Hardware and Software Implementation of Canny Edge Detection Algorithm

Meghanand Kumar¹, Sejal Singh²
meghanandpk@gmail.com, sejalssdd@gmail.com

¹Department of Instrumentation, Cochin University of Science and Technology, Kochi
²Department of Electronics and Telecommunication, Indian Institute of Engineering Science and Technology, Shibpur

Abstract. Digital Image consists of some informative pixels and some redundant pixels. Edge Detection is the process of preserving informative pixels and eliminating redundant data which improves bandwidth and storage efficiency. A sudden change in pixel intensity level is defined as an Edge. The main motive of edge detection is to simplify the image data in order to minimize the amount of data to be processed. Theoretically, from the literature, it is seen that Canny edge detection is the most accurate algorithm. It is also insensitive to Noise. The only disadvantage of the Canny algorithm is its high computational complexity which limits its maximum frequency of application with high latency and low throughput. Therefore, in this work, a trade-off between Accuracy and Complexity is made. An efficient canny algorithm is designed using MATLAB for software implementation and the same is implemented using Xilinx System Generator to realize on hardware (FPGA). Xilinx System Generator utilizes the JTAG Hardware co-simulation approach for hardware realization. Accuracy is compromised in order to make it more efficient in terms of resource utilization than the conventional one. It is found that the designed hardware architecture is faster than many of the existing architectures.

Keywords: Image Processing, Edge Detection, Canny Algorithm, Xilinx System Generator (XSG), MATLAB, Simulink, Virtex-7 FPGA

1 Introduction

Pixels are the principal unit of any Digital Image. Each pixel has some intensity value which varies between 0-255. Edges are nothing but pixels that carry structural information of an Image. Edges in an image are an unexpected change in intensity values of images. By simply observing at the variation in the intensity values, the conclusion can be made that the point or pixel is of edge. Edges generally occur on the boundary between two different regions in an image.
Edge detection is a fundamental tool in image processing so as to process further computer vision applications like smart transportation, programmed machine assessment, driverless automobiles, and wise robot control frameworks. Various edge detection techniques are used for this which is shown in Fig. 1. The various steps of Edge Detection are Smoothing, Enhancement, Thresholding, and Localization [1]. The result of any edge detection technique is to perceive or find the edges of various objects [2]. Recognition and classification of images are fundamentally done through the object edge information. The human eye can easily play out this assignment yet the machine can't play out this assignment without edge information. Therefore, Edge Detection is a significant and essential step in any image processing application [3].

Fig. 1. Classification of Edge Detection Technique.

Xilinx System Generator provides a common environment for MATLAB/Simulink and ISE Design Suite. XSG provides an efficient way of designing complex algorithms. It automatically produces the required Hardware Description Language (HDL) alongside a Test bench. Field Programmable Gate Array (FPGA) is utilized for prototyping purposes. Virtex-7 FPGA is programmed by downloading generated bit stream file. System Generator Token provides various types of Compilation.

2 Conventional Canny Edge Detection Algorithm

Of all the edge detection techniques, Canny is the most accurate edge detection technique. The Canny algorithm is also called as the optimal edge detector. It is proposed by John Canny in 1986. At that time, classical operators such as Robert, Prewitt, and Sobel were known. His interest was to improve performance of these operators. All the mathematical formulation and analysis required to support his theory is given in [4]. Canny proposed few criteria for improving classical methods of edge detection. First one is Low Error Rate. It emphasizes on minimizing Errors [2]. Second one is Localization of Edges [2]. There should not be any difference between located and actual edges. Third and final criterion is only one response to single edge [2]. It makes Canny the most accurate edge detection technique. Fig. 2 shows the block diagram of Conventional Canny Edge Detector.
In this step, using the Gaussian function is an important choice for the classical canny edge detection algorithm, which can smooth images for removing the noise before edge detection. Classical Canny algorithm generally uses 1-D Gaussian functions to smooth image and denoise.

