I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Maria Pantoja

ENTITLED

An Efficient IPB-Picture Transcoder with Joint Pixel and Transform Domain Processing

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING

Thesis Advisor
Dr. Nam Ling

Thesis Reader
Dr. Silvia Figueira

Chair of Department
Dr. Weijia Shang

Thesis Reader
Dr. JoAnne Holliday

Thesis Reader
Dr. Tokunbo Ogunfunmi

Thesis Reader
Dr. Weijia Shang
An Efficient IPB-Picture Transcoder with Joint Pixel and Transform Domain Processing

By

Maria Pantoja

Dissertation

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Engineering in the School of Engineering at Santa Clara University, 2009

Santa Clara, California
Dedicated to David
Acknowledgements

I would like to take this opportunity to acknowledge and highlight the contributions of those who help me succeed at Santa Clara University. My advisor, Nam Ling, I am grateful for his patience in reading my partially developed ideas, for giving a decent form to all my papers and for working so hard to make the papers better and always asking the important questions.

I would like to thank Hari Kalva and Jae-Beom Lee for giving me different topics that could be interesting for a PhD research from an industry and academic point of view. I not only owe to them the idea of transcoding video for my research topic but also their support from the beginning to the end, without them I know for sure I would not have finished this thesis.

There are other people who made my stay at Santa Clara a wonderful experience and I would like to take this opportunity to thank my students at SCU. I did really enjoy being a TA and helping beginners to learn programming. I always thought it will be hard to teach young people (everybody is afraid of teenagers) and I always thought that at some point I will find at least one student that will make my life hell and I would have to say to the people that advised me against rejecting the Packard fellowship to become a teaching assistant (TA) “You were right”. But I have found nothing like that. I can say that all of the students that I have encountered at SCU were happy, polite, diligent, and hard working (ok not all of them worked hard but most), some of them loved programming, some of them hated programming, but hopefully at the end of the quarter they all learned something new and useful. Looking at them I know that I do not have anything to fear
for my son. There is a big chance that he will grow up to be a wonderful teenager, something that most people think is impossible, especially in America. And I could not thank enough my Ph.D. committee, Drs. Silvia Figueira, JoAnne Holliday, Shoba Krishnan, Tokunbo Ogunfunmi, and Weijia Shang, for their invaluable support and suggestions throughout the program and their willingness to spend their hard earned vacation time for me.

Above all, I owe my Ph.D. success to the help provided by my husband Miguel, he supported me in any way that he could, he is the most elegant programmer in the world (no contest for the rest of the world’s programmers), and constantly showed me how most complicated concepts could be explained in a way that anybody interested could understand them. This work would have never happened if it wasn’t for my parents taking care of my son and for my two sisters, Ana, and Marina, always happy, fun, and fashionable.

I would also like to thank the Dolphin South End (DSE) runners of San Francisco, I did discover that nothing helps me more relax than running, and running with the DSE has been my medicine for every Sunday, thanks to them that I could start fresh and with full energy on Mondays.

This work was financially supported in part by fellowships from Packard foundation and the department TASHip and to them would be forever grateful.
ABSTRACT

The standardization of video compression technology gave margin to a revolution in broadcast television and home entertainment. Current standards (MPEG-1, MPEG-2, MPEG-4, H.261, H.263 and H.264/MPEG-4 AVC) as well as the recent VC-1 have contributed to such revolution. The need to move data among devices with different capabilities and standards creates a need for transcoding. Transcoders convert a video stream from one format to another, involving two different standards, but they can also be used to perform resolution conversion, and rate control. This dissertation covers transcoding among different standards (VC-1 to H.264), resolution, and rate control transcoding.

This work presents an efficient transcoding algorithm from VC-1 video to H.264 video, includes a complete transcoder that addresses I, P, B, and interlaced pictures. The proposed algorithm also presents a solution to several transcoding problems such as transcoding with resolution conversion, quality enhancement, and rate control transcoding. The main differences between the two standards were analyzed and a solution for transforming from one to the other is presented. The dissertation proposes a joint of two approaches, pixel domain transcoding and transforms domain transcoding, that tradeoff between quality and complexity. The pixel domain transcoding part exploits the variable size transform used in VC-1 to select the variable block size for motion compensation in H.264. Since VC-1 uses an adaptive block size integer transform, the transform domain transcoding part proposes an algorithm to transcode the transform coefficients from VC-1 to those for H.264. The typical drift error of transform domain transcoding is also analyzed. High Profile features of both standards are also discussed in
this dissertation; in particular, adaptive size transform and interlacing. Experimental results of the two disjoint approaches, sole pixel domain transcoding and sole transform domain transcoding, are presented. These experiments show that the pixel domain approach reduces computational complexity compared to a fully cascaded transcoder by about 60% with negligible drop in PSNR, while the transform domain approach further reduces the complexity but with a PSNR drop of around 1 dB. Finally, a joint pixel/transform domain transcoder is proposed with drift error compensation when reducing complexity is the main objective of the transcoder. In contrast, pixel domain transcoder may be a good candidate when maintaining the PSNR is the highest goal.

This dissertation also presents a solution for the case in which the transcoder is used for resolution conversion (up/down sampling). We discuss an efficient transcoding algorithm from VC-1 to H.264 video with down/up sampling. Video compression usually uses quantization and other techniques that result in lower video quality. To address this problem a novel super-resolution (SR) algorithm based on irregular sampling (IS) is presented. The high-resolution (HR) frame is obtained as an interpolation of one or more previous frames; the resulting interpolated frame has samples non-uniformly spaced in the areas where movement happened. To reconstruct the irregular sampled frame we use a well known irregular sampling algorithm (modified to perform in 2D space). SR algorithms are in general computationally expensive; to address this problem we also present a hardware feasibility study. The proposed solution is not intended for any specific application but we have specifically tested the algorithm in a transcoding application. Experimental results show that the proposed algorithm can successfully remove artifacts and generate high video quality.
Lastly, this dissertation also covers the case in which the transcoder is used as a rate control transcoder. For rate-control video transcoding the optimal video quality is obtained while maintaining a target bit rate for the transcoded output. The relationship between the rate control in VC-1 and H.264 was studied and using the rate control information from the VC-1 encoded video, we accelerate the rate control algorithm in H.264. The key idea for the rate-control transcoder is to find the relationship between quantization parameters (QP) in VC-1 to QP in H.264 so the first can be used to estimate the second, short cutting the need for complex estimation in H.264. Experiments show that the previous idea works well only if the QP ranges from 10 to 29, to solve this; a medium complexity algorithm was also developed. The key idea for the medium complexity algorithm is to use mean absolute differences (MAD) and sum of absolute transform differences (SATD) calculated in VC-1, for frames transcoded using a pixel/transform domain transcoder respectively, to estimate the complexity of the macroblock (MB) in H.264. To transcode the entire QP range we propose a combination of low/medium complexity tools. Experimental results show that the proposed rate control for transcoding is less complex than that for a fully cascaded decoder while maintaining the target bit rate and PSNR.
# Index

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dedications</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>iv</td>
</tr>
<tr>
<td>Abstract</td>
<td>vi</td>
</tr>
<tr>
<td>Index</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xiv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xviii</td>
</tr>
<tr>
<td>Chapter 1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Related Work</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Thesis Outline</td>
<td>11</td>
</tr>
<tr>
<td>1.3.1 Format Conversion Transcoding and Transcoding VC-1 to H.264 Baseline and High Profile</td>
<td>11</td>
</tr>
<tr>
<td>1.3.2 Resolution Conversion Transcoding</td>
<td>12</td>
</tr>
<tr>
<td>1.3.3 Rate Control Transcoding</td>
<td>14</td>
</tr>
<tr>
<td>Chapter 2 Overview of Video Coding Standards</td>
<td>18</td>
</tr>
<tr>
<td>2.1 Compression Tools for VC-1 Progressive</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Compression Tools for H.264</td>
<td>19</td>
</tr>
<tr>
<td>Chapter 3 Pixel Domain Transcoder</td>
<td>23</td>
</tr>
<tr>
<td>3.1 I-Frames</td>
<td>24</td>
</tr>
<tr>
<td>3.1.1 Intra Macroblock</td>
<td>24</td>
</tr>
<tr>
<td>3.1.1.1 Intra Macroblock Mode Mapping</td>
<td>26</td>
</tr>
</tbody>
</table>
3.1.1.2 Prediction Mode Computation for Intra MBs . 26

