International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-4 Number-2 Issue-15 June-2014

Image Processing Algorithms – A Comprehensive Study

Mahesh Prasanna K[1], Shantharama Rai C[2]

Abstract

Digital image processing is an ever expanding and dynamic area with applications reaching out into our everyday life such as medicine, space exploration, surveillance, authentication, automated industry inspection and many more areas. These applications involve different processes like image enhancement and object detection [1]. Implementing such applications on a general purpose computer can be easier, but not very time efficient due to additional constraints on memory and other peripheral devices. Application specific hardware implementation offers much greater speed than a software implementation. With advances in the VLSI (Very Large Scale Integrated) technology hardware implementation has become an attractive alternative. Implementing complex computation tasks on hardware and by exploiting parallelism and pipelining in algorithms yield significant reduction in execution times [2].

Keywords

Digital Image Processing, Application Specific Integrated Circuits, Digital signal processors, Field Programmable Gate Arrays, Hardware Design Language.

1. Introduction

This work is structured into two parts. The first part gives a comprehensive study of image processing algorithms. Given the importance of digital image processing and the significance of their implementations on hardware to achieve better performance, this work addresses image processing algorithms like median filter, morphological processing, convolution operation and edge detection. The second part covers topics concerning a range of technologies used in the hardware implementation of typical image processing systems, e.g. image sensors, signal processing units, memory technologies and displays. Comparisons are made in various technologies regarding performance, area and power consumption cost etc.

Manuscript received May 18, 2014.

Mahesh Prasanna K, Department of Information Science & Engineering, Alva's Institute of Engineering & Technology, Moodbidri, India.

Shantharama Rai C, Department of Electronics & Communication Engineering, Canara Engineering College, Benjanapadavu, India.

2. Image Processing Algorithms

This section discusses the most commonly used image processing algorithms like, 1) Filtering, 2) Morphological Operations, 3) Convolution, and 4) Edge detection.

2.1 Filtering

The classification of image processing filters is presented in Figure 2.1. In the following section one of the filters (Median Filter) is explained in detail.

2.2 Median Filter

A median filter is a non-linear digital filter which is able to preserve sharp signal changes and is very effective in removing impulse noise (or salt and pepper noise) [1]. An impulse noise has a gray level with higher or lower value that is different from the neighbourhood point. Linear filters have no ability to remove this type of noise without affecting the distinguishing characteristics of the signal. Median filters have remarkable advantages over linear filters for this particular type of noise. Therefore median filter is very widely used in digital signal and image/video processing applications. A standard median operation is implemented by sliding a window of odd size (e.g. 3x3 window) over an image. At each window position the sampled values of signal or image are sorted, and the median value of the samples replaces the sample in the center of the window as shown in Figure 2.2.

The main problem of the median filter is its high computational cost; for sorting *n* pixels, the time complexity is *O(n log n)*, even with the most efficient sorting algorithms. When the median filter is carried out in real time, the software implementation in general-purpose processors does not usually give good results. The execution times can be reduced by implementing median filters on FPGAs.

2.3 Morphological Operations

The word morphology commonly denotes a branch of biology that deals with the form and structure of

animals and plants. A mathematical morphology is a tool for extracting image components that are useful in the representation and description of region space. The term morphological image processing refers to a class of algorithms that transforms the geometric structure of an image. Morphology can be used on binary and gray scale images, and is useful in many areas of image processing, such as skeletonization, edge detection, restoration and texture analysis.

A morphological operator uses a structuring element to process an image. The structuring element is a window scanning over an image, which is similar to the pixel window used in the median filter. The structuring element can be of any size, but 3x3 and 5x5 sizes are common. When the structuring element scans over an element in the image, there may be instances where the structuring element completely fits inside the object or does not fit inside the object.

The two principal morphological operations are *dilation* and *erosion*. Dilation allows objects to expand, thus potentially filling in small holes and connecting disjoint objects. Erosion shrinks objects by etching away (eroding) their boundaries. These operations can be customized for an application by the proper selection of the structuring element, which determines exactly how the objects will be dilated or eroded. The Figure 2.3 shows the example of dilation and erosion.

