ISSN 2395-1621

Implementation of Canny Edge Detection Algorithm on Real Time Platform

[#1]Mr. Prasad Khadke, [#2]Prof. S.R.Thite

[1]khadkepm@gmail.com
[2]srthite1988@gmail.com

[#1]Electronics and telecommunication Department, Savitribai Phule Pune University
Pune, Maharashtra, India

ABSTRACT

Edge detection is primary stage in image processing algorithms which helps in image enhancement, image segmentation, tracking and image coding. It also helps in object recognition. The canny edge detection algorithm due to its best performance, it is widely used in computer vision to locate sharp discontinuities in image intensity. In this paper, we are implementing a canny edge detection algorithm on FPGA kit so as to reduce the efforts of traditional canny edge detection algorithm such as threshold selection and time consumption.

***Keywords*— Canny, FPGA.**

ARTICLE INFO

Article History :

Received :18th June 2015

Received in revised form :

19th June 2015

Accepted : 23rd June 2015

Published online :

30th June 2015

I. INTRODUCTION

Edge detection is the basic operation and fundamental tool in image processing, machine vision and computer vision. In the areas of feature detection and feature extraction, this aims to identify points in a digital image at which the image brightness changes sharply with discontinuities and has wide application in research area. Many edge detection algorithms have been proposed such as Robert detector, Prewitt detector, Kirsch detector, Gauss-Laplace detector and Canny edge detector but due to its good performance Canny algorithm has been widely used in the field of image processing. Edges define the boundaries between regions in an image, which helps with segmentation and object recognition. They can show where shadows fall in an image or any other distinct change in the intensity of an image. The recent study on shows that the traditional Canny edge detector has two shortcomings. The threshold of the algorithm needs to be set by manual. Secondly, the algorithm is very time consuming and cannot be implemented in real time. A new self-adapt threshold Canny algorithm is proposed and a pipelined implementation on FPGA is designed to overcome the above disadvantages. Compared with the implementation in a PC based system, pipelined implementation on FPGA takes much less implementation time and can therefore be used for the mobile robot vision system which is very strict for the real-time performance of its vision system. Generally image processing algorithms are implemented on DSP kits. Image processing algorithms are repetitive in nature and require more computation. The alternative choice is to implement algorithm on the very expensive application specific integrated circuits (ASIC) or Field Programmable Gate Array (FPGA). Since FPGAs offer the features like reprogram ability flexibility parallelism short development time and computational power these FPGAs are best suited to implement image processing algorithms.

The Canny edge detection algorithm is known to many as the optimal edge detector. Canny's intentions were to enhance many edge detectors present at the age. The algorithm runs in 5 separate steps:

1. **Smoothing:** It is inevitable that all images taken from a camera will contain some amount of noise. Preventing that noise is mistaken for edges but this noise must be reduced. Therefore the image is first smoothed by applying a Gaussian filter to input image.

2. **Finding gradients:** The edges should be marked where the gradients of the image has large magnitudes. The Canny algorithm basically finds edges where the grayscale intensity of the image changes the most. These areas are found by determining gradients of the image. Gradients at each pixel in the smoothed image are determined by applying the Sobel-operator.

3. **Non-maximum suppression:** The purpose of this step is to convert the "blurred" edges in the image of the gradient magnitudes to "sharp" edges. Basically this is done by preserving all local maxima in the gradient image and deleting everything else.

4. **Double thresholding:** Potential edges are determined by thresholding. The edge-pixels remaining after the non-maximum suppression step are still marked with their strength pixel-by-pixel. Many of these will probably be true edges in the image, but some may be caused by noise or color variations for instance due to rough surfaces. The simplest way to discern between these would be to use a threshold, so that only edges which are stronger, for that a certain value would be preserved.

