Journal of Computing and Information Technology - CIT 15, 2007, 2, 143–149
doi:10.2498/cit.1000899

Real-time Face Recognition Using SIMD and VLIW Architecture

Srinivasa Kumar Devireddy [1], Iyyanki V. Murali Krishna[2] and Venkateswara Rao Tiruveedhula[1]

[1]Department of Computer Science and Engineering, Loyola Institute of Technology and Management, India
[2]Jawaharlal Nehru Technological University, India

There is a rapidly growing demand for using intelligent cameras for various applications in surveillance and identification. Most of these applications have real-time demands and require huge processing capacity. Face recognition is one of those applications highly in demand. In this paper we show that we can run face recognition in real-time by implementing the algorithm on an architecture which combines a massively parallel processor with a high performance Digital Signal Processor.

In this paper we focus on the INCA+ intelligent camera. It contains a CMOS sensor, a Single Instruction Multiple Data (SIMD) processor [1] and a Very Long Instruction Word (VLIW) processor. The SIMD processor enables high-performance pixel processing and detects the interesting (face) regions from the video. It sends the regions of interest to the VLIW processor, which performs the actual face recognition using a neural network. With this architecture we perform face recognition from a 5-persons database at more than 200 faces per second. The performance is better than most recent high-end professional systems [2].

Keywords: face detection, face recognition, smart camera, intelligent camera, SIMD processor, VLIW processor, real-time image processing

1. Introduction

Recently, face detection and recognition is becoming an important application for intelligent cameras. Face detection and recognition require lots of processing performance if real-time constraints are taken into account [3].

Face detection is the detection of faces in the scene from video data, it is usually done using color and/or feature segmentation. Face recognition is the actual recognition of the person based on the pixels that span the face region found in the detection process. Face recognition is usually performed by either neural network matching or by feature measurement and matching through a database. For robust recognition, the face needs to be at a proper angle and completely in front of the camera. Also, the size of the face has to be spanning the correct range of pixels. If the face portion of the image does not contain enough pixels, a reliable detection cannot be made.

What we want to show in this paper is that it is possible, for smart camera architectures, to achieve good, real-time face recognition results. A "smart camera" is hereby defined as a stand-alone programmable device with a size equal to or smaller than a typical video surveillance camera. In our situation it is programmed in such a way that video goes in and the names of recognized people come out.

One of the main platforms for proper face recognition is using an Intelligent Camera (INCA+) see Figure 1 [4]. This camera houses a CMOS

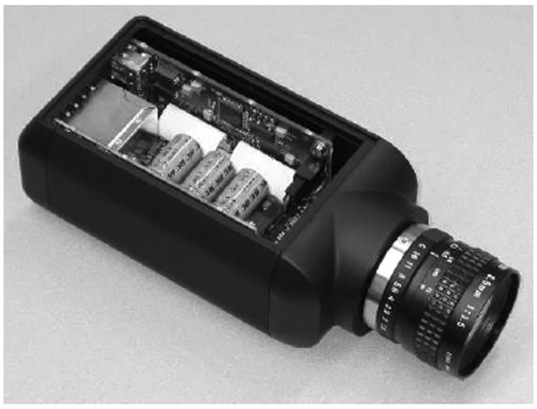

Figure 1. INCA camera.

sensor, a parallel processor for pixel-crunching and a DSP for the high-level programs. With this platform we can easily map face recognition into a smart camera (INCA+), which performs in real-time.

The contents of this paper are as follows: in Section 2 we explain the architecture of camera, in Sections 3 and 4 we describe the algorithm that we used for face detection and recognition respectively. The results are given in Section 5 and conclusions are drawn in Section 6.

2. Architecture

Face recognition consists of a face detection and face recognition. In the detection part faces are detected in the scene and the relevant parts of the image are forwarded to the face recognition process where the found face is matched to a database with a set of stored faces in order to recognize and put identification to it.

These two parts of the algorithms work on different data structures. While the detection part works on all pixels of the captured video and is pixel-oriented (low level image processing), the recognition part is working on face objects and is face-oriented (high level image processing). The detection part has to do similar operations for all pixels in the scene to determine if the pixel belongs to a face-blob or not. While we have a high amount of pixels in a live video stream, the operations are simple and similar for each pixel, allowing data-level parallelism.

The data rate in the recognition part is not that high, it only works on a few hundred faces per second, but it has a high amount of operations in an iterative way while a database is "scanned". Because of the higher complexity of the instructions and the combination with an operating system, this part of the algorithm is best mapped on a task-parallel architecture.