2.3.1 The Dilation

The *dilation* process is performed by laying the structuring element B on the image A and sliding it across the image. The operation performed is described in a sequence of steps:

1. If the origin of the structuring element coincides with a 'white' pixel in the image, there is no change; move to the next pixel.
2. If the origin of the structuring element coincides with a 'black' pixel in the image, make black all pixels from the image covered by the structuring element.

With A and B as sets in Z^2, the dilation of A by B is – $A \oplus B = \{Z \mid (\hat{B})_Z \cap A \neq \emptyset\}$ i.e., the dilation of A by B is the set of all displacements, Z, such that B and A overlap by at least one element.

The structuring element can have any shape. Typical shapes are presented in Figure 2.4.

2.3.2 The Erosion

The *erosion* process is similar to dilation, but we turn pixels to 'white', not 'black'. As before, slide the structuring element across the image and then follow these steps:

1. If the origin of the structuring element coincides with a 'white' pixel in the image, there is no change; move to the next pixel.
2. If the origin of the structuring element coincides with a 'black' pixel in the image, and at least one of the 'black' pixels in the structuring element falls over a white pixel in the image, then change the 'black' pixel in the image (corresponding to the position on which the center of the structuring element falls) from 'black' to a 'white'.

With A and B as sets in Z^2, the erosion of A by B is – $A \ominus B = \{Z \mid (B)_Z \cap A^C = \emptyset\}$ i.e., the erosion of A by B is the set of points, z, such that B, translated by Z, is contained in A.

2.3.3 Opening and Closing

The two basic operations, dilation and erosion, can be combined into more complex sequences. The most useful of these for morphological filtering are called opening and closing. *Opening* consists of an erosion followed by a dilation and can be used to eliminate all pixels in regions that are too small to contain the structuring element. In this case the structuring element is often called a probe, because it is probing the image looking for small objects to filter out of the image. *Closing* consists of a dilation followed by erosion and can be used to fill in holes and small gaps.

Closing and opening will generate different results even though both consist of erosion and dilation. Opening generally smoothes the contour of an object, breaks narrow isthmuses, and eliminates thin protrusions. Closing also tends to smooth sections of the contour but, as opposed to opening, it generally fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour.

2.4 Convolution

Convolution is another commonly used algorithm. It is from a class of algorithms called spatial filters. Spatial filters use a wide variety of *masks*, also known as *kernels*, to calculate different results, depending on the function desired. For example, certain masks yield smoothing, while others yield low pass filtering or edge detection.

The convolution algorithm can be calculated in the following manner. For each input pixel window, the values in that window are multiplied by the convolution mask. Next, those results are added

together and divided by the number of pixels in the window. This value is the output for the origin pixel of the output image for that position. Mathematically, this is represented using the following equation. $y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} A(k_1, k_2) k(n_1 - k_1, n_2 - k_2)$ where A is the input image and k is the convolution kernel.

The input pixel window is always the same size as the convolution mask. The output pixel is rounded to the nearest integer. As an example, Figure 2.5 shows an input pixel window, the convolution mask, and the resulting output. This convolution mask in this example is often used as a noise-cleaning filter. The results for this algorithm carried over an entire input image will result in an output image with reduced salt-and-pepper noise. An important aspect of the convolution algorithm is that it supports a virtually infinite variety of masks, each with its own feature. This flexibility allows for many powerful uses.

2.5 Edge Detection

Edges are places in the image with strong intensity contrast. Edges often occur at image locations representing object boundaries; edge detection is extensively used in image segmentation when we want to divide the image into areas corresponding to different objects. Representing an image by its edges has the further advantage that the amount of data is reduced significantly while retaining most of the image information. Edges can be detected by applying a high pass frequency filter in the Fourier domain or by convolving the image with an appropriate kernel in the spatial domain. In practice, edge detection is performed in the spatial domain, because it is computationally less expensive and often yields better results. Since edges correspond to strong illumination gradients, the derivatives of the image are used for calculating the edges.

