DESIGNING AND OPTIMIZATION OF CODEC H-263 FOR MOBILE APPLICATIONS

Mr. Sourabh Rungta Dr.Neeta Tripathi* Mr.Arvind K Verma Dr.Anupam Shukla****

*RCET DURG
**ABV-IIITM GWALIOR

Abstract
Mobile phones size transformed today's communication industry with their continuous reduction in size and weight, Personal Digital Assistants (PDAs) such as pocket PCs and palms have grown in popularity over the past years. Due to the limited processing capability, memory constraints, and the power budget of mobile clients, coders, decoders and renderer are very difficult to implement on wireless handheld PDAs. In this paper we proposed an algorithm to develop a software which implement H.263 Decoder (& renderer) and Encoders so that the software implementation must be highly optimized to achieve "reasonable" video quality.

Keywords *- Optimize, Personal Digital Assistants (PDA), Applications, H.263, Multimedia, Encoder, Decoder, Renderer, MobiFDR.*

1. Introduction

The H.263 design original goal of this endeavor was to design a video coding standard suitable for applications with bit rates around 20 Kbits/s (the so-called very-low-bit-rate applications), it became apparent that H.263 could provide a significant improvement over H.261 at any bit rate. In essence, H.263 combines the features of H.261 with several new methods, including the half-pixel motion compensation first found in MPEG-1 and other techniques. Compared to an earlier standard H.261, H.263 can provide 50% or more savings in the bit rate needed to represent video at a given level of perceptual quality at very low bit rates. H.263 provides superior coding efficiency to that of H.261 at all bit rates (although not nearly as dramatic an improvement when operating above 64 Kbits/s). H.263[1,2] represents today's state of the art for standardized video coding. Essentially any bit rate, picture resolution, and frame rate for progressive-scanned video content can be efficiently coded with H.263. H.263 is structured around a "baseline" mode of operation, which defines the fundamental features supported by all decoders, plus a number of optional enhanced modes of operation for use in customized or higher performance applications. Because of its high performance, H.263 was chosen as the basis of the MPEG-4 video design, and its baseline mode is supported in MPEG-4 without alteration.

Many of its optional features are now also found in some form in MPEG-4. In addition to the baseline mode, H.263 includes a number of optional enhancement features to serve a variety of applications. The original version of H.263 had about four such optional modes. The latest version of H.263, known informally as H.263+ or H.263 Version 2, extends the number of negotiable options to 16. These enhancements provide either improved quality or additional capabilities to broaden the range of applications. Among the new negotiable coding options specified by H.263 Version 2, five of them are intended to improve the coding efficiency. These are the advanced intra coding mode, alternate inter VLC mode, modified quantization mode, deblocking filter mode, and improved PB-frame mode. Three optional modes are especially designed to address the needs of mobile video and other unreliable transport environments. They are the slice structured mode, reference picture selection mode, and independent segment decoding mode. The temporal, SNR, and spatial scalability modes support layered bitstream scalability, similar to those provided by MPEG-2. There are two other enhancement modes in H.263 Version 2: the reference picture resampling mode and reduced-resolution update mode. The former allows a previously coded picture to be resampled, or warped, before it is used as a reference picture. Another feature of H.263 Version 2 is the use of supplemental information, which may be included in the bitstream to signal enhanced display capabilities or to provide tagging information for external use. One use of the supplemental enhancement information is to specify the chroma key for representing transparent and semitransparent pixels. Each optional mode is useful in some applications, but few manufacturers would want to implement all of the options. Therefore, H.263 Version 2 contains an informative specification of three levels of preferred mode combinations to be supported[3]. Each level contains a number of options to be supported by an equipment manufacturer. Such information is not a normative part of the standard. It is intended only to provide manufacturers some guidelines as to which modes are more likely to be widely adopted across a full spectrum of terminals and networks. Three levels of preferred modes are described in H.263 Version 2, and each level supports the optional modes specified in lower levels.

Due to the limited processing capability, memory constraints, and the power budget of mobile clients, Full Duplexing multimedia coders, decoders and renderer are

Manuscript received March 5, 2009
Manuscript revised March 20, 2009

very difficult to implement on wireless handheld PDAs. The project is especially aimed at development of a Full Duplexed Mobile Video Decoder and Renderer system so that videos in H.263 / H.264+[10,12,14] formats, which are popularly used in the various mobile phones, PDAs and wireless devices, can be encoded (recorded) using the on-board hardware and played back at the same time. Applications of such a system would be widespread in the current scenario where mobile value added services including content streaming, mobile TV streaming, video conferencing, e-learning, telemedicine etc are being commercially deployed. This system could be useful component in all these application services and could thus be made available to technical users at reasonable prices.

