# IMPROVED MACHINE LEARNING TECHNIQUES FOR LOW COMPLEXITY MPEG-2 TO H.264 TRANSCODING USING OPTIMIZED CODECS

*Chris Holder†, Tao Pin‡,and Hari Kalva†*
†Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL
‡Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China

## ABSTRACT

This paper discusses techniques for efficiently implementing a Mpeg-2 to H.264 video transcoder. The transcoding results reported in the literature are based on a reference implementation and may not reflect the true performance gains obtained in real world systems. We have developed low complexity transcoding algorithms and have implemented these solutions using highly optimized encoder and decoder implementations available from Intel. The transcoding algorithms are based on exploiting the mode decision knowledge inherent in the decoded MPEG-2 data. Machine learning techniques are used to make accurate and low-complexity H.264 MB encoding mode decisions. The results show that the proposed transcoder reduces the complexity by 50% without a significant loss in PSNR. This performance improvement in production quality transcoders, and demonstrates the practicality of machine learning based video transcoding algorithms.

***Index Terms***— H.264, Mpeg-2, transcoding, machine learning

## 1. INTRODUCTION

The demand for MPEG-2 to H.264 video transcoding has been clearly demonstrated and this is an active area of research. This paper presents transcoding using a highly optimized and production quality H.264 encoder and MPEG-2 decoder. Most papers that discuss this topic have been using the un-optimized reference implementations: JM implementation of H.264 encoder and MPEG simulation group's implementation of MPEG-2 decoder[3]. For example, the actual complexity and time performance of the JM 14.0 is roughly 4 times slower than the Intel IPP implementation using only one core. The performance gains reported thus may not reflect the gains in real world products. The developers of x264 demonstrated that the performance of their H.264 implementation is faster and has quality very close to that of the JM implementation[1]. After this being said, this paper uses Intel's IPP implementation of H.264 because the MSU's H.264 codec comparisons show that Intel's performance is close to the x264 implementation[6]. Even though an open-source project (handbrake) for MPEG-2 to H.264 was analyzed for a set of standards, the current plan is to make a more universal transcoder as Intel IPP supports multiple video formats in the baseline transcoder. A universal transcoder will give the opportunity to more researchers to re-use the structures for passing transcoder information for their own implementation in other standards. The algorithm which was chosen for implementation in this paper also had to be universal and ability to adapt easily. Overall, reduction in transcoder complexity will reduce the current cost for large scale transcoding.

The algorithm used for this transcoder is based on machine learning similar to the transcoders reported in [3]. This implementation was done on the JM H.264 encoder and was the first generation of machine learning algorithms used for mode decisions for transcoding. Many new training techniques and variables for the machine learning will be presented in this paper. The variables and techniques presented in this paper are a novelty over the past Mpeg-2 to H.264 papers. The techniques are compared in training complexity, quality, and speed. The variables have been changed to all integer values and all floats have been removed. Validation of using the J4.8 machine learning algorithm (C4.5 variant) is much more apparent. Most of the other machine learning algorithms have too much complexity for implementation in an efficient encoder; for example, neural networks or genetic algorithms.

## 2. MACHINE LEARNING BASED TRANSCODING

This section provides a brief overview of several of the machine learning algorithm used. In [3], the variables used were all based on the luma residue from the input Mpeg-2 video. The means and variances of the 16 4x4 blocks, the mean of the entire 16x16 block, and the coded block pattern were given to the J4.8 algorithm to generate decision trees (an if-else tree for implementation). A total of 37 different variables were used to generate the tree. The training was only done on one frame of the sequence called Flower Garden (CIF). In this paper we use 131 relevant attributes to improve the mode decision process. Larger training sets were also used to improve the performance. However, because of space considerations, all the variables are not listed here. The additional variables used include DCT coefficients and neighboring MB information. The training was much more strenuous; the training set had three different video sizes: CIF, 4CIF, and 420p; the test data for finding the best tree was the full sequence of 10 different clips; the two frame types (P,B) were split into two different training and testing datasets. The three main techniques

This work was supported in part by the National Science Foundation under OISE-0730065. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

978-1-4244-2559-4/09/$25.00 ©2009 IEEE

implemented were: One Frame Trained [3], Mode Separation [5], and on-line multi-tree boosting.

**2.2 Machine Learning Techniques**

**Mode Separated -** This method separated all the possible MPEG-2 modes for training which is based on the hypothesis that the H.264 mode decisions for each of the MPEG -2 modes are different and vary with the MPEG-2 MB mode. This approach will cause the trees to be more biased towards skipped MBs for skipped MPEG-2 blocks, and inter type blocks for inter blocks in MPEG-2. MPEG-2 has five modes for P frames and nine modes for B frames. A total of fourteen trees were trained using the first 2000 MBs of a sequence, this was done for 10 sequences of various sizes, and to decide on the trained tree, the trees were tested against all dataset of ten full sequences.

