syst. Appl.* **2023**, *222*, 119853. [CrossRef]
10. Xu, G.; Wang, Y.; Chen, F.; Shen, H.; Li, X. Accurate and Robust Three-Intersection-Chord-Invariant Ellipse Detection. *IEEE Trans. Image Process.* **2025**, *34*, 2392–2407. [CrossRef] [PubMed]
11. Hauschild, M.; Pelikan, M. An introduction and survey of estimation of distribution algorithms. *Swarm Evol. Comput.* **2011**, *1*, 111–128. [CrossRef]
12. Mühlenbein, H.; Paaß, G. From recombination of genes to the estimation of distributions I. Binary parameters. In *Proceedings of the Parallel Problem Solving from Nature—PPSN IV*; Voigt, H.M., Ebeling, W., Rechenberg, I., Schwefel, H.P., Eds.; Springer: Berlin/Heidelberg, Germany, 1996; pp. 178–187.
13. Deliparaschos, K.M.; Oliva, G. FPGA-Accelerated-Greedy Ant Colony Optimisation for Maritime Bearings-Only Nonlinear Target Motion Analysis. *IEEE Access* **2025**, *13*, 190027–190039. [CrossRef]
14. Rojas-Muñoz, L.F.; Sánchez-Solano, S.; García-Capulín, C.H.; Rostro-González, H. Embedded system implementation of an evolutionary algorithm for circle detection on programmable devices. *Comput. Electr. Eng.* **2022**, *99*, 107714. [CrossRef]
15. Iñiguez-Lomeli, F.J.; Garcia-Capulin, C.H.; Rostro-Gonzalez, H. A hardware architecture for single and multiple ellipse detection using genetic algorithms and high-level synthesis tools. *Microprocess. Microsyst.* **2024**, *111*, 105106. [CrossRef]
16. Zeng, K.; Ma, Q.; Wu, J.W.; Chen, Z.; Shen, T.; Yan, C. FPGA-based accelerator for object detection: A comprehensive survey. *J. Supercomput.* **2022**, *78*, 14096–14136. [CrossRef]
17. Martelli, S.; Marzotto, R.; Colombari, A.; Murino, V. FPGA-based robust ellipse estimation for circular road sign detection. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, San Francisco, CA, USA, 13–18 June 2010; pp. 53–60. [CrossRef]
18. Dohi, K.; Hatanaka, Y.; Negi, K.; Shibata, Y.; Oguri, K. Deep-pipelined FPGA implementation of ellipse estimation for eye tracking. In Proceedings of the 22nd International Conference on Field Programmable Logic and Applications (FPL), Oslo, Norway, 29–31 August 2012; pp. 458–463. [CrossRef]
19. Ayachi, R.; Afif, M.; Said, Y.; Abdelali, A.B. Traffic Sign Detection for Green Smart Public Transportation Vehicles Based on Light Neural Network Model. In *Computational Intelligence Techniques for Green Smart Cities*; Lahby, M., Al-Fuqaha, A., Maleh, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 95–106. [CrossRef]
20. Oloumi, F.; Rangayyan, R.M. Detection of the temporal arcade in fundus images of the retina using the hough transform. In Proceedings of the 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Minneapolis, MN, USA, 3–6 September 2009; pp. 3585–3588. [CrossRef] [PubMed]
21. Oloumi, F.; Rangayyan, R.M.; Ells, A.L. Parametric representation of the retinal temporal arcade. In Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine, Corfu, Greece, 3–5 November 2010; pp. 1–4. [CrossRef]
22. Maalmi, K.; El Ouaazizi, A.; Benslimane, R.; Lew Yan Voon, L.; Diou, A.; Gorria, P. Detecting parabolas in ultrasound B-scan images with Genetic-Based Inverse Voting Hough Transform. In Proceedings of the 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing, Orlando, FL, USA, 13–17 May 2002; Volume 4, pp. IV-3337–IV-3340. [CrossRef]
23. Zhang, S. Real Time Image Processing on FPGAs. Ph.D. Thesis, University of Liverpool, Liverpool, UK, 2018.
24. Mansouri, A.E.; Youssef, H.B.; Saidi, L. Real-time object detection using Xilinx Zynq FPGA and tensor flow framework. *Int. J. Res. Circuits Devices Syst.* **2024**, *5*, 13–18. [CrossRef]
25. Zhang, J.; Xiong, S.; Liu, C.; Geng, Y.; Xiong, W.; Cheng, S.; Hu, F. FPGA-Based Feature Extraction and Tracking Accelerator for Real-Time Visual SLAM. *Sensors* **2023**, *23*, 8035. [CrossRef] [PubMed]
26. Padua, B. Hybrid implementation of evolutionary algorithms in FPGAs for automatic generation of morphological image filters. *Prz. Elektrotech.* **2020**, *1*, 14–18. [CrossRef]
27. Mazouz, A.; Le, D.H.; Nguyen, V.T. An FPGA compiler for on-the-fly adaptive CNN deployment and reconfiguration. *arXiv* **2025**, arXiv:2504.08534. [CrossRef]
28. Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [CrossRef]

29. Wang, T. "Retinal Vessel", Mendeley Data, V1. 2023. Available online: https://data.mendeley.com/datasets/frv89hjgrr/1 (accessed on 29 April 2026).
30. Xie, S.; Tu, Z. Holistically-Nested Edge Detection. *arXiv* **2015**, arXiv:1504.06375. [CrossRef]
31. Balla, S. Deep Learning Based Edge Detection Using HED. 2021. Available online: https://github.com/bnsreenu/python_for_microscopists/tree/master/290-Deep%20Learning%20based%20edge%20detection%20using%20HED (accessed on 5 January 2026).
32. DigitalSreeni. Deep Learning Based Edge Detection Using Holistically Nested Edge Detection (HED)—Video Tutorial. YouTube. 2021. Available online: https://www.youtube.com/watch?v=UIrvEG9Oj1s (accessed on 5 June 2026).
33. Petcher, P.; Dixon, S. Parabola detection using matched filtering for ultrasound B-scans. *Ultrasonics* **2012**, *52*, 138–144. [CrossRef] [PubMed]
34. Jafri, M.Z.M.; Deravi, F. Efficient algorithm for the detection of parabolic curves. In *Proceedings of the Vision Geometry III*; Melter, R.A., Wu, A.Y., Eds.; International Society for Optics and Photonics, SPIE: Bellingham, WA, USA, 1995; Volume 2356, pp. 53–61. [CrossRef]
35. de Macedo, M.M.G.; Salas, J.; Conci, A. Hough Transform Voting Scheme for Detection of Parabolas and Open Conics in Images. In *Proceedings of the ICGG 2020-Proceedings of the 19th International Conference on Geometry and Graphics*; Cheng, L.Y., Ed.; Springer: Cham, Switzerland, 2021; pp. 407–418.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.