The Canny edge detection algorithm is considered a "standard method" and is used by many researchers, because it provides very sharp and thin edges. The Canny operator works in a multi-stage process. Canny edge detection uses linear filtering with a Gaussian kernel to smooth noise and then computes the edge strength and direction for each pixel in the smoothed image. This is done by differentiating the image in two orthogonal directions and computing the gradient magnitude as the root sum of squares of the derivatives. The gradient direction is computed using the arctangent of the ratio of the derivatives. Candidate edge pixels are identified as the pixels that survive a thinning process called non-maximal suppression. In this process, the edge strength of each candidate edge pixel is set to zero if its edge strength is not larger than the edge strength of the two adjacent pixels in the gradient direction. Thresholding is then done on the thinned edge magnitude image using hysteresis. In hysteresis, two edge strength thresholds are used. All candidate edge pixel values below the lower threshold are labelled as non-edges, and the pixels values above the high threshold are considered as definite edges. All pixels above low threshold that can be connected to any pixel above the high threshold through a chain are labelled as edge pixels. The schematic of the canny edge detection is shown in Figure 2.6.

2.5.1 Smoothing

In the first stage, a Gaussian convolution mask is used for smoothing. The effect of Gaussian convolution is to blur an image. The degree of smoothing is determined by the standard deviation of the Gaussian.

2.5.2 Gradient Calculation

After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image. Most edge detection methods work on the assumption that an edge occurs where there is a discontinuity in the intensity function or a very steep intensity gradient in the image. Most edge-detecting operators can be thought of as gradient-calculators. Since the gradient is a continuous-function concept and images are discrete functions, we have to approximate it. Since derivatives are linear and shift-invariant, gradient calculation is most often done using convolution. Numerous kernels have been proposed for finding edges, some of the kernels are: Roberts Kernel, Kirsch Compass Kernel, Prewitt Kernel, Sobel Kernel, and many others.

The Prewitt kernels are based on the simple idea of the central difference between rows for horizontal gradient and difference between columns for vertical gradient.

$$\frac{\partial I}{\partial x} \approx \frac{I(x+1,y) - I(x-1,y)}{2}$$

$$\frac{\partial I}{\partial y} \approx \frac{I(x,y+1) - I(x,y-1)}{2}$$

The convolution masks of Figure 2.7 are derived from these equations. These convolutions are used for calculating the horizontal and vertical gradients.

2.5.3 Magnitude and Phase

Convolution of the image with horizontal and vertical gradients produces horizontal gradient (dx) and vertical gradient (dy) respectively. The absolute gradient magnitude (|G|) is calculated by the mean square root of the horizontal (dx) and vertical (dy) gradients. That is, $|G| = \sqrt{dx^2 + dy^2}$. To reduce the computational cost of magnitude, it is often approximated with absolute sum of the horizontal and vertical gradients ($| G |=| dx | + | dy |$).

The direction of the gradient (θ) is calculated by arctangent of the vertical gradient to the horizontal gradient: $\theta = \arctan(dy / dx)$

Since arctangent is a very complex function and also requires floating point numbers, it is very difficult to implement such functions on FPGA. Instead, the value and sign of the components of the gradient is analyzed to calculate the direction of the gradient.

2.5.4 Non-Maximum Suppression

Once the direction of the gradient is known, the values of the pixels found in the neighbourhood of the pixel under analysis are interpolated. The pixel that has no local maximum gradient magnitude is eliminated. The comparison is made between the actual pixel and its neighbours, along the direction of the gradient. For example, if the approximate direction of the gradient is between 00 and 450, the magnitude of the gradient at $P_{x, y}$ is compared with the magnitude of the gradient at adjacent points.

2.5.5 Threshold

The output image of non-maximum suppression stage may consist of broken edge contours, single edge points which contribute to noise. This can be eliminated by thresholding with *hysteresis*. Two thresholds are considered for hysteresis, one high threshold other low threshold. If any edge response is above a high threshold, those pixels constitute definite edge output of the detector for a particular scale. Individual weak responses usually correspond to noise, but if these points are connected to any of the pixels with high threshold, they are more likely to be actual edges in the image. Such connected pixels are treated as edge pixels if their response is above a low threshold.

To get thin edges two thresholds (high threshold (TH) and low threshold (TL) are used. If the gradient of the edge pixel is above the TH, it is considered as an edge pixel. If the gradient of the edge pixel is below TL then it is unconditionally set to zero. If the gradient is between these two, then it is set to zero unless there is a path from this pixel to a pixel with a gradient above TH; the path must be entirely through pixels with gradients of at least TL.