3.1.2 Mixed MB Mode Mapping ................................. 28

3.2 P-Frames .......................................................... 29

3.2.1 Inter MB Mode Mapping ................................. 29

3.2.2 MV Mapping ..................................................... 34

3.3 Overlap Smoothing Transform ................................. 34

3.4 B-Frames .......................................................... 35

3.5 Loop Filtering ..................................................... 37

3.6 Intensity Compensation ................................. 40

3.7 High Profile Features ........................................... 42

3.7.1 Adaptive Size Transform for H.264 High Profile .. 42

3.7.2 Interlacing ..................................................... 43

Chapter 4 Transform Domain Transcoder .......................... 49

4.1 Coefficient Conversion ........................................... 51

4.1.1 VC-1 to H.264 Transform ................................... 51

4.2 Fast Algorithm Implementation ................................ 54

4.3 Drift Error .......................................................... 57

4.3.1 Integer Pixel Displacement ................................... 59

4.3.2 ½ Pixel Accuracy ............................................... 63

4.3.3 ¼ Pixel Accuracy ............................................... 65

4.3.4 Error Drift Amount ............................................ 65

4.4 High Profile Features ........................................... 66

4.4.1 Adaptive Size Transform ................................... 66
4.4.2 Interlacing ......................................................... 66

Chapter 5 Results for Pixel and Transform Domain Transcoder ..... 68

5.1 Results ............................................................... 72

5.1.1 Pixel Domain Results.............................................. 72

5.1.1.1 I-Frames Results.............................................. 72

5.1.1.2 P-Frames Results.............................................. 73

5.1.1.3 B-Frames Results.............................................. 76

5.1.1.4 Transform Size in H.264 and Interlace Results 76

5.1.2 Transform Domain Results ...................................... 76

5.1.3 Complexity ....................................................... 78

5.2 Summary ............................................................. 89

Chapter 6 Resolution Conversion ...................................... 90

6.1 Downsampling ....................................................... 90

6.2 Upsampling .......................................................... 92

6.3 Quality Enhancement using Irregular Sampling ................. 93

6.3.1 Introduction to Super Resolution .............................. 93

6.3.2 Irregular Sampling Theory .................................... 96

6.3.3 Irregular Sampling for Video Quality Enhancement .... 98

6.3.4 Different Interpolation Tech. ................................. 101

6.3.5 Use of Different Number of Frames ......................... 102

6.3.6 Hardware Feasibility Study ................................. 102

6.4 Results .............................................................. 103

6.4.1 Subjective Quality Evaluation................................. 104
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4.2 PSNR Performance</td>
<td>104</td>
</tr>
<tr>
<td>6.4.3 Time Complexity Evaluation</td>
<td>105</td>
</tr>
<tr>
<td>6.4.3 Comparison with Other SR Algorithms</td>
<td>106</td>
</tr>
<tr>
<td>6.5 Summary</td>
<td>115</td>
</tr>
<tr>
<td>Chapter 7 Rate Control Transcoding</td>
<td>116</td>
</tr>
<tr>
<td>7.1 Rate Control in VC-1 and H.264</td>
<td>116</td>
</tr>
<tr>
<td>7.1.1 VC-1 Rate Control</td>
<td>116</td>
</tr>
<tr>
<td>7.1.2 H.264 Rate Control</td>
<td>117</td>
</tr>
<tr>
<td>7.2 Low Complexity Rate Control</td>
<td>118</td>
</tr>
<tr>
<td>7.2.1 Frame Layer Rate Control</td>
<td>119</td>
</tr>
<tr>
<td>7.2.2 MB Layer Rate Control</td>
<td>120</td>
</tr>
<tr>
<td>7.2.3 Low Rate Control Tools</td>
<td>121</td>
</tr>
<tr>
<td>7.2.4 Rate Control in Transcoding</td>
<td>123</td>
</tr>
<tr>
<td>7.3 Medium Complexity Rate Control</td>
<td>123</td>
</tr>
<tr>
<td>7.3.1 Complexity Estimation</td>
<td>124</td>
</tr>
<tr>
<td>7.3.2 Texture Bit Allocation</td>
<td>124</td>
</tr>
<tr>
<td>7.4 Results</td>
<td>126</td>
</tr>
<tr>
<td>7.4.1 Low Complexity Tools</td>
<td>127</td>
</tr>
<tr>
<td>7.4.2 Medium Complexity Tools</td>
<td>128</td>
</tr>
<tr>
<td>7.5 Summary</td>
<td>131</td>
</tr>
<tr>
<td>Chapter 8 Conclusions</td>
<td>132</td>
</tr>
</tbody>
</table>
8.1 Summary of Techniques for Standard Independent Transcoding. 134

8.1.1 Pixel Domain Transcoding ............................................. 134

8.1.1.1 I-Frames ............................................................. 134

8.1.1.2 P-Frames ............................................................. 135

8.1.1.3 B-Frames ............................................................. 135

8.1.2 Transform Domain Transcoding ................................. 135

8.1.3 Resolution Conversion ............................................. 136

8.1.4 Rate Control ........................................................... 136

8.1.5 Quality Enhancement ................................................. 136

8.2 Future of Transcoding ..................................................... 136

References ........................................................................... 139

Glossary ............................................................................... 149

Publications ................................................................. 155

Biography .......................................................................... 157
LIST OF FIGURES

Figure 1. Transcoder ............................................................... 3
Figure 2. A transcoding application scenario ................................ 3
Figure 3. Reference cascade pixel-domain transcoder for VC-1 to H.264 .... 10
Figure 4. Transform-domain transcoder for VC-1 to H.264 .................... 10
Figure 5. VC-1 to H.264 transform domain transcoding for P/B frames ...... 12
Figure 6. VC-1 to H.264 transcoder block diagram ........................... 12
Figure 7. VC-1 to H.264 transcoder with resolution conversion .............. 16
Figure 8. VC-1 to H.264 rate control transcoder .............................. 17
Figure 9. VC-1 MB types for MC ................................................. 19
Figure 10. H.264 MB types for MC .............................................. 21
Figure 11. VC-1 to H.264 intra MB transcoding in pixel domain .......... 25
Figure 12. Intra MB mode mapping according to MB variance for sequence Foreman (QP=28) ......................................................... 25
Figure 13. Possible prediction directions for H.264 intra 4x4 mode .......... 26
Figure 14. H.264 MC prediction for 16x16 MBs ............................... 26
Figure 15. Examples of Mode 0 vertical MC prediction ......................... 27
Figure 16. Inter MB mode transcoding with QP<th and VC-1 1-MV ....... 32
Figure 17. Inter MB mode transcoding with QP<th and VC-1 4-MV ....... 32
Figure 18. Inter MB mode transcoding with QP>th ............................ 33
Figure 19. Vertical and horizontal edges representation for overlapping transform .................................................................................. 34
Figure 20. B-Frame prediction in VC-1 .......................................... 37
Figure 21. B-Frame prediction in H.264 ........................................ 37

Figure 22. Effect of overlap transform and deblocking filtering

(Foreman QCIF) Difference images d) and e) highlight the
edges improved by the process. ........................................... 38

Figure 23. In-loop vertical filtering on 4-pixel segment ...................... 40

Figure 24. In-loop horizontal filtering on the 3rd pixel pair ................. 40

Figure 25. Transform size mapping ........................................... 42

Figure 26. Flowchart for adaptive transform size ............................ 43

Figure 27. Interlace picture coding mode supported by VC-1 4:2:0

luma and chroma temporal and vertical sample positions (where
from left to right is shown a top field, bottom field, top field, and
bottom field). ................................................................. 44

Figure 28. VC-1 field picture reference. ...................................... 44

