Canny Genetic Programming (CGP) for Edge Detection

Daniel van Zyl
Dept of Computer Science
University of Pretoria
Pretoria, South Africa
u21446459@tuks.co.za

Thambo Nyathi
Dept of Computer Science
University of Pretoria
Pretoria, South Africa
0000-0003-4676-6063

Nelishia Pillay
Dept of Computer Science
University of Pretoria
Pretoria, South Africa
0000-0003-3902-5582

***Abstract*—Edge detection is a foundational task in image processing. Modern methods are driven by deep learning, which typically requires large amounts of labelled training data. Additionally, these methods require large computational resources for training. Conventional methods, although not as accurate as deep learning approaches, offer quick execution times and may not always require labelled data. Among conventional methods, the Canny Edge Detector is an industry standard. Despite its efficiency and widespread adoption, the Canny detector has notable limitations. Its gradient-based formulation struggles with semantic boundary localisation. Furthermore, CED is vulnerable to noise interference, which results in spurious edge results. Additionally, as a parameterised algorithm, the Canny detector's performance is dependent on its manual configuration. Selection of optimal parameters is context-dependent and lacks an automated framework for diverse noise profiles. This study proposes a hybrid approach that combines the Canny Edge Detector and Genetic Programming for edge detection. To evaluate the proposed approach, a subset of images was obtained from the Berkeley Segmentation Dataset. The performance of edge maps evolved by the proposed approach was compared with that of the standard Canny detector. Three metrics, namely recall, precision, and F-measure, were used for comparison. On average across all images, the proposed approach was found to perform significantly better than the standard Canny algorithm. Additionally, the computational training times were found to be within reason.**

***Index Terms*—Edge Detection, Genetic Programming, Canny Edge Detector, Image processing.**

I. Introduction

Edge detection is a foundational technique in image processing and computer vision. It is mainly used to identify the boundaries of objects within an image. Edge detection methods work by detecting sharp discontinuities in pixel intensity, colour, or texture. These methods are broadly categorised as either conventional (hand-crafted) or deep learning-based. [1]. Conventional approaches are deterministic algorithms built on mathematical operations. They are designed to calculate the rate of change (the gradient) of pixel intensity values. The principal idea is that a rapid change in intensity corresponds to an edge. Mathematically, this is calculated by convolving a small, pre-defined matrix called a filter with the image. The Sobel [2] and Prewitt [3] filters are classic examples of this. These filters are explicitly designed to approximate the image's first derivative. The gradients are calculated along both the horizontal and vertical axes to derive their combined magnitude and directional vectors. A high magnitude signifies a likely edge pixel. Second-derivative methods such as the Marr-Hildreth [4] go a step further by evaluating zero-crossings, which pinpoint the edge's centre. State-of-the-art deep learning approaches, such as the Holistic-nested Edge Detector (HED) [5], treat the image as a single unit instead of pixel-by-pixel [6].

Among conventional techniques, the Canny Edge Detector (CED) [7] is considered an industry standard. It remains relevant, serving as a crucial benchmark for evaluating new algorithms [8]. The Canny detector is a multi-stage algorithm where the strict order of execution is critical, as each step prepares the data for the next. Its primary advantages over deep learning approaches include high execution speed and the elimination of training requirements, enabling immediate deployment in real-world applications. As a result, the Canny detector is frequently integrated into high-level computer vision pipelines such as those for object detection, where it serves as a reliable preprocessing step for edge segmentation [9].

Despite its widespread use, the Canny detector suffers from inherent limitations. Depending on the type and configuration of the noise filtering technique used, it is sensitive to noise [10]. Furthermore, manually configuring the algorithm's parameters for optimal performance across different images is a significant challenge [11]. Owing to its widespread adoption, improvements of the Canny detector remain a subject of active research [12], [13]. This study proposes using Genetic Programming (GP) [14] to automatically evolve and optimise the Canny edge detector. We hypothesise that GP can perform structural optimisation to discover a functionally better variant of the original algorithm. To validate our approach, the evolved detector was compared to the conventional Canny method on benchmark image data. The performance of the evolved detector was evaluated using precision, recall and F-measure metrics. Experimental results demonstrate that the GP-evolved Canny outperforms the standard Canny detector across all evaluated metrics, achieving statistically significant improvements.

A. Motivation

The aim of this study is not to evolve detectors that compete with deep learning architectures, but those that can seamlessly replace the Canny detector within established image processing pipelines. We aim to provide a more robust adaptive version of Canny that addresses its inherent vulnerabilities.

The main contribution of this research is an evolutionary edge detector framework for simultaneously optimising the structural and parametric aspects of the Canny Edge Detector using Genetic Programming.

The rest of the paper is structured as follows: Section II provides the background, and Section III presents the proposed approach. Sections V and VI outline the results and conclusion, respectively.

II. Background

This section presents a review of fundamental concepts, including edge detection, the Canny edge detector, enhancements to the Canny operator proposed in the literature, an introduction to genetic programming, and its application in image processing.

A. Edge Detection

Edge detection is the process of identifying and extracting boundary information in an image. It works by analysing pixel intensities to determine where there is a drastic change in intensities between pixels. A pixel is classified as an edge pixel if its intensity changes drastically compared to the surrounding pixels. Edge detection often forms the foundation of image-processing technologies. The three common stages of edge detection are preprocessing, feature extraction, and post-processing [15]. Various edge detection techniques have been developed; among the most widely adopted is the Canny edge detection algorithm.