5. **Edge tracking by hysteresis:** Final edges are determined by suppressing all edges that are not connected to a strong edge. Strong edges are interpreted as "certain edges", and can immediately be included in the final edge image. Weak edges are included if and only if they are connected to strong edges. The logic is of course that noise and other small variations are unlikely to result in a strong edge with proper adjustment of the threshold levels. Thus strong edges will only be due to true edges in the original image. The weak edges can either be due to true edges or noise and color variations.

II. PROPOSED CANNY EDGE DETECTION ALGORITHM.

Canny developed an approach to derive an optimal edge detector to deal with step edges corrupted by a white Gaussian noise. The original Canny algorithm consists of the following steps:

a) Calculating the horizontal gradient Gx and vertical gradient Gy at each pixel location by Convolving with gradient masks.

b) Computing the gradient magnitude G and direction θG at each pixel location).

c) Applying Non-Maximal Suppression (NMS) to thin edges. This step involves computing the gradient direction at each pixel. If the pixel's gradient direction is one of 8 possible main directions, the gradient magnitude of this pixel is compared with two of its immediate neighbors along the gradient direction and the gradient magnitude is set to zero if it does not correspond to a local maximum. For the gradient directions that do not coincide with one of the 8 possible main directions, an interpolation is done to compute the neighboring gradients.

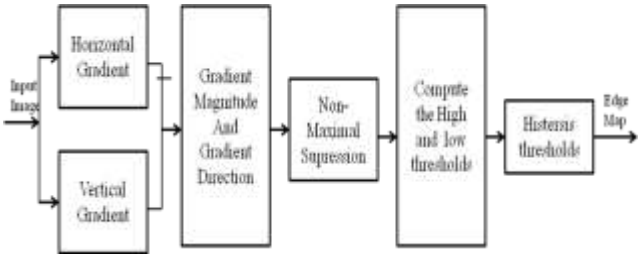

Fig.1 Block diagram of Canny edge detection algorithm.

d) Computing high and low thresholds based on the histogram of the gradient magnitude for the entire image. The high threshold is computed such that a percentage P1 of the total pixels in the image would be classified as strong edges. In other words, the high threshold corresponds to the point at which the value of the gradient magnitude cumulative distribution function (CDF) equals to P1. The low threshold is computed as a percentage P2 of the high threshold. The values of P1 and P2 are typically set as 20% and 40%, respectively.

e) Performing hysteresis thresholding to determine the edge map. If the gradient magnitude of a pixel is greater than the high threshold, this pixel is considered as a strong edge. If the gradient magnitude of a pixel is between the low threshold and high threshold, the pixel is labeled as a weak edge. Strong edges are interpreted as "certain edges", and can be immediately included in the final edge images. Weak edges are included if and only if they are connected to strong edges.

A block diagram of the Canny edge detection algorithm is shown in Fig. 1. In this original Canny edge detection algorithm, the gradient calculation is performed by using Finite-Impulse Response (FIR) gradient masks designed to approximate the following 2D sampled versions of the partial derivatives of a Gaussian function:

$$F_x(x,y) = -\frac{x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = (-x e^{-\frac{x^2}{\sigma^2}})(\frac{1}{\sigma^2} e^{\frac{y^2}{\sigma^2}}) \quad (1)$$

$$F_y(x,y) = -\frac{y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = (-y e^{-\frac{y^2}{\sigma^2}})(\frac{1}{\sigma^2} e^{\frac{x^2}{\sigma^2}}) \quad (2)$$

Where σ is the standard deviation of the Gaussian function. The size of the gradient masks used by the Canny edge detector is usually implemented as a function of the chosen σ, with larger values of σ yielding larger masks. However, the best choice of σ is image-dependent and can be selected by the user based on knowledge of the present noise characteristics or the size of desired objects in the image. The parameter σ can also be set by a separate application that estimates the noise and scale of objects in the image.

