# Adapting x264 to Asynchronous Video Telephony for the Deaf

**Zhenyu Ma** and William D. Tucker
University of the Western Cape, Computer Science Department
Private Bag X17, Bellville 7535 South Africa
Phone: +27 21 959 3010 Fax: +27 21 959 1274 Email: {**zma**, btucker}@uwc.ac.za

**Abstract—Deaf people want to communicate remotely with sign language. Sign language requires sufficient video quality to be intelligible. Internet-based real-time video tools do not provide that quality. Our approach is to use asynchronous transmission to maintain video quality. Unfortunately, this entails a corresponding increase in latency. To reduce latency as much as possible, we sought to adapt a synchronous video codec to an asynchronous video application. First we compared several video codecs with subjective and objective metrics. This paper describes the process by which we chose x264 and integrated it into a Deaf telephony video application, and experimented to configure x264 optimally for the asynchronous environment.**

SATNAC Classification: Innovation and Regulatory – Telecommunications Developments and Inventions

Keywords: H.264, x264, asynchronous, latency, Deaf telephony, Quality of Service, video

## I. INTRODUCTION

Deaf people have access to information and communication technology (ICT) even though they are limited by their deafness to non-audio ICT. The MobileASL project provided the Deaf people with real time sign language communication over cellular telephones using the most popular and newest video codec—H.264 [1]. The H.264/AVC standard was first published in 2003 and was built on previous standards like MPEG-2 and MPEG-4 [11]. H.264 offered better compression, transmission and storage of video. In comparison to other sophisticated codecs like DivX and XviD, H.264 has been adopted for much synchronous communication, including IPTV, due to its low bit rate transmission.

In most cases, synchronous video based on these codes is not good enough for intelligible real-time sign language communication. This paper describes a project with the goal to improve Deaf video communication by adapting synchronous codecs for an asynchronous exchange of video where quality of the sign language video is improved at the expense of some additional delay. The intention is to minimise that delay as much as possible while retaining as much video quality as possible to support sign language. Thus, this paper explains how to find the most likely codec candidate to adapt for asynchronous Deaf video telephony.

The paper is organized as follows. Section II provides some background on the project. Section III provides a survey of related work. Section IV states the motivation of the approach in terms of project goals. The implementation process is described in section V. The experimental process of testing, data collection and analysis is presented in section VI. Finally, conclusions and future work are discussed in sections VII and VIII, respectively.

## II. BACKGROUND

For several years, we have worked with the Deaf Community of Cape Town (DCCT), a Deaf NGO situated in Newlands, Cape Town. The Deaf would like to communicate with their own language—sign language. Sign language video consists of detailed movements associated with facial expression, mouth shape and figure spelling from the point of perceptual view [9]. Hence, it demands much better quality than that offered by tools like Skype and Camfrog.

Deaf users currently use these tools at DCCT, but they complain about the size of video pictures, blurring of fast-speed motion, and jerkiness of some sequences. Synchronous video communication routinely consumes fifteen to thirty frames per second in order to provide a decent frame rate with minimal delay. This is adequate for hearing users but Deaf users are more concerned with picture quality than with delay since a tiny visual gesture may be the key to understanding an entire sequence. Therefore asynchronous communication offers a way to improve quality.

We piloted asynchronous video telephony for the Deaf in 2006 [7]. It was a peer-to-peer asynchronous video communication tool implemented in Java Media Framework (JMF). We used the JPEG codec supported by JMF. The quality of the video was deemed (by users) to be acceptable, but the delay also increased. That delay was unavoidable due to the recording and playing processes but was somewhat controllable by the users. The only real opportunities to decrease delay were to speed up the video compression and transmission.

## III. RELATED WORK

Video codecs have worked in two ways: temporal and spatial compression. Both schemes achieved “lossy” compression; meaning redundant or unnoticeable (to the viewer) information was discarded. In addition, all discarded information was non-retrievable.

Temporal compression dealt with related information that appeared in different frames and was not necessarily rebuilt for continuity to human eyes, such as background relative to foreground. In such cases, the compression algorithm compared the first frame, known as a key frame, with the next. The changed information was kept and a large portion of the file was deleted. If the scene changed, the algorithm tagged another key frame for the new scene and continued the process until the last frame was reached.

