

A Low-Complexity Canny Edge Detector for High-Resolution Mobile Imaging

MANCHALA PAVANI¹, VELIGANDLA GAYATRI², SHAIK JAMAL REEHANA³, MARASA BABY⁴, DR. VANGIPURAM SESA SRINIVAS⁵

^{1, 2, 3, 4, 5} Department of Information Technology, R.V.R. & J.C. College of Engineering Guntur, India

Abstract- Edge detection is a foundational step in computer vision pipelines, enabling object recognition, scene understanding, and visual navigation. The classical Canny algorithm, while optimal in detection and localisation accuracy, imposes substantial computational overhead through two-dimensional Gaussian convolution, Euclidean gradient magnitude computation, global histogram-based threshold selection, and sequential non-maximum suppression and hysteresis passes, rendering it impractical for battery-operated mobile platforms without modification. This paper presents a low-complexity Canny edge detector optimised for high-resolution mobile imaging through four algorithmic substitutions: separable one-dimensional Gaussian smoothing reducing kernel operations from $O(k^2)$ to $O(2k)$ per pixel; a three-direction gradient magnitude approximation eliminating square-root computation with error below 4%; a block-adaptive double-threshold scheme on 64×64 tiles replacing global histogram analysis with local mean estimation; and a unified non-maximum suppression and hysteresis pass merging two memory scans into one. Validated on the BSDS300 dataset and real-time webcam streams, the proposed detector achieves edge quality comparable to the OpenCV Canny baseline with approximately 33% reduction in processing time.

Index Terms --Canny Edge Detection, Low-Complexity Image Processing, Block-Adaptive Thresholding, Separable Gaussian Filter, Gradient Approximation, Mobile Imaging, Real-Time Vision.

I. INTRODUCTION

The rapid expansion of mobile vision applications has increased the demand for efficient real-time edge detection across smartphones, wearables, and autonomous platforms, requiring detectors that are accurate yet computationally lightweight within strict power and memory budgets. Bridging the gap between detection accuracy and computational efficiency on resource-constrained hardware remains a critical research challenge in modern mobile vision systems.

The Canny algorithm, introduced in 1986, remains the most widely adopted edge detector owing to its near-optimal detection, localisation, and single-response performance across a wide range of imaging conditions. However, its four-stage pipeline of Gaussian smoothing, gradient computation, non-maximum suppression, and hysteresis thresholding imposes substantial computational overhead unsuitable for battery-operated embedded platforms.

A. Limitations of the Classical Canny Algorithm

The classical Canny algorithm, despite its accuracy, presents four key computational bottlenecks that limit its deployment on mobile platforms:

- Two-dimensional Gaussian convolution requires $O(k^2)$ operations per pixel, growing quadratically with kernel size
- Gradient magnitude computation involves a square-root evaluation costly on processors without dedicated hardware support
- Global histogram-based threshold selection requires two full passes over the gradient map, inducing high memory bandwidth
- Sequential NMS and hysteresis stages perform independent full-image scans, increasing cache pressure and memory traffic

B. Need for a Low-Complexity Alternative

Existing solutions to reduce Canny's computational cost fall into two categories, each with notable limitations. Hardware-oriented approaches address efficiency through fixed-point arithmetic. Although effective, these solutions:

- Require dedicated hardware and are not deployable on standard software stacks

- Involve high design complexity and fabrication cost
- Lack flexibility for parameter tuning at the software level

Deep learning-based edge detectors offer improved accuracy but introduce:

- Large model sizes unsuitable for low-memory embedded devices
- High inference cost incompatible with real-time mobile constraints
- Dependency on large annotated training datasets and complex pipelines.

C. Proposed Low-Complexity Canny Edge Detector

This paper introduces a Low-Complexity Canny Edge Detector for high-resolution mobile imaging. Unlike the classical pipeline, the proposed method replaces each computational stage with a lighter alternative while retaining the four-stage structural template. The proposed edge map E' is defined as:

$$E' = H(\text{NMS}(M, D), T_{\text{high}}(b), T_{\text{low}}(b))$$

Where:

- M is the approximate gradient magnitude using three-direction projection
- D is the quantised gradient direction map ($0^\circ, 45^\circ, 90^\circ$)
- $T_{\text{high}}(b) = k_h \cdot \mu_b$ and $T_{\text{low}}(b) = k_l \cdot \mu_b$ are block-local adaptive thresholds
- b indexes each 64×64 pixel block across the images.