The different aspects of the two algorithmic tasks have made us choose for a dual processor the approach where the low-level image processing part of the face detection part is mapped on a massively parallel processor "Xetal" [1] working in SIMD (Single Instruction Multiple Data) mode. The high level image processing part of recognition is mapped on a high-performance fully programmable DSP core "TriMedia" [5]. This DSP has a VLIW (Very Long Instruction Word) architecture where instruction fetch and data fetch and processing are performed in a pipelined fashion.

For the defined task the two processors can be simply connected in series as in Figure 2. The Xetal does the face detection, the TriMedia does face recognition and the operating system also runs on TriMedia.

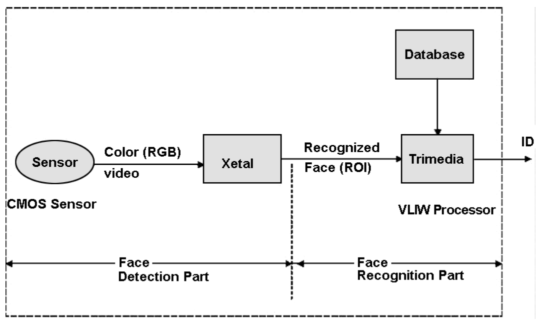

Figure 2. Architecture of the INCA camera.

First part of the architecture is CMOS sensor, it can send image to the SIMD processor. The sensor can take up to 30 frames per second with a resolution of $640 \cdot 480$ pixels. The output of sensor is RGB Bayern pattern where each pixel has one element. Moreover, if we want to have other elements we should interpolate with neighbor pixels (see Figure 3).

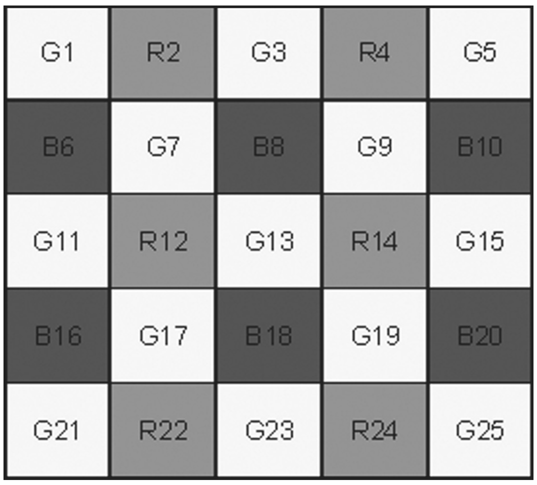

Figure 3. RGB Bayern pattern.

The Xetal processor exploits SIMD parallelism. It contains 320 pixel level processors and each pixel processor is responsible for 2 columns of image. It can handle up to 1000 instructions for each pixel at the same time. It has also 16

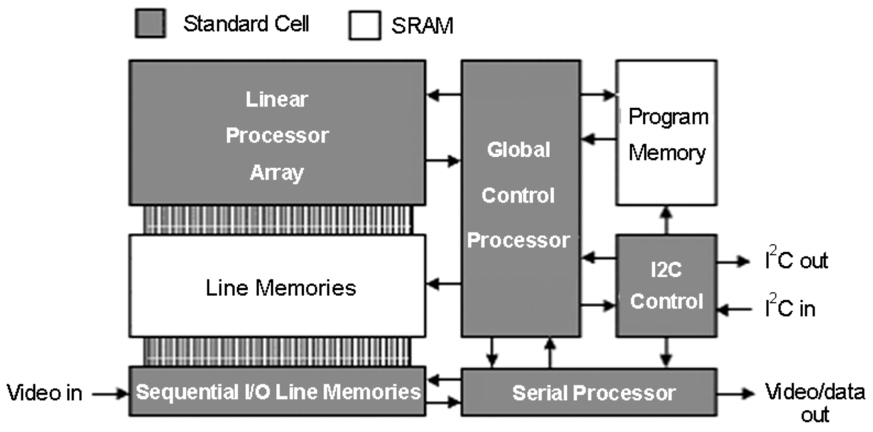

Figure 4. Xetal architecture.

line memories to save information [1]. Figure 4 shows the Architecture of Xetal processor.

This processor directly reads the pixels from the CMOS image sensor and performs the face detection part. Coordinates and sub regions of the image where prospective faces are found are forwarded to the TriMedia. The TriMedia exploits limited instruction level parallelism; it can handle 5 operations in parallel. This processor scales the sub regions and matches them to the faces in its database. In the fashion of a real "smart" camera, only IDs are reported to the user.