Figure 29. VC-1 luminance macroblock structure for frame picture. ..... 45

Figure 30. H.264 PAFF and MBAFF field coding ......................... 47

Figure 31. Flowchart for interlaced video transcoding. ..................... 47

Figure 32. Distribution of the number of neighboring field-encoded

MBs when the “brute force” H.264 method chooses frame or field
mode for P-slice (sequence “Football”). ................................... 48

Figure 33. 8-point and 4-point inverse transform in VC-1 ................. 50

Figure 34. H.264 transform matrices ......................................... 51

Figure 35. Fast algorithm for the transform domain VC-1 to HT
conversion................................................................. 57
Figure 36. MC in the DCT transform domain ........................................... 59

Figure 37. Motion compensation in the HT domain for integer displacement ... 63

Figure 38. Predicted block for ½ pixel accuracy for VC-1 transform size=8x8. 63

Figure 39. Percentage of correctly predicted MB in different sequences for
every frame. ......................................................................................... 74

Figure 40. R-D performances for the proposed and reference cascade
transcoders. ......................................................................................... 83

Figure 41. PSNR with/without using transcoding. ........................................ 85

Figure 42. The 5th frame of the video sequence using H264 to encode
it (first column) and using the transcoding algorithm described in
this thesis (second column) .................................................................. 87

Figure 43. Proposed transcoder architecture. Focus is on reducing
complexity while maintaining the PSNR/bitrate performance
using pixel and transform domain transcoder ........................................ 89

Figure 44. Down sampling an 8x8 block in VC-1 to a 4x4 block
in H.264 in the transform domain ....................................................... 90

Figure 45. Result of using downsampling by four on an image ................. 91

Figure 46. Up sampling an 8x8 block in VC-1 to a 4x4 block in H.264 in
the transform domain ......................................................................... 92

Figure 47. Result of using upsampling by 4 on an image ......................... 93

Figure 48. Super resolution example .................................................... 95

Figure 49. Video quality enhancement using irregular sampling ........... 98

Figure 50. Magnitude response \( N=2^w-1=19 \) (2 sides) ...................... 100
**Figure 51.** Magnitude response in 3-D .............................................. 101

**Figure 52.** Data flow diagram ........................................................ 103

**Figure 53.** Hardware block diagram ................................................ 103

**Figure 54.** Video quality enhancement using irregular sampling for

“BeforeMan” .................................................................................. 109

**Figure 55.** Video quality enhancement using irregular sampling for

“Car” .......................................................................................... 112

**Figure 56.** Video quality enhancement using irregular sampling for

“Train” ......................................................................................... 115

**Figure 57.** Frame layer rate control .................................................. 119

**Figure 58.** QP vs. bit-rate for VC-1 and H.264 ................................. 121

**Figure 59.** $Q_{VC-1}$ vs. $Q_{H.264}$ and second-degree polynomial .... 121

**Figure 60.** QP vs. bit-rate for VC-1 (using low rate tool) and H.264 ... 122

**Figure 61.** $Q_{VC-1}$ vs. $Q_{H.264}$ and second-degree polynomial ....... 123

**Figure 62.** Flow chart of rate control algorithm ............................... 126

**Figure 63.** The number of bits obtained by the proposed and reference
cascade algorithm........................................................................... 131
LIST OF TABLES

Table I. Brief summary of VC-1 and H.264 features related to transcoding .......... 22

Table II. Percentages of different VC-1 transform sizes chosen as different

H.264 MC Block Sizes by H.264 encoder for “Claire” at QP=28 .......... 30

Table III. Threshold values ............................................................. 69

Table IV. VC-1 encoder configuration .................................................. 70

Table V. H.264 encoder configuration .................................................. 71

Table VI. Percentages of correctly predicted MBs ................................. 73

Table VII. PSNR (dB)/Bitrate (kbits/sec) for I-P-frames with respect

to cascade (Baseline) ................................................................. 75

Table VIII. PSNR (dB)/Bitrate (kbits/sec) for I-P-B-frames with respect

to cascade (Main) ........................................................................ 75

Table IX. PSNR(dB)/Bitrate(kbits/sec) for transform transcoder

with respect to cascade (Baseline) .................................................. 78

Table X. Time taken by transcoder ..................................................... 86

Table XI. PSNR (dB) improvements .................................................... 105

Table XII. Computational time complexity ........................................... 106

Table XIII. Comparisons with other SR methods on PSNR (dB) ............... 107

Table XIV. Transcoding experimental results ....................................... 127

Table XV. Rate-control transcoding time units ...................................... 128

Table XVI. Transcoding bit-rate and PSNR results .................................. 129

Table XVII. Rate control transcoding time units .................................... 129
Chapter 1

Introduction

1.1 Background

There has been an extensive increase in applications requiring multimedia capabilities, from mobile phones through Internet video to high definition broadcast television (HDTV). The high definition Blu-ray disc format has mandated MPEG-2, H.264, and VC-1 as video compression formats. The VC-1 video compression standard [1] developed by Microsoft has been standardized by SMPTE and adopted by the DVD Forum for high definition DVDs. VC-1 is also expected to be deployed as a key engine in satellite TV, IP set-top boxes and high-definition DVD recorders. H.264 [2] standard, which is developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG), is being adopted for a broad range of video applications including digital TV and mobile video services. To adapt to the different requirements of each video application and the devices that play them, transcoders are key. Transcoders convert a video stream from one format to another. There are several video standards in the market. This work focuses on two of the standards, VC-1 and H.264, but many of the features presented here can be reused for
transcoding between other video formats. VC-1 and H.264 were selected for several reasons. First, because they are both mandatory in Blu-ray DVD and therefore the industry has to support both and is interested in their transcoding; second, because VC-1 has been the subject of academic studies far less often than other standards (i.e. MPEG-2); and last, because VC-1 is a less complex standard than H.264 and transcoding from a less complex standard to a more complex one is a more interesting academic problem than the reverse. Transcoding may involve two codecs, changing input bitstream to one codec to an output bitstream for another codec, or just one codec in which a different output resolution than the original one is needed (figure 1). For example, when displaying video on mobile devices with low display capabilities, reducing the spatial resolution of the original bitstream is necessary (i.e. down-sampling). On the other hand, some applications require to display Standard Definition TV (SDTV) bitstreams on High Definition (HD) devices or to display low quality videos generated by mobile (for example, YouTube videos) devices on high quality displays, increasing spatial resolution of original bitstream is thus necessary (i.e. up-sampling), which implies that a higher resolution frame has to be obtained by up sampling from its low resolution version. The addition of high frequency information in such cases is a well-known ill-posed problem as it increases the noise. In this dissertation we first find an algorithm to perform transcoding between the two standards and then solve the inherited problems of transcoding when used for resolution conversion, quality enhancement or rate control. Figure 2 presents a general transcoding scenario.
1.2 Related Work