B. Canny Edge Detector

According to [7], generally, an edge detection technique must meet the following three performance criteria:

a) *Good detection* the approach must have a high probability of correctly identifying true edges and a low probability of misclassifying non-edges as edges.
b) *Good localisation* points identified as edges must be as close as possible to the centre of the real edge.
c) *Non-duplication* only one true edge should be identified and marked.

Algorithm 1 outlines the steps of the Canny edge detector. The algorithm's first step, after the image is input, is a noise-reduction step. Noise may be present in an image due to several factors, such as poor illumination during image capture or hardware limitations of the device. If an image contains noise, the edge detection results may yield unreliable outputs, such as false edges or distorted true edges. Noise reduction is performed by convolving the image's pixel matrix with a Gaussian filter. This process, known as Gaussian smoothing or blurring, is the most commonly used approach for noise reduction for the Canny algorithm. An adjustable filter smoothing parameter, σ, determines the Gaussian filter radius. The blurring effect is directly proportional to σ. The smaller the value of σ, the smaller the blur effect. The value of σ is a design decision and depends on the image and the amount of noise present. Unfortunately, this smoothing process can also blur image edges, which is an undesirable effect [16]. The second step of the algorithm is to locate the regions where the pixel intensity changes the most. This is achieved by computing the local gradient vectors using first-order difference filters. The Sobel filter is the most commonly used first-order derivative filter at this stage of the algorithm [17]. Roberts and Prewitt filters are other examples of first-order derivative filters that can be used [18]. It has been shown that first-order derivative filters are vulnerable to noise [17]. Thus, improvements of the Canny algorithm using the second-order derivatives [18] have been proposed. The third step of the algorithm is non-maximum suppression, a technique used to thin out the detected edges in an image by preserving only the local maxima of the gradient magnitude. The image from Step 2 contains non-uniform edges, as some are thick and others are thin. Non-maximum suppression is applied to address these issues. It is used to isolate candidate edge pixels identified as part of an edge by suppressing all non-maximum pixels along the direction of the gradient. In so doing, it preserves only pixels representing a local maximum along the identified edge. The magnitude and direction are important at this stage. If a pixel has a higher intensity than its neighbours along the gradient vector, it is retained as part of the edge. The intensity values of its neighbours are set to zero. Step 4, known as double thresholding, identifies three pixel types: strong, weak, and non-relevant. Two threshold values, t_{hi} and t_{lo}, where ($t_{hi} > t_{lo}$)are specified. Pixels whose intensity values lie above t_{hi} are considered strong, and those that lie between t_{hi} and t_{lo} are regarded as weak. Pixels whose intensity values are less than t_{lo} are classified as irrelevant (i.e., non-edges). Due to its reliance on user-fixed double threshold values, the Canny algorithm is considered non-adaptable. The final step, *edge tracking by hysteresis*, converts weak pixels identified in step 3 into strong pixels. A weak pixel is converted into a strong pixel only if any one of its neighbours is strong.

Algorithm 1 Canny Edge Detector Algorithm [7]

1: **Input** Image
2: Filter the input image using a Gaussian function.
3: Calculate the image gradient.
4: Apply non-maximum suppression.
5: Verify and connect the edges.
6: **Output** EdgedImage

A critical survey of recent literature indicates a shift from the standard static Canny towards adaptive hybrid frameworks. The advancements mostly target three notable weaknesses of the Canny detector, namely, noise sensitivity, suboptimal threshold selection and limitations of gradient-based localisation. To address the limitations of the Gaussian noise filter, more sophisticated filters have been proposed as a replace-

ment, such as the Meanshift and Guided filters [1], [19]. To eliminate the need for manual configuration, metaheuristics have been employed as shown in [20] and [21], where a Genetic Algorithm and Particle Swarm Optimisation were respectively used to evolve optimal upper and lower hysteresis thresholds. While in [22], Ant Colony Optimisation was used to dynamically determine the edge paths instead of the standard hysteresis approach. In [18], replacing the standard image gradients approach with the use of Gravitational field intensity was proposed. Hybridising the standard Canny with other detectors, such as the Hough Transform, has been proposed to improve its performance based on structural improvements [23].

Current literature mainly proposes the replacement of Canny components and/or parameter optimisation. The fundamental logic of the Canny detector remains static, with the proposed approaches searching for optimal values, such as scalar values of hysteresis thresholding. If the underlying Canny pipeline is not suited for a specific noise profile, no amount of parameter tuning can be applied to overcome the structural deficiencies. We hypothesise that the use of Genetic Programming can overcome these deficiencies through the addition of structural optimisation. Genetic Programming is able to treat the edge detection process as an evolvable computational tree. GP has the potential to discover non-linear combinations of gradients and filters which can address the limitations of the Canny detector. The next section presents genetic programming.

C. Genetic Programming

Genetic programming is an evolutionary algorithm that searches a program space. GP represents individuals as syntax trees, where internal nodes are usually functions, and the leaf nodes are terminals obtained from the problem.