3. Face Detection

In face detection, we take an image from the sensor and detect and localize an unknown number (if any) of faces. Before enabling the detection and localization, the image should be segmented to the region which face should be there. This is done by colour specific selection. By removing too small regions and enforcing a certain aspect ratio of the selected region of interest (ROI) the detection becomes more reliable. We detect faces in the image by searching for the presence of skin-tone coloured pixels or groups of pixels. The representation of pixels as they are delivered by the colour interpolation routines from the CMOS sensor image is in RGB form. This is not very suitable for characterizing skin colour. The components in RGB space not only represent colour, but also luminance, which varies from situation to situation by going to a normalized colour domain this effect is minimized. The effect on the skin detection is that when a light condition changes the skin tone has also changed colour. With this feature the detection decreases in reliability. To solve this obstacle, there is the need to find a colour domain which separates the luminance with the colour. The YUV colour domain is suitable for the detection because it separates the luminance (Y) with the true colours (UV). The YUV colour domain is also 3-dimensional and it has the shape of a cube. Y is the luminance and it is the brightness of a colour image as it would be displayed in a black and white monitor. The U and V are just components of the colour signal so that the colour image can be reconstructed. The Y value can vary from 0 to 255 whereas the U and the V can have values from -128 to 128.

By using the YUV colour domain, not only the detection has become more reliable, but the skin tone indication has become easier, because the skin tone can now be indicated on a 2 dimensional space. The skin tone region is a square on the UV spectrum (Figure 5) and every not-skin colour out of the skin box is seen as non face

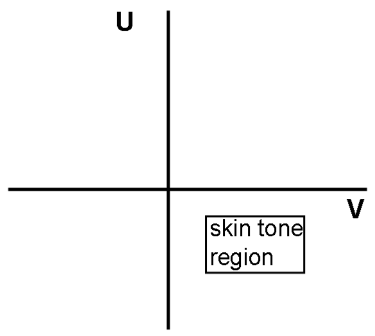

Figure 5. Skin region in UV spectrum.

(Figure 6). The face detection part only sends luminance and coordinate of face to recognition part.

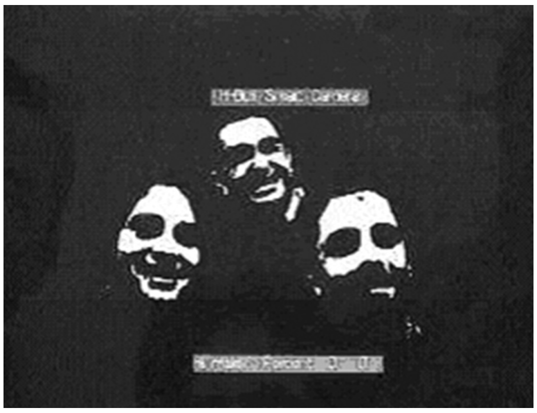

Figure 6. Skin tone result.

4. Face Recognition

The main goal of this section is to introduce the face recognition process. Through this process, the face detected in the previous section is identified with respect to the face database. For this purpose a Radial Basis Function (RBF) neural network is used [6]. The reason behind using an RBF neural network is its ability for clustering similar images before classifying them. RBF-based clustering received wide attention in the neural networks community. Apart from good clustering capabilities, RBF net-works have a fast learning speed, and a very compact topology.

4.1. Architecture of RBF Neural Networks

An RBF neural network structure is demonstrated in Figure 7.

Its architecture is similar to that of a traditional three-layer feed forward neural network. The input layer of this network is a set of n units, which accepts the elements of an n-dimensional input feature vector (here, the RBF neural network input is the face which is gained from the face detection part. Since it is normalized with a $64 \cdot 72$ pixel face, it follows that $n = 4608$). The input units are completely connected to the hidden layer with m hidden nodes. Connections between the input and the hidden layers have fixed unit weights and, consequently, it is

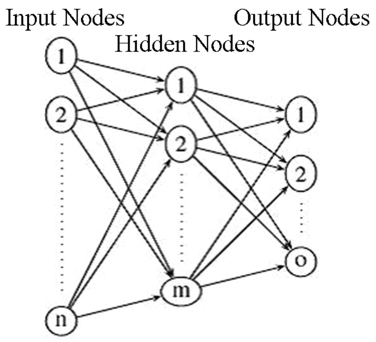

Figure 7. Architecture of RBF neural networks.

not necessary to train them. The purpose of the hidden layer is to cluster the data and decrease its dimensionality. The RBF hidden nodes are also completely connected to the output layer. The number of outputs depends on the number of people to be recognized (for example, for 100 persons $o = 100$). The output layer provides the response to the activation pattern applied to the input layer. The change from the input space to the RBF unit space is nonlinear, whereas the change from the RBF hidden unit space to the output space is linear.