Spatial compression used a different approach to delete information that was common to the entire file or an entire sequence within the file. The algorithm also looked for redundant information, but it defined an area in terms of coordinates instead of indicating each pixel in the area. This approach originated from image processing where the encoders only considered the data that was contained within a single picture and bore no relationship to other frames in a sequence.

### *A. DivX and XviD*

Modern video codecs require flexibility, efficiency and robustness [5]. Both DivX and XviD, based on the MPEG-4 standard, met these demands. They originated from OpenDivX, and then broke into two branches until DivX became commercial software (www.divx.org). XviD remained an open source effort (www.xvid.org).

The DivX codec implemented lossy MPEG-4 Advanced Simple Profile (ASP), where quality was balanced against file size. DivX has proven quite popular, with releases for Windows, Linux and Mac. Recently, DivX also released DivX Web Player that provided 720 pixels HD playbacks live inside major web browsers.

While DivX has long been renowned for its excellent video quality, its counterpart equivalent XviD offers even more advanced quality. Founded in 2001, early XviD implemented MPEG-4 Simple Profile (SP) de/encoding. XviD 1.0 introduced MPEG-4 ASP compression including advanced coding tools like B-frames, quarter-pixel motion compensation and so forth. In later versions, additional features included MPEG-4 advanced video coding, high profile and dramatic compression performance advances.

### *B. H.264 and x264*

The latest well-known standard was H.264, developed by the Joint Video Team (JVT) [5] and its full name was MPEG-4 AVC, Advanced Video Coding defined in MPEG-4 Part 10 [2]. With a high compression ratio, flexibility and extensibility, many applications have adopted the H.264 standard. In comparison to previous standards, H.264's compression achieved over two times than that of MPEG-2 and almost double than that of MPEG-4 [16]. Meanwhile, the penalty was increased CPU power and the amount of time required.

H.264 employed techniques inherited from previous standards, such as basic video coding functions, motion estimations, motion compensations, transformation and quantisation. The basic structure of H.264 was motion-compensated transform, based on a block coding approach that divided a frame into macro blocks (MB). Additional features were variable block-size motion compensation with the block size as small as 4x4 pixels[11], more complex intra-frame compression, multiple reference frames, B-frame as reference and enhanced entropy coding methods—Context-Adaptive Variable-Length Coding (CAVLC) and Context-Adaptive Binary Arithmetic Coding (CABAC).

x264 was an open source encoder of H.264. In other words, H.264 was the standard while x264 was a product that implemented H.264. x264 has been used in many popular applications such as ffdshow, ffmpeg and MEncoder. According to a recent study at Moscow State University (MSU), x264 showed better quality than several commercial H264/AVC encoders [11]. Other results proved that the x264 codec yielded significantly better subjective quality than other widespread codecs such as DivX, XviD and WMV [14]. x264's high performance is ascribed to its flexibility in rate control, motion estimation (ME), MB mode decision, quantisation and frame type decision algorithms.

### *C. MobileASL Project*

MobileASL, a Deaf telephony project, employed x264 as a video encoder on a cell phone to give the Deaf people access to real-time mobile communication in their preferred language [1]. The proposed outcome of the MobileASL project was to maintain the intelligibility of sign language communication while maximally compressing the video sequence for stringent rate constraints and effectively simplifying the compression algorithm enough to reduce power consumption [2].

The MobileASL project focused on a Region of Interest (RoI) encoding process that contributed to a low bit rate for real-time transmission. This kind of encoding made for a differential resolution within the frame; that is, high resolution for the RoI parts and low resolution for non-RoI parts. The research utilised an eye tracker to collect eye movements of Deaf people watching sign language videos. Over 95% of the gaze points fell within the signer's face region, specifically on or near the lower part of the face [7]. It turned out that subtle changes in facial expression substantially changed the meaning of a hand gesture. For example, a gaze in a particular direction indicated different pronouns and raising one's eyebrows indicated a question.

### *D. Quality of Service*

Subjective quality measurement has been adopted for Quality of Service (QoS) evaluation. Mean Opinion Score (MOS) was one of the subjective methods, and defined a scaled opinion of controlled playback of spoken material [4]. This approach also worked for video. The MSU team used subjective video quality measurements to help obtain user opinions, including Stimulus Comparison Adjectival Categorical Judgement (SCACJ) from ITU-R, Double Stimulus Continuous Quality Scale (DSCQSII) from ITU-R, and Subjective Assessment of Multimedia Video Quality (SAMVIQ) from the European Broadcast Union (EBU).