**On-line Multi-Tree Boosting -** This last method uses five different trees which were trained using various amounts of MBs from the ten sequences, then tested against the sequences separately, and the top 5 trees from each frame type were chosen. Boosting is a machine learning algorithm which uses several different weak machine learning algorithms or datasets to produce a stronger one. This is a variation which is based on game theory [4]. The first P and B frame are encoded without using machine learning for the mode decision, then the same frame is run through the five different trees and if the tree has a correct response, a vote is added. The next frame's MBs will use these five trees and the one with the most votes will be used to determine the H.264 MB mode.

Different variations of these methods were evaluated and will be shown in results section. These methods are the basis of the machine learning techniques used.

## 3. RESULTS

The results were obtained from an Intel Core 2 duo CPU 1.6 Ghz with 2 GB of RAM. The transcoder was run as a single core and compiled as a 32 bit application. The H.264 motion search was set to log search mode with a range of 8 pixels; only one reference frame was used for our experiments. In order to demonstrate flexibility of the machine learning can be generalized to other sequences than the ones trained on, the sequences used are all 4CIF and none of the trees used are in the sequence.

The results showed a significant speed-up of the transcoding process by using machine learning. This demonstrates time gains which can be performed by re-use of decoder data. The encoder speed-up of these sequences demonstrates the practicality of machine learning based solutions in production quality transcoders. Figure 1 shows the PSNR curve for the sequence City. Table 1 shows the PSNR loss and complexity reduction for sequence City and Soccer.

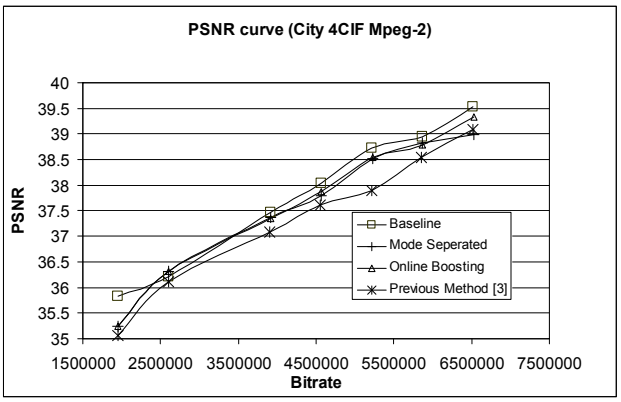


Figure 1. RD performance of transcoding algorithms

Table1: Transcoder performance comparison

| | Harbor | | City | |
|---|---|---|---|---|
| | Time % | Δ PSNR | Complexity Loss | ΔPSNR |
| Previous Method [3] | -61.42% | -0.37 | -58.60% | -0.48 |
| Mode Separated | -50.59% | -0.29 | -46.64% | -0.24 |
| Online Boosting | -45.11% | -0.27 | -57.81% | -0.19 |

## 4. CONLCUSION

This paper presents algorithms for low complexity transcoding implemented in optimized and production quality codecs. We show that machine learning based algorithms reduce the complexity significantly. New machine learning approaches were developed to improve the transcoder performance. While the complexity reduction is 10% lower than similar algorithms implemented using reference software, our approach shows that machine learning based transcoding reduces the complexity by over 50% even when using optimized, production quality codecs. Our new machine learning methods also reduce the PSNR loss compared to previous methods.

## 5. REFERENCES

[1] Loren Merritt and Rahul Vanam "Improved Rate Control and Motion Estimation for H.264 Encoder" Proceedings of the ICIP 2007

[2] A. Vetro, C. Christopulos, and H. Sun, "Video transcoding architectures and techniques: An overview," IEEE Signal Process. Mag., vol. 20, no. 2, pp. 18–29, Mar. 2003.

[3] G. Fernandez-Escribano et. al., "Very Low Complexity MPEG-2 to H.264 Transcoding Using Machine Learning," Proc. of the ACM Multimedia 2006, pp. 931-940.

[4] Yoav Freund and Robert E. Schapire, "Game theory, on-line prediction and boosting," In Proceedings of the Ninth Annual Conference on Computational Learning Theory, pages 325-332, 1996.

[5] C. Holder and H. Kalva, "H.263 to VP-6 Video Transcoder," Proceedings of SPIE/VCIP 2008, January 2008.

[6] "MPEG-4 AVC/ H.264 video codec comparison," CS MSU Graphics & Media Lab Video Group, December 2007, http://compression.ru/video/codec_comparison .