2. H.263 Encoder and Decoder

A number of video coding standard exist, each of which is designed for a particular type of application: for e.g. JPEG for still images, MPEG2/4 for digital television and H.261 for ISDN video conferencing. H.263 standard specifies the requirements for a video encoder and decoder[6,7,9]. It does not describe the encoder or decoder themselves: instead, it specifies the format and content of the encoded (compressed) stream. A typical encoder (as in figure 1) and decoder (as in figure 2) are described here:

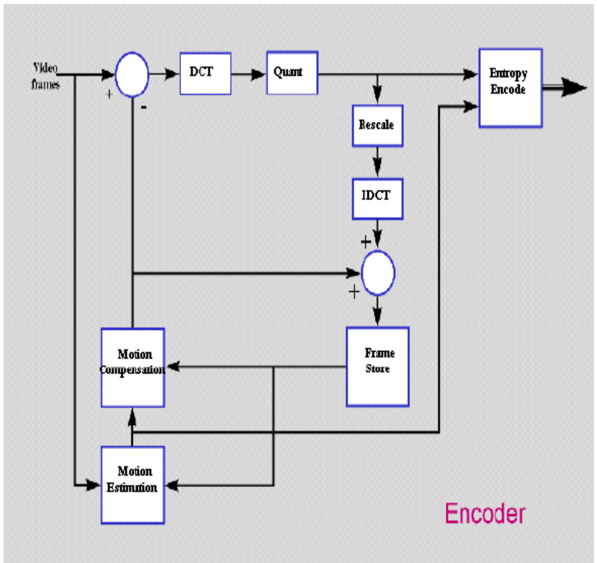

Figure 1. H.263 Encoder

H.263 Encoder:

Motion estimation and compensation: The first step in reducing the bandwidth is to subtract the previous frame from the transmitted frame from the current frame so that only the difference or residue needs to be encoded and transmitted. This means that areas of the frame that do not change (for example the background) are not encoded. Further reduction is achieved by attempting to estimate where areas of the previous frame have moved to in the current frame (motion estimation) and compensating for this movement (motion compensation). The motion estimation module compares each 16×16 pixel block (macroblock) in the current frame with its surrounding area in the previous frame and attempts to find a match. The matching area is moved into the current macroblock position by the motion compensator module. The motion compensated macroblock is subtracted from the current macroblock. If the motion estimation and compensation process is efficient, the remaining "residual" macroblock should contain only a small amount of information.

Virtually all motion estimation algorithms in video communication have been developed for coding purposes with different objectives. In the first stage of this method, called motion estimation (ME), the motion of objects between a reference frame and the current frame is estimated. This motion information is then used in the second stage, called motion compensation (MC), to move the objects of the reference frame to provide a prediction for the current frame. They aim at minimizing the prediction error after motion-compensation so that only a comparatively small residue must be encoded. By removing the high temporal redundancy present in video sequences. High compression ratios can be achieved. Classical approaches to motion estimation belong to the group of nonparametric techniques because their only interest is in computing the motion field. All motion estimation methods rely on the principle of intensity conservation; that is, they more or less implicitly assume that the luminance of pixels does not change along their motion trajectories. *Discrete Cosine Transform (DCT):* The DCT transform a block of pixel values (or residual values) into a set of "spatial frequency" coefficients. The DCT operates on a 2-dimensional block of pixel (rather than on a 1-dimensional signal) and is particularly good at "compacting" the energy in the block of values into a small number of coefficients. This means that only a few DCT coefficients are required to recreate a recognizable copy of the original block of pixels.

Quantization: For a typical block of pixels, most of the coefficients produced by the DCT are close to zero. The Quantizer module reduces the precision of each coefficient so that the non-zero coefficients are set to zero and only a few significant non-zero are left. This is done in practice by dividing each coefficient by an integer scale factor and truncating the result. It is important to realize that the Quantizer "throws away" information.