A formal description of the standards decoders can be found in [1] for VC-1 and [2] for H.264. A more general overview of the standards can be found in [39, 40] for VC-1 and [41] for H.264. Two main algorithms are used for transcoding [3] – pixel domain transcoding and transform domain transcoding. The pixel domain transcoder is composed of a decoder and an encoder, in which the input video is fully decoded, followed by an accelerated encoding stage that re-uses the information gathered during decoding. Figure 3 shows the general architecture of a VC-1 to H.264 transcoder with a VC-1 decoding
stage and a H.264 encoding stage (cascade pixel domain transcoder). In contrast, the
transform domain transcoder partially decodes the bit stream to the transform coefficients
and converts these transform coefficients from one standard to the other using matrix
multiplication techniques (figure 4). Transcoding tools and algorithms have been
proposed to transcode video from H.263 [4], MPEG-4 [5, 67, and 64], and MPEG-2 [6, 7,
8, 9, 10, and 74] to the H.264 format. An outline of technical issues and research results
related to transcoding can be found in [66], this article also discusses techniques for
reducing complexity and techniques for improving video quality. A general performance
evaluation specifically for H.264 transcoding algorithms is explained in [71] and a mixed
requantization technique is also proposed. In work [4] a fast cascaded pixel-domain
transcoder from H.263 to H.264 for both intra- and inter-frame coding is proposed. In [5]
a temporal resolution reduction transcoding method that transforms an MPEG-4 video
bitstream into an H.264 video bitstream is proposed. The block mode statistics and
motion vectors in the MPEG-4 bitstream are utilized in the H.264 encoder for block
recording system that employs MPEG-2 to H.264/AVC transcoding to achieve efficient
storage of broadcast streams. In [7] a macroblock (MB) mode estimation technique for
MPEG-2 to H.264 intra frame video transcoding is presented. The estimated mode is
used to constrain the MB prediction mode computation just to the estimated MB mode. In
[8] a novel technique for transcoding MPEG-2 to H.264 based in machine learning is
introduced. Work done in [9] proposes a motion vector and coding mode re-use process
for MPEG-2 to H.264 main profile transcoding. Article [10] investigates the issues on
transcoding MPEG-2 into H.264 in transform domain with consideration of drift error
due to the mismatch of motion compensation, and proposes a transform domain solution to transcode MPEG-2 into H.264. In [63] a transcoding algorithm from H.264 to MPEG-4 based on the block occurrence probability with motion vector (MV) refinement determining the optimal search window is presented. In [64] the algorithm proposed in [63] is extended to also perform resolution conversion. In [65] the reverse algorithm (MPEG-4 to H.264) to [63] is presented using four types of MV interpolation methods to avoid performing brute-force motion estimation again in H.264. In [68], a pixel-domain transcoding method is proposed for conversion of H.264/Advanced video coding (AVC) baseline profile and MPEG-4 visual simple profile (VSP), DCT-domain transcoding is not employed in this case, due to the nonlinear loop filtering included in the H.264/AVC standard. For MPEG-2 to H.264 transcoding, [69] presents an intra frame video transcoder. The DCT coefficients gathered from the MPEG-2 process are used to estimate the directional features in a picture. The estimated directional features are then used to compute the intra prediction modes in the H.264 encoding stage. [70] shows that just reusing MVs in transcoding is non-optimal and they propose a fast-search adaptive motion vector refinement, the case of frame rate conversion (with dropped frames) is also analyzed and a motion vector composition method to compose a MV from the incoming MV is proposed.

In [11], an algorithm is developed to perform transcoding with spatial resolution downscaling from MPEG-2 to VC-1. In [12], a pixel domain transcoding algorithm from VC-1 to H.264 is presented, and the transform domain transcoder is presented in [13]. There are several works that address the fundamental problem of converting MPEG discrete cosine transform (DCT) coefficients to H.264 transform coefficients entirely in
the transform domain by using matrix multiplication [14, 15 and 16]. In [14] a general method for converting 8-tap DCT coefficients to 4-tap integer transform coefficients is presented. In [15] a fast coefficient conversion method for transform domain MPEG-2 to H.264 is developed and tested. Article [16] presents an algorithm to convert DCT coefficients to H.264 integer transform coefficients. The topic of transcoding interlaced video has also been studied in several works. In [17], a fast DV (interlaced) to MPEG-4 (progressive) downscaling transcoder is implemented. In [18], an HDTV to SDTV spatial transcoder is implemented considering that both input and output bit streams are interlaced. The advanced/high profiles for video standards are used mainly for high quality requirement applications such as broadcast TV, HDTV, and Blu-ray disc storage.

The problem of the deblocking filter in transcoding has been studied in previous works. In [42] a description of a programmable VC-1 de-blocking filter architecture with capabilities to support different standards is presented. The architecture has been modeled, simulated and implemented in hardware description language (Register Transfer Level). Results show a threefold performance improvement as compared to solutions where filtering algorithms are otherwise not hardwired. Results also point to parallelism based on existing data flow and show that real-time requirements can be met. Paper [43] describes the adaptive deblocking filter used in the H.264/MPEG-4 AVC video coding standard. The filter performs simple operations to detect and analyze artifacts on coded block boundaries and attenuates those by applying a selected filter. Work done in [44] analyzes the problem in which DCT coefficients are subject to a requantization process (this happens during transform domain transcoding), which may render the deblocking
ineffective. The paper proposes to consider deblocking in conjunction with the requantization process therefore accelerating the transcoding processes.

To be able to use the transcoder for resolution conversion we need first to find an efficient method for resizing. Transcoding tools and algorithms with resizing in the DCT-domain have been proposed for 2D images and video [19, 20]. Transcoding and up/down sampling usually results in lower output quality and hence we propose a quality enhancement of the output bitstream. The key idea for our video quality enhancement is using irregular sampling techniques. Irregular Sampling has been used for 2D images quality enhancement [21-25] but it has not been used in video quality enhancement. There are several articles that explain in depth irregular sampling (IS) theory [21, 22], and in particular how to use of IS for static image quality enhancement [23-25]. The problem of video resolution enhancement is currently of great importance due to the emergence of high definition (HD) displays, a good technical overview of super resolution (SR) can be found in [35, 36]. Many video SR algorithms borrow from still image SR solutions, but video processing presents several complications, different objects move at different speed in the scene, movement of objects can result in parts of the scene being cover/uncovered and the need for faster processing. [53] examined the LR image acquisition process as a whole, the method assumes affine motion in the sequence and considers the point spread function to be known. [37] proposed an example based image magnification method using a large database to add high frequency information. For scalable video coding and distributed video coding, some frames (key frames) are sent at a high resolution (HR), [54] uses this key frames to super-resolve the low resolution (LR) non-key frames, the motion estimation process in performed using
blocks of band-pass versions of the frames, rather than low pass ones. [55] uses an adaptive framework where the target frame is divided into adaptive-sized blocks, which are classified into categories by their features, and then different conventional SR algorithms are applied to the blocks in different categories, a de-block process is also applied to reduce the block edge effects. In [56] a solution is presented where a relatively simple block-processing linear minimum mean squared error (LMMSE) spatial-domain interpolation is used; the solution is similar to previous approaches for still-image scenarios but with modifications to improve performance and tackle some complications that arise specifically for the case of video. In [52], an iterative and temporally recursive technique is used to improve the resolution of a video sequence.

To use the transcoder for rate control we need to find good compatibilities among the rate control tools used by the video standards being transcoded. Standards usually only specify the decoder side of video compression, the implementation of the encoder is left out of the standard to allow flexibility and optimizations by third-party. But since the efficiency of the decoder greatly depends on the encoder implementation, standard organizations usually issue non-normative guidance to aid in implementation for the encoder. As part of this guide the Join Video Team (JVT) standard organization adopted a rate control algorithm described in [26] for H.264, other rate control algorithms adopted for different standards are TMN8 [27] for H.263+, and VM8 [28], one of the rate control algorithms adopted by MPEG-4. An overview of rate constrained coder and comparation of video coding standards can be found in [51]. The rate control scheme for transcoding is different from that for video encoder since the information obtained from the encoder-decoder part of the input bitstream can be reused to accelerate the rate control
performance of the transcoded output bitstream. There are several works about rate-control transcoding between different transcoders H.263 to H.264 [29] and MPEG-2 to H.264 [30, 31]. In [29] a low-complexity rate control scheme for transcoding from H.263 to H.264/AVC is proposed. The relationship between the rate of the pre-coded video and both the rate and distortion of the transcoded video are studied. A row-layer bit allocation is introduced to perform average rate shaping across a row of MBs. Estimation error diffusion is also explained. Article [30] proposes a new frequency domain complexity estimation scheme and an algorithm for adaptive bit allocation during transcoding. In conjunction with frame skipping mechanism, the proposed algorithm adaptively determines spatial coding parameters to realize very low target bitrate for MPEG-2 to MPEG-4 transcoding. In [31] an efficient rate control method for MPEG-2 to H.264/AVC transcoding is proposed. Specifically, the values of quantization parameters for I and B frames are computed adaptively based on the side information from the precoded video. Article [32] presents an efficient rate control scheme for the H.264/AVC video coding in low-delay environments and proposes an enhancement to the buffer-status based H.264/AVC bit allocation method. Paper [33] presents a frame-layer rate control for H.264/AVC that computes the Lagrange multiplier ($\lambda_{\text{MODE}}$) for mode decision by using a quantization parameter (QP) which may be different from that used for encoding. In order to obtain an accurate QP for a frame, a complexity-based bit-allocation scheme and a QP adjustment method is employed. Paper [34] aims to improve video distortion, due to high motions or scene changes, by more accurately predicting frame complexity using the statistics of previously encoded frames using mean absolute difference (MAD) ratio as a measure for global frame encoding complexity. In [74] a
frame layer adaptive rate control MPEG-2 to H.264 is proposed. In [75] a macroblock layer rate control transcoder from MPEG-2 to AVS (a video coding standard developed in China) is explained. Given the existence of channel errors that can easily corrupt video quality, there is also the need to make the bitstream more resilient to transmission errors. In [67] an overview of the error resilience tools found in today’s video coding standards and a description of a variety of techniques that may be used to achieve error resilient video transcoding is provided.

