Design and Implementation of Edge Detection Algorithm in dsPIC Embedded Processor

Prashan Premaratne*, Sabooh Ajaz*, Ravi Monaragala+, Nalin Bandara+, Malin Premaratne++
*School of Electrical Computer and Telecommunications Engineering
University of Wollongong, North Wollongong, NSW 2522, Australia.
Email: prashan@uow.edu.au
+Centre for Research and Development, Army Cantonment, Panagoda, Sri Lanka.
++Advanced Computing and Simulation Laboratory (AXL)
Department of Electrical and Computer Systems Engineering
Monash University, Clayton, Victoria, Australia.

***Abstract*—The research presented here is an attempt to use a very basic, low cost and non-specialized microcontroller for image processing tasks. The applications emanating from such an attempt will result in inexpensive face detection, intelligent motion sensors to low cost vehicle counting systems. We have been able to develop a system based on Microchip dsPIC microcontroller that implements edge detection of still images.**

Hardware-based signal processors such as Texas Instrument DSP (Digital Signal Processing) or Field Gate Arrays (FPGA) are generally an expensive solution for image processing applications. On the other hand a conventional 8-bit microcontroller doesn't have enough capability to handle memory intensive DSP algorithms. In this regard, Microchip offers a tradeoff between cost and performance. Although performance does not compete with TI DSPs or FPGAs, the proposed system yet provides a sound platform to perform Signal processing directly on embedded hardware. Our research presents a preliminary approach to perform any type of image processing task using microchip 16-bit Microcontrollers and 16-bit digital signal controllers. Even though this attempt is aimed at Edge Detection, the research opens up possibilities for numerous other algorithms of signal and image processing that can be implemented using the same low cost hardware.

I. Introduction

THIS research reports one of the first attempts to implement image processing algorithms embedded on hardware provided by low cost processors from Microchip corporation. The only resources available for such an effort are confined to FPGA systems [5, 6] or costly DSP processors. The discussion here will detail the development of a low cost hardware based platform to perform image processing.

One common factor that lies with most of the signal and image processing algorithms is that it is highly computationally memory intensive. Specifically for image processing, the memory is the key requirement. Embedded hardware always faces memory limitations. Although microchip digital signal controller dsPIC [7] is fully capable of implementing most signal processing tasks, it falls short of on-chip memory resources for image processing applications. If the system needs to incorporate a camera and an LCD to capture data and display it, the microcontroller needs to be interfaced with a special co-processor SSD1928 [8] to overcome this memory limitation. Figure 1. shows a diagram interfacing dsPIC with SSD1928 camera and LCD. This co-processor takes the load off the main processor so that it can fully utilize its resources on image processing algorithm implementation. After lengthy deliberations, we have decided to implement Canny Edge Detection in order to demonstrate the image processing capabilities of the dsPIC.

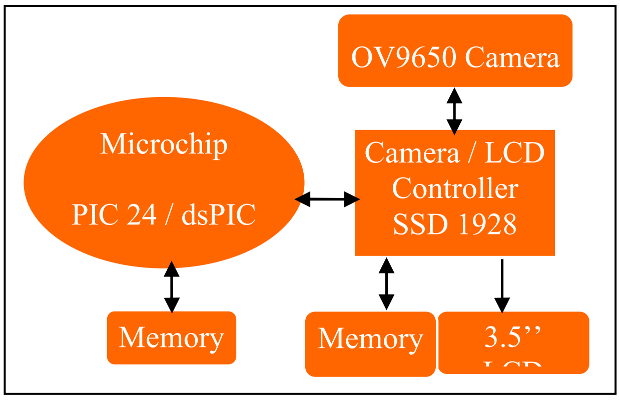


Fig. 1. Overview of the hardware setup

There are large numbers of edge detection techniques available. The Quality of edge detection can be measured from several criteria objectively. Some criteria are proposed in terms of mathematical measurement [11], some of them are based on application and implementation requirements. In all five cases given below, a quantitative evaluation of performance requires the use of images where the true edges are known.