Entropy encoding: An entropy encoder (such as Huffman encoder) replaces frequently-occurring values with short binary codes and replaces infrequently-occurring values with longer binary codes. The entropy encoding in H.263 is based on this technique and is used to compress the quantized DCT coefficients. The result is a sequence of variable length binary codes. These codes are combined with synchronization and control information to form the encoded H.263 bitstream.

Frame store: The current frame must be stored so that it can be used as a reference when the next frame is encoded. Instead of simply copying the current frame in to a store, the quantized coefficients are re-scaled; inverse transformed using an Inverse DCT and added to the motion-compensated reference block to create a reconstructed frame that is placed in a store. This ensures that the contents of the frame store in the decoder. When the next frame is encoded, the motion estimator uses the contents of this frame store to determine the best matching area for motion compensation.

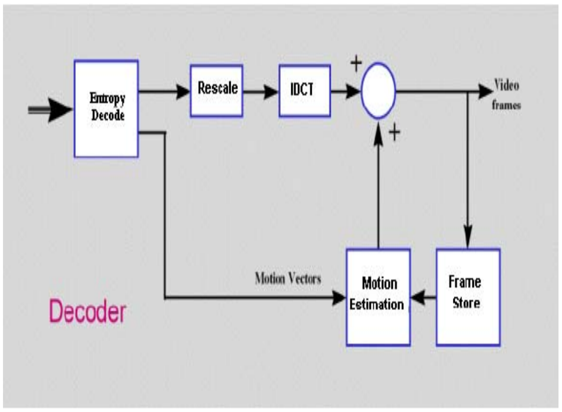

Figure2. H.263 Decoder

H.263 Decoder:

Entropy Decode: The variable-length codes that make up the H.263 bit stream are decoded in order to extract the coefficient values and motion vector information.

Rescale: This is the “reverse” of quantization: the coefficients are multiplied by the same scaling factor that would be used in the Quantizer. However, because the Quantizer discarded the fractional remainder, the rescaled coefficients are not identical to the original coefficients.

Inverse Discrete Cosine Transform: The IDCT reverses the DCT operation to create a block of samples: these correspond to the difference values that were produced by the motion compensator in the encoder.

Motion compensation: The difference values are added to a reconstructed area from the previous frame. The motion vector information is used to pick the correct area. The result is a reconstruction of the original frame: note that this will not be identical to the original because of the “lossy” quantization stage, i.e. the image quality will be poorer than the original. The reconstructed frame is placed in a frame store and it is used to motion-compensate the next received frame.

3. Implementations Issues

3.1 Real-time video communications

Many issues need to be addressed in order to develop a video encoder and decoder that can operate effectively in real time. These include:

Bit rate control: Practical communications channels have a limit to the number of bits that they can transmit per second. The basic H.263 encoder generates a variable number of bits for each encoded frame[11,13]. If the motion estimation/compensation process works well then there will be few remaining non-zero coefficients to encode. However, if the motion estimation does not work well (for example when the video scene contains complex motion), there will be many non-zero coefficients to encode and so the number of bits will increase.

In order to "map" this varying bit rate to (say) a CBR channel, the encoder must carry out **rate control**. The encoder measures the output bit rate of the encoder. If it is too high, it increases the compression by increasing the quantizer scale factor: this leads to more compression (and a lower bit rate) but also gives poorer image quality at the decoder. If the bit rate drops, the encoder reduces the compression by decreasing the quantizer scale factor, leading to a higher bit rate and a better image quality at the decoder.

Synchronization: The encoder and decoder must stay in synchronization, particularly if the video signal has accompanying audio. The H.263 bitstream contains a number of "headers" or markers: these are special codes that indicate to a decoder the position of the current data within a frame and the "time code" of the current frame. If the decoder loses synchronization then it can "scan" forward for the next marker in order to resynchronize and resume decoding. It should be noted that even a brief loss of synchronization can cause severe disruption in the quality of the decoded image and so special care must be taken when designing a video coding system to operate in a "noisy" transmission environment.

Audio and multiplexing:

The H.263 standard describes only video coding. In many practical applications, audio data must also be compressed, transmitted and synchronized with the video signal. Synchronization, multiplexing and protocol issues are covered by "umbrella" standards such as H.320 (ISDN-based videoconferencing), H.324 (POTS-based video telephony) and H.323 (LAN or IP-based videoconferencing. Audio coding is supported by a range of standards including G.723.1. Other, related standards cover functions such as multiplexing (e.g. H.223) and signaling (e.g. H.245).