Algorithm 2 Genetic Programming Algorithm [14]

```
Initialise a population of individuals (program).
Evaluate the fitness of each program.
while Stopping criterion is not met do
    Select parents.
    Create offspring using the crossover operator.
    Create offspring using the mutation operator.
    Evaluate the fitness of each offspring.
    Update the population.
end while
return fittest individual
```

The algorithm is initialised by generating a population of randomly generated individuals (programs). Each program is executed on a target problem, and a fitness function is then used to evaluate the fitness of each program. If the desired fitness is not obtained, the algorithm enters a loop, and a selection method is used to select individuals from the current population to act as parents to generate offspring. The offspring are evolved using genetic operators, namely crossover and mutation. The fitness of each offspring is evaluated, and they replace individuals of the old population. This process iterates until the target fitness is obtained or a specified stopping criterion is achieved.

D. Genetic Programming and Edge Detection

Research has shown that genetic programming is highly effective for classification and regression tasks. Edge detection can be considered as a classification task in which a pixel must be categorised as either part of an edge or a non-edge. Most studies utilising Genetic Programming for edge detection use GP to manipulate a portion of the image, typically a window with a pre-specified size (e.g., 3×3), and examine the pixels within the selected area. The majority of studies use GP to evolve mathematical expressions or image filter programs that determine whether a pixel is part of an edge. The filters could be pre-existing or evolved by GP. In [24], an approach was proposed utilising an unsupervised learning approach that combined Genetic Programming and Artificial Ant Colony. This method operated on a single image without requiring ground-truth data to evolve detectors. The approach metaphorically represented edges in the image as "food" for artificial ants to detect. The algorithm employed a function set comprising logical operations and subprograms that evaluated pixel characteristics and performed actions based on randomly generated conditions. The GP-based approach outperformed the Sobel filter. In [25], an extension of the approach outlined in [24], termed Modified Genetic Programming (MGP), was presented. Key features of MGP were centred on its ability to simultaneously evolve edge detectors and rules, leveraging strongly typed GP. Notably, MGP's computational efficiency is maintained by avoiding the need to restart for subsequent detector evolution. One of the strengths of GP is feature extraction. In [26], the authors used GP to extract a set of raw pixels from an image to construct first and second-order filters. In a regression setup, GP reduces the error margin between the ground-truth edge image and the technique it employs. In [27], GP was used to fine-tune numerous Gaussian filter parameters, which were then applied for edge detection.

While GP for edge detection has shown significant promise, it has yet to achieve widespread adoption as a standard industry technique. This is due to the subjective nature of edges in images. No discernible pattern can be predicted regarding edges, and the sensitivity of variance in pixel intensities compounds the difficulty. To improve its edge detection capabilities, we aim to combine GP with the Canny edge detector.

E. Genetic Programming Automated Algorithm Design

Due to the flexibility of its representation, GP is an effective approach for the automated design of other algorithms [28]. An advantage of GP in automated design is that it can simultaneously search for optimal parameters and structures. According to [29], to effectively solve a problem using GP, one must choose the most suitable variant from the several that exist. To preserve algorithmic integrity when using Genetic Programming for automated design, Strongly Typed Genetic Programming (STGP) [30] is commonly used to ensure that all generated programs are syntactically well-formed. Genetic

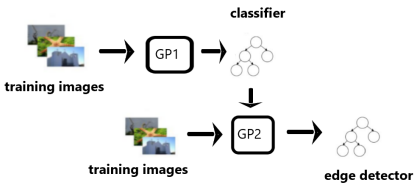

Fig. 1: Architecture Proposed Approach

Programming was first demonstrated for automated design by evolving entire analogue electrical circuits, including both their structure and component values [31]. In [32], [33], strongly-typed GP was used to evolve image classification algorithms. The approach specified a function set of image processing operations and used the input image as a terminal, allowing GP to evolve both the algorithmic structure and the hyperparameters.

III. Canny Genetic Programming (CGP)

This study proposes using GP in two stages, as illustrated in Figure 1. In the first stage, GP1, GP evolves classifiers for edge detection, similarly to the approach presented in [34]. The best evolved classifier is then passed on to GP2. The second stage, GP2, uses Strongly Typed Genetic Programming (STGP) to evolve a complete Canny-like edge detection pipeline. By enforcing type constraints, STGP ensures structural validity, enforcing that every evolved program adheres to the Canny detector pipeline.

A. Tree Generation

To ensure structural diversity for both GP1 and GP2, the initial population is created using the grow initialisation method.

B. Function and Terminal Sets

The source input image is represented as a greyscale intensity matrix $\mathbf{X}$. Formally, defined as:

$$\mathbf{X} \in \mathbb{Z}^{n \times m}, \quad \text{where } x_{i,j} \in \{0, 1, \ldots, 255\} \tag{1}$$

where n and m represent the image height and width in pixels, respectively. Each element $x_{i,j}$ denotes the 8-bit greyscale intensity value at the spatial coordinate (i, j). To ensure the evolved expressions produced valid pixel intensities, the output of the function set was constrained to the interval $[0, 255]$ via a saturation function $\sigma(x) = \max(0, \min(255, x))$.

The terminal set of GP1 contains the input image and one or more random ephemeral constants (ephem), in the range [-10, 10]. The GP1 terminal set is: $\{x, ephem\}$.

The function set consisted of standard arithmetic operators: $\{+, -, \times, /, \text{sqrt, abs}\}$. To ensure numerical stability, a protected division operator was implemented. Rather than evaluating the divisor strictly against zero, a safety buffer defined by a small positive constant ϵ was employed. This prevents the generation of excessively large values or floating-point overflows that typically arise from near-zero divisors. The operator is formally defined as $f(a, b) = a/b$ if $|b| > \epsilon$, and 1 otherwise. In this implementation, ϵ was assigned a value of 10^{-6}. Consequently, any divisor falling within the interval $[-\epsilon, \epsilon]$ was treated as an exceptional case, returning a protected value of 1 to maintain the integrity of the fitness evaluation.