Traditionally, objective quality measurements were performed by Mean Square Error (MSE) metrics: Signal to Noise Ratio (SNR) and Peak Signal to Noise Ratio (PSNR). PSNR was widely used to evaluate quality because of its simplicity, not because it took into account properties of the human visual system (HVS) [10]. In addition, Structural Similarity Index (SSIM) [11] and Video Quality Metric (VQM) [17] were other video quality evaluation methods. SSIM index was a combined value reflecting three components—luminance similarity, contrast similarity and structural similarity. This measurement was based on exploiting structural distortion instead of the error, and gave a correlation to the subjective impressions because the human vision system was highly attentive to structural information from the viewing field and not the errors. VQM was based on Discrete Cosine Transforms (DCT) video quality evaluation, corresponding to human perception. Results from [17] showed that VQM had a high correlation with subjective video quality assessment.

## IV. Motivation

Asynchronous video was a promising opportunity to increase subjective and objective quality for sign language telecommunication, with the obvious detrimental factor of an increase in latency. We wanted to learn if Deaf people considered this approach better than the synchronous video tools available to them on the Internet. Thus, asynchronous video quality needed to be evaluated, with respect to both video compression and the resulting latency.

We worked with several key DCCT members that had significant experience with SMS, Instant Messaging and Internet video conferencing. Overall, they preferred to communicate in their native tongue, sign language. They found Internet video difficult in many of the same ways hearing people found early voice over IP (VoIP) systems difficult: distorted words and variable delays that interfered with the natural conversation rhythm.

Preliminary asynchronous video experiments at DCCT taught us that asynchronous video communication must alter from information delivery to information interchange. That meant asynchronous communication needed to take more synchronous aspects into consideration, including user interface issues as well as reducing latency. This project addressed both of those issues. We redesigned the user interface and experimented with synchronous video codecs in an asynchronous exchange environment. We emphasised the latter in order to spend less time computing the compression algorithm and aimed for a small resultant file to spend less time on transmission. The next section describes the experimental implementation and the results are discussed in section VI.

## V. Implementation

The implementation aims were to minimise latency and maximise video quality. To this end, we built a tool to compare several video encoders: x264, XviD and DivX. The playback process employed the ffdshow package to decode a compressed video file and play it to the user. Therefore, there were no comparisons performed at playback, only for encoding. We employed both subjective and objective measurements with Deaf users and automated tools, respectively. The overall flow of the application developed to carry out the experiments is shown in Figure 1.

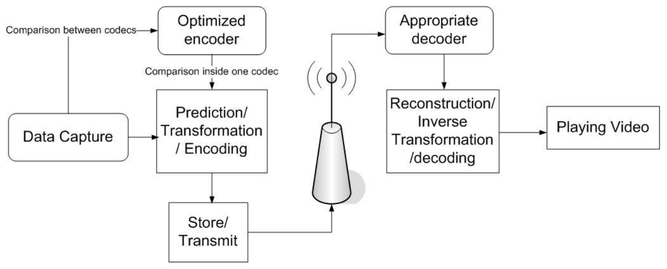
Comparison between codecs
Optimized encoder
Comparison inside one codec
Data Capture
Prediction/ Transformation / Encoding
Store/ Transmit
Appropriate decoder
Reconstruction/ Inverse Transformation /decoding
Playing Video

*Figure 1: Overview of the asynchronous video telephony tool for the Deaf shows the key stages involved with the compression and transmission of asynchronous video.*

The application was built with the Microsoft Visual C++ environment with the DShow API enabled. Simple presence and File Transfer Protocol (FTP) services were used. The application aimed to provide simple and easy interfaces for the Deaf user. There were three layers to the application: user interactive layer, video manipulation layer and transmission control layer. These layers are described in the following subsections.

### A. *User interactive layer*

The user interactive layer concerned the user interface. A user sent a connection request to another user, and a connection was established once the remote user accepted the request. Then the users could exchange sign language videos. Figure 2 shows the main window of this system. The main window of the application provides notification to the users by flicking the message box and the small coloured icon in the system icon tray. Event-driven message appears inside the message box to response the users to notify the arrival of new video file with flicking both the message box content and the small icon in the system icon tray. The buttons were quite simple: *capture* allowed the user to start recording sign language, *transmit* sent the message, *play* displayed a newly received video, and *replay* permitted to view the latest previous video again. A message box provided messages and hints to help the user.