3. Procedures for Hardware Implementation

There are two types of technologies available for hardware design. Full custom hardware design also called as Application Specific Integrated Circuits (ASIC) and semi-custom hardware device, which are programmable devices like Digital signal processors (DSPs) and Field Programmable Gate Arrays (FPGA's). Full custom ASIC design offers highest performance, but the complexity and the cost associated with the design is very high. The ASIC design cannot be changed and the design time is also very high. ASIC designs are used in high volume commercial applications. In addition, during design fabrication the presence of a single error renders the chip useless. DSPs are a class of hardware devices that fall somewhere between an ASIC and a PC in terms of the performance and the design complexity. DSPs are specialized microprocessors, typically programmed in C, or with assembly code for improved performance. It is well suited to extremely complex math intensive tasks such as image processing. Field Programmable Gate Arrays are reconfigurable devices. Hardware design techniques such as parallelism and pipelining techniques can be developed on a FPGA, which is not possible in dedicated DSP designs. Implementing image processing algorithms on reconfigurable hardware minimizes the time-to-market cost, enables rapid prototyping of complex algorithms and simplifies debugging and verification. Therefore, FPGAs are an ideal choice for implementation of real time image processing algorithms [3]. A comparison is made for the areas where each of these technologies prevails [4]. This is shown in Table 1 and 2.

No perfect technology exists that is competent in all areas. For a balanced embedded system design, a combination of some of the alternative technologies is a necessity. FPGAs have traditionally been configured by hardware engineers using a Hardware Design Language (HDL). The two principal languages used are Verilog HDL (Verilog) and Very High Speed Integrated Circuits (VHSIC) HDL (VHDL), which allows designers to design at various levels of abstraction. Verilog and VHDL are specialized design techniques that are not

immediately accessible to software engineers, who have often been trained using imperative programming languages. Consequently, over the last few years there have been several attempts at translating algorithmic oriented programming languages directly into hardware descriptions. C-based hardware descriptive languages have been proposed and developed since the late 1980s. Some of the C-based hardware descriptive languages include Cones [5], HardwareC [6], Transmogrifier C [7], SystemC [8], OCAPI [9], C2Verilog [10], Cyber [11], SpecC [12], Nach C [13], CASH [14]. A new C like hardware description language called Handel-C introduced by Celoxica [2, 15], allows the designer to focus more on the specification of an algorithm rather than adopting a structural approach to coding.

Application Specific Integrated Circuits (ASICs) represent a technology in which engineers create a fixed hardware design using a variety of tools. Once a design has been programmed onto an ASIC, it cannot be changed. Since these chips represent true, custom hardware, highly optimized, parallel algorithms are possible. However, except in high-volume commercial applications, ASICs are often considered too costly for many designs. In addition, if an error exists in the hardware design and is not discovered before product shipment, it cannot be corrected without a very costly product recall. Digital Signal Processors (DSPs) are a class of hardware devices that fall somewhere between an ASIC and a PC in terms of performance and design complexity. They can be programmed with either assembly code or the C programming language, which is one of the platform's distinct advantages. Hardware design knowledge is still required, but the learning curve is significantly lower than some other design choices, since many engineers have knowledge of C prior to exposure to DSP systems. However, algorithms designed for a DSP cannot be highly parallel without using multiple DSPs. Algorithm performance is certainly higher than on a PC, but in some cases, ASIC or FPGA systems are the only choice for a design. Still, DSPs are a very common and efficient method of processing real-time data. Field Programmable Gate Arrays (FPGAs) represent reconfigurable computing technology, which is in some ways ideally suited for video processing. Reconfigurable computers are processors which can be programmed with a design, and then reprogrammed (or reconfigured) with virtually limitless designs as the designer's needs change. FPGAs generally consist of a system of logic blocks (usually look up tables and flip-flops) and some amount of Random Access Memory (RAM), all wired together using a vast array of interconnects. All of the logic in an FPGA can be rewired, or reconfigured, with a different design as often as the designer likes. This type of architecture allows a large variety of logic designs dependent on the processor's resources, which can be interchanged for a new design as soon as the device can be reprogrammed.