![Figure 3. Reference cascade pixel-domain transcoder for VC-1 to H.264](image)

**Figure 3.** Reference cascade pixel-domain transcoder for VC-1 to H.264

![Figure 4. Transform-domain transcoder for VC-1 to H.264](image)

**Figure 4.** Transform-domain transcoder for VC-1 to H.264
1.3 Thesis Outline

1.3.1 Format Conversion Transcoding and transcoding from VC-1 to H.264 Baseline and High Profiles.

In this dissertation, we first analyze the problem of transcoding VC-1 to H.264 in the pixel domain (figure 3) and in the transform domain (figures 4, and 5), presenting both solutions. For the pixel domain transcoder in H.264, mode decisions for intra macroblock (MB) and inter MB are computationally intensive since each of the different modes has to be checked to select the best coding mode. The key idea is to propose an early termination algorithm of the H.264 encoding based on incoming VC-1 encoded bitstreams. The proposed approach uses the variable transform sizes used in VC-1 to determine the sizes of the blocks for motion estimation in H.264. With this approach, instead of evaluating all the possible block sizes for motion estimation, the transcoder evaluates a single block size determined based on the transform size used in VC-1. For the transform domain transcoder, the proposed approach is based on matrix multiplication in the transform domain. On top of that, a fast implementation of the algorithm proposed is also studied in this dissertation. Transform domain usually results in drift error propagation; this dissertation also analyzes and compensates the drift specific error for VC-1 to H.264 transcoder (figure 5). The two approaches are compared and a performance optimized VC-1 to H.264 transcoder is proposed. The block diagram of the proposed transcoder is presented in figure 6.
1.3.2 Resolution Conversion Transcoding (Up/Down Sampling)

This dissertation presents an algorithm to perform resolution conversion transcoding from VC-1 to H.264. For spatial downscaling the DCT decimation scheme has been proven to achieve significantly better visual quality compared to the schemes using pixel domain low pass filtering followed by down sampling. On the other hand, some applications
require to display SD bitstreams on HD devices or to display low quality videos generated by mobile devices in high quality displays, increasing the spatial resolution of the original bitstream is thus necessary (e.g. up sampling), which implies that a higher resolution frame has to be obtained by up sampling from its low resolution version. To study the problem we first apply a down/up sampling algorithm for VC-1 to H.264 transcoding. We then analyze the problem of quality enhancement in transcoding VC-1 to H.264 when down/up sampling is used. To transcode I-frames from VC-1 to H.264 we use the pixel domain transcoder described in Chapter 3 changed to perform down/up sampling (figure 7). To transcode P/B frames the transform domain transcoder described in chapter 4 changed to perform down/up sampling is used. After transcoding with resolution conversion we perform quality enhancement. The problem of video resolution enhancement is currently of great importance due to the emergence of high definition (HD) displays and the need for synthetic zooming of region of interest for surveillance, medical, and satellite imaging. A good technical overview of super-resolution (SR) can be found in [35, 36, and 37]. The addition of high frequency information is a well-known ill-posed problem as it increases the noise. This high frequency additional information may be obtained from a group of several shifted versions of the low resolution images, a collection of optimal estimated filters selected for specific image content, or from a training set that combines low and high resolution image pairs [35]. Many video SR algorithms borrow from still image SR solutions, but video processing presents several complications. Different objects move at different speed in the scene; movement of objects can result in parts of the scene being covered or uncovered and could be complicated by the need for faster processing. In this dissertation, a novel super-
resolution algorithm based on irregular sampling (IS) is presented. The high-resolution (HR) frame is obtained as an interpolation of one or more previous frames; the resulting interpolated frame has samples non-uniformly spaced in the areas where movement happened. To reconstruct the irregular sampled frame we use well-known irregular sampling algorithms (modified to perform in 2D space). There are several articles that explain irregular sampling theory in depth [21-23], and in particular how to use IS for static image quality enhancement [24, 25]. As part of the proposed SR algorithm we need motion vectors (MVs) that describe the trajectory of objects in the scene. Block-based motion estimation proposed by video compression standards [1, 2] are not very suitable for our proposed algorithm since the MVs may not represent the actual motion. Finding true motion vectors is beyond the scope of this work, any true-MV algorithm could work with our proposed SR method. For the proof of concept we use the true MV algorithm described in [38] since it is very efficient and hardware friendly.

1.3.3 Rate Control Transcoding

The rate control scheme for transcoding is different from that for video encoder since the information obtained from the encoder-decoder part of the input bitstream can be reused to accelerate the rate control performance of the transcoded output bitstream. For this work we use the pixel domain transcoder described in section 3 to transcode I-frames, and the transform domain transcoder described in section 4 to transcode P/B frames, from VC-1 to H.264. Figure 8 represents the transcoder used. The main idea is to reuse the rate control information generated by VC-1 encoder to accelerate the rate control module in H.264. We develop two algorithms. A low complexity algorithm whose main idea is to find a mapping between QP in VC-1 and QP in H.264 is developed. The low complexity
rate control algorithm has the limitation that only works for QP_{H.264} ranges of 10 to 29. To increase the range we implement a more complex rate-control (RC) transcoder that does map the complexity estimators (MAD). There are several problems that we have to solve to implement it; for the transform domain transcoder, used for the P/B frames, where motion estimation (ME) is not used, it is difficult to get MAD values, therefore to estimate the complexity for the frame we use the sum of absolute transform differences SATD calculated using the sum of the absolute values of the frequency transform of the residual. SATD predicts more accurately quality from the standpoint of objective and subjective metrics, but is more complex to calculate. For pixel domain transcoding, used here for I frames, we use MAD for its simplicity of calculation. Results show that by using a combination of the low and median complexity rate control tools we can successfully transcode the entire QP range without significant loss in video quality.
Figure 7. VC-1 to H.264 pixel domain transcoder with resolution conversion
Figure 8. VC-1 to H.264 rate control transcoder
Chapter 2

Compression Tools for VC-1

2.1 Compression Tools for VC-1 Progressive

This section provides a brief overview of VC-1 with an emphasis on the features that impact transcoding. Like most video coding standards, VC-1 is based on the principles of hybrid video coding -- motion compensated transform coding. VC-1 has five picture types: I, P, B, BI and skipped P. Similar to MPEG standards, an I frame has all MBs that are intra coded, the P frame has MBs that are intra or predicted from previous frames, and the B MBs are bi-directionally predicted The BI frames are special B frames with only intra MBs and can be used in places of B frames, when the B frame coding is inefficient. If there are big continuous scene changes and B frames cannot capture any similarity from two reference frames, BI can be used to improve coding performance. Since BI does not have the overhead for motion compensation, the syntax is optimized for such scenarios. A BI frame is not used as a reference frame. Skipped P frame is signaled when the frame is exactly the same as the previous reference and can be decoded by copying previous reference frames. Intra MBs in VC-1 do not use prediction in the pixel domain and use a fixed size transform of size 8x8. The inter-coded MBs (such as those in P/ B
frames) can use up to four different transform sizes for the residuals: 8x8, 4x8, 8x4, and 4x4. Transform block size can change adaptively in P/B frames with four different size options, while the size of motion compensation is either 16x16 or 8x8 in VC-1 (figure 9). A good overview of VC-1 can be found in [39, 40]. VC-1 has three profiles. The main differences among them are that for Simple profile, where there is no loop-filter, no intensity compensation, no B frames, and no interlacing. For Main profiles, there is no interlacing. Advanced profile uses all features in VC-1. In this chapter our discussion focuses on transcoding from VC-1 Simple profile to H.264 Baseline profile, VC-1 Main profile to H.264 Main profile, and VC-1 Advanced profile to H.264 High profile. A high level comparison of VC-1 and H.264 features is shown in Table I.