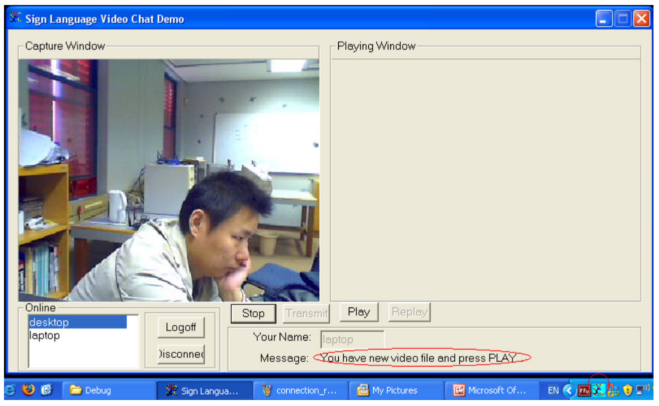

*Figure 2: Main window of the asynchronous video application with presence service.*

### B. *Video manipulation layer*

The video manipulation layer dealt with video capture, compression and playback processes. These processes were hidden from the user; only the message box told the user what was going on at any given moment. The compression process only ran after the capture process terminated. This avoided compressing unwanted video files if the user wanted to recapture the video and overwrite one that was not satisfactory.

The process to get a synchronous codec to work in asynchronous mode was complicated. For example, x264 involved open source code, x264vfw API and x264vfw.dll library. x264vfw API used the libx264.lib that was generated by compiling x264 source code. Then, having compiled x264vfw, x264vfw.dll was generated and could be used for video communication. Without that specific library, the application would not work, throwing a "No preferred codec found!" error.

### *C. Transmission layer*

The transmission layer consisted of the transmission protocol, FTP, and the orchestration of user notification messages. The application was intended for a wireless network environment and FTP was deemed acceptable. The main purpose of the application was to compare various codecs in order to determine which one would be best suited for asynchronous video telephony for Deaf people.

## VI. TESTING, DATA COLLECTION AND ANALYSIS

The first step in choosing a codec was to compare codecs in a controlled manner. We compared DivX, XviD and x264 codecs. After choosing the most appropriate codec, the task remained to adapt it into the asynchronous video application and optimise its performance. During the experimental phase, we captured a raw video as a reference. The video file had 640,198,656 bytes, was 112 seconds long, and 2819 frames. All comparisons were based on this reference video. The playback rate was fixed to 25.17 fps, and the compression rate varied depending on the compression algorithm. From a practical point of view, CPU utilization constrained compression time and the transmission process quite a bit. Therefore, comparison candidate files and corresponding log files were created with the CPU as idle as possible.

### *A. Different codecs comparison tests*

Three codecs, DivX, XviD and x264, were plugged into a simple video testing tool and corresponding data was recorded into log files. The performance of each codec was evaluated subjectively with MOS provided by users and objectively with the MSU video quality measurement tools.

The user sample for the subjective inter-codec evaluation was 17 Deaf participants. The experimentation required a sign language interpreter whose role was to explain the procedure and relate user opinions back to the researcher. The participant was shown a series of videos, each constructed with a different codec as well as the reference uncompressed reference video. Each participant was asked two questions for each video: one about blurring and the other about understanding the content. The participant did not know which video was which, and gave a scaled mark for each question. The results are shown in Figure 3. Deaf people considered XviD and x264 to be quite similar. They definitely thought that DivX video was worse than the other two. Overall, XviD appeared slightly stronger than x264.

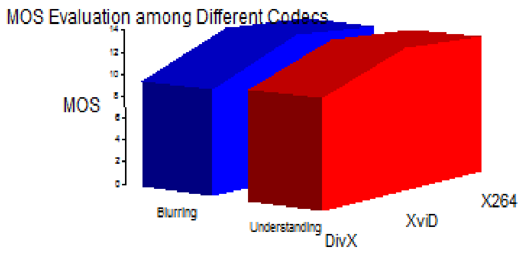

*Figure 3: MOS results on codec comparison with regard to blurring and understanding video content.*

The objective evaluation ran the three codecs through a battery of tests in the automated MSU suite, namely PSNR, SSIM and VQM. x264 emerged as the stronger candidate and supports the positive regard from the Deaf users. Figure 4 shows the objective evaluation results.