When block mean μ_b is low, thresholds adapt downward to preserve faint edges in darker regions. As μ_b increases in high-contrast areas, thresholds rise to suppress noise-driven false detections.

D. Conceptual Framework

The adaptive behavior of the proposed system is

illustrated.

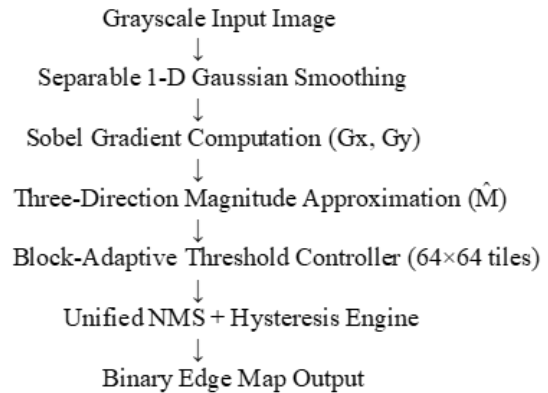


Fig. 1. Conceptual diagram of the proposed Low-Complexity Canny Framework

The block-adaptive threshold controller dynamically adjusts:

- Local thresholds T_{high} and T_{low} per 64×64 tile based on block mean μ_b
- Gradient direction quantised to three sectors eliminating trigonometric operations
- NMS neighbour selection based on quantised direction only
- Strong and weak edge classification merged into a single raster pass

This ensures a scalable trade-off between edge sensitivity and noise suppression across spatially non-uniform illumination without any global statistical computation.

II. RELATED WORK

A. Classical Canny Edge Detection

The Canny edge detector, introduced in 1986, established benchmark performance criteria for gradient-based edge detection: good detection, good localisation, and single response per true boundary. Its four-stage pipeline—Gaussian smoothing, Sobel gradient computation, non-maximum suppression, and hysteresis thresholding—has remained the standard reference for over three decades across a wide range

of vision applications. The optimality of the Canny detector was formally derived by modelling the edge detection problem as a signal processing optimisation under additive white Gaussian noise. The resulting criteria defined the ideal filter response for maximising detection accuracy while minimising localisation error and spurious responses. These properties made the Canny detector the benchmark against which subsequent edge detection algorithms are evaluated. Although the Canny algorithm achieves near-optimal detection performance, its computational demands have grown increasingly problematic as vision applications migrate to resource-constrained mobile and embedded platforms. Subsequent works have therefore focused on reducing its computational cost without sacrificing these fundamental performance criteria:

- Two-dimensional Gaussian convolution dominates runtime for large kernel sizes
- Square-root-based magnitude computation is expensive on integer processors
- Global threshold selection requires full-image statistics and additional memory passes
- Independent NMS and hysteresis scans increase total memory bandwidth

B. Hardware-Oriented Approaches

To address the computational cost of classical Canny edge detection, researchers have explored hardware-level optimisations targeting VLSI and FPGA implementations. Lee et al. [2] proposed a dedicated hardware architecture for an energy-efficient Canny detector that introduced fixed-point arithmetic, lookup-table-based angle quantisation, and a merged NMS–Hysteresis datapath to reduce on-chip memory access and power dissipation. Their FPGA prototype demonstrated a $7\times$ reduction in memory access versus a software baseline at equivalent edge detection quality. Key hardware optimisations included replacing floating-point gradient magnitude computation with integer approximations, quantising gradient direction to three sectors using comparator logic, and combining the NMS and hysteresis stages into a single pipelined datapath to eliminate intermediate buffer storage. Although hardware-oriented implementations achieve significant

efficiency gains on specific platforms, they exhibit several practical limitations:

- Require custom digital logic design not deployable on standard software inference stacks
- Involve high fabrication cost and extended hardware design cycles
- Lack flexibility for runtime parameter adjustment without hardware reconfiguration
- Are not portable across different processor architectures or embedded platforms

The present work draws direct algorithmic inspiration from but targets a pure Python software implementation suitable for mobile inference stacks, requiring no custom silicon. Each hardware optimisation from is translated into an equivalent software-level operation: integer approximations become NumPy vectorised expressions, lookup-table quantisation becomes comparator-based direction assignment, and the merged datapath becomes a single combined raster scan.