4. Conclusions and Future Work

The most feasible solution for improving the performance of image processing systems is by implementing the image processing algorithms in hardware. The introduction of reconfigurable devices and system level hardware programming languages has further accelerated the design of image processing in hardware. Most of the system level hardware programming languages introduced and commonly used in the industry are highly hardware specific and requires intermediate to advance hardware knowledge to design and implement the system. In order to overcome this bottleneck various C-based hardware descriptive languages have been proposed over the past decade. These languages have greatly simplified the task of designing and verifying hardware implementation of the system. However, the synthesis process of the system to hardware was not completely addressed and was conducted using manual methods resulting in duplication of the implementation process. Handel-C is a new C-based language proposed that provides direct implementation of hardware from the C-based language description of the system. Handel-C language and the IDE tool introduced by Celoxica Ltd. provide both simulation and synthesis capabilities.

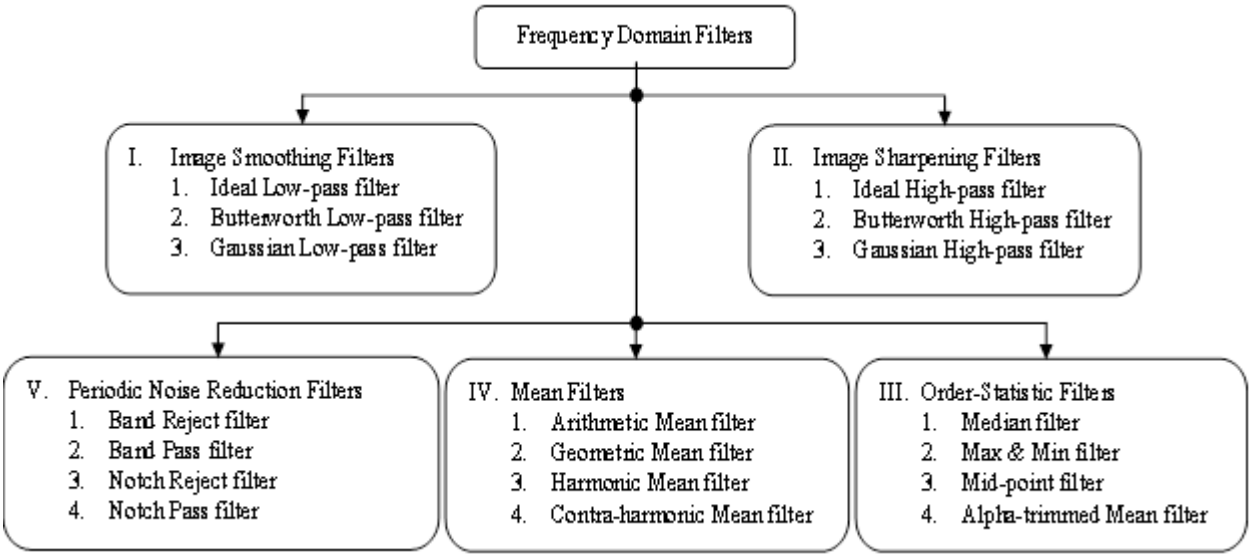


Figure 2.1: Classification of Image Processing Filters

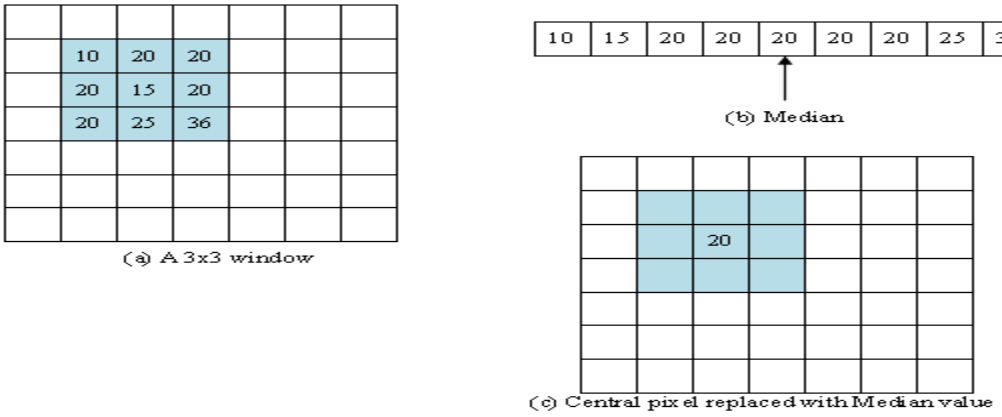
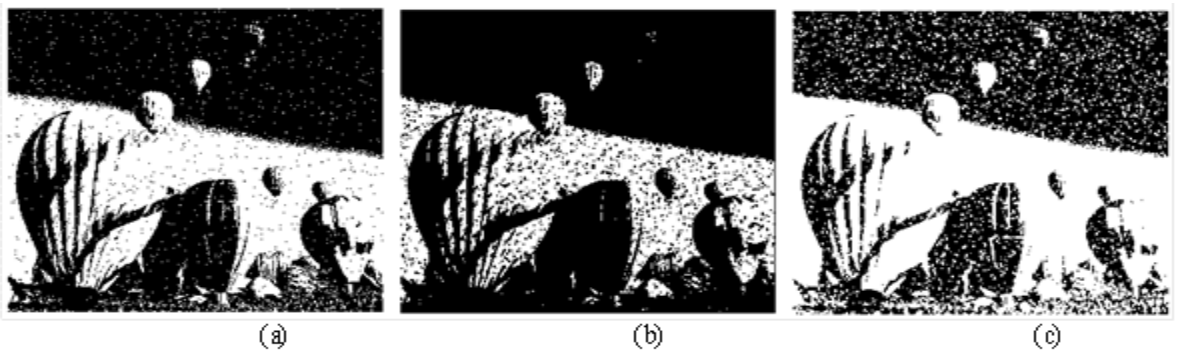


Figure 2.2: Median Filter.



(a) Original Image, (b) Resulting Image after dilation, (c) Resulting Image after erosion.

Figure 2.3: Example of dilation and erosion.

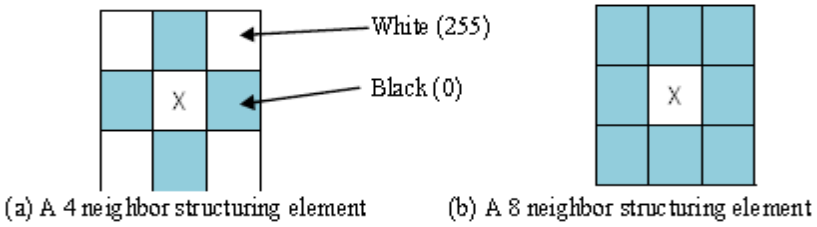


Figure 2.4: Typical shapes of structuring elements (B)

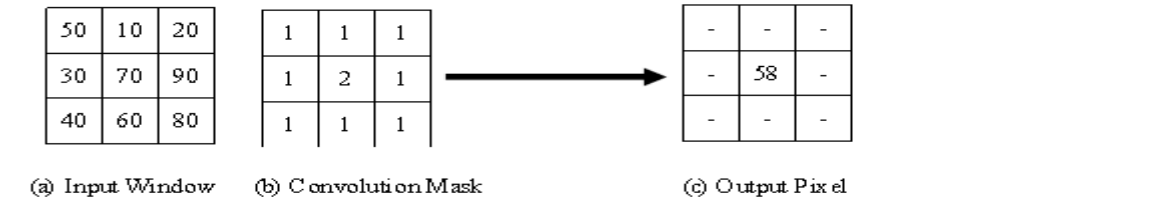


Figure 2.5: Convolution Algorithm Example.

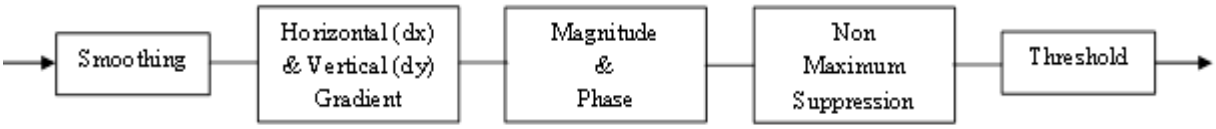


Figure 2.6: Schematic of Canny Edge Detection.

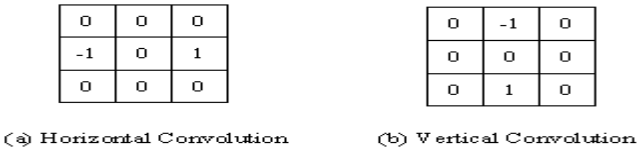


Figure 2.7: Gradient of Image.