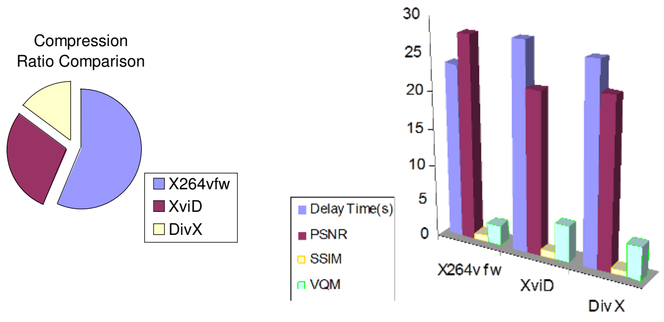

*Figure 4: Compression ratio comparison pie and other comparison metrics between DivX, XviD and x264 computed with the MSU video quality measurement tools. A video file encoded in x264 appeared to have a higher compression ratio with fairly similar PSNR values.*

### *B. x264 internal comparison tests*

The inter-codec comparison tests indicated that x264 was a worthwhile candidate to adapt into the asynchronous video telephony tool. The next step was to figure out how to configure x264 to achieve the best performance in its adapted asynchronous usage. x264 was based on conventional block-based motion-compensated video coding, and supported a number of configuration parameters that are summarised in Table 1. These parameters and their characteristics helped improve coding efficiency and retain reliable quality [16].

**Table 1 x264 parameters and their characteristics**

| Parameters | Characteristics |
|---|---|
| Integer Motion Estimation (ME) | dia: diamond search with radius1<br>hex: hexagonal search with radius 2<br>umh: uneven multi-hexagon search<br>Chroma: enabled or disabled |
| reference frame | up to 16 reference frames for motion compensation |
| B-frame | multiple B-frames with adaptive or non-adaptive decisions |
| direct Motion Vector (MV) prediction modes | spatial, temporal and auto |
| Entropy coding | CAVLC: luminance and chrominance residual encoding<br>CABAC: dynamically chooses probability module for encoding, depending on current content and previous encoded content |
| In-the-loop deblocking filtering | Enabled or disabled |

Intra-codec comparison tests were performed with x264 by varying the parameters laid out in Table 1. We continued to use the MSU video quality measurement tests for PSNR, SSIM index and the VQM value. We also built some tests of our own, including compression ratio (CR), compression time (CT), transmission time (TT) and delay time (DT). The test comparisons concentrated on the increment and decrement percentages of the all of these metrics. During the intra-codec testing phase, the optimisations were also adapted into the asynchronous video application, which meant that the application code was adjusted accordingly.

Motion Estimation (ME) played a significant role in the encoding process. It divided the moving picture into several MBs or blocks and searched each MB or block to find the corresponding position in the adjacent frame, and then calculated the relative spatial offset from the difference. That offset was the Motion Vector (MV). The ME method to find the MV was the search method and affected the encoding efficiency. Figure 5 shows the comparison of different search methods (dia, umh and hex). The dia search method made x264 more efficient. Then, disabling the chroma during ME decreased delay time 6.463% without degrading video quality with respect to the comparison metrics.

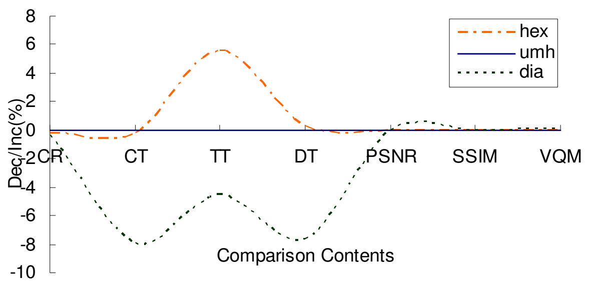

*Figure 5: ME methods comparison between dia, umh and hex.*

The reference frame test sought an optimal number of reference frames. In earlier standards, the number was typically one, or in the case of conventional B-frame, two. x264 allowed up to 16 reference frames to be used. That could lead to modest improvements in bit rate and quality. In most cases, it was not necessary to use so many reference frames. Figure 6 shows the results of the reference frame test. We took one reference frame as a baseline, the solid blue line in the x-axis. The more reference frames are chosen, the greater the compression ratio is, the smaller the file size is, and the less time transmission takes. However, the compression process is complex and takes longer to calculate the residues from different reference frames. In this case, we chose two reference frames for sake of saving latency.