i) Good detection: There should be a minimum number of false edges. Usually, edges are detected after a threshold

operation. The high threshold will lead to less false edges, but it also reduces the number of true edges detected.
ii) Noise sensitivity: The Edge detector should have some mechanism to remove noise or at least reduce noise to some acceptable level.
iii) Good localization: The edge location must be reported as close as possible to the correct position, i.e. edge localization accuracy (ELA).
iv) Orientation sensitivity: The operator not only detects edge magnitude, but it also detects edge orientation correctly. Orientation can be used in post processing to connect edge segments, reject noise and suppress non-maximum edge magnitude.
v) Speed and efficiency: The algorithm should be fast enough to be usable in an image processing system. An algorithm that allows recursive implementation or separately processing can greatly improve efficiency.

I. HARDWARE CONSIDERATIONS

This project uses Matlab where an image is treated as a matrix. However, in hardware the pixels are stored in memory and is not similar to the matrix structure of Matlab. In hardware (Microchip dsPIC), each memory location is identified by a unique address [15]. For a QVGA image of size 320x240, Matlab stores the data in a matrix where as in hardware, the data is stored in a memory location unlike in a matrix. As precursor to implement an algorithm such as Canny edge detection in hardware, it would first be implemented in Matlab without using built in functions. This will then be easily transferred to C or C++ implementation that can be converted to machine code.

A. *Canny's Implementation in Matlab*

Canny edge detection comprises of following steps:

1. **Smoothing:** Blurring of the image to remove noise.
2. **Finding gradients:** The edges should be marked where the gradients of the image has large magnitudes.
3. **Thresholding:** Potential edges are determined by Thresholding

It is inevitable that all images taken from a camera will contain some amount of noise. In order to prevent noise being mistaken for edges, noise must be reduced. Therefore the image is first smoothed by applying a Gaussian filter. The mask of Gaussian filter of size 5x5 with a standard deviation of 1.4 is shown below [16].

$$GaussianMask = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Gaussian can be calculated using the relationship:

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

Where x and y represent the coordinate distance from the centre pixel (X, Y). Mask (x-1,y-1) can be calculated as follows:

$$g(-1,-1) = e^{-\frac{((-1)^2+(-1)^2)}{2(1.4)^2}} = 0.7748$$

It is evident that all such values are in fractional form. In order to perform any kind of mathematical operation on fractions, a floating point number system is needed. Even though Matlab has ample resources to implement floating point calculations, hardware implementation requires converting fractions to integers for hardware platforms. A possible solution to the problem is denoted using dot points.

- Multiply mask with 255 $\rightarrow(2^8 -1)$ for 8bit per pixel case
- Round off fractions to nearest integer
- Scale answers by sum of all weights.

A diagrammatic view of the edge detection process is shown in Fig. 2.

B. *Properties of Gaussian Mask/Filter*

Smoothing or blurring of high contrast regions is an essential step before applying any type of gradient mask on the image. Gaussian mask which is effectively a gradient mask is highly sensitive to noise due to its derivative nature. Smoothing therefore will result in better edge detection. The following properties can be associated with Gaussian mask:

- Pixel intensity is inversely proportional to distance from origin
- It Blurs the image to produce smoothing and reduces the Noise
- The distribution has a mean Zero (Centered around origin) and Standard Deviation has to be defined
- All values are positive
- Output on smooth regions unchanged