In GP2, the system evolves a complete Canny-like pipeline. The terminal set contains the input image and a single random float between [0, 1] used for dynamic parameter tuning of the filters. GP2 terminal set $\{x, rfn\}$.

The function set consists of core Canny operators, i.e., filters, gradient, direction, non-maximum suppression, double thresholding and the best classifier evolved in GP1. The functions and their arity were specified as follows

- Filter: 3
- Magnitude: 1
- Direction: 1
- Non-Maximum Suppression: 2
- Double Thresholding: 3
- GP1 component: 3
- COMB: 2

The best individual produced by GP1 gets transferred to GP2 as a function. The functions set is presented as: $\{FILTER, MAG, DIR, NMS, DT, GP1, COMB\}$. The *COMB* component is only used as the root of the tree. The reason is that the tree structures tended to be very linear, and introducing the COMB function as the root allowed the trees to exhibit a better branching structure.

1) Fitness Function: The fitness function used by both GP1 and GP2 algorithms was proposed in [34] and is shown in equation 2.

$$Fit = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{r_i p_i}{\alpha r_i + (1 - \alpha) p} \right) \tag{2}$$

Where *N* is the number of training images r_i and p_i are the recall and precision for image i, respectively. α is used to balance precision and recall. A high recall should correspond to correctly classified edge pixels, and a high precision to fewer pixels incorrectly classified as non-edge pixels. The goal is to prioritise recall because there are more non-edge pixels in an image, so the chance of high precision is good. Hence, α was set to 0.8 (as in [34]) for GP1 and recall and precision were kept equally important for GP2.

C. Selection Method

Tournament selection is used as the selection method for both GP1 and GP2.

D. Genetic Operators

Subtree crossover is used in both GP1 and GP2, where random points between parents are selected, and their respective subtrees are swapped. Grow mutation is used, which involves selecting a random point in a parent and replacing the subtree at that point with a newly generated subtree. For both GP1 and GP2, elitism is used to preserve the best-performing individuals.

E. Population Update and Termination

A generational evolutionary approach is used to update both genetic programming approaches, with the total number of generations used as the termination criterion of the algorithms.

IV. Experimental Setup

This section presents the experimental setup. It involves a description of the image dataset and the experimental parameters.

A. Baseline Canny

The industry standard Auto-Canny detector, which is available as a Python library, was adopted as the baseline for comparative analysis. This implementation uses a median-based heuristic to automate threshold selection. We refer to this as Canny.

B. Statistical Tests

To evaluate the statistical significance of performance between Canny Genetic Programming and Canny, the non-parametric one-tailed Wilcoxon signed-rank test was used. The test was conducted at the 95% confidence level.

C. Image Dataset

The Berkeley Segmentation Dataset (BSD500) [35] is a primary benchmark for edge detection due to its diverse collection of natural images that reflect real-world visual complexity. It also offers human-annotated ground truth images, enabling objective performance comparison between different algorithms. This study used 19 training images and 11 test images from the BSD500 dataset. Training image IDs: {2092, 8049, 8143, 12003, 12074, 15004, 15088, 16052, 20008, 22013, 22090, 22093, 23025, 23080, 23084, 24004, 24063, 25098, 26031 }. Testing image IDs: {2018, 3063, 6046, 8068, 10081, 16068, 41029, 41085, 41096, 43033, 43051 }. These images were selected based on their diverse scenes and varying levels of visual complexity.

D. Parameter Settings

The Canny parameters are summarised in Table I.

CGP parameters were tuned using the one-factor-at-a-time (OFAT) method on images {2092, 12074, 24063, 25098, 22090}. Starting with baseline values from the literature [26], each parameter was independently varied while holding the others constant. The value for each parameter that yielded the best performance was then selected as its final setting. Table II outlines the parameters used for both the GP implementations.

TABLE I: Parameters for Canny Detector

Parameter	Notation	Value
Input Image	$\mathbf{X}$	Greyscale ($n \times m$)
Median Intensity	v	median($\mathbf{X}$)
Sigma Coefficient	σ	0.33
Lower Threshold	T_L	$\max(0, (1-\sigma) \cdot v)$
Upper Threshold	T_H	$\min(255, (1+\sigma) \cdot v)$

TABLE II: Canny Genetic Programming Parameters

Parameter	GP1 Value	GP2 Value
Population size	100	100
Initial tree depth	4	6
Crossover rate	0.65	0.55
Mutation rate	0.3	0.4
Elitism rate	0.05	0.05
Tournament size	3	3
Max generations	30	30
α parameter	0.8	0.6

E. Technical Specifications

The algorithm was developed and tested on an ASUS Vivobook Pro notebook with the following specifications: CPU: AMD Ryzen 7 5800H with Radeon Graphics, 3.2 GHz, 8 Cores, RAM: 16 Gb, GPU: NVidia GeForce RTX3050 Ti Laptop GPU 4 Gb, Storage: 500 Gb NVMe SSD, OS: Windows 11 Home

V. Results

The best-performing Canny Genetic Programming (CGP) detector from thirty independent runs was compared against the standard Canny algorithm. Performance was measured using precision, recall, and F-measure, calculated against the ground-truth images.