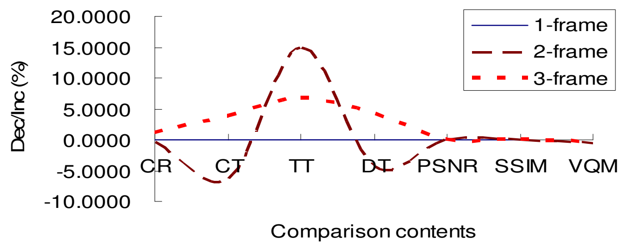

*Figure 6: Number of reference frames comparison between one to three references.*

The B-frame test considered the number of B-frames. Figure 7 shows the results of the comparison test. Having two B-frames showed a sharp curve on the graph. The adaptive B-frame decision algorithm had a strong tendency to avoid B-frames during fades. From this test, disabling adaptive B-frame favourably reduced delay time 1.86% and increased PSNR 0.0112%, SSIM 0.0103% and VQM 0.406% because only one B-frame was used. If adaptive B-frame decision-making were enabled, fast movement areas of the video suffered.

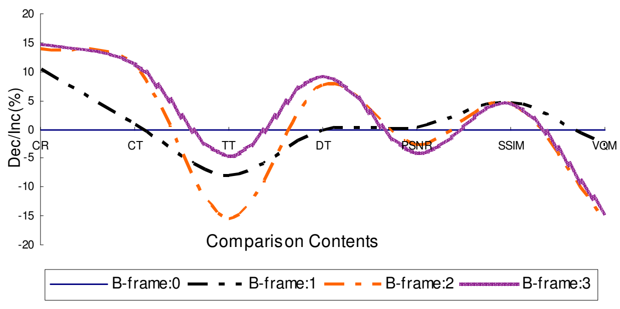

*Figure 7:* Comparison on B-frame numbers: none, 1, 2 and 3.

In order to find out the better mode to direct MV searching, the MV test compared several modes: spatial, temporal and auto. As mentioned earlier, the default MV was calculated from relative spatial offsets. Figure 8 shows that the auto mode performed well to direct MV prediction. Obviously, the auto mode might be spatial or temporal depending on the complexity of the contents in the current frame (or field). The auto algorithm decided the mode for error concealment accordingly.

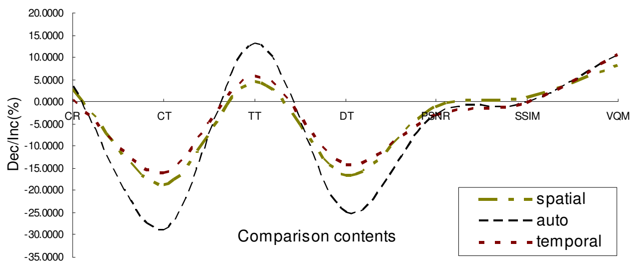

*Figure 8: Comparison between the modes that direct MV prediction.*

The CAVLC and CABAC tests indicated that CABAC was more complex but more efficient than CAVLC. Compression under CABAC comprised an 8% increment of delay for 0.018dB improvement of PSNR in comparison to CAVLC. Since CABAC was a lossless algorithm to compress syntax elements into probabilities in a given context, it needed to take more time in the compression process.

An in-the-loop deblocking filter prevented the blocking of artefacts incurred from spatial motion vector prediction that were common to other DCT-based image compression techniques. The compression speed penalty had a heavy impact on latency.

Unfortunately, x264 did not contain all of the features that H.264 has, such as Switching I-frame (SI) and Switching P-frame (SP) slices, Flexible Macroblock Ordering (FMO), Arbitrary Slice Ordering (ASO), Redundant Slices (RS) and Data Partitioning (DP) and so on. However, from the characteristics x264 provides so far, the project saw some great changes in latency and quality after adapting x264 into asynchronous use for Deaf telephony.

## VII. CONCLUSION

We chose x264 to provide low latency and high quality for asynchronous video telephony. The adaptation process configured x264 with: diamond search motion estimation without chrominance; two reference frames; one B-frame without self-adaptiveness; automatic motion vector mode; CAVLC entropy coding; and some other minor factors. This configuration of x264 provided fast compression, fast transmission and high quality playback with less complex calculations. Thus, x264 enabled this project to move toward providing better quality asynchronous video communication for the Deaf.