![Figure 9. VC-1 MB types for MC](image)

### 2.2 Compression Tools for H.264

H.264 syntax supports I, P, and B frames. A macroblock can be coded as intra or inter MB mode. There are two classes of intra MBs 16x16 and 4x4. In contrast with previous standards where only some of the DCT coefficients can be predicted from neighboring intra MB, in H.264 prediction is always used in the spatial domain by referring to neighboring samples of already coded blocks. For inter coded MBs H.264 has nine different motion compensation modes ranging from 4x4 to 16x16 (figure 10). In H.264
MC is performed with quarter-pixel accurate MV. B frames utilize two different reference picture buffers and four different types of predictions are supported: list 0, list 1, bipredictive, and direct. In direct prediction no delta motion vector is transmitted, furthermore there are two methods for obtaining prediction signal in B frames, temporal and spatial direct prediction. For the quantization of transform coefficients, H.264 uses scalar quantization without dead-zone; and one of the 52 quantizers can be selected for each MB by the quantization parameter. Two methods of entropy coding are supported. The default entropy coding methods uses a single infinite-extended codeword set for all syntax elements but for residual data. It is called context adaptive variable length coding (CAVLC). The efficiency of the entropy coding can be improved if the context adaptive binary arithmetic coding (CABAC) is used. For removing block edge artifacts, H.264 included a deblocking filter, which is applied inside of the motion prediction loop. In H.264, three profiles are defined. The baseline profile includes all described features except B-frames, CABAC, and interlace coding tools. Since the main target application area of Baseline profile is the interactive transmission of video, it is used for videoconferencing applications. In the comparison for video streaming and entertainment applications, which allow a larger delay, the Main profile of H.264 is used. The Main profile adds support for B-frames, CABAC entropy coding method, as well as the interlaced coding tools. A more detailed explanation of the H.264 and VC-1 standards parts affected by the proposed transcoding algorithm will be given in each of the individual sections describing the transcoder.
Figure 10. H.264 MB types for MC
**TABLE I.**
**BRIEF SUMMARY OF VC-1 AND H.264 FEATURES RELATED TO TRANSCODING**

<table>
<thead>
<tr>
<th></th>
<th>VC-1</th>
<th>H.264</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simple</td>
<td>Advanced Profiles</td>
</tr>
<tr>
<td>Picture type</td>
<td>I, P, Skip</td>
<td>I, P, Skip, B, BI</td>
</tr>
<tr>
<td>Transform size</td>
<td>Adaptive: 4x4, 4x8, 8x4, 8x8</td>
<td>Fixed: 4x4</td>
</tr>
<tr>
<td>Transform</td>
<td>Integer, similar to DCT</td>
<td>Integer, similar to DCT</td>
</tr>
<tr>
<td>Intra prediction</td>
<td>AC Coefficients (transform domain)</td>
<td>Directional predictors (spatial domain)</td>
</tr>
<tr>
<td>Motion comp. frames</td>
<td>16x16, 8x8</td>
<td>7 variable block sizes</td>
</tr>
<tr>
<td>Reference frames</td>
<td>Max 2</td>
<td>Max 16 (each dir.)</td>
</tr>
<tr>
<td>YUV Format</td>
<td>4:2:0</td>
<td>4:2:0</td>
</tr>
<tr>
<td>Overlap Smoothing</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>In-Loop deblocking</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Intensity Compensation</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Skipped MB</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Range Reduction</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>Quantizer Range</td>
<td>1-21</td>
<td>1-51</td>
</tr>
<tr>
<td>Quantizer</td>
<td>Uniform/non-uniform</td>
<td>Uniform</td>
</tr>
<tr>
<td>Interlace</td>
<td>Fame/Field MB level</td>
<td>Fame/Field Super MB level</td>
</tr>
</tbody>
</table>
Chapter 3

Pixel Domain Transcoder

The transcoding algorithm discussed in this section assumes full VC-1 decoding down to the pixel level, followed by a reduced complexity H.264 encoding. Figure 1 shows reference transcoder architecture. The data gathered during the VC-1 decoding stage is used to accelerate the H.264 encoding stage. YUV format in VC-1 is 4:2:0, therefore the transcoder output is also 4:2:0. To get other YUV format, quality enhancement techniques would have to be used. VC-1 does not have a fixed group of pictures (GOP) structure and H.264 [2] does not require it either, although rate control tools (not part of the standard) do require a repeating GOP structure. For experiments reported in this paper, the rate control tool is turned off and frame types in VC-1 are mapped to the same/similar type frames in H.264. BI frames are mapped to B frames since the positions of the BI frames are in B frames, and to benefit from the advantages BI frames present [19]. There is no skipped frame in H.264 but it has skipped MBs. In order to map skipped frames in VC-1, all MBs of a skipped frame are mapped to skipped MBs in H.264. Pixel domain design performs direct MB mode mapping with the same reference frames used in VC-1.
3.1 I-Frames

3.1.1 Intra Macroblocks

3.1.1.1 Intra MB Mode Mapping

An intra MB in the incoming VC-1 bitstream is transcoded as an H.264 intra MB. A VC-1 intra MB uses only 8x8 size transform. An H.264 Baseline intra MB can use either intra 4x4 or intra 16x16 for regions that require less details. Based on uniformity of the incoming texture, these VC-1 intra 8x8 block sizes are mapped to either intra 16x16 or intra 4x4 modes in H.264. We use an approximation of the variance of the 16x16 MB (four 8x8 intra blocks) as an estimate for the uniformity of the MBs. The variance of a 16x16 MB is strongly correlated with the variance of the 4 DC coefficients of its non-overlapping 8x8 blocks (figure 11). A variance threshold ($th_{intra}$) is experimentally determined as a function of the quantization parameter (QP). The variance thresholds used are largely content independent and the PSNR drop caused by this variance-based estimation is negligible [7]. The process can be described as follows.

\[
\begin{align*}
\text{calculate: } & \sigma = E[(DC_i - \mu)^2], \\
& \text{where } E[x] \text{ is the expected value of } x, \mu \text{ is the mean, and } DC_i \text{ is} \\
& \text{the DC coefficient of block } i(1..4) \\
\begin{cases}
\text{if } \sigma \leq th_{intra} & \text{Intra 16x16} \\
\text{else} & \text{Intra 4x4}
\end{cases}
\end{align*}
\]

In figure 12 we can see the results of the previously described algorithm, for blocks of size 16x16 the variance is very low and with a small range [0-10]. This graphs help determine the value of the threshold $th_{intra}$.
H.264 High profile also uses intra 8x8. The test that we performed on several video sequences (Logo, Foreman qcif, Claire qcif, Walk qcif and Foreman cif) shows that H.264 uses on average intra 8x8 block size around 50% of the time, while intra 4x4 is used 33% and intra 16x16, 17%. To be able to transcode to the High profile case, we also
need to map for the intra 8x8 by using two experimentally obtained thresholds ($th_{\text{intra}}$, $th_{\text{intra HP}}$).

3.1.1.2 Prediction Mode Computation for Intra MBs

H.264 allows intra-frame prediction. Prediction is always conducted in the spatial domain by referring to neighboring samples of already coded blocks.

![Figure 13](image1.png)

**Figure 13.** Possible prediction directions for H.264 intra 4x4 mode

![Figure 14](image2.png)

**Figure 14.** H.264 MC prediction for 16x16 MBs.

H.264 uses nine prediction modes for 4x4 intra blocks (figure 13), while for regions that require less details H.264 uses 16x16 blocks with only four prediction modes (figure 14). An example of how the prediction direction is used can be found in figure 15.
Since VC-1 transform is an approximation of the DCT, the directional features can be estimated in VC-1 using an approach similar to the one developed for MPEG-2 to H.264 transcoding [7].