Table 1: Comparisons of different types of signal processing technologies.

Technology	Performance	Power	Flexibility	Price
ASIC	Excellent	Good	Poor	Excellent
DSP	Excellent	Excellent	Excellent	Excellent
FPGA	Excellent	Fair	Excellent	Poor

Table 2: Comparisons between ASICs and FPGAs.

Performance Metric	ASICs	FPGAs
Power	Low	High
Flexibility	Low	High
Clock Speed	High	Low
Logic Integration	High	Low
Integrated Features	Low	High
Back-end Design Effort	High	Low
Unit Cost with Volume Production	Low	High

References

[1] John C. Ross. Image Processing Hand book, CRC Press. 1994.
[2] Peter Mc Curry, Fearghal Morgan, Liam Kilmartin. Xilinx FPGA implementation of a pixel processor for object detection applications. In the Proc. Irish Signals and Systems Conference, Volume 3, Page(s):346 – 349, Oct. 2001.
[3] Stephen D.Brown, R.J. Francis, J.Rose, Z.G.Vranesic. Field Programmable Gate Arrays, 1992.
[4] L. Adams. (2002, November) Choosing the right architecture for real-time signal processing designs. [Online]. Available: http://focus.ti.com/lit/an/spra879/spra879.pdf
[5] E. Stroud, R. R. Munoz, and D. A. Pierce. Behavioral model synthesis with cones. Design & Test of Computers, 5(3):22–30, July 1988.
[6] D. C. Ku and G. De Micheli. HardwareC: A language for hardware design. T.R. CSTL-TR-90-419, Stanford University, CA, Aug. 1990.
[7] D. Galloway. The Transmogrifier C hardware description language and compiler for FPGAs. In Proc. FCCM, pp. 136–144, Napa, CA, 1995.
[8] T. Grotker, S. Liao, G. Martin, and S. Swan. System Design with SystemC. ISBN 1-4020-. 7072-1, Mar 4, 6. Design space exploration, Mar 11, 13. Spring Break. 9. Kluwer, 2002.
[9] P. Schaumont et al. A programming environment for the design of complex high speed ASICs. In Proc. DAC, pp. 315–320, 1998.
[10] D. Soderman and Y. Panchul. Implementing C algorithms in reconfigurable hardware using C2Verilog. In Proc. FCCM, pp. 339–342, 1998.
[11] K.Wakabayashi. C-based synthesis experiences with a behavior synthesizer, "Cyber". In Proc. DATE, pp. 390–393, 1999.
[12] D. D. Gajski, J. Zhu, R. D¨omer, A. Gerstlauer, and S. Zhao. SpecC: Specification Language and Methodology. Kluwer Academic Publishers, 2000 [11] P.
[13] T. Kambe et al. A C-based synthesis system, Bach, and its application. In Proc. ASP-DAC, pp. 151–155, Yokohama, Japan, 2001.
[14] M. Budiu and S. C. Goldstein. Compiling application-specific hardware. In Proc. FPL, LNCS 2438, pp. 853–863, Montpellier, France, 2002.
[15] Celoxica, http://www.celoxica.com. Handel-C Language Reference Manual, 2003. RM-1003-4.0.

Mr. Mahesh Prasanna K., working as Associate Professor & HOD, Department of Information Science & Engineering, Alva's Institute of Engineering & Technology, Moiodbidri. He received his BE in Electronics & Communications Engineering from Mangalore University; M.Tech. in Computer Science & Engineering from VTU, Belgaum. Currently he is doing his research work in the field of Image Processing. He fields of interest are Artificial Intelligence, Control Systems, Embedded Systems, Fuzzy Logic, Image Processing, etc. His published several papers on National and International Journals.

Dr. Shantharama Rai C., working as Professor & Head, Department of Electronics & Communications Engineering, Canara Engineering College, Benjanapadavu. He received his BE in Electrical & Electronics Engineering from Mangalore University; M.Tech. from NITK, Surathkal, and Ph.D. from VTU, Belgaum. His fields of interest are Artificial Intelligence, Control Systems, Embedded Systems, Fuzzy Logic, Image Processing, etc. He is the member of several professional organizations. He published several papers on National and International Journals.