3.2 Software implementations:

Functions such as motion estimation, variable length encoding/decoding and the DCT require a significant amount of processing power to implement. However, with recent developments in processor technology, it is possible to encode and decode H.263 video in real time on general-purpose processors such as the Pentium family.

A software implementation must be highly optimized to achieve "reasonable" video quality This involves a number of steps such as choosing fast algorithms for processor-intensive functions, minimizing the number of move or copy operations and unrolling loops.

3.3 Hardware implementations

For high quality video, or in applications where a powerful processor is not available, a hardware implementation is the solution. A typical hardware CODEC might use dedicated logic for the computationally intensive parts of the system (such as the motion estimator/compensator, DCT, quantizer and entropy encoder) with a control module that schedules events and keeps track of the encoding and decoding parameters[4,5,8]. A programmable controller is advantageous because many of the encoding parameters (such as the rate control algorithm) can be modified or adapted to suit different environments. Recently, logic core (Intellectual Property core) implementations of H.263 have become available.

4. Problems of the Mobile platforms

When we switch from one platform to another such as PC platforms to devices special care has to be taken during the development process because the mobile platform have limitations in performance, power and memory resources[8,9]. Limitations that should be faced by all mobile application programmers.

A. Limited Speed Control Processing Units

Many of the current high-end mobile phones operate at relatively slow clock speed. This means that many complex calculations required by some applications, such as graphics, are limited in the speed with which they can be performed. CPUs on mobile phones generally have small caches due to which the complex algorithms requiring more data moves accelerates demands on the CPU. Some specialists expect that high speed mobile processors are likely to appear in the next couple of years and for these processors to exist we have to overcome the problems of battery life and heat generation.

B. Limited Bandwidth

In complex applications data transferred between the CPU and memory, or I/O peripherals, could easily create problem due to limited bandwidth. Too narrow or too slow bus limits have significant impact on the overall efficiency and throughput.

Various mobile venders are trying to optimize bus speed for better performance but developers have to be aware of this limitation while programming their application by meeting the critical data throughput boundaries and not overloading the data path.

C. Limited Memory

The memory space available is always been an issue in mobile environment. This is no longer a critical issue with the tremendous fall in prices of flash memory and the introduction of hard-drive-based mobile phones. However, memory has to be used carefully and used wisely in any mobile application because the sizes available are insignificant in comparison to the nature of applications using memory this might cause memory fragmentation, memory leaks and frequent system crashing.

5. The Proposed Improvement in H.263

Now day’s Mobile phones have evolved rapidly and are now required to perform increasing amounts of processing and data usage more than their predecessors. Applications ranging from teleconferencing to telecommunication all operation on this compact but limited platform creating considerable demands of its memory and power resources. To make sure these resources are not wasted programmers should guarantee efficient performance of their programs by providing the optimized balance of processing consumption and code size. In other words they must develop designs that take proper account of the platform constraints and operating parameters.

With the potential of even greater speeds in the near future, low-cost multimedia solutions would be possible since audio and video decompression would be done on the native processor without any additional hardware. Optimization can occur in different stages of the development cycle and in different areas. For example, the target architecture can be upgraded, algorithms could be modified, compilers’ optimization power can be turned on, and coding practices might be subject to amendment. Upgrading the mobile architecture is not practical for the general developer as it would generally require significant modifications of internal chip architectures or device structures. Algorithms on the other hand are dependent on the context and field they are being used in and are therefore often application dependent. Compilers usage relies on programmers’ taste and preference. However, coding style is generic to every project and therefore is a worthy topic of study. In this paper we will address this generic issue for the wider benefit and improvement of most mobile application developers.

Keeping in mind the observations outlined above, we would proposed to develop a software video library (MobiFDR) that would

1. Provide a common architecture under which multiple audio and video codecs and renderers could be accessed.
2. Be the lowest, functionally complete layer in the software video codec hierarchy.
3. Be fast, extensible, and thread-safe, providing reentrant code with minimal overhead
4. Provide an intuitive, simple, flexible, and extensible application programming interface (API) that supports a client-server model of multimedia computing.
5. Provide an API that would accommodate multiple upper layers, allowing foe easy and seamless integration into multimedia products.

Whilst there is a number of software platforms for mobile development we focus the most popular, the Symbian OS, which utilizes a specialized version of C++. Symbian is divided into different versions to cover a number of user interfaces and in this paper we concentrate on the most common which is Series 60, although this is for illustration and the techniques could be applied almost generically.