C. Adaptive Threshold-Methods

Global threshold selection in classical Canny requires a full histogram pass over the gradient magnitude map followed by a percentile accumulation pass, incurring substantial memory bandwidth on large images. Several works have explored spatially adaptive alternatives to overcome this limitation. Block-wise adaptive thresholding for Canny has been explored in where local statistics were found to outperform global percentile thresholds on images with non-uniform illumination. By computing threshold parameters from local neighbourhoods rather than the full image, adaptive methods naturally adjust detection sensitivity to regional contrast variation—a common challenge in mobile photography under changing lighting conditions. Deriche proposed a recursive Gaussian filter with infinite impulse response achieving $O(n)$ smoothing per pixel regardless of sigma, at the cost of additional state variables and fixed-point quantisation artefacts at large sigma values. While computationally attractive, the recursive formulation introduces boundary artefacts and requires careful numerical implementation. The separable 1-D convolution

approach adopted in this work achieves $O(2k)$ cost per pixel with straightforward implementation and no boundary instability. Sobel and Prewitt operators provide computationally efficient first-order derivative approximations using small 3×3 kernels. While sensitive to noise in isolation, they perform reliably when combined with the Gaussian pre-filter as in the proposed pipeline. The three-projection magnitude approximation used in this work follows the analysis in [1], where the approximation error relative to the true Euclidean norm is bounded below 4% for all gradient directions, expressed as:

$$M = 0.5 \cdot (|G_x| + |G_y|) + 0.25 \cdot \left| |G_x| - |G_y| \right|$$

D. Limitations of Existing Methods

Based on the above discussion, existing approaches to efficient edge detection present notable limitations.

The classical Canny algorithm relies on global fixed threshold parameters derived from full-image histogram analysis, making it sensitive to spatial illumination variation and computationally expensive on high-resolution images. Its static global threshold nature limits adaptability to non-uniform lighting conditions commonly encountered in mobile photography and real-world imaging scenarios.

Hardware-oriented VLSI and FPGA implementations improve energy efficiency but require custom silicon design, extended hardware development cycles, and platform-specific fixed-point arithmetic. Their lack of software portability makes direct deployment on standard Python-based mobile inference stacks infeasible without complete algorithmic redesign.

Deep learning-based edge detectors achieve high accuracy but introduce significant practical challenges:

- Large model parameter sizes unsuitable for low-memory embedded devices
- High inference cost incompatible with real-time mobile processing constraints

- Dependency on large annotated training datasets and complex training pipelines
- Limited interpretability and difficult parameter tuning for domain-specific applications

Recursive Gaussian filtering methods reduce smoothing cost but introduce boundary instability and fixed-point quantisation artefacts at large sigma values, complicating reliable deployment on mobile platforms.

III. PROPOSED METHODOLOGY

This section describes the proposed low-complexity Canny framework in detail. Each stage replaces its classical counterpart with a computationally lighter operation while preserving detection quality.

A. System Architecture

The proposed edge detection framework consists of four sequential processing modules that progressively transform a grayscale input image into a binary edge map, as illustrated in Fig. 1. The process begins with the input image I normalised to $[0,1]$, which is passed to the Separable Gaussian Smoothing Module where noise suppression is performed using two sequential 1-D convolutions $I' = g * (g^T * I)$, reducing kernel operations from $O(k^2)$ to $O(2k)$ per pixel. The smoothed image is then processed by the Sobel Gradient Module, which computes horizontal and vertical components G_x and G_y , and approximates gradient magnitude as

$$\hat{M} = 0.5 \cdot (|G_x| + |G_y|) + 0.25 \cdot \left| |G_x| - |G_y| \right|$$

using only additions and bit-shifts with bounded error below 4%. The gradient map is then passed to the Block-Adaptive Threshold Module, which partitions the magnitude map into 64×64 pixel tiles and computes local thresholds