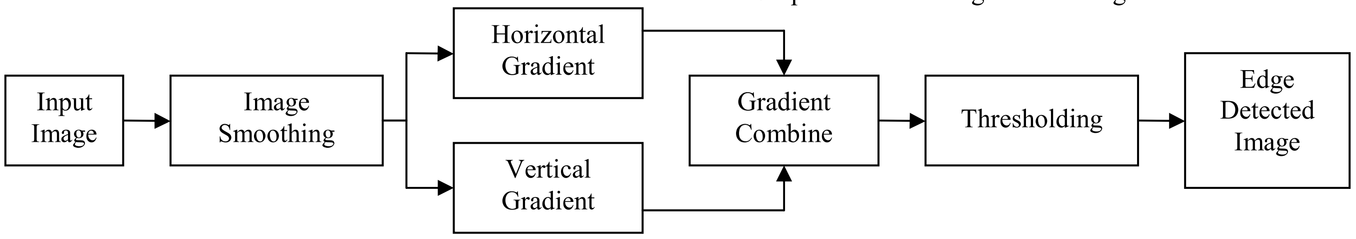


Fig. 2. Block diagram of an edge detector

- Blurs high-contrast regions
- Larger mask leads further smoothing
- Offers complete circular symmetry.

C. *Finding Gradients*

The Canny algorithm generally 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 Sobel mask (Fig. 3.) [16].

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Fig. 3. Sobel convolution mask for horizontal(x) and vertical(y) edges

The first step is to approximate the gradient in the x- and y-direction respectively by applying the mask shown in Figure 3. The gradient magnitudes (also known as the edge strengths) can then be determined as a Euclidean distance measure by applying the law of Pythagoras as shown in Equation below [16].

$$G = \sqrt{\left(G_x^2 + G_y^2\right)} \qquad (2)$$

It is sometimes simplified by applying Manhattan distance measure as shown in Equation below to reduce the computational complexity.

$$|G| \approx |G_x| + |G_y| \qquad (3)$$

D. *Thresholding*

Gradient image may contain true edges, but some edges may be caused by noise or color variations due to uneven surfaces. The simplest way to discern between these would be to use a threshold, so that only edges stronger than a certain value would be preserved. Edge pixels stronger than the threshold are marked and the ones weaker than the threshold are suppressed. The effect on the test images with thresholds of 30 is shown in Figures 4 and 5.

II. HARDWARE IMPLEMENTATION

The hardware implementation of the projects will involve image acquisition using a camera, main processor to implement edge detection algorithm and an image processor to render the image on a display and a display device such as an LCD to view the processed image. Next sections will discuss these components in detail followed by programming procedure for image processing algorithms.

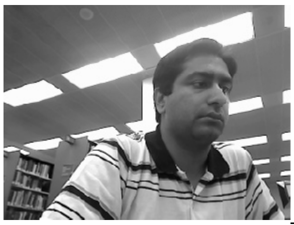


Fig. 4(a) Original Image

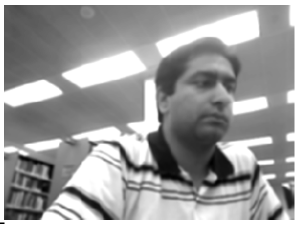


Fig. 4(b) Gaussian smoothed image

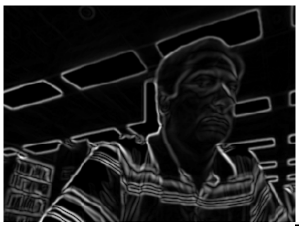


Fig. 5(a). Gradient Approximation of Image using Sobel kernal

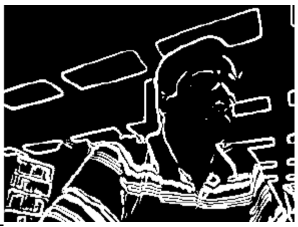


Fig. 5(b). Final edge detected image after thresholding

A. *Image Acquisition Device*

The image is acquired through a CMOS camera with digital outputs (Not PAL/NTSC based output used in conventional cameras) [17]. The camera being used is a Omni Vision OV9650 Color CMOS SXGA (1.3 Mega Pixel) Camera Chip which supports following resolutions [18]:
SXGA (1280x1024) , VGA (640x480), CIF (352x288), QVGA(320x240), QCIF(176x144), QQVGA (160x120), and QQCIF (88x72).

