Control Theory and Informatics
ISSN 2224-5774 (print) ISSN 2225-0492 (online)
Vol 2, No.1, 2012

Digital Image Processing for Camera Application in Mobile Devices Using Artificial Neural Networks

Sachin P. Kamat

Samsung India Software Operations Pvt. Ltd., Bangalore, 560052, India

* E-mail: sachin.kamat@samsung.com

Abstract

This paper utilizes artificial neural networks based image processing techniques for low capacity and resource constrained devices like mobile phones for camera applications. The system is trained to develop the operating matrix, called the function matrix, by using artificial neural network theory, from a sample input and output image matrix. This is done when the system is in idle mode. After having obtained the function matrix, it can be very conveniently operated upon any other input image matrix by simple multiplication to obtain the desired modification in the input image in real time. Computer simulation results are provided to prove the concept.

Keywords: image processing, artificial neural networks, mobile devices, camera application

1. Introduction

Camera is an integral part of most mobile devices coming to market in recent times. High end mobile devices come with high resolution cameras and powerful sensors that can process large size images instantly through the on-chip image signal processors (ISP) and digital signal processors (DSP). They support several kinds of image effects like edge detection, smoothening, filtering, embossing, pencil sketch, etc. These operations are computationally intensive and require high speed CPUs and digital signal processors to perform them in real time. Thus these operations are not generally supported by low-end mobile devices which have a basic and low resolution sensor that can cater to only very basic image and colour effects. It is also not possible to perform all these image processing operations in software on the mobile device without loss of real time constraints and hence most of the operations need to be carried out on desktop computers using the bundled software tools.

This paper describes the concept of using artificial neural networks for image processing whereby the embedded system is trained to develop the operating matrix, called the function matrix, by using artificial neural network theory using a sample input and output image matrix. Having obtained the function matrix, it can be very conveniently operated upon any other input image matrix by simple multiplication to obtain the desired modification in the input image. The conventional methods of image processing operations involve convolving the given image matrix with a predetermined function matrix depending upon the effect required. This is a computationally intensive operation and it becomes necessary to know the function matrix before hand. In addition, the matrix required for convolution varies with the type of processing required. Image processing using artificial neural networks does not involve the conventional convolution techniques. On the other hand, the system is trained to generate the function matrix depending upon the type of output required. Several methods and techniques of achieving the same are already available and this paper attempts to utilize the results of artificial neural network based image processing techniques for a real life use case scenario.

The rest of the paper is organized as follows. Section 2 describes the existing methods of image processing, artificial neural networks based method is described in section 3. Experimental results are elaborated in section 4 and conclusions are given in section 5.

2. Conventional Methods

This section describes the existing methods involved in the processing of the image data.

The conventional methods of processing the image involve the formation of a matrix (usually a [3x3] or [9x9] matrix) called the function matrix. The elements of this matrix depend on the type of processing required. For example, for edge detection, the matrix is one which is determined by the type of algorithm used for edge detection namely, Sobel, Prewitt, Roberts, log, zero cross, etc. Hence, this matrix is not unique and depends on the output required. In addition, the processing method is meant to determine this matrix and as such varies depending on the application. One needs to write a different program for different application to determine the matrix. Once the function matrix is determined, the next step is to convolve the above matrix with the given input matrix to obtain the required output image matrix. The disadvantage of this method is that one can modify the images to certain predetermined format only. If multiple effects are to be given, then the image needs to be modified several times. This makes the methods computationally intensive and not suitable for limited resource devices.

3. Artificial Neural Network Method

The processor is trained to form the function matrix (weight matrix) by comparing the sample input matrix and sample output matrix. Hence, the essential requirement of this technique is a sample input image matrix and a desired sample output image matrix. Once these two images are available, the system is trained to generate the function matrix by adaptive processing and once the function matrix is generated, any number of input images can be converted to the required format. The program is very much generic and can be used for any type of processing like edge detection, smoothening, filtering, etc. without any modifications. Hence, the only requirement of this method is that of an initial input and output sample images. The function matrix can also be generated on a separate high capacity system and used by the mobile device for processing the final images. Depending upon the type of effects needed, several such matrices can be stored and appropriately loaded when the desired effect is selected from the camera application.