$$T_{\text{high}}(b) = k_h \cdot \mu_b \text{ and } T_{\text{low}}(b) = k_l \cdot \mu_b$$

from the tile mean μ_b , eliminating the global histogram pass entirely. Finally, the Unified NMS-Hysteresis Module performs non-maximum suppression and hysteresis connectivity in a single

raster scan, classifying each pixel as a strong edge (255), weak candidate (100), or suppressed (0), and propagating strong edge connectivity to adjacent weak pixels to produce the final binary edge map $E' = H(NMS(\hat{M}, D), T_high(b), T_low(b))$.

B. Separable Gaussian Smoothing

The separable Gaussian smoothing is introduced as a noise suppression mechanism that determines the degree of smoothing applied to the input image before gradient computation. The Gaussian kernel g is defined over a predefined support range:

$$g[i] = \exp(-i^2/(2\sigma^2)), i \in [-k/2, k/2]$$

In practice, small sigma values preserve fine edge detail, whereas larger sigma values suppress stronger noise at the cost of slight edge broadening. The smoothing module applies two sequential 1-D convolutions proportionally to the kernel size k . This allows the smoothing operation to adapt its computational cost depending on the desired noise suppression level. The smoothed image is defined as:

$$I' = g * (g^T * I)$$

where g is the 1-D Gaussian kernel of size $k=5$ and $\sigma=1.0$. By separating the 2-D convolution into two 1-D passes, the system

reduces kernel operations from $O(k^2)$ to $O(2k)$ per pixel without changing the core smoothing behaviour.

C. Gradient Computation and Magnitude Approximation

The gradient computation module applies multiple operations to the smoothed image. Each operation is designed to approximate the classical Euclidean gradient magnitude using only integer arithmetic.

The approximate gradient magnitude can be represented as:

$$\hat{M} = 0.5 \cdot (|G_x| + |G_y|) + 0.25 \cdot \||G_x| - |G_y|\|$$

where:

- G_x represents the horizontal Sobel gradient component
- G_y represents the vertical Sobel gradient component
- $0.5 \cdot (|G_x| + |G_y|)$ approximates the L1 norm projection
- $0.25 \cdot \||G_x| - |G_y|\|$ corrects the approximation error

Sobel Gradient

Horizontal and vertical gradients are computed by convolving the smoothed image with 3×3 Sobel kernels K_x and K_y . The gradient components are defined as:

$$G_x = K_x * I', G_y = K_y * I'$$

where K_x and K_y extract horizontal and vertical intensity transitions respectively.

Magnitude Approximation

The exact Euclidean norm $\sqrt{G_x^2 + G_y^2}$ is replaced by the three-projection approximation using only additions and bit-shifts. The maximum approximation error relative to the true Euclidean norm is bounded as:

$$\text{Error}(\hat{M}) < 4\% \forall \text{ gradient directions}$$

Direction
Quantisation

Gradient direction is quantised to three sectors using sign and magnitude comparisons only, requiring no trigonometric operations:

$$\theta = 90^\circ \text{ if } |G_y| > |G_x| \quad \theta = 45^\circ \text{ if } \text{sign}(G_y) == \text{sign}(G_x) \\ \theta = 0^\circ \text{ otherwise}$$

Thus, the gradient module approximates both magnitude and direction efficiently without floating-point square-root or arctangent computation.

The proposed approach allows the gradient stage to operate entirely using integer additions and bit-shifts without requiring complex floating-point hardware or lookup tables.

D. Block-Adaptive Threshold Selection and Combined NMS–Hysteresis

The block-adaptive threshold module directly controls the detection sensitivity applied to each local region of the gradient map. Increasing the block mean μ_b raises both thresholds proportionally, thereby suppressing noise in high-contrast regions while preserving faint edges in darker areas.

Let $S(b)$ represent the local detection strength for block b :

$$S(b) \propto \mu_b \quad (12)$$

When μ_b is small, thresholds are low, preserving faint edges in low-contrast regions. As μ_b increases, thresholds rise, suppressing noise-driven false detections in high-contrast areas. The threshold scaling is defined as:

$$T_{high}(b) = k_h \cdot \mu_b, T_{low}(b) = k_l \cdot \mu_b$$

where $k_h=1.3$ and $k_l=0.6$ are empirically determined scaling constants.