Horizontal/Vertical Gradient

The gradients in horizontal and vertical directions, $f_x(x, y)$ and $f_y(x, y)$ respectively, for each pixel $(x, y)$ is calculated by convolving the images with gradient masks.

Gradient Magnitude and Direction Calculation

After obtaining $f_x(x, y)$ and $f_y(x, y)$, each pixel calculates the magnitude and direction. The magnitude defines the strength at the edge, and direction defines the orientation of the edge present at each pixel. The magnitude $mag(x, y)$ can be given as,

$$mag(x, y) = \sqrt{f_x(x, y)^2 + f_y(x, y)^2} \quad (1)$$

The direction $\theta(x, y)$ can be given as,

$$\theta(x, y) = \tan^{-1}\left(\frac{f_y(x, y)}{f_x(x, y)}\right) \quad (2)$$

Non-Maximum Suppression

The edges are made thin by converting the blurred edges of the image gradients into sharp edges. This can be done by suppressing the minima (preserving local maxima) in the gradient image.

![Fig. 2. Block diagram of Conventional Canny Edge Detector.](image)
• High/Low Thresholds Computation

The potential edges are determined by high and low threshold. These values are calculated based on gradient magnitude histogram for the whole image. The high threshold value is chosen based on $P_1$ percentage of the total number of pixels. The lower threshold value is $P_2$ percentage of a high threshold value. The percentage values are calculated by using the cumulative distribution function (CDF). The values of $P_1$ are $P_2$ are typically chosen as 20% and 40% respectively.

• Hysteresis Thresholding

Finally, the hysteresis thresholding creates the continuous edge map by removing (i) the edge discontinuities present within an image, and (ii) the edges caused due to noise and illumination variation. The gradient magnitude of each pixel is compared with the high and low threshold values. If the gradient magnitude is greater than the high threshold value, then it is marked as a strong edge [5]. If the gradient magnitude is lower than the low threshold value, then it is marked as a weak edge. If the gradient magnitude value is in between high and low threshold value, then further investigation is done with eight neighbour’s values. If they are connected, then it is interpreted as a strong edge otherwise it is taken as a weak edge.

3 Proposed Canny Edge Detection Algorithm

Proposed design of canny algorithm utilizes trade-off between Accuracy and Complexity. Accuracy has been compromised in order to make conventional Canny less complex. Proposed design skips gradient direction calculation part and deals with magnitude calculation only. Rest all the steps are same for the proposed algorithm as that of conventional one.

3.1 Software Implementation

The proposed canny edge detection algorithm is implemented through MATLAB code. MATLAB R2013a is used to carry out the process.

3.2 Hardware Implementation

The same algorithm is implemented using Xilinx System Generator which utilizes JTAG Hardware co-simulation approach for hardware realization. The System used to design proposed Canny Edge Detection model should have licensed versions of MATLAB and Integrated Software Environment (ISE) Design Suit. Operating System used here is Windows10. System Generator provides a library of various Xilinx Blocks and Simulink Blocks. These Xilinx blocks use Fix Point Data Type which is required for hardware implementation whereas; Simulink blocks use Floating Point Data type by default. Thus, Gateway Blocks are used for the connection between
Simulink blocks and Xilinx blocks. System Generator can generate HDL code automatically from the created model. The automatically generated code in VHDL/Verilog can be synthesized along with its test bench and implemented on FPGA using bit stream file. System Generator Token allows user to perform various types of compilation. The Xilinx System Generator is a plug-in to Simulink that enables designer to develop high-performance DSP systems for Xilinx FPGAs [6]. Its flow is given in Fig. 3.

![System Generator Design Flow Diagram](image)

**Fig. 3.** System Generator Design Flow

Image data serialization & de-serialization is completed in Software (Simulink) and edge detection is implemented in hardware. The primary objective of this paper is to implement the edge detection algorithm and find a better solution for existing system. Hardware Software Co-Simulation Co-Design is a powerful way to realize this. To start the co-simulation process, the user must indicate the amount of time that the system should run to successfully obtain all output data. To determine the run time for the system, use the following equation:

\[
T = W^2 + 2 \times W + 30\]  

(3)

This equation will provide adequate time for the system, assuming that the input image is a square image and “W” represents the pixel width of the image. This method allows one to do without any camera or VGA-monitor interfacing which minimizes the complexity of whole system setup.