TABLE III: Performance Comparison of CGP and Canny Detector

Image ID	Precision		Recall		F-measure	
	CGP	Canny	CGP	Canny	CGP	Canny
10081	**0.040**	0.035	**0.316**	0.142	**0.071**	0.056
16068	0.007	**0.140**	**0.108**	0.099	0.013	**0.024**
2018	**0.108**	0.077	**0.429**	0.191	**0.172**	0.110
3063	**0.043**	0.018	**0.488**	0.100	**0.079**	0.030
41029	**0.057**	0.042	**0.448**	0.212	**0.101**	0.071
41085	**0.027**	0.020	**0.182**	0.091	**0.047**	0.033
41096	**0.010**	0.009	**0.318**	0.154	**0.020**	0.016
43033	**0.035**	0.032	**0.235**	0.146	**0.060**	0.053
43051	**0.020**	0.013	**0.453**	0.017	**0.039**	0.024
6046	**0.029**	0.024	**0.292**	0.178	**0.053**	0.043
8068	**0.084**	0.058	**0.529**	0.203	**0.145**	0.090

TABLE IV: Aggregated Performance Metrics Across All Test Images

Metric	CGP (Proposed)	Canny (Baseline)
Average Precision	0.041 ± 0.03	$\mathbf{0.042 \pm 0.036}$
Average Recall	$\mathbf{0.345 \pm 0.130}$	0.139 ± 0.057
Average F-measure	$\mathbf{0.072 \pm 0.0489}$	0.050 ± 0.029

The proposed CGP approach evolved detectors that consistently outperformed the standard Canny algorithm across the 11 test images, as detailed in Table III. In the recall metric, the CGP method performed better than Canny on every image. In several instances (e.g., images 3063, 43051, and 8068), the recall was more than double that of the Canny method. In terms of precision, the CGP method outperformed the Canny detector on 10 out of the 11 images. Although the margins

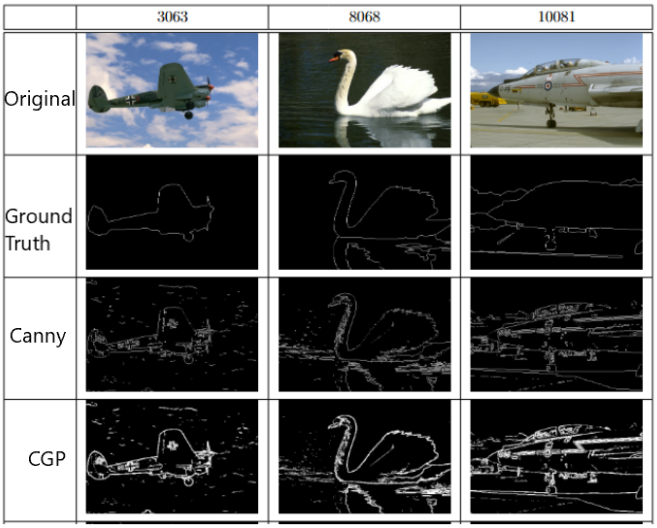

Fig. 2: Detected Edges Image 3063, 8068, 10081.

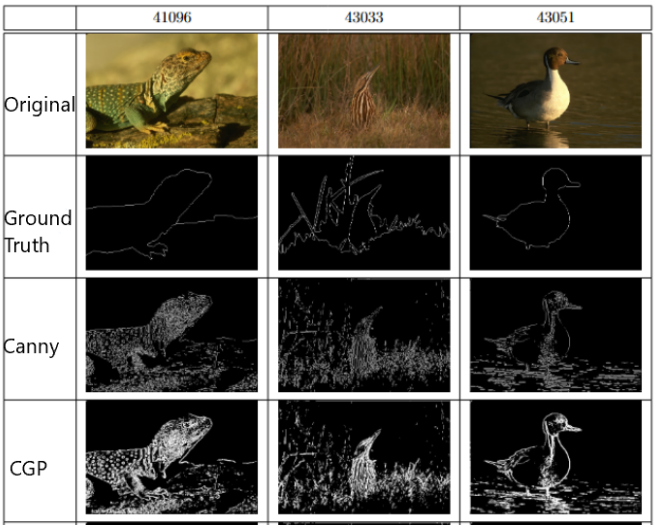

Fig. 3: Detected Edges Image 41096, 43033, 43051.

of improvement in precision were generally more modest than those for recall. The F-measure provides a balanced assessment of precision and recall. The CGP method achieved a higher F-measure on 10 of the 11 images, indicating an overall better performance in generating accurate and complete edge maps. On image 16068, the Canny method achieved higher precision (0.140 vs. 0.007) and a better F-measure (0.024 vs. 0.013) than the CGP approach. The reason for this can be that the stripes of the zebra are more defined in the output image, and as a result, the pixels were classified as false positives, which decreased the precision of the CGP edge detector. Overall, the CGP evolved detectors were more robust and effective, consistently improving edge recall with minimal compromise in precision. These results are in support of the CGP approach based on the characteristics presented in Section II-B on the requirements of a good edge detector. Across 30 independent runs, the average training time on the 19 images was 24.65 minutes.