The RBF neural network is a class of neural networks, where the activation function (basis function) of the hidden units is known by the distance between the input vector and a prototype vector. The activation function of the RBF hidden node is stated as follows [7]:

$$F_i(x) = G_i(||x - c_i||^2/\sigma_i),\ i = 1, 2, \ldots, m \quad (1)$$

where x is an n-dimensional input feature vector (normalized face $64 \cdot 72$), c_i is an n-dimensional vector called the center of the RBF hidden node, σ_i is also an n-dimensional vector called the width (also called radius) of RBF hidden node and m is the number of the hidden nodes.

Normally, the activation function G of the hidden nodes is selected as a Gaussian function with mean vector c_i and variance vector σ_i as follows:

$$F_i(x) = e^{\left(\frac{-||x-c_i||^2}{\sigma_i^2}\right)} \qquad i = 1, 2, \ldots, m \quad (2)$$

Because the output units are linear, the response of the k'th output unit (among the o number of outputs) for input x is given as:

$$Out_k(x) = B_k + \sum_{i=1}^{c} F_i(x) \cdot W(i,k), \quad k = 1, 2, \ldots, o \tag{3}$$

where $W(i,k)$ is the connection weight of the i'th RBF hidden node to the k'th output node and B_k is the bias of the k'th output.

4.2. Using RBF Neural Network

The first step in face recognition is normalizing the region of interest (Figure 8) to the size of the faces stored in the identification database ($64 \cdot 72$ pixels) and after that feeding them to the neural network input. Subsequently, we calculate the output for each person, and we consider the maximum value between the outputs and report that as the recognized person. Figure 9 shows the main kernel for using the RBF neural network.

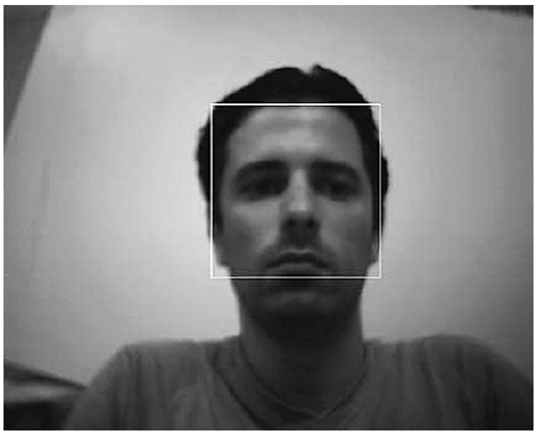

Figure 8. Region of interest result.

```
// compute output of hidden node
L1:
for 0 < i < Number_Hidden_Node
{
sum =0
for 0<j< Number_Input_Node(64*72=4608)
{
temp = data[j]-center_value[i][j]
temp = temp * temp
temp = temp / sigma_value[i][j]
sum = sum+temp
}
out_hiddennode[i] = exp(-sum)
```

```
// compute output
L2:
for 0 < i < Number_Output(5 person)
{
sum =0
for 0 < j < Number_Hidden_Node(20)
{
sum = sum +
(out_hiddennode[i][j]*weight[i][j])
}
sum = sum + bias_value[i]
output[i] = temp
}
```

Figure 9. Kernel for RBF neural network.

5. Measurements and Performance

In this section we evaluate the performance of our algorithm. Since the face recognition, and not the detection part, turned out to be the major bottleneck we concentrate on that part first. At the end of this section we evaluate the overall performance and recognition rate.

5.1. Face Recognition

The algorithms described for using the RBF neural network have certain demands on the processing power, bandwidth and flexibility of the architectural template. To measure the performance, we first need to extract the kernel loops and loop-nests, which are done in the RBF neural network. If we take the example of face recognition, the input is a normalized face ($64 \cdot 72$ pixels, hence 4608 inputs), there are m hidden nodes (resulting in $4608 \cdot m$ weights between the input and hidden layers), and o output nodes, depending on the number of people to be recognized (hence $m \cdot o$ weights between the hidden and output layers).