The mapping from VC-1 intra blocks to H.264 intra blocks is done as follows:

\[
\begin{align*}
    d_x &= \sum_{u=1}^{7} |F(u,0)| \quad \text{and} \quad
d_y &= \sum_{v=1}^{7} |F(0,v)| \quad ,
\end{align*}
\]

where \(F(x,y)\) are the transform coefficients for the MB at row \(x\), column \(y\).

For VC-1, 8x8 intra blocks are mapped to H.264 4x4 intra blocks by

\[
\begin{cases}
    \text{if } dx \leq 1 \text{ and } dy \leq 1 & \text{Prediction mode = DC prediction} \\
    \text{since DC prediction is good for blocks with little or no local activity} \\
    \text{else} & \\
    \theta = \frac{180}{\pi} \arctan \left( \frac{\sum_{u=1}^{7} |F(u,0)|}{\sum_{v=1}^{7} |F(0,v)|} \right) \quad \text{and set Prediction mode = } i
\end{cases}
\]

\[\quad \text{for } |\theta - \text{dir}[i]| \leq 11.25\]
where $\text{dir}[i] = \{90, 0, 180, 45, -45, -67.5, -22.5, 67.5, 22.5\}$ the orientations allowed by H.264 for 4x4 intra blocks. Through experimentation we have determined the threshold (= 1) for $d_x$ and $d_y$.

For VC-1, 8x8 intra blocks are mapped to H.264 16x16 intra blocks by

$$\begin{array}{ll}
\text{if } dx \leq 1 \text{ and } dy \leq 1 & \text{ Prediction mode } = \text{DC prediction} \\
\text{else if } \left( \begin{array}{l}
\text{Center pixel is in same plane as the plane } \\
\text{formed by using three corners of the block.} (*)
\end{array} \right) & \text{ Prediction mode } = \text{Plane prediction} \\
\text{else if } dx > 0 \text{ and } dy \leq th_{\text{intraPred}} & \text{ Prediction mode } = \text{Horizontal prediction} \\
\text{else } dx \leq th_{\text{intraPred}} \text{ and } dy > 0 & \text{ Prediction mode } = \text{Vertical prediction}
\end{array}$$

where $th_{\text{intraPred}}$ is the value for the threshold is determined through extensive experimentation.

(*) Since Plane prediction works well for gently changing luminance, to generate the plane equation:
1. Produce the three corners of the block. $P=(0,0,z1)$, $Q=(0,N,z2)$, and $R=(N,0,z3)$.
2. Produce vectors $PQ=(0,N,z2-z1)$ and $PR=(N,0,z3-z1)$.
3. Calculate the cross product $n=PQ\times PR=(N(z3-z1),N(z2-z1),N^2)$.
4. Generate plane equation $N(z3-z1)x + N(z2-z1)y + N^2(z-z1)=0$.
5. Generate center point $c=(Xc,Yc,Zc)$, and is in the plane if $(z3-z1)*Xc+(z2-z1)*Yc+N*Zc=N*z1$.

In above steps N, z1, z2, and z3 are integer values.

3.1.2 Mixed Macroblock Mode Mapping

The inter MBs in the P/B pictures in VC-1 can have up to three 8x8 blocks coded as intra. This is a mixed mode intra case that does not exist in H.264. We map this VC-1 mixed mode MB to intra/inter MB in H.264 as follows:
if 16x16 Mixed MB in VC-1 has:

<table>
<thead>
<tr>
<th>a) 38x8 intra blocks and</th>
</tr>
</thead>
<tbody>
<tr>
<td>18x8 inter block :if (TS=8x8) ----&gt; 1 16x16 H.264 intra macroblock</td>
</tr>
<tr>
<td>else -----&gt; 16 4x4 H.264 intra blocks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b) 2 8x8 intra blocks and</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 8x8 inter block :if (TS=8x8) ----&gt; 1 16x16 H.264 intra macroblock</td>
</tr>
<tr>
<td>else -----&gt; 16 4x4 H.264 intra blocks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c) 3 8x8 intra blocks and</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 8x8 inter block :if (TS=8x8) ----&gt; 4 8x8 H.264 inter blocks</td>
</tr>
<tr>
<td>else -----&gt; 16 4x4 H.264 inter blocks</td>
</tr>
</tbody>
</table>

where TS is the transform block size. For example, a VC-1 MB with 3 8x8 intra blocks and one 8x8 inter block with transform size 8x8 will be mapped to 1 16x16 H.264 intra macro block.

### 3.2 P-Frames

#### 3.2.1 Inter Macroblock Mode Mapping

An inter-coded MB in the incoming VC-1 bitstream is coded as inter MB in H.264. Inter MBs are used in P and B frames, the particularities of B frames are studied in section 3.4. Which transform blocks used to code MC residuals in VC-1 indicates continuous regions and performing motion compensation with that block size is likely to find a better match and improve the prediction. As a proof of concept, we performed a simple experiment to check how well the VC-1 transform size (TS) relates to the H.264 motion compensation block size (BS). The results for sequence “Claire” at QP=28 are shown in Table II (encoder/decoder configuration as per Tables IV and V).


<table>
<thead>
<tr>
<th>VC-1 Block Size</th>
<th>H.264 Block Sizes</th>
<th>8x8</th>
<th>8x4</th>
<th>4x8</th>
<th>4x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>16x16</td>
<td>40.1</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>&lt;1</td>
</tr>
<tr>
<td>16x8</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>&lt;1</td>
</tr>
<tr>
<td>8x16</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>&lt;1</td>
</tr>
<tr>
<td>8x8</td>
<td>10.5</td>
<td>5.2</td>
<td>2.8</td>
<td>&lt;1</td>
<td></td>
</tr>
<tr>
<td>8x4</td>
<td>&lt;1</td>
<td>7.1</td>
<td>3.3</td>
<td>&lt;1</td>
<td></td>
</tr>
<tr>
<td>4x8</td>
<td>&lt;1</td>
<td>2.8</td>
<td>3.3</td>
<td>&lt;1</td>
<td></td>
</tr>
<tr>
<td>4x4</td>
<td>3.2</td>
<td>3.1</td>
<td>2.3</td>
<td>6.1</td>
<td></td>
</tr>
</tbody>
</table>

Motion blocks are usually larger than transform sizes; and to compensate for this we also use the block size of the VC-1 block as explained in the algorithm as follows. Inter MBs in VC-1 have two motion compensation modes (MCMs) – 1 motion vector (MV) mode and 4 MV mode. The 1 MV mode is usually selected in VC-1 for areas that are relatively uniform. We map these 1 MV MBs in VC-1 to inter block sizes 16x16, 16x8, or 8x16 in H.264. We use the VC-1 block TS as a measure of homogeneity in the block to be able to differentiate the three different block sizes. The 4 MV mode is usually selected in VC-1 for areas that have non-uniform motion. To determine a proper mode mapping for MBs, the 16x16, 16x8, and 8x16 modes are eliminated for such non-uniform MBs. The coding mode of a MB is then mapped to one of the other variable block sizes allowed for H.264 (4x4, 4x8, 8x4 and 8x8) motion compensation based on the transform.
size used in VC-1. To improve the performance of the algorithm, we make the mode mapping for inter MBs adaptively according to the quantization parameter (QP). We observed that at lower QP, H.264 maps a VC-1 8x8 inter block into H.264 inter 8x8 approximately 25% of the time, while at higher QPs this percentage is approximately 40%. The adaptive algorithm takes the previous observation into consideration and further enhances mapping of the inter MBs as follows. We set a threshold $th_{inter}$, obtained by experimentation.