The camera also supports following output formats [18]:

- YUV/YCbCr → 8 bits, 4:2:2 (Interpolated color)
- GRB 8 bits → 4:2:2 (Interpolated color)
- RGB565 →5-bit R, 6-bit G, 5-bit B
- RGB555 →5-bit R, 5-bit G, 5-bit B
- Raw RGB →10/8 bits (Bayer filter color)

We have decided to use RGB565 with QVGA (320x240) resolution as it is commonly used. It is very important to select the bits per pixel and the resolution based on the onboard memory available. As the resolution and bits per pixel increase, the memory requirement also increases. It is important to select a resolution such that the processor is not overloaded. QVAG resolution of 320 * 240 will result in a resolution of 76800 pixels and considering RGB565 with 16 bits per pixel will require only 2 bytes per pixel. This will result in a total memory requirement of 76800 * 2 equaling 153600 Bytes or 150KB for each image and further memory for the processed image which will be used for edge detection as discussed before.

Another important aspect of any live image capturing device is its "Frame Rate" or sampling rate. It specifies how often an image is refreshed. Normally 30 frames per second are required to view a good motion picture. Since we are only focused on edge detection of a still image, a lower refresh rate would be adequate.

B. Image Processor

The main image processor that we are using in the project is Microchip's 16-bit PIC24/dsPIC. SSD1928 co-processor [9] is sharing load from the main processor [7, 20]. Tasks assigned to co-processor and the main processor can be categorized as follows:

Tasks assigned to co-processor SSD1928

1. Interface with OV9650 camera
2. Interface with 3.5'' LCD
3. Store QVGA picture in RGB565 format inside on chip image buffer.
4. Continuously update LCD from on chip image buffer.
5. Update on chip image buffer from the camera using I2C protocol according to predefine frame rate.

Tasks assigned to main processor PIC24/dsPIC

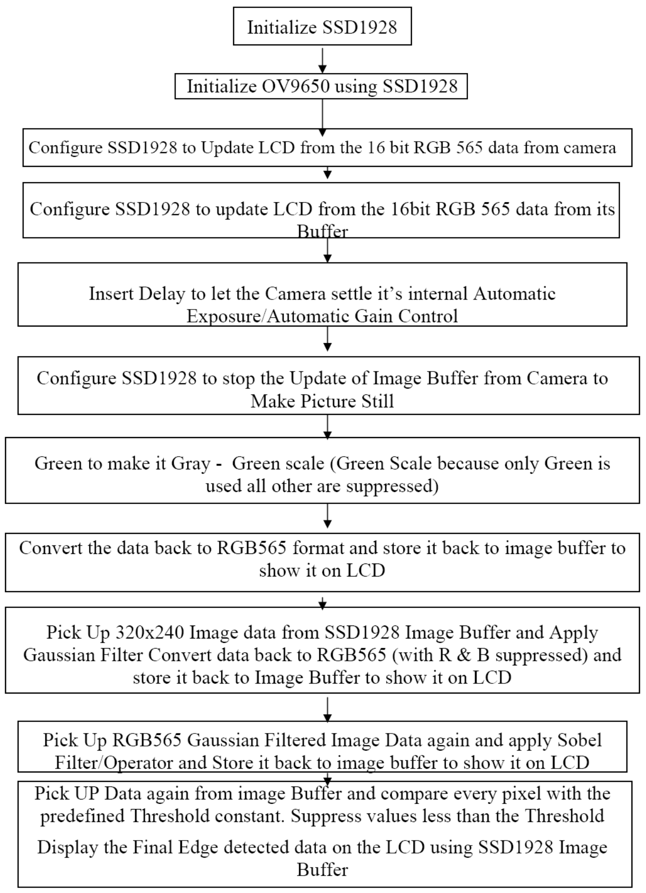
1. Configure co-processor for the operations / tasks mentioned above.
2. Taking data from image buffer of co-processor and performing image processing on that, then storing the processed image back to image buffer to show it on LCD.
3. Handling serial communication with laptop to send image data to software (Matlab) if required.