TABLE V: WILCOXON TEST RESULTS ($\alpha = 0.05$)

Statistic	**Precision**	**Recall**	**F-measure**
W-statistic	1.0	0.0	1.0
Critical Value ($n = 11$)	13.0	13.0	13.0
p-value	< 0.01	< 0.01	< 0.01
Significance ($W \leq W_{crit}$)	**Significant**	**Significant**	**Significant**

A one-tailed Wilcoxon signed-rank test (n=11) confirmed that the performance gains of the proposed method in precision, recall, and F-measure were statistically significant. The computed w-statistics (1 for precision, 0 for recall, and 1 for F-measure) were all below the critical value of 13, showing that the improvements were not due to random chance. From Table IV, the CGP detector has a 44% improvement over Canny with respect to F-measure, which indicates that the detector is more robust to noise. With respect to the precision metric, the Canny detector has slightly better performance, which also indicates

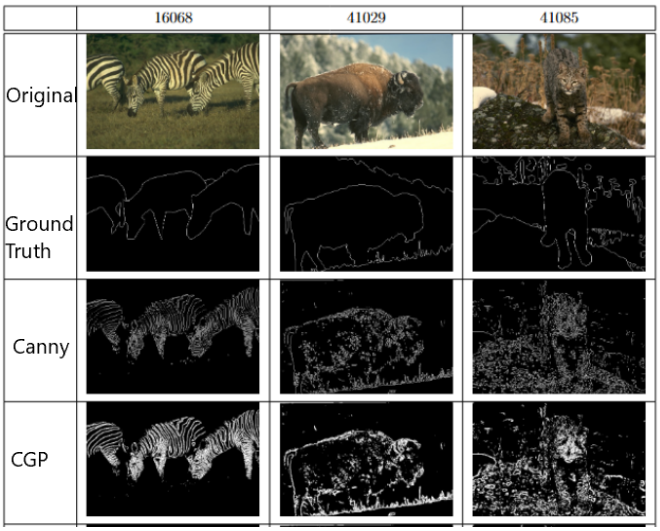

Fig. 4: Detected Edges Image 16068, 41029, 41085.

that CGP does, in fact, detect proper edges. The recall value indicates that CGP is better at identifying and preserving true edges.

A. *Quality Assessment*

The visual outputs of the 11 test images are shown in Figs. 2–5. The figures present a side-by-side comparison of the source images, ground truth and the resulting edge maps from the Canny detector and evolved CGP detector. From the edge maps, the CGP-evolved edge maps exhibit a more pronounced structural definition than those produced by the Canny detector. The edge maps show that CGP achieves higher boundary continuity than the Canny detector. However, it is also apparent that the CGP evolved edge detectors were also more sensitive than the Canny detector. For example, in image 43051, the CGP edge map contains edges of reflection from the dam, which are minimal in the Canny edge map. This

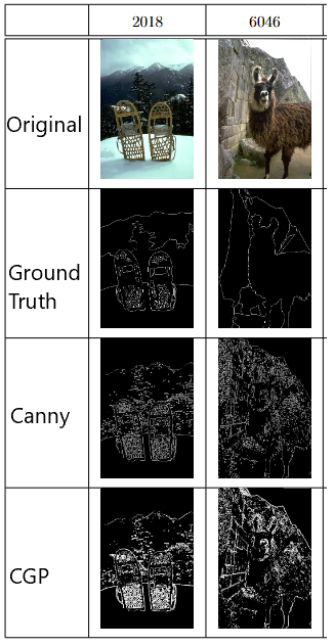

Fig. 5: Detected Edges Image 2018, 6046

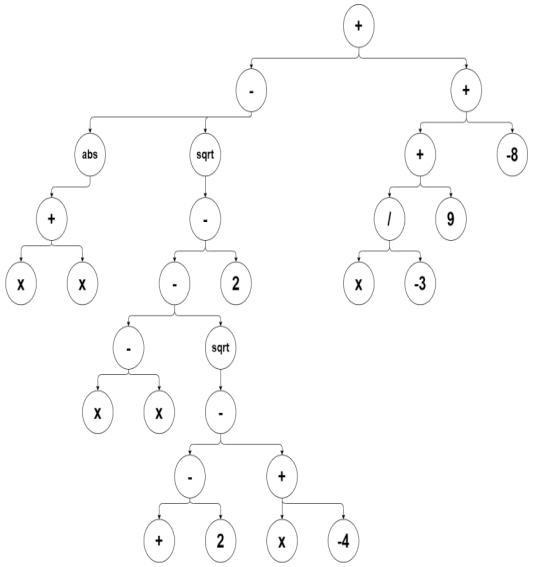

Fig. 6: GP1 Output Classifier

can be attributed to the fitness function, which was tailored more to the accurate identification of a pixel belonging to an edge. Quantitative analysis of the edge maps was done using Structural Similarity Index (SSIM), Peak Signal-to-Noise ratio (PSNR) and the Feature Similarity Index (FSIM) [36]. Higher values on each metric indicate better performance. Each edge map was compared to the ground-truth on the quantitative metrics. Each edge map was post-processed using a morphological thinning filter [37]. This was performed to normalise edge thickness to enable fair comparison. Table III