Fig. 4 shows design for proposed Canny Algorithm. It consists of various blocks which are designed as follows.
Proposed design for Canny Edge Algorithm Model

- **Image from file Block**

‘Image from File’ is utilized to access the input images where image file path is given and data type of image data is selected. If the image is an M x N x P array, then the block outputs a color image in which M x N are the number of rows & columns in each color pan.

- **Color Space Conversion and Resize Block**

After getting the image from file, the image is converted to gray image using color space conversion and then a resize block is used to change the pixel value to the required size. In our work, the algorithm is designed for 512 x 512 images so the image is resized to 512 x 512.

- **Simulink Image Pre-Processing**

Simulink Image Pre-Processing serializes the image data for the hardware as Xilinx System Generator serial Data transfer. The data which is in 2-D is converted into 1-D by ‘Convert 2-D to 1-D’ Simulink block whereas ‘Transpose’ block transposes the image data. ‘Frame Conversion’ & ‘Unbuffer’ block together sends the 1-D image data frame by frame serially to the hardware input. The ‘Unbuffer’ block converts the frame to scalar samples output at a higher sampling rate. Fig. 5 shows Simulink Image Pre-Processing block.
Xilinx System Generator Token

Xilinx System Generator token is used for FPGA Hardware functionality generation. It can generate (i) HDL, (ii) Bit stream, (iii) NGC & (iv) Hardware Co-Simulation from Xilinx System Generator blocks of Simulink. Both VHDL & Verilog can be generated by the token. It also allows 3rd party synthesis tool such as Synplify Pro from Synopsys Inc. [6]

Gateway In/Gateway Out

Gateway In and Gateway Out blocks are one of the basic blocks of System Generator which works as input and output respectively for the hardware design. These blocks are utilized to convert Simulink data type to Xilinx data type and vice-versa. These blocks characterize top level input & output port respectively in the HDL design generated by System Generator. [6]

Resource Estimator

Resource estimator is the block through which one can estimate the resources used in the design. It displays the no. of slices, IOBs, LUTs etc. used by the design.

Gaussian Filter Block

Proposed design uses 5x5 Gaussian filter (see Fig. 6) for smoothing the image. This filter is already present in the Xilinx Reference Block set of Simulink library. It has total nine applications out of which we used Gaussian filtering. It acts as a low pass filter and thus removes Noise (i.e., high frequency components).

Fig. 5. Simulink Image Pre-processing block.

Fig. 6. Design for 5x5 Gaussian Filter.
Gradient Magnitude Calculation

3 x 3 Sobel operator mask is used to find the horizontal and vertical gradient of the image. Buffering scheme is used to find the horizontal and vertical gradient of the image. Fig. 7 & Fig. 8 show horizontal and vertical gradient filters for sobel operator respectively. In order to calculate the resultant magnitude, square root operation is needed to be performed. But the cost paid to realize square root operation is high, so low-cost hardware implementation technique named square approximation suggested in [8] is adopted to compute the magnitude $\text{mag}(x, y)$. It is given as,

$$\text{mag}(x, y) \approx \max((0.875a + 0.5b))$$

where, $a = \max(f_x(x, y), f_y(x, y))$ and $b = \min(f_x(x, y), f_y(x, y))$
• Non-Maximal Suppression (NMS)

Gradient output contains thick edges due to smoothing image. This block (designed as shown in Fig. 9) provides edge thinning. It compares all neighboring pixel values to the central pixel and suppresses it to zero value if it is not maximum. Otherwise, it is given to next block. In conventional Canny, comparison is done with the pixels in the direction of gradient. This change in the design degrades accuracy of proposed Canny.