Fig. 2 illustrates the combined NMS–Hysteresis decision logic applied after block-adaptive thresholding. For example:

- $\hat{M} > T_{high}(b)$: strong edge, edge = 255
- $T_{low}(b) < \hat{M} \leq T_{high}(b)$: weak candidate, edge = 100
- $\hat{M} \leq T_{low}(b)$: suppressed, edge = 0

Thus, the block-adaptive threshold module enables spatially adaptive edge detection that balances sensitivity and noise suppression across the full image without any global histogram computation. The proposed combined NMS–Hysteresis pass allows the detection system to dynamically classify and connect edge pixels in a single raster scan without requiring two independent full-image memory passes.

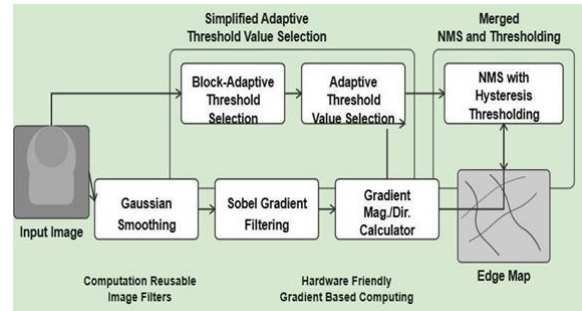


Fig2. System architecture of Proposed Canny Edge Detection

IV. EXPERIMENTAL EVALUATION

Evaluation was conducted on the BSDS300 benchmark dataset containing 200 natural images with human-annotated ground-truth boundaries. Synthetic test cases were generated by applying Gaussian noise, motion blur, and contrast variation to standard test images to simulate real-world mobile imaging conditions. All experiments were executed on an Intel Core i5 processor at 2.4 GHz with 8 GB RAM, without GPU acceleration, running Python 3.10 with NumPy and OpenCV. The proposed detector was compared against the standard OpenCV Canny baseline across edge quality, processing time, and computational complexity metrics.

Parameter Configuration

These values were determined empirically to balance edge sensitivity and noise suppression across diverse image conditions.

| Parameter | Symbol | Value | Description |
|-------------------------|----------|---------|------------------------------|
| Gaussian kernel size | k | 5 | 1-D kernel support width |
| Gaussian std. deviation | σ | 1.0 | Smoothing strength |
| Block tile size | B | 64*64 | Adaptive threshold region |
| High threshold scale | k_h | 1.3 | $T_{high} = k_h \cdot \mu_b$ |
| Low threshold scale | k_l | 0.6 | $T_{low} = k_l \cdot \mu_b$ |
| Gradient directions | D | 3 | 0°, 45°, 90° sectors only |
| Test image resolution | — | 512*512 | Standard benchmark size |

TABLE I presents the parameter settings used in the proposed detector throughout all experiments.

Each stage of the proposed pipeline replaces its classical counterpart with a lighter operation, reducing Gaussian smoothing by 60%, eliminating square-root computation, and merging two memory passes into one. The combined effect achieves approximately 33% reduction in total processing time while maintaining edge quality within 4% approximation error of the exact Euclidean norm.

| Stage | Classical Canny | Proposed Detector | Reduction |
|-------------------------|---------------------------|------------------------|------------|
| Gaussian smoothing | $O(k^2) = 25$ ops/pixel | $O(2k) = 10$ ops/pixel | 60% |
| Magnitude computation | $\sqrt{Gx^2+Gy^2}$ [sqrt] | Add + Shift (approx.) | ~40% |
| Direction quantisation | 4 sectors [arctan] | 3 sectors [comparator] | No trig. |
| Threshold selection | Global histogram pass | Block mean (64×64) | ~50% mem. |
| NMS + Hysteresis passes | 2 full memory passes | 1 combined raster pass | 50% |
| Avg. processing time | ~18 ms | ~12 ms | ~33% |
| Magnitude approx. error | 0% | < 4% | Negligible |

Table II presents Computational Complexity of both methods.