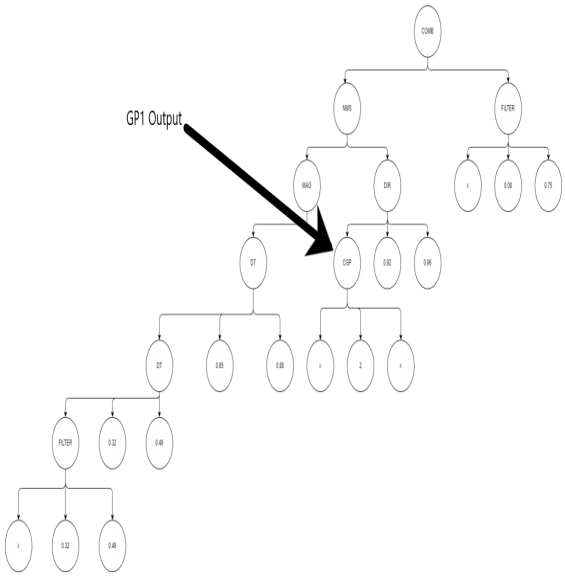

Fig. 7: CGP Output

TABLE VI: COMPARATIVE STRUCTURAL SIMILARITY AND SIGNAL QUALITY ANALYSIS

Image	SSIM		PSNR (dB)		FSIM	
	CGP	Canny	CGP	Canny	CGP	Canny
10081	**0.6219**	0.5924	11.6855	**12.0488**	0.8010	**0.8120**
16068	**0.5888**	0.5849	10.9526	**11.9741**	0.7358	**0.7689**
2018	**0.6254**	0.5527	**11.5639**	10.9647	**0.7789**	0.7441
3063	**0.8354**	0.7720	**15.8373**	15.6764	**0.9141**	0.9037
41029	**0.6785**	0.5781	**12.9447**	11.9435	**0.8243**	0.7794
41085	**0.4511**	0.3057	**10.7035**	9.8407	**0.6721**	0.5995
41096	**0.5945**	0.5064	**10.9604**	10.5856	**0.7492**	0.7120
43033	**0.5990**	0.4594	**11.4950**	10.8358	**0.7593**	0.7110
43051	**0.7389**	0.6664	**13.4433**	12.7995	**0.8723**	0.8543
6046	**0.5218**	0.3339	**11.0591**	9.9433	**0.7326**	0.6421
8068	**0.7678**	0.7228	**13.8422**	13.6345	**0.8733**	0.8603

shows the results of the comparison. With respect to SSIM, the CGP edge maps had a better SSIM on all images, while on both the PSNR and FSIM metrics, CGP performed better on 9 of the 11 images. On average across all images, the proposed approach performed better as shown in Table VII.

TABLE VII: AGGREGATE STATISTICAL PERFORMANCE SUMMARY FOR STRUCTURAL AND SIGNAL METRICS

Model	SSIM ($\mu \pm \sigma$)	PSNR ($\mu \pm \sigma$)	FSIM ($\mu \pm \sigma$)
Canny	0.55 ± 0.14	11.84 ± 1.65	0.76 ± 0.089
CGP	$\mathbf{0.64 \pm 0.11}$	$\mathbf{12.23 \pm 1.53}$	$\mathbf{0.79 \pm 0.07}$

*Values represent Mean (μ) $\pm$ Standard Deviation (σ) for $N = 11$ images.

Fig. 6 is an output classifier of GP1, which is used as a function for GP2 as shown in Fig. 7.

VI. CONCLUSION AND FUTURE WORK

This study proposed Canny Genetic Programming, an evolutionary approach that optimises the architecture of a standard

Canny edge detector. To prove the efficacy of the proposed approach, the performance of the best evolved detector was compared with the standard Canny detector on benchmark images from the BSD500 dataset. The results showed that overall, CGP can evolve edge maps that perform better than standard Canny on the precision, recall and F-measure metrics. Visually, the evolved edge maps also appear more distinctly, showing better localisation than the Canny edge produced edge maps. Future work will include adding some post-processing elements to the Genetic Programming pipeline. Although the images from the BSD500 provide diverse complexity, future work will also involve testing on images from different domains, such as hyperspectral and medical images.

REFERENCES

[1] J. Jing, S. Liu, G. Wang, W. Zhang, and C. Sun, "Recent advances on image edge detection: A comprehensive review," *Neurocomputing*, vol. 503, pp. 259–271, 2022.

[2] O. R. Vincent, O. Folorunso *et al.*, "A descriptive algorithm for sobel image edge detection," in *Proceedings of informing science & IT education conference (InSITE)*, vol. 40, 2009, pp. 97–107.

[3] L. Yang, X. Wu, D. Zhao, H. Li, and J. Zhai, "An improved prewitt algorithm for edge detection based on noised image," in *2011 4th International congress on image and signal processing*, vol. 3. IEEE, 2011, pp. 1197–1200.

[4] W. E. L. Grimson and E. C. Hildreth, "Comments on" digital step edges from zero crossings of second directional derivatives," *IEEE transactions on pattern analysis and machine intelligence*, no. 1, pp. 121–127, 1985.

[5] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1395–1403.

[6] R. Sun, T. Lei, Q. Chen, Z. Wang, X. Du, W. Zhao, and A. K. Nandi, "Survey of image edge detection," *Frontiers in Signal Processing*, vol. 2, p. 826967, 2022.

[7] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[8] S. Xu, P. Zhang, X. Song, and R. Zhang, "Application of improved canny algorithm in image edge detection," in *2024 9th International Conference on Computer and Communication Systems (ICCCS)*. IEEE, 2024, pp. 13–18.