5.2. Adapting for Real-time Performance

We observed that most of the running time of the algorithm was spent on calculating the output of hidden nodes and calculating the output (see Figure 9). For example, if the number of hidden nodes is equal to 20 and we want to recognize faces of five persons, the number of executed instructions on a TriMedia (166 MHz.) is about $12 \cdot 10^6$, and the number of cycles (taking memory delays into account) is about $15 \cdot 10^6$, which corresponds to 90 ms. This is far from real-time, therefore we employed several optimizations like:

- Replace all division operations in the program.
- Use single precision floating point instead of double precision.
- Use local variables instead of global variables.
- Perform loop-unrolling.

Then the number of executed instructions is reduced to $4 \cdot 10^5$ and the number of cycles is reduced to $7 \cdot 10^5$(thus, resulting in 4.2 ms execution time).

5.3. Complexity

The execution time in the RBF loop nests is related to *m*, *n*, and *o* (see Figure 9) where *n* is the number of input nodes, *m* is the number of hidden nodes and *o* is the number of output nodes (see Figure 7). The time complexities for the first (L1) and second (L2) loop nests are:

$$T_{L1} = O(m.n),\ T_{L2} = O(m.o) \tag{4}$$

Therefore, the total time complexity is given by:

$$T(m.n.o) = O(m.n + m.o) = O(m.(n + o)) \tag{5}$$

It is easily seen that the memory size required for allocating all variables has the same complexity:

$$M(m.n.o) = O(m.(n + o)) \tag{6}$$

Because m is independent of *n* and *o* (*m* is more or less constant, and equal to the number of characteristics in a face), execution time and memory size are linear in *n* and *o* [8].

5.4. Overall Practical Performance

Our algorithms have been mapped to a handheld camera device as shown in Figure 1. After programming using a host computer and a fire wire or Ethernet link, the camera starts running the face recognition application. Because faces are recognized at video rate and with more possible faces per frame (a maximum recognition rate of 230 faces per second is possible), an operation system running in the camera has to control the reporting process. The operating system obtains the IDs of the recognized person and monitors the reliability of recognition as reported by the face recognition part. If this is high enough, a person is positively identified and will also not be reported in subsequent frames until he/she leaves the scene or another person shows up.

The overall performance we reach ranges from a recognition rate of 97% with a false detection rate of 1 out of 20 to a recognition rate of 90% with a false detection rate of 1 out of 50 depending of the settings. These numbers are for a real-time (up to 230 faces per second) stand-alone system with 5 stored "identifiable" faces.

6. Conclusions and Future Work

Face recognition is becoming an important application for smart cameras. However, till now, the processing required for real-time detection prohibits integration of the whole application into a small sized, consumer type of camera. This paper showed that by:

1. Proper selection of algorithms, both for face detection and recognition,
2. Adequate choice of processing architecture, supporting both SIMD and ILP types of parallelism,
3. Tuning the mapping of algorithms to the selected architecture,

this integration can be achieved. We implemented the algorithms on a small smart camera. As a result we can recognize one face per 4.2 ms, when we are searching for 5 persons, with 90% recognition rate and only 1% failure rate.

Future research will focus on further tuning the mapping of the algorithms, e.g. by replacing floating point operations with fixed point, trying other (cheaper) activation functions (see eq. 2), and further parallelization of the RBF neural network. This should allow for further speedups needed when searching in much larger databases that can contain large numbers of identifiable faces. Furthermore, the recognition will be enhanced by using multiple cameras with different viewpoints also for detection part add automatic white -balance control, to prevent the sensibility of the light.

References

[1] A. ABBO, R. KLEIHORST, Smart cameras: Architectural challenges. *Proceedings of ACIVS 2002 (Advanced Concepts for Intelligent Vision Systems)*, (2002), Gent, Belgium.

[2] ELECTRONIC PRIVACY INFORMATION CENTER, (2006). `www.epic.org/privacy/facerecognition`

[3] B. L. E. HJELMAS, Face detection: a survey. *Computer Vision and Image Understanding*, Vol. 83, pp. 236–274, (2001).

[4] CENTRE FOR INDUSTRIAL TECHNOLOGY, (2006). `www.cft.philips.com/industrialvision`

[5] TRIMEDIA TECHNOLOGIES, (2006). `www.trimedia.com`

[6] J. HADDADNIA, K. FAEZ, P. MOALLEM, Human face recognition with moment invariants based on shape information. *Proceedings of the International Conference on Information Systems, Analysis and Synthesis*, vol. 20, (Orlando, Florida USA), International Institute of Informatics and Systemics (ISAS'2001), (2001).