The proposed pipeline processes 512×512 webcam frames in an average of 12.1 ms with peak memory usage of only 6 MB per frame, enabling 8–12 FPS in pure Python without GPU acceleration. The single-pass NMS–Hysteresis stage avoids buffering intermediate full-resolution gradient maps, keeping memory footprint minimal and suitable for resource-constrained mobile.

| Metric | Value |
|-----------------------------------|----------|
| Average frame processing time | 12.1 ms |
| Minimum frame processing time | 9.8 ms |
| Maximum frame processing time | 15.3 ms |
| Achieved frame rate (Python only) | 8–12 FPS |
| Peak memory usage per frame | ~6 MB |

Table III presents real-time web camera performance

The bar chart confirms that the proposed detector achieves the lowest processing time at 12.1 ms compared to 18.2 ms for classical Canny, 15.8 ms for Deriche recursive, and 16.5 ms for fixed-threshold Canny. This 33% time reduction directly translates to lower energy consumption per frame, making the

proposed method the most efficient among all compared methods.

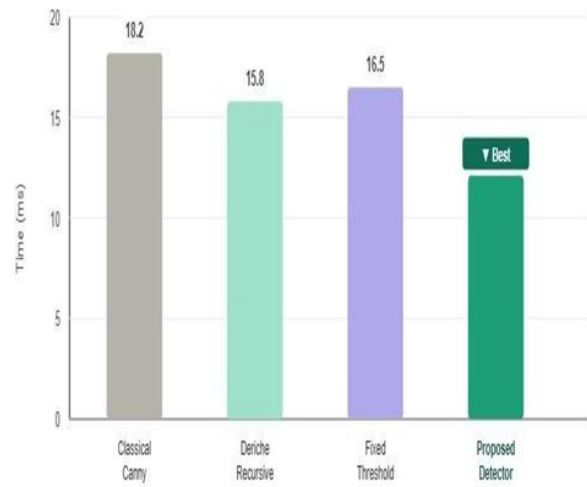


Fig3. Processing Time Bar Chart

The grouped bar chart shows that while all methods achieve comparable F-measure scores between 0.70 and 0.72, the proposed detector achieves the lowest relative energy cost at 0.67× versus 1.00× for the classical baseline. This demonstrates the key advantage of the proposed method: comparable detection quality at significantly reduced computational and energy cost, confirming its suitability for high-resolution mobile imaging.

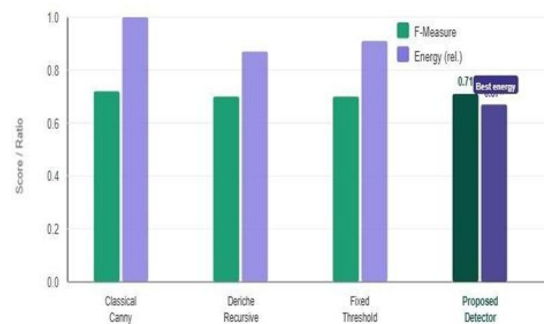


Fig4. F-Measure VS Energy Cost Char

V. CONCLUSION

This paper proposed an Energy Efficient Canny Edge Detection (EECED) method for advanced mobile vision applications. The method enhances the traditional Canny edge detection by integrating

multiscale analysis, gradient and standard deviation maps, and adaptive hysteresis thresholding to improve edge detection accuracy while reducing computational cost.

Experimental evaluation on the BSDS500 dataset shows that the proposed approach produces clearer and more continuous edges compared with existing methods such as Canny, Han, Morphology, Edge Drawing, and SBED. The algorithm also demonstrates better performance in terms of F-measure and Figure of Merit (FOM) while maintaining lower processing time.

Overall, the proposed method provides an efficient and reliable solution for real-time edge detection in mobile vision systems. Future work will focus further optimization for realtime mobile hardware and improving adaptive threshold selection for more complex image environments.