Fig. 9. Design for Non-Maximum Suppression Block

• Hysteresis Thresholding

It is the final block in canny algorithm design which uses two thresholds i.e., high and low. Also, it eliminates streaking effect caused due to single threshold when it is subjected to noisy image. If gradient value exceeds high threshold then it is considered as Strong Edge (255) and when it is less than Low Threshold then it is considered as weak edge (suppressed to 0). Fig. 10 gives design of this block.

Fig. 10. Design for Hysteresis Thresholding Block.
JTAG Co-Sim

JTAG Co-Sim is the Hardware Co-Simulation Block generated by System Generator Token after the simulation of the whole system. This is the block that configures the FPGA, gets its output and displays it in Simulink.

Simulink Image Post-Processing

Simulink Image Post-Processing, which is inverse of Image Pre-Processing, is utilized to transform the image data back to floating type. To de-serialize the image data, at first, ‘Buffer’ block is used to convert the scalar samples to frame output at lower sampling rate along with the ‘Convert 1-D to 2-D’ then the image data is transposed by the ‘Transpose’ block. Fig. 11 shows Simulink Image Post-Processing block.

4 Results and Analysis

4.1 Matlab Results

In Canny Edge Detection Algorithm, edges are mainly found by using gradient based edge detector Sobel operator. So one of our approaches was to find out impact of different gradient operators on the Canny. So Robert, Prewitt operators were implemented into canny algorithm in addition to Sobel and outputs were checked how they behave on the experimenting images. From visual analysis of images showed in Fig. 12, it is seen that with the use of Prewitt operator in Canny, the performance of Canny degrades. In Canny, Prewitt becomes noise prone as it identifies false edges and Canny final output fails to recognize original edges. On the other hand, performance of Robert operator in Canny is better in than Prewitt as it does not detects any false edge though it fails to detect low contrast edges. So only a canny with sobel operator is used for hardware implementation, as it is more accurate.
4.2 Hardware Results

The proposed design is compiled using hardware co-simulation compilation given in system generator token. Output results are observed in MATLAB Environment using Hardware co-simulation compilation. Generated bit stream file is dumped to Virtex-7 FPGA board through JTAG cable. Fig. 13 shows output results for each step of proposed canny algorithm. It can be observed that gradient output contains thick edges due to image smoothing. These edges are made thin using NMS block. Further it is given to Hysteresis thresholding which converts image into binary form as a final output edge map. Device (xc7v585t-1ffg1157) utilization summary for the proposed canny algorithm is shown in the Table 1.

![Fig. 4](image1.png)

**Fig. 4.** Canny edge detection output of ‘Lena’ image using different edge operator: (a) Robert (b) Prewitt (c) Sobel.

![Fig. 5](image2.png)

**Fig. 5.** Output results for each step of proposed Canny in MATLAB Environment: (a) Input Images (b) Gaussian smoother output (c) Gradient output (d) NMS output (e) Canny output.
Table 1. Device Utilization Summary.

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Flip-Flops</td>
<td>1066</td>
<td>728400</td>
<td>0.14%</td>
</tr>
<tr>
<td>Total No. of LUTs</td>
<td>879</td>
<td>364200</td>
<td>0.24%</td>
</tr>
<tr>
<td>Total No. of IOBs</td>
<td>82</td>
<td>600</td>
<td>13%</td>
</tr>
<tr>
<td>No. of BRAMs</td>
<td>17</td>
<td>795</td>
<td>2.1%</td>
</tr>
<tr>
<td>No. of DSPs</td>
<td>5</td>
<td>1260</td>
<td>0.39%</td>
</tr>
</tbody>
</table>

5 Conclusions

Proposed Canny algorithm is more efficient in terms of resource utilization than the conventional one. It is implemented on Virtex-7 FPGA by utilizing JTAG Hardware Co-Simulation compilation using XSG. An efficient Canny is designed by compromising the accuracy. It makes proposed design simpler and also enhances its limitation in terms of maximum frequency of operation. In future, the designed algorithm can be used for real-time application using ASIC based platform.

References

7. https://www.egr.msu.edu/classes/ece480/capstone/fall10/group03/ganley_t_app_note.pdf