III. FLOW CHART.

Edge is the boundary between two regions in image which relatively differentiate gray-level properties. In digital image edge is defined as a sharp change in intensity between neighboring pixels. Basically image is composed of object and background. Recognition of object shape from its background is important in some image processing

applications where only shape of object matters and no any other information of image. This process is known as edge detection. A system block diagram explains basic idea about proposed hardware implementation.

a) Input image is a digital image which is of different formats such as .jpg, .bmp, .jpeg, .tiff, .gif, .png which are stored in memory. These images are color as well as black and white images. These images are converted from RGB to gray color using the MATLAB code.

b) In photography and computing, a grayscale digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest. Grayscale images are also called monochromatic, denoting the absence of any chromatic variation. Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum and in such cases they are monochromatic proper when only a given frequency is captured. But also they can be synthesized from a full color image; see the section about converting to grayscale. In computing, although the grayscale can be computed through rational numbers, image pixels are stored in binary, quantized form. Some early grayscale monitors can only show up to sixteen different shades, but today grayscale images intended for visual display are commonly stored with 8 bits per sampled pixel, which allows 256 different intensities to be recorded, typically on a non-linear scale. Technical uses often require more levels, to make full use of the sensor accuracy and to guard against round off errors in computations. Sixteen bits per sample is a convenient choice for such uses, as computers manage 16-bit words efficiently.

Conversion of a color image to grayscale is not unique; different weighting of the color channels effectively represents the effect of shooting black-and-white film with different-colored photographic filters on the cameras. A common strategy is to match the luminance of the grayscale image to the luminance of the color image. There are various ways to convert color values to grayscale. Anyone can use depending on the user's needs.

RGB Averaging Formula:

gs = (r+g+b) / 3;

Here averages of all three colors are calculated and saved in output image. Above formula can also be written as:

gs = r * 0.33 + g * 0.33 + b * 0.33;

i.e. 33% of all colors is used to compose final 100% grayscale component.

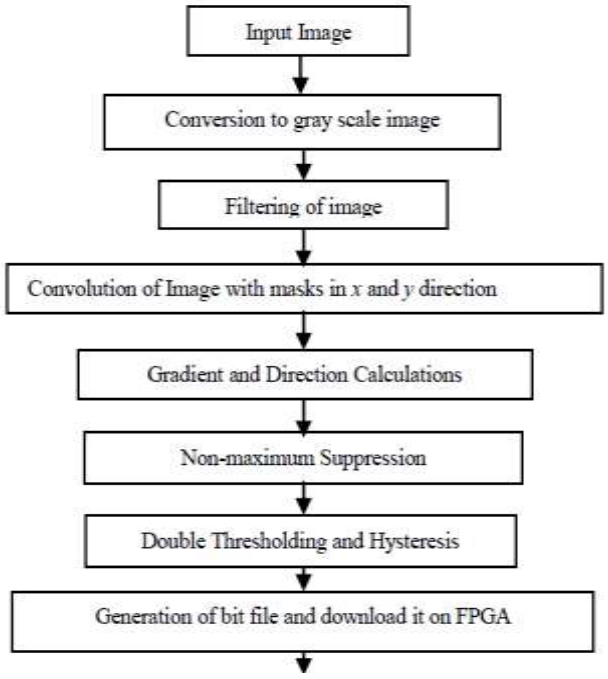

Fig.2 Flow of Canny Edge Detection Algorithm.

c) The Gaussian filter can be computed using a simple mask, it is used exclusively in the Canny algorithm. Once a suitable mask has been calculated, the Gaussian smoothing can be performed using standard convolution methods. A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. The larger the width of the Gaussian mask, the lower is the detector's sensitivity to noise. The localization error in the detected edges also increases slightly as the Gaussian width is increased.

d) After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image. The Sobel operator performs a 2-D spatial gradient measurement on an image. Then, the approximate absolute gradient magnitude at each point can be found.

e) The architecture of NMS block is shown in figure 4.3. It consists of window 3x3 unit, selector unit, arithmetic and comparator unit. Window 3×3 unit holds the 8 pixel values at a time and feeds the input to the selector unit and the middle value is given to the comparator unit.