The information retrieved demonstrates the effectiveness of the prescribed programming technique which would helpful in creating the optimized software. The optimizations we will consider have been split into four categories:

A) Object-oriented optimization
B) Memory optimization
C) Coding Style Optimization, and
D) Optimizations in Compiler.

6. Software Performance and usability

A better performing software product is one that saves time for the user. Time is a precious resource for many computer users and much time is wasted on software that is slow, difficult to use, incompatible or error prone. All these problems are usability issues, software performance should be seen in the broader perspective of usability. The following list points out some typical sources of frustration and waste of time for software users as well as important usability problems that software developers should be aware of to overcome the problems caused by the hardware implementation of H.263 and in optimizing the MobiFDR.

Big runtime frameworks- The .NET framework and the Java virtual machine are frameworks that typically take much more resources than the programs they are running. Such frameworks are frequent sources of resource problems and compatibility problems and they waste a lot of time both during installation of the framework itself, during installation of the program that runs under the framework, during start of the program, and while the program is running. The main reason why such runtime frameworks are used at all is for the sake of cross-platform portability. Unfortunately, the cross-platform compatibility is not always as good as expected. The portability could be achieved more efficiently by better standardization of programming languages, operating systems, and API's.

Installation problems- The procedures for installation and un-installation of programs should be standardized and done by the operating system rather than by individual installation tools due to which we proposed MobiFDR.

Compatibility problems- All software should be tested on different platforms, different screen resolutions, different system color settings and different user access rights. Software should use standard API calls rather than self-styled hacks and direct hardware access. Available protocols and standardized file formats should be used. Web systems should be tested in different browsers, different platforms, different screen resolutions, etc.

Copy protection- Some copy protection schemes are based on hacks that violate or circumvent operating system standards. Such schemes are frequent sources of compatibility problems and system breakdown. Many copy protection schemes are based on hardware identification. Such schemes cause problems when the hardware is updated. Most copy protection schemes are annoying to the user and prevent legitimate backup copying without effectively preventing illegitimate copying. The benefits of a copy protection scheme should be weighed against the costs in terms of usability problems and necessary support.

Hardware updating- The change of a hard disk or other hardware often requires that all software be reinstalled and user settings are lost. It is not unusual for the reinstallation work to take a whole workday or more. Current operating systems need better support for hard disk copying.

Security- The vulnerability of software with network access to virus attacks and other abuse is extremely costly to many users.

Background services- Many services that run in the background are unnecessary for the user and a waste of resources. Consider running the services only when activated by the user. Take user feedback seriously. User complaints should be regarded as a valuable source of information about bugs, compatibility problems, and usability problems and desired new features. User feedback should be handled in a systematic manner to make sure the information is utilized appropriately. Users should get a reply about investigation of the problems and planned solutions. Patches should be easily available from a website.

Conclusion

In this paper we proposed to develop an software library which try to remove hardware based implementation in which the same component cannot be replicated for two tasks. The hardware components would put these tasks in a queue and perform these tasks one by one (rather than together). In case of a two-way video streaming, the task of encoding and decoding are to be performed simultaneously this is thus not possible. The proposed work, thus, is to develop a software implementation (MobiFDR) for H.263 Decoder (& renderer) and Encoders so that this can be executed simultaneously.

References

[1] Gary J. Sullivan, Pankaj Topiwala, and Ajay Luthra "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions" Applications of Digital Image Processing XXVII Special Session on Advances in the New Emerging Standard: H.264/AVC, August, 2004

[2] V.Nguyen and Y.Tan, "Efficient H.263 To H.264/AVC Video Transcoding Using Enhanced rate control", *IEEE International Symposium* on Volume, Issue, 23-26 May 2005 Page(s): 904 - 907 Vol. 2.

[3] T.Ye, Y.Tan and P.Xue, "Enhanced H.263 to H.264/AVC Video Transcoding With Adaptive Intra-mode Decision", *IEEE Information, Communications and Signal Processing, 2005* Fifth International Conference, on Volume, Issue, 06-09 Dec. 2005, page(s)1130- 1134 .

[4] C.Wang, G.Sung, and J.Li, "Codec design for variable length to fixed-length data conversion for H.263", *IEEE Proceedings of the 2006 International conference* on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'06).