[7] Y.-H. HU, J.-N. HWANG, EDS. , Handbook of neural network signal processing. *CRC Press*, (2002).

[8] H. H. FATEMI, R. P. KLEIHORST, P. JONKER, Real time face recognition on a smart camera. *Proceedings of ACIVS 2003 (Advanced Concepts for Intelligent Vision Systems)*, (2003), Gent, Belgium.

[9] H. BROERS, W. CAARLS, P. J. R. KLEIHORST, Architecture study for smart cameras. *Proceedings of EOS conference on Industrial imaging and machine vision*, European optical society, (Jun 2005), Munich, Germany, pp. 39–49.

[10] V. S. CHOWDARY, A. A. ABBO, Dynamic power management for the Xetal SIMD Processor. *Proceedings of ProRISC 2004*, (2004). `www.stw.nl/Programmas/Prorisc`

[11] K. N. TRUONG, G. D. ABOWD, INCA: A software infrastructure to facilitate the construction and evolution of Ubiquitous capture & Access applications. *ubicomp-f2004/papers/14-truoyk-pervasive2004*, (2004). `www.cs.cmu.edu`

[12] W. CAARLS, P. P. JONKER, H. CORPORAAL, Data and Task Parallel Image Processing on a Mixed SIMD-ILP Platform using Skeletons and Asynchronous RPC. *Proceedings of the 5th progess symposium on Embedded Systems*, (2004). `www.ph.tn.tudelft.nl/~wcaarls/smartcam/files/pw2004.pdf`

[13] M. HEIJLIGERS, PHILIPS NL, Xetal-II: A low power multi-processing SIMD architecture. *MEDEA+ Design Automation Conference, Yachthotel Chiemsee*, (2006), Germany. `www.infineon.com/medea-dac_2006/`

Received: July, 2006
Accepted: October, 2006

Contact addresses:

Srinivasa Kumar Devireddy
Department of Computer Science and Engineering
Loyola Institute of Technology and Management
Sattenapalli-522412
Guntur (Dt.), Andhra Pradesh, India
e-mail: `srinivaskumar_d@yahoo.com`

Iyyanki V. Murali Krishna
Center for Spatial Information Technology
Jawaharlal Nehru Technological University
Hyderabad, Andhra Pradesh, India
e-mail: `ivm@ieee.org`

Venkateswara Rao Tiruveedhula
Department of Computer Science and Engineering
Loyola Institute of Technology and Management
Sattenapalli-522412
Guntur (Dt.), Andhra Pradesh, India
e-mail: `tv_venkat@yahoo.com`

SRINIVASA KUMAR DEVIREDDY receivied his B.E. in Computer Science from Karnataka University, Dharwad, India in 1992 and M.S. in Software Systems from Birla Institute of Technology and Science, Pilani in 1995. He is a senior member of IEEE. He is working as Professor in the Department of Computer Science and Engineering Loyola Institute of Technology and Management, Sattenapalli – 522412, Guntur (Dt.), India. Presently, he is a Ph.D. candidate in Jawaharlal Nehru Technological University, Hyderabad.

DR. IYYANKI V. MURALI KRISHNA obtained his M. Tech from Indian Institue of Technology (IIT), Madras and Ph.D. from Indian Institute of Science (I.I.Sc.), Bangalore. He is working as a Professor in the Center for Spatial Information Technology, Jawaharlal Nehru Technological University, Hyderabad. At present he is the Director of Research and Development wing of Jawaharlal Nehru Technological University, Hyderabad. He has over 30 years of experience in teaching and research in the field of Information Technology. Dr. Murali Krishna successfully guided many research scholars for the award of Ph.D. He received the award of the Best Teacher for the year 2004-2005 from the Government of Andhra Pradesh, India.

DR. VENKATESWARA RAO TIRUVEEDHULA obtained his B.E. E.C.E. in 1977 from Andhra University, Visakhapatnam, M.E. in Computer Science in 1979 from P.S.G. College of Technology, Coimbatore, India and Ph.D. in Computer Engineering in 1992 from Wayne State University, Detroit, USA. He worked as a senior faculty member in both USA and India. At present he is working as Professor and Principal of L.I.T.A.M., Guntur. He guided some research scholars for the award of Ph.D. He has many international publications to his credit.