For the implementation of image processor, a 3-layer feed forward neural network model has been made use of. There are 9 input nodes in layer 1, 9 hidden nodes in the 2nd layer and 9 nodes in the output layer. The 9 inputs are the 9 pixel values of the image data obtained in [3x3] format. The neural network structure used in this technique is shown in figure 1.

The input values range from 0 to 1, corresponding to the colour of the pixel. Moreover, since the output of the network is the corresponding [3x3] matrix of the output image, the output will also range from 0 to 1.

The activation function for the neuron is a sigmoid as shown in figure 2.

The activation function is described as follows.

$$f(net) = 1/(1+\exp(-net)) \tag{1}$$

The network is trained using the error back propagation algorithm. Since it is a universal approximator, the network can perform any linear or non-linear point-by-point operation.

For training the network, a sample image and the desired output image needs to be given in a very simple format. These images are then loaded into the system. The neural network maps the sample image through it, and if it finds that its output and the expected output do not match then it modifies itself according to the learning rule. The network takes a [3x3] matrix of the input and the output and scans over the whole image, similar to the convolution process. In order to perform a good and effective training process, most of the possibilities must be presented to the network.

Hence, the training set may not be able to follow all the desired properties, unless given enough experience. Once the training is over, the modified network parameters can be used to work on any other image to produce the desired output. The input image is fed to the neurons in the form of a [3x3] matrix, and the output of the neurons is transformed into a [3x3] matrix and stored as the output image. This process is carried out repeatedly, until the whole image is scanned. If the output that we get does not have a particular property, then the same neural network can be further trained for that specific property, while the earlier

properties still remain embedded in the network. Hence, we can modify and train the network for more and more properties at a time.

3.1 Formulation

The mathematical formulation required to implement the image processing algorithm using neural networks is presented below.

Let the vector $x = [X_i]_{9x1}$ be the input to the neural network, where i stands for the i^{th} input or pixel of the [3x3] matrix.

$v = [V_{ij}]_{9X9}$ and $w = [W_{ij}]_{9x9}$ are the weight matrices, where ij stands for the weight between the i^{th} input and the j^{th} neuron. $[net_j]$ is the net input to the j^{th} neuron, and is given by

$$\lfloor net_j \rfloor_{9x1} = \lfloor v_{ij} \rfloor_{9x9} * [x_i]_{9x1} \quad (2)$$

Or,

$$\lfloor net_j \rfloor = \lfloor w_{ij} \rfloor * [y_i] \quad (3)$$

Then the output of the neuron is given by the vector $[O_j]$ as,

$$[O_j]_{9x1} = [f(net_j)]_{9x1} \quad (4)$$

3.2 Processing

If a network is already trained then the output of the network is obtained by feed forward method as

$$[y] = f([v] * [x]) \quad (5)$$

$$[z] = f([w] * [y]) \quad (6)$$

Where [y] is the output of the hidden layer and [z] is the output of the network.

3.3 Training

To train the network, weight matrices w and v are initialized with some random values. Next, the error back propagation algorithm is used to train the network as follows

$$[\Delta w] = ([d] - [z]) \bullet (1 - [z]).[z] \quad (7)$$

$$[\Delta v] = [y] \bullet (1 - [y]) \bullet ([w]^T * [\Delta w]) \quad (8)$$

Where,

[d] is the desired output matrix

[Δw] is the change in w

[Δv] is the change in v

The weights of the system are updated as

$$[w] = [w] + c([\Delta w] * [y]^T) \quad (9)$$

$$[v] = [v] + c([\Delta v] * [x]^T) \quad (10)$$

In (9) and (10), c is called the learning constant and its value is positive.

3.4 Implementation

The image to be processed and the desired output images are split into [3x3] matrices, then rearranged in a [9x1] format and stored in **x** and **d** respectively. They are used to train the network. After each update, the new output is calculated. If the error between the output image and the desired image is smaller than Δ, next training sample is taken, else the training is repeated.