[5] Hwang, T.Oh, H.Jung and S.Ha, "Conversion of Reference C Code to Dataflow Model: H.264 Encoder case study", *IEEE Design Automation, 2006.* Asia and South Pacific Conference on Volume, Issue, 24-27 Jan. 2006 Page(s): 6 pp.

[6] H.kim and M.kim "The Design of Single Encoder and Decoder for Multi-view Video" *SPRINGER*, PSIVT 2006, LNCS 4319, pp. 732-741, 2006.

[7] Bialkowski, J.; Barkowsky, M.; Kaup, A. "Overview of Low-Complexity Video Transcoding from H.263 to H.264**"** *Multimedia and Expo, 2006 IEEE International Conference* on Volume , Issue , 9-12 July 2006 Page(s):49 - 52

[8] Chehimi, Fadi and Coulton, Paul and Edwards, Reuben (2006) "C++ Optimisations for Mobile Applications". In: The Tenth *IEEE International Symposium on Consumer Electronics*, June 29 – July 1, 2006, St.Petersburg , Russia.

[9] Zhihang Wang, Xiangyang Ji, Wen Gao, Qingming Huang and Debin Zhao "Effective algorithms for fast transcoding of AVS to H.264/AVC in the spatial domain" *Journal Multimedia Tools and Applications Springer* Netherlands, Volume 35, Number 2 / November, 2007 page(s):175 - 202.

[10] Pasqualini, S.Paola Pierleoni Fioretti, F.Andreoli, A.Univ. Politec. delle Marche, Ancona; "Adaptive Threshold For Intra Frame Prediction In H.263 To H.264 Smart-Transcoder" *Advanced Communication Technology, 2008. ICACT 2008.* 10th International Conference on 17-20 Feb. 2008
Volume: 2, On page(s): 1439-1444.

[11] Busschaert, H.J.; Reusens, P.P.; Dartois, L.; Desperben, L.Custom "A power efficient channel coder/decoder chip for GSM terminals**"** *Integrated Circuits Conference*, 1991., Proceedings of the IEEE 1991 Volume , Issue , 12-15 May 1991 Page(s):7.8/1 - 7.8/4.

[12] Gary Sullivan, "Recommended Simulation Common Conditions for H.26L Coding Efficiency Experiments on Low Resolution Progressive Scan Source Material," VCEG-N81, *14th meeting: Santa Barbara*, USA. Sept. 2001.

[13] Keman Yu, Jiangbo Lv, Jiang Li and Shipeng Li. Practical Real-TimeVideo Codec for Mobile Devices. Proceedings of 2003 IEEE InternationalConference on Multimedia and Expo, ICME 2003, USA, pages 509-512, 2003.

[14] B. Ashwani, Devesh Kandpal, Mayank Srivastava, Dr. Anupam Shukla "An Efficient Mode Selection Algorithm for H.264 encoder for Application in Low Computational power devices**"**

Anupam Shukla was born on 1st January 1965, at Bhilai (CG). He is presently working as an Associate Professor (Information Communication & Technology Deptt) at Atal Bihari Vajpayee Indian Institute of Information Technology & Management, (ABVIIITM), Gwalior (MP). He completed PhD (Electronics & Telecommunication) in the area of Artificial Neural Networks in the year 2002 and ME (Electronics & Telecommunication) Specialization in Computer Engineering in the year 1998 from Jadavpur University, Kolkata. He stood first position in the university and was awarded with gold medal. He completed BE (Hons) in Electronics Engineering in 1988 from MREC, Jaipur. He has teaching experience of 19 years. His research area includes Speech recognition, Artificial neural Networks, Image Processing & Robotics. He published around 57 papers in national/international journals and conferences

Arvind Kumar Verma was born on 26th September 1984, at Bhilai (C.G). He is presently pursuning M.Tech from Atal Bihari Vajpayee Indian Institute of Information Technology and Management, (ABV-IIITM), Gwalior (MP). He completed BE in Information Technology in 2006 from GEC, Bilaspur.

Sourabh Rungta is presently working as an Reader (Computer Science and Engineering Departtment) in RCET, Durg (CG). He completed M.Tech (Hons) in 2004. He completed BE in 1998. He has teaching experience of 5 years. He published around 5 papers in national/international conferences and journals.

Neeta Tripathi is principle of RCET, Durg. She has teaching experience of 20 years. She published around 30 papers in National/international conferences and journals. Her contributed research area includes speech recognition.