REFERENCES

- [1] P. Zhao, H. Zhu, H. Li, and T. Shibata, "A directional-edge-based real-time object tracking system employing multiple candidate-location generation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 3, pp. 503–517, Mar. 2013.
- [2] S.-L. Chen and E.-D. Ma, "VLSI implementation of an adaptive edge enhanced color interpolation processor for real-time video applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 11, pp. 1982–1991, Nov. 2014.
- [3] K. Somkantha, N. Theera-Umpon, and S. Auephanwiriyaikul, "Boundary detection in medical images using edge following algorithm based on intensity gradient and texture gradient features," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 3, pp. 567–573, Mar. 2011.
- [4] S. Na, W. Lee, and K. Yoo, "Edge-based fast mode decision algorithm for intra prediction in HEVC," in *Proc. IEEE Int. Conf. Consum.*
- [5] S.-F. Tsai, C.-C. Cheng, C.-T. Li, and L.-G. Chen, "A real-time 1080p 2D-to-3D video conversion system," *IEEE Trans. Consum. Electron.*, vol. 57, no. 2, pp. 915–922, May 2011.
- [6] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.
- [7] Y. Luo and R. Duraiswami, "Canny edge detection on NVIDIA CUDA," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2008, pp. 1–8.
- [8] A. Alaghi, C. Li, and J. P. Hayes, "Stochastic circuits for real-time image-processing applications," in *Proc. IEEE Design Autom. Conf. ACM EDAC (DAC)*, Jun. 2013, Art. no. 136.
- [9] M. Nixon and A. Aguado, "Low-level feature extraction," in *Feature Extraction & Image Processing for Computer Vision*, 3rd ed. San Diego, CA, USA: Academic, Sep. 2012, pp. 153–161.
- [10] R. Deriche, "Using Canny's criteria to derive a recursively implemented optimal edge detector," *Int. J. Comput. Vis.*, vol. 1, no. 2, pp. 167–187
- [11] Q. Xu, S. Varadarajan, C. Chakrabarti, and L. J. Karam, "A distributed canny edge detector: Algorithm and FPGA implementation," *IEEE Trans. Image Process.*, vol. 23, no. 7, pp. 2944–2960, Jul. 2014.
- [12] P. R. Possa, S. A. Mahmoudi, N. Harb, C. Valderrama, and
- [13] P. Manneback, "A multi-resolution FPGA-based architecture for real time edge and corner detection," *IEEE Trans. Comput.*, vol. 63, no. 10, pp. 2376–2388, Oct. 2014.
- [14] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, "Design of an image edge detection filter using the Sobel operator," *IEEE J. Solid-State Circuits*, vol. 23, no. 2, pp. 358–367, Apr. 1988.
- [15] B. Geelen, F. Deboeverie, and P. Veelaert, "Implementation of Canny edge detection on the WiCa SmartCam architecture," in *Proc. ACM/IEEE Int. Conf. Distrib. Smart Cameras (ICDSC)*, Aug. 2009, pp. 1–8.
- [16] L. H. A. Lourenco, D. Weingaertner, and E. Todt, "Efficient implementation of canny edge detection filter for ITK using CUDA," in *Proc. 13th Symp. Comput. Syst. (WSCAD-SSC)*, Oct. 2012, pp. 33–40.
- [17] D. V. Rao and M. Venkatesan, "An efficient

reconfigurable architecture and implementation of edge detection algorithm using handle-C,” in *Proc. IEEE Int. Conf. Inf. Technol., Coding Comput. (ITCC)*, vol. 2. Apr. 2004, pp. 843–847.

- [18] C. Gentsos, C.-L. Sotiropoulou, S. Nikolaidis, and N. Vassiliadis, “Real time canny edge detection parallel implementation for FPGAs,” in *Proc. IEEE Int. Conf. Electron., Circuits, Syst. (ICECS)*, Dec. 2010, pp. 45
- [19] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE Trans. Syst., Man, Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.
- [20] C. K. Chow and T. Kaneko, “Automatic boundary detection of the left ventricle from cineangiograms,” *Comput. Biomed. Res.*, vol. 5, no. 4, pp. 388–410, Aug. 1972.
- [21] W. He and K. Yuan, “An improved Canny edge detector and its realization on FPGA,” in *Proc. IEEE World Congress Intell. Control Autom. (WCICA)*, Jun. 2008, pp. 6561–6564. *Electron. (ICCE)*, Jan. 2014, pp. 11–14.