It is quite evident that by using a co-processor, the load on the main processor considerably reduces freeing up resources to perform any type of signal / image processing.

C. Programming with MPLAB

The hardware programming is handled by MPLAB integrated environment with MPLAB C compiler for PIC24/dsPIC and Microchip Graphics Library [21-23]. MPLAB Integrated Development Environment (IDE) is a free, integrated toolset for the development of embedded applications employing Microchip's PIC 24 and dsPIC microcontrollers. MPLAB IDE runs as a 32-bit application on MS Windows, is easy to use and includes a host of free software components for fast application development and super-charged debugging. MPLAB IDE also serves as a single, unified graphical user interface for additional Microchip and third party software and hardware development tools. Moving between tools is a snap, and upgrading from the free software simulator to hardware debug and programming tools is done in a flash because MPLAB IDE has the same user interface for all tools [21].

The MPLAB C Compiler for PIC24 / dsPIC (also known as MPLAB C30) is a full-featured ANSI compliant C compiler for the Microchip 16-bit devices. MPLAB C is fully compatible with Microchip's MPLAB Integrated Development Environment, allowing source level debugging with the, MPLAB ICD 2 In-Circuit Debugger and MPLAB SIM Simulator [22].



Initialize SSD1928
Initialize OV9650 using SSD1928
Configure SSD1928 to Update LCD from the 16 bit RGB 565 data from camera
Configure SSD1928 to update LCD from the 16bit RGB 565 data from its Buffer
Insert Delay to let the Camera settle it's internal Automatic Exposure/Automatic Gain Control
Configure SSD1928 to stop the Update of Image Buffer from Camera to Make Picture Still
Green to make it Gray - Green scale (Green Scale because only Green is used all other are suppressed)
Convert the data back to RGB565 format and store it back to image buffer to show it on LCD
Pick Up 320x240 Image data from SSD1928 Image Buffer and Apply Gaussian Filter Convert data back to RGB565 (with R & B suppressed) and store it back to Image Buffer to show it on LCD
Pick Up RGB565 Gaussian Filtered Image Data again and apply Sobel Filter/Operator and Store it back to image buffer to show it on LCD
Pick UP Data again from image Buffer and compare every pixel with the predefined Threshold constant. Suppress values less than the Threshold
Display the Final Edge detected data on the LCD using SSD1928 Image Buffer

Fig. 6. Program flow

The features of the MPLAB C compiler can be highlighted as follows:

- ANSI compliant with standard, math, memory, data conversion and math libraries.
- Optimized to generate as much as 30% less code than other 16-bit MCU compilers
- Strong support for in-line assembly when total control is absolutely necessary
- Peripheral library for quick coding using Microchip device peripherals
- Allows code and data to be located at absolute addresses
- Supports advanced code size optimizations
- Support for DSP accumulator registers from the C language (For dsPIC digital signal controllers)
- Support for DSP intrinsic (functions) from the C language. DSP intrinsic map directly to native dsPIC assembly language instruction.

The Microchip Graphics Library is highly modular and is optimized for Microchip's 16- and 32-bit microcontrollers. It is compatible with almost all well-known image co-processors and can also be used in any image co-processor with slight modifications. The Microchip graphics library supports [23] Graphics, Fonts, User interface for Touch sensing, Image and animation.

MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost effective high-speed hardware debugger/programmer for Microchip Digital Signal Controllers (DSC). It debugs and programs PIC controllers and dsPIC with the powerful, yet easy-to-use graphical user interface of MPLAB Integrated Development Environment (IDE). It can be connected to PC through high-speed USB 2.0 interface while being connected to the target with a compatible connector [24].