## VIII. Future work

Research on asynchronous Deaf video telephony must continue. Synchronous communication is more attractive to end users despite its difficulties. If the end user would not notice the delay, asynchronous technology could become widely accepted for Deaf communication. Simulating a synchronous environment with asynchronous technology is the next step and will involve more codec optimisation to reduce bit rate, decrease compression time and increase the compression ratio so as to enable such services to run on mobile devices.

## Acknowledgments

The authors thank the staff and members of Deaf Community of Cape Town (DCCT) for their participation in the project. We also thank Telkom, Cisco and THRIP for financial support via the Telkom Centre of Excellence (CoE) programme.

## References

[1] A. Cavender, R.E. Ladner, and E.A. Riskin, "MobileASL: intelligibility of sign language video as constrained by mobile phone technology," *8th International ACM SIGACCESS Conference on Computers and Accessibility*, Portland, Oregon, USA, 2006, pp. 71-78.

[2] J.W. Chen, C.Y. Kao and Y.L. Lin, "Introduction to H.264 Advanced Video Coding". *2006 Conference on Asia South Pacific design automation*, New York, NY, USA, 2006, pp. 736-741.

[3] F.M. Ciaramello and S.S. Hemami, "Complexity constrained rate-distortion optimization of sign language video using an objective intelligibility metric," *Proceedings of SPIE Vol. 6822, Visual Communication and Image Processing 2008*, San Jose, CA, January 2008.

[4] ITU, "Mean Opinion Score (MOS) terminology", P.800.1 (03/2001), 2003.

[5] Y.V. Ivanov and C.J. Bleakley, "Dynamic Complexity Scaling fo rReal-Time H.264/AVC video Encoding," *MM'07*, Augsburg, Bavaria, Germany, September 23-28, 2007, pp. 962-970.

[6] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H264/ISO/IEC 14496-10 AVC)", Document JVT-G050d35.doc, 7th Meeting: Pattaya, Thailand, March 2003.

[7] G. Lu, "Communication and Computing for Distributed Multimedia Systems", Artech House, Inc., Norwood, MA, 1996, pp. 63-90.

[8] Z. Ma and W.D. Tucker, "Asynchronous Video Telephony for the Deaf," *South African Telecommunications Networks & Applications Conference*, Mauritius, 9-13 September 2007.

[9] L. J. Muir and I.E.G. Richardson, "Perception of sign language and its applications to visual communications for deaf people," *Journal of Deaf Studies and Deaf Education*, volume 10, 2005, pp. 390-401.

[10] O. Nemcic, M. Vranje and S. Rimac-Drlje, "Comparison of H.264/AVC and MPEG-4 Part 2 Coded Video", *49th International Symposium ELMAR-2007 focused on Mobile Multimedia*, Zadar, Croatia, 12-14 September 2007, pp. 41-44.

[11] I. Richardson, "An Overview of H.264 Advanced Video Coding," Vcodex White Paper, March 2007.

[12] G.J. Sullivan, P. Topiwala, and A. Luthra, "The H264/AVC advanced video coding standard: Overview and introduction to the fidelity arrange extensions," *SPIE conference on Digital Image Processing*, August 2004.

[13] D. Vatolin, O. Petrov, A. Parshin and A. Titarenko, "MPEG-4 AVC/H.264 video codec comparison," Computer Science Department, Moscow State University, Graphics and Media Lab, December 2005.

[14] D. Vatolin, O. Petrov, A. Parshin and A. Titarenko, "MSU Subjective Comparison of Modern Video Codecs," Computer Science Department, Moscow State University Graphics and Media Lab, January 2006.

[15] Z. Wang, L. Lu, and A.C. Bovic, "Video quality assessment using structural distortion measurement," *Signal Processing: Image Communication special issue on Objective video quality metrics*, vol. 19, no.2, pp. 121-132, February 2004.

[16] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," in *IEEE transactions on Circuits and Systems for Video Technology, vol. 13, no. 7*, 2003, 560-576.

[17] F. Xiao, "DCT-based video quality evaluation", student final project Digital Video Processing (EE392J), 2000.

## Authors

Zhenyu Ma is a Masters student with the Broadband Applications Network Group (BANG) in the Department of Computer Science at the University of the Western Cape (UWC). He is currently working on a video relay service for the Deaf.

William D. Tucker is a senior lecturer of Computer Science at UWC. He leads BANG research. His PhD on communication abstractions is near completion at the University of Cape Town.