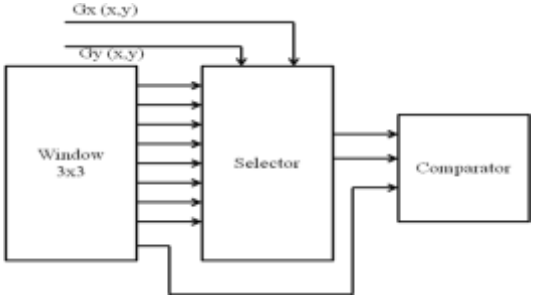

Fig.3 Non-maximal suppression unit.

f) The architecture of pipelined thresholding is shown in figure 4.4. The output of the non maximu

suppression unit contains some spurious edges. Hence the hysteresis thresholding is performed to reduce those effects. The threshold is calculated based on the gradient histogram i.e. we need the histogram of the image after the NMS operation. In non maximal suppression block the operation is performed to thin the edges as well as to detect all possible edges.

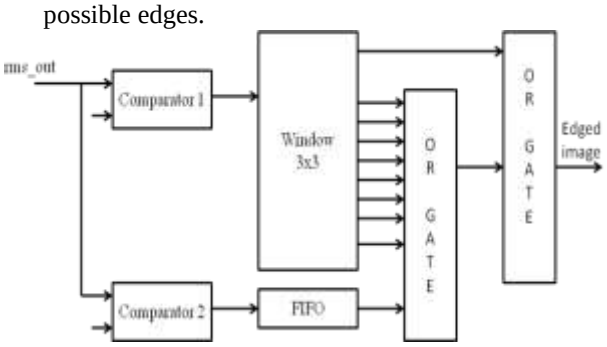

Fig. 4 Block diagram of pipeline thresholding.

To identify correct edges this algorithm considers two thresholds ie. High threshold and low threshold.

IV. RESULTS.

a) Original image.

b) Gray scale image.

c) Sdge detected image.

V. CONCLUSION

The original Canny algorithm relies frame-level statistics to predict the high and low thresholds and thus has latency proportional to the frame size. In order to reduce the large latency and meet real-time requirements, we are implementing it on FPGA. To meet the requirements an adaptive threshold selection method is proposed that predicts the high and low thresholds of the entire image while only processing the pixels of an individual block. This will in three benefits:

a. A significant reduction in the latency.

b. Better edge detection performance.

c. The possibility of pipelining the Canny edge detector with other block-based image codes.

REFRENCES

[1]. Housam Khalifa Bashier, Lau Siong Hoe , Pang Ying Han, IEEE July 2014 “Graphical Password: Pass-Images Edge Detection.”

[2]. Kiranjeet Kaur, Sheenam Malhotra,IJAIEM Volume 2, Issue 4, April 2013 “A Survey On Edge Detection Using Different Techniques.”

[3]. S.Lakshmi, Jeppiar Engineering College Chennai, IJCA CASCT 2010 “A study of Edge Detection Techniques for Segmentation Computing Approaches.”

[4]. Luc Vincent, member, IEEE April 1993 “Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms.”

[5]. Qian Xu, Srenivas Varadarajan, Chaitali Chakrabarti, Fellow, IEEE, and Lina J. Karam, Fellow, IEEE “A Distributed Canny Edge Detector: Algorithm and FPGA Implementation”.

[6]. Alasdair Mc Andrew. “Introduction to Digital Image Processing with MATLAB”.

[7]. Ms. P.H. Pawar, Prof. R. P. Patil, IJECS “FPGA Implementation of Canny Edge Detection Algorithm.”

[8]. Ehsan Nadernejad, Applied Mathematical Sciences, Vol. 2, 2008, no. 31, 1507 - 1520 “Edge Detection Techniques: Evaluations and Comparisons”.

[9]. Dr.S.Vijayarani, Mrs.M.Vinupriya, IJIRCE“Performance Analysis of Canny and Sobel Edge Detection Algorithms in Image Mining.”