Once the training is complete, the weight matrices can be used to perform the trained operation on any other input images. To do that, first the given image is split into [3x3] blocks and processed upon in the feed forward mode. The output $[z]_{(9x1)}$ is then rearranged into a [3x3] matrix and is used to replace the input matrix in its respective position. After all the blocks of the image are processed, we get the desired output image.

The camera sensors generally produce output in RGB as well as YCbCr formats. If the processing does not involve modifying the colour components and YCbCr format is used as input, then only the Y component can be used for operating with the function matrix and later re-composed with the Cb and Cr components. This reduces the overall computations involved in the processing.

4. Experimental Results

In this section the simulation results of the algorithm described in section 3 for detecting the edges of the input image are presented. Two simple sample images were created using a software drawing tool on a computer. One of them, the input, had shaded geometric objects (Figure 3(a)), and the other, the output image, had only the edges of the input image (Figure 3(b)). The two images were loaded into an array and [3x3] matrices were derived from the input and the output arrays. These were then processed using the formulation mentioned in section 3.1 and the resulting weight matrix stored for further usage as the edge detector.

The edge detector matrix is then retrieved into **v** and **w**. The new image to be processed is loaded onto an array and processed according to the implementation formulation. The output is obtained in the form of edges of the input image. Figures 4(a) and 4(b) show the input and the result.

5. Conclusion

This technique of image processing allows the user to add any desired effect to the image without any prior database. Hence the users themselves can program the image processor. Since the matrix once computed can be used on any other image, the time and complexity of it is very less. Thus this is highly suitable for mobile devices. This technique can be used for the edge detection, digital filters, colour detection, colour transformation, colour edge detection, etc. Thus this method can be an efficient substitute for hardware ISPs and advanced image sensors.

References

Castleman, K. R. (1998), "Digital Image Processing", *Prentice Hall.*

Etemad, K. & Chellappa, R. (1993), "A Neural Network Based Edge Detector", *IEEE International Conference on Neural Networks,* 1, 132-137.

Kamat, S. P. & Jayakumar, A. (2004), "A Novel Method for Colour Edge Detection in an Image Using Neural Networks Approach", *Proc. of Global Signal Processing Expo and Conference, GSPx 2004.*

Terry, P. J. & Vu, D. (1993), "Edge Detection Using Neural Networks", *1993 Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, 1, 391-395.

Zurada, J. M. (1994), "Introduction to Artificial Neural Systems", *Jaico publishing house.*