The core advantage of in-circuit debugger is its ability to perform real time debugging. It gives the user full control over program execution. Executing program code can start / stop anywhere in the program. It also provides the values of the program variables at runtime. This is an extremely important tool for embedded system development using dsPIC because it can give us an insight into the program flow. The program flow of the edge detection process in depicted in Fig. 6.

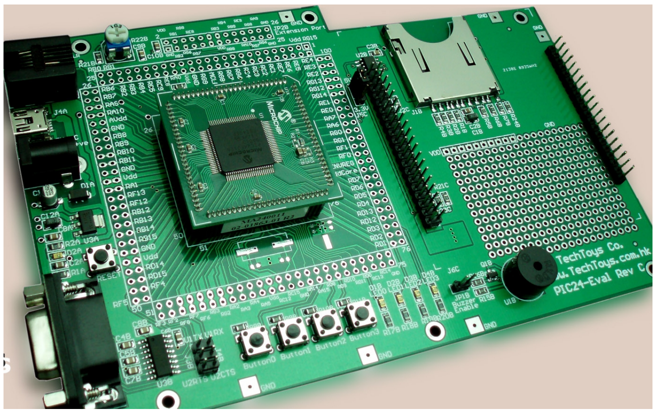


Fig. 7. Evaluation platform for Microchip 100-pin microcontroller

D. *Features of Microchip's 16-bit dsPIC/PIC 24*

The notable features of the Microchip 100-pin microcontroller evaluation platform (Fig. 7.) used in the project are given below [20]:

- 16 Bit Operations which are most suitable for calculations involving values more than Decimal 255.
- Upto 256K Bytes of on Chip Program Memory.
- Upto 32K Bytes of On chip RAM.
- On chip oscillator with PLL (Phase Locked Loop) to make speed nearly equal to 40 MIPS (Million Instructions per second)
- In Circuit Debugger Support using Microchip ICD2 / ICD3
- Assembly & C Language Support. C language is particularly helpful, when implementing filtering tasks as these computationally intensive processes. Microchip C-30 Compiler supports Integer Arithmetic, Fixed Point Arithmetic and Floating Point Arithmetic [25].
- dsPIC also supports Vector based calculations, which is particularly useful in Signal Processing.

We have used Microchip graphics library to configure coprocessor and to control LCD and Math library to perform all the calculation work. We have also used integer arithmetic to perform the calculations. Hence the fraction gets automatically neglected. But there is microchip library support available for fixed point and floating point arithmetic. These libraries can be used to get more accuracy in the results and to suppress the noise. Figures 8 and 9 depict the results of embedded processing

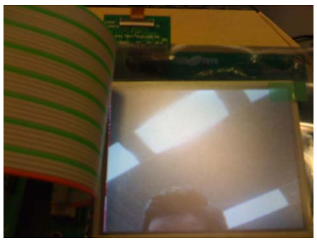


Fig. 8(a) Captured image using camera

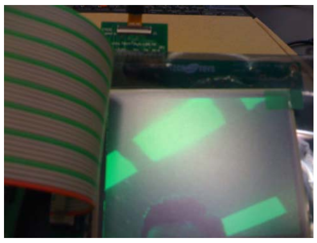


Fig. 8(b). Image converted to green scale and smoothed

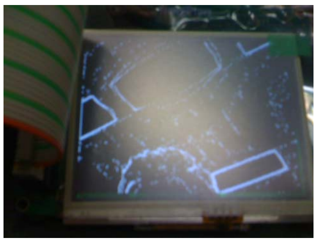


Fig. 9(a) Edge detected image on LCD

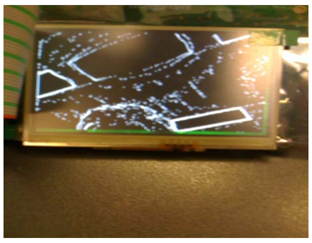


Fig. 9(b) Edge detected image on LCD (another view)

III. CONCLUSION

The goal of this research project was to develop low cost embedded hardware based edge detector. Edge detection using dsPIC processor can be regarded as a starting point of implementing complex embedded vision and image processing algorithms. This attempt can be extended to applications such as object counting, face detection, feature detection etc.