[9] T. Sugashini and G. Balakrishnan, "Visually impaired object segmentation and detection using hybrid canny edge detector, hough transform, and improved momentum search in yolov7," *Signal, Image and Video Processing*, vol. 18, no. Suppl 1, pp. 251–265, 2024.

[10] P. Topno and G. Murmu, "An improved edge detection method based on median filter."

[11] F. Wu, C. Zhu, J. Xu, M. W. Bhatt, and A. Sharma, "Research on image text recognition based on canny edge detection algorithm and k-means algorithm," *International Journal of System Assurance Engineering and Management*, vol. 13, no. Suppl 1, pp. 72–80, 2022.

[12] J. O. Maranga, J. J. Nnko, and S. Xiong, "Learned active contours via transformer-based deep convolutional neural network using canny edge detection algorithm," *Signal, Image and Video Processing*, vol. 19, no. 3, p. 222, 2025.

[13] J. Xiong, D. Wang, J. Yin, and R. Wu, "Precise z-block positioning and dimension measurement using improved canny edge detection and sub-pixel contour fitting," *The Journal of Supercomputing*, vol. 81, no. 1, pp. 1–31, 2025.

[14] J. R. Koza, "Genetic programming as a means for programming computers by natural selection," *Statistics and computing*, vol. 4, pp. 87–112, 1994.

[15] G. Papari and N. Petkov, "Edge and line oriented contour detection: State of the art," *Image and Vision Computing*, vol. 29, no. 2-3, pp. 79–103, 2011.

[16] M. Nikolic, E. Tuba, and M. Tuba, "Edge detection in medical ultrasound images using adjusted canny edge detection algorithm," in *2016 24th telecommunications forum (TELFOR)*. IEEE, 2016, pp. 1–4.

[17] C. Ma, W. Gao, L. Yang, and Z. Liu, "An improved sobel algorithm based on median filter," in *2010 2nd International Conference on Mechanical and Electronics Engineering*, vol. 1. IEEE, 2010, pp. V1–88.

[18] W. Rong, Z. Li, W. Zhang, and L. Sun, "An improved canny edge detection algorithm," in *2014 IEEE international conference on mechatronics and automation*. IEEE, 2014, pp. 577–582.

[19] Y. Lijun, L. Mengbo, W. Tongxin, B. Youfeng, L. Junhui, and J. Yi, "Geo-information mapping improves canny edge detection method," *IET Image Processing*, vol. 17, no. 6, pp. 1893–1904, 2023.

[20] R. Liu and J. Mao, "Research on improved canny edge detection algorithm," in *MATEC Web of Conferences*, vol. 232. EDP Sciences, 2018, p. 03053.

[21] Y. Gao and F. Gao, "Optimize the edge detection algorithm of canny operator threshold selection," in *2024 International Conference on Artificial Intelligence and Power Systems (AIPS)*. IEEE, 2024, pp. 280–287.

[22] K. Benhamza and H. Seridi, "Canny edge detector improvement using an intelligent ants routing," *Evolving Systems*, vol. 12, pp. 397–406, 2021.

[23] R. Song, Z. Zhang, and H. Liu, "Edge connection based canny edge detection algorithm," *Pattern Recognition and Image Analysis*, vol. 27, pp. 740–747, 2017.

[24] W. Fu, M. Johnston, and M. Zhang, "Unsupervised learning for edge detection using genetic programming," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 117–124.

[25] W. Fu, B. Xu, e. Zhang, and M. Johnston, "Fast unsupervised edge detection using genetic programming [application notes]," *IEEE Computational Intelligence Magazine*, vol. 13, no. 4, pp. 46–58, 2018.

[26] W. Fu, M. Johnston, and M. Zhang, "Genetic programming for edge detection: a global approach," in *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 254–261.

[27] ——, "Genetic programming for edge detection: a gaussian-based approach," *Soft Computing*, vol. 20, pp. 1231–1248, 2016.

[28] S. Nguyen, Y. Mei, B. Xue, and M. Zhang, "A hybrid genetic programming algorithm for automated design of dispatching rules," *Evolutionary computation*, vol. 27, no. 3, pp. 467–496, 2019.

[29] C. Kuranga and N. Pillay, "A comparative study of genetic programming variants," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2022, pp. 377–386.

[30] D. J. Montana, "Strongly typed genetic programming," *Evolutionary computation*, vol. 3, no. 2, pp. 199–230, 1995.

[31] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane, "Automated design of both the topology and sizing of analog electrical circuits using genetic programming," in *Artificial intelligence in design'96*. Springer, 1996, pp. 151–170.

[32] D. Herrera-Sánchez, H.-G. Acosta-Mesa, and E. Mezura-Montes, "Auto machine learning based on genetic programming for medical image classification," in *Mexican International Conference on Artificial Intelligence*. Springer, 2023, pp. 349–359.

[33] D. Herrera-Sánchez, H.-G. Acosta-Mesa, E. Mezura-Montes, and A. Márquez-Grajales, "Shifting color space for image classification using genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 283–286.

[34] W. Fu, M. Johnston, and M. Zhang, "Genetic programming for edge detection: a global approach," in *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 254–261.

[35] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[36] S. Manikandan, K. Ramar, M. W. Iruthayarajan, and K. Srinivasagan, "Multilevel thresholding for segmentation of medical brain images using real coded genetic algorithm," *Measurement*, vol. 47, pp. 558–568, 2014.

[37] R. C. Gonzalez, *Digital image processing*. Pearson education india, 2009.