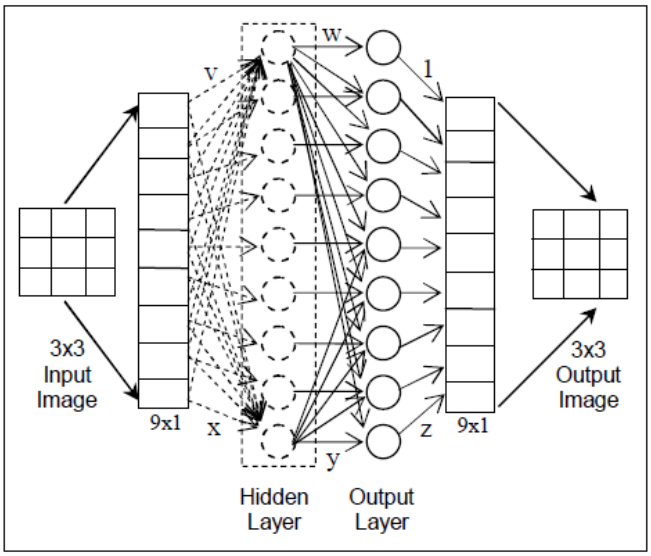

Figure 1. Neural Network Structure

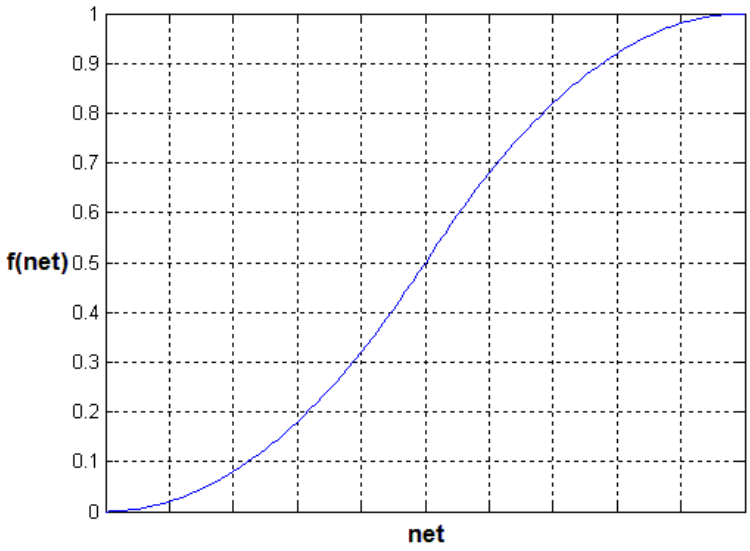

Figure 2. Sigmoid Activation Function for the Neuron

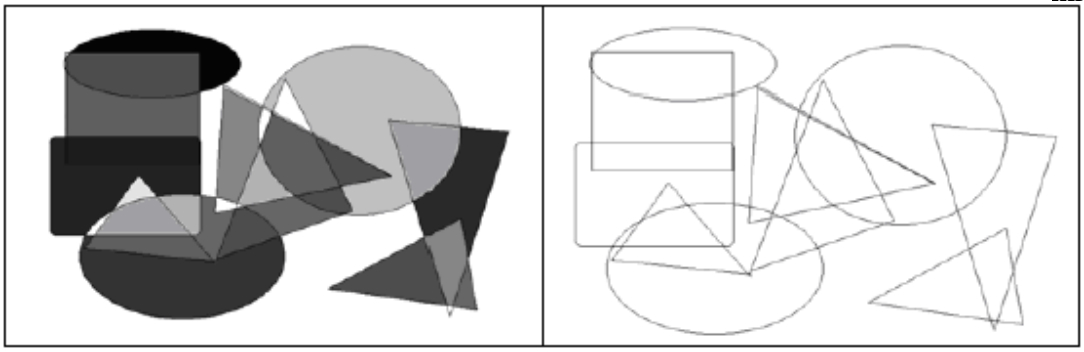

Figure 3. (a) Input Image for Training the Neural Network (b) Output Image for Training the Neural Network

Set of input and output images used for training the neural network. The output image consists of the edges of the image in the input.

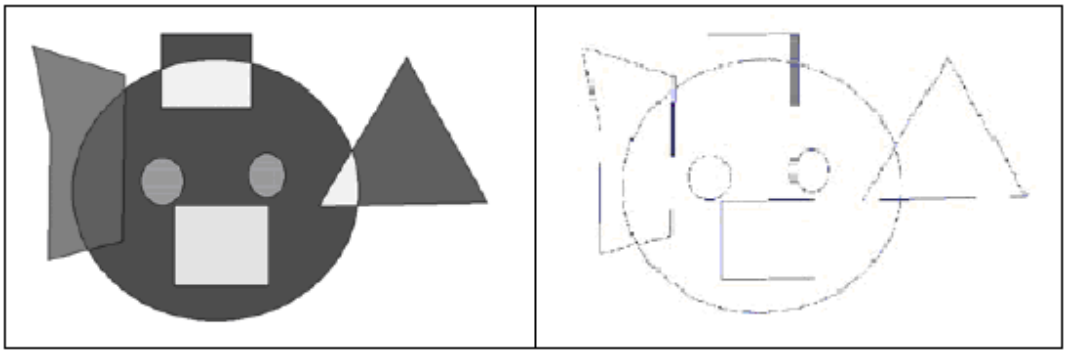

Figure 4. (a) Input Image for Processing (b) Output Image after Processing

Output image obtained from the neural network after feeding the input image. The network has been trained to produce edges of the input image.