The project utilized only few of the features of Microchips dsPIC. There is still a lot of room for enhancements. For example, integer arithmetic was used for all the calculations resulting in noise in edge detected image which can be easily reduced by using floating point arithmetic or by using fixed point arithmetic. There are various dsPIC features available, which can perform vector math. The vector math is an essential part of signal / image processing that can be used to speed up the processing.

The edge detection implanted in this project is only capable of operating on still images and not on motion video due to speed limitation. It is not possible to processes 320x240 16 bit data three times with 40 MIPS. For real time edge detection, FPGA or II DSP can be used with very cost.

REFERENCES

[1] N. G. See and C. K. hiang. "Edge Detection using supervised Learning and voting scheme". *Nanyang Technological University*, National university of Singapore, Singapore.

[2] Md. S. Bhuiyan, Y. Iwahori, and A. Iwata. "Optimal edge detection under difficult imaging conditions". *Lecture Notes in Computer Science*, 1997, vol. 1352/1997, pp. 25-32.

[3] Available: http://www.mathworks.com/products/image/

[4] http://www.mathworks.com/access/helpdesk/help/toolbox/images/edge.html

[5] J. Hamon, V. Fristot and R. Rolland,"FPGA implementation of a real time multi-resolution edge detection video filter" *8th European Workshop on Microelectronics Education*, 2010.

[6] H. S. Neoh, A. Hazanchu. "Adaptive Edge Detection for Real-Time Video Processing using FPGAs" Available http://www.uweb.ucsb.edu/~shahnam/AEDfRTVPUF.pdf

[7] sPIC® Digital Signal Controllers http://ww1.microchip.com/downloads/en/DeviceDoc/DS-70095K.pdf http://www.mathworks.com/access/helpdesk/help/toolbox/images/edge.html

[8] Available: http://www.solomon ystech.com/products/ssd1928.htm.

[9] Available: www.solomon-systech.com/pdf/AppNote_SSD1928_10.pdf

[10] http://www.pages.drexel.edu/~weg22/edge.html

[11] V. Mittal., "Edge Detection Technique using Fuzzy Logic". Master of Engineering Thesis, Thapar University Patiala

[12] Z. Yaniv., Edge Detection. Lecture Notes School of Engineering and Computer Science The Hebrew University, Jerusalem, Israel.

[13] I. Sobel., An isotropic 3x3 image gradient operator. In H. Freeman, editor, Machine Vision for Three-Dimensional Scenes, pages 376--379. Academic Press, 1990.

[14] J.F. Canny, A computational approach to edge detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence* (IEEE TPAMI), Vol. 8(6), pp. 769-798, 1986.

[15] Page 33, dsPIC33FJXXXGPX06/X08/X10 datasheet. Available:: http://ww1.microchip.com/downloads/en/ DeviceDoc/70286C.pdf

[16] http://www.cvmt.dk/education/teaching/f09/VGIS8/ AIP/canny_09gr820.pdf

[17] Omni Vision OV9650 Color CMOS SXGA 1.3Mega Pixel Camera Chip, http://www.techtoys.com.hk/

[18] Omni Vision OV9650 Register Set, Available: http://www.techtoys.com.hk/Components/OV9650_MOD/OV9650_DS%20(1.3).pdf

[19] RGB565 Color Model. Available: wikipedia.org/wiki/RGB_color_model

[20] dsPIC33FJXXXGPX06/X08/X10 Data Sheet, Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70286C.pdf

[21] MPLAB integrated development environment, Available: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002Asdsa

[22] MPLAB C comiler for dsPIC / PIC 24, Available: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010065Sa

[23] Microchip Graphics Library, Available: https://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2608&page=1&param=en532061Dsa