If $Q P \leq th_{inter}$

$$\begin{aligned}
\text{If } MCM_{VC-1} = 1 MV \text{ (figure 16)} & \quad \text{ (figure 16)} \\
\begin{cases}
\text{If } TS = 8x8 & BS_{H.264} = 16x16 \\
\text{If } TS = 4x8 & BS_{H.264} = 8x16 \\
\text{If } TS = 8x4 & BS_{H.264} = 16x8 \\
\text{Else} & BS_{H.264} = 8x8 \\
\end{cases}
\end{aligned}$$

$$\begin{aligned}
\text{If } MCM_{VC-1} = 4 MV \text{ (figure 17)} & \quad \text{(figure 17)} \\
\begin{cases}
\text{If } TS = 8x8 & BS_{H.264} = 8x8 \\
\text{If } TS = 4x8 & BS_{H.264} = 4x8 \\
\text{If } TS = 8x4 & BS_{H.264} = 8x4 \\
\text{Else} & BS_{H.264} = 4x4 \\
\end{cases}
\end{aligned}$$

else

\begin{aligned}
\text{If } MCM_{VC-1} = 1 MV & \quad BS_{H.264} = 16x16 \\
\text{If } MCM_{VC-1} = 4 MV & \quad BS_{H.264} = 8x8 \\
\begin{cases}
\text{If } TS = 8x8, 4x8, \text{ or } 8x4 & BS_{H.264} = 8x8 \\
\text{Else} & BS_{H.264} = 4x4 \\
\end{cases}
\end{aligned}

Note that the proposed transcoder does not perform resolution conversion or bit rate scaling.
Figure 16. Inter MB mode transcoding with QP<\text{th} and VC-1 1-MV

Figure 17. Inter MB mode transcoding with QP<\text{th} and VC-1 4-MV
To further improve the performance of H.264, skipped MBs are also added to reduce the number of operations needed to code the video stream. Skipped MBs represent the case in which the motion vector can be extrapolated from its neighbors and the prediction error is zero. To achieve the same complexity reduction and to be able to map to skipped MBs in H.264, the following operations are performed. First, skipped frames in VC-1 are P frames that are identical to the reference frame; skipped frames can be treated conceptually as copying the reference frame. Therefore, if the frame in VC-1 is a skipped frame, all MBs in the frame are mapped to skipped MBs in H.264. Second, if the MB is a skipped MB in VC-1, the MB is coded as H.264 skipped. Third, the motion vectors for the neighboring MBs are inspected for VC-1 -- if they are the same and the prediction error is zero, the MB is mapped to skipped MB in H.264. The impact in the quality of reusing skipped MBs can be seen in figure 40 for the Logo sequence that contains many skipped MBs and is low bit rate.
3.2.2 MV Mapping

Once the MB coding mode is mapped, the next step is to determine the motion vectors for the MB (including skipped MB). VC-1 calculates MVs in a similar way as H.264 and we can reuse them as follows. For inter 16x16 and the VC-1 4 MV mode mapped to four 8x8 blocks in H.264, the motion vectors are used without any refinement. For other block sizes, the VC-1 motion vector is used as a seed and the motion vectors are classified in two different groups according to the MV magnitude (Equation 2) into low motion and high motion. If the vector is classified as low motion, using a threshold \( th_{MV} \) (determined experimentally), a refinement window of size 2x2 (multiplied by a factor \( f \) which depends on the sequence resolution) is selected; else the refinement window becomes 5x5 (also multiplied by factor \( f \)). The reference frames used in VC-1 are also selected as references in H.264.

\[
MV \text{ magnitude} = \sqrt{MV_X^2 + MV_Y^2} \tag{2}
\]

3.3 Overlap Smoothing Transform

In VC-1, overlap smoothing transform (OLT) is used to reduce blocking artifacts in intra data and used only for high quantization values. The application of OLT is signaled at the

![Figure 19. Vertical and horizontal edges representation for overlapping transform](image)
MB level. This allows for overlap smoothing to be switched on or off (high texture areas) selectively. The process requires that vertical edges be filtered first (a0, a1, b0, b1) followed by the horizontal edges (p0, p1, q1, q0) as shown in figure 19. In figure 19, pixels a0 and a1 are to the left and b1, b0 to the right of the vertical block edge. The pixels marked by p0, p1, q1, and q0 are horizontal edges. The filters applied to both vertical and horizontal stripes are given by a 4x4 matrix lapped inverse transform \( P_i \) [1]:

\[
\begin{pmatrix}
    y_0 \\
    y_1 \\
    y_2 \\
    y_3
\end{pmatrix} =
\begin{pmatrix}
    7 & 0 & 0 & 1 \\
    -1 & 7 & 1 & 1 \\
    1 & 1 & 7 & -1 \\
    1 & 0 & 0 & 7
\end{pmatrix}
\begin{pmatrix}
    x_0 \\
    x_1 \\
    x_2 \\
    x_3
\end{pmatrix} +
\begin{pmatrix}
    r_0 \\
    r_1 \\
    r_0 \\
    r_1
\end{pmatrix} \gg 3
\]

Where \( P_i = \frac{1}{8}
\begin{pmatrix}
    7 & 0 & 0 & 1 \\
    -1 & 7 & 1 & 1 \\
    1 & 1 & 7 & -1 \\
    1 & 0 & 0 & 7
\end{pmatrix} \]

The values \((x_0, x_1, x_3, x_4)\) correspond to the original pixels being filtered (a0, a1, b1, b2) for the vertical edges and \((p0, p1, q1, q0)\) for the horizontal edges. The rounding parameters \(r_0\) and \(r_1\) both take alternate values \((3, 4)\) or \((4, 3)\) depending on the indexed columns/rows being, respectively, even or odd. The matrix is invertible so the original data can be recovered. For transcoding if the block has OLT applied to it we have to first apply the inverse transform and then continue with the transcoding as described in section 3.1.

### 3.4 B-Frames

B-frames are not part of the Baseline profile, but for completeness of the transcoder we also propose a method to transcode B-frames. B-frames, in both standards, can use four
different motion compensation modes: forward (prediction using previous frame), backward (prediction using future frame), interpolated (in VC-1 an average and in H.264 a weighted average of previous and future frames), and direct (MVs are computed based on previously decoded frames). In VC-1 B-frames use 16x16 MBs, and there is no 4 MV motion compensation mode. B-frames cannot be used for prediction (figure 20). Direct mode takes advantage of the basic assumption that objects tend to move with relatively constant velocity which allows motion information for the current MB to be predicted using simple interpolation and therefore not having to encode additional data. To improve direct mode prediction VC-1 explicitly codes B frames temporal position relative to its two reference frames -- this generalizes the direct mode prediction model from constant velocity to variable velocity. MBs can be intra, inter, or skipped. In H.264, the difference between B and P frames is that in B frames some MBs may use a weighted average of two distinct motion-compensated predicted values (figure 21). MBs in B-frames can be coded using intra, inter, or direct mode. Direct mode can be temporal (uses data from previous frames) or spatial (uses data from neighboring MB and list1 MV). Statistical data performed in [41] for several video sequences, show that on average a large percentage of MBs in H.264 B frames are direct mode. All VC-1 B-frame direct and skipped MBs are mapped to temporal direct and B slice skipped MBs in H.264. To map to spatial direct mode, we use as an approximation the same method described for skipped MB. The motion vectors for the neighboring MBs are inspected for VC-1. If they are the same and the prediction error is zero, the MB is mapped to spatial direct in H.264.
3.5 Loop Filtering

In figure 22 [42] we can see the effects of using OLT together with deblocking filter, the noise is much lower when using both filters at the same time. And probably this is the reason why VC-1 performs so well on subjective quality tests, and all standards describe a deblocking filter as part of the decoder.
Figure 22. Effect of overlap transform and deblocking filtering (Foreman QCIF)

Difference images d) and e) highlight the edges improved by the process.

For Main and Advanced profiles in VC-1, in-loop filter is used. It consists of checking if the discontinuity among two adjacent blocks is above a threshold, and if so, averages the pixels. As shown in figure 23, the 8-pixel pair on the vertical direction is
filtered at all 8x8 block boundaries. They are divided into two 4-pixel segments with the third pixel pair filtered. The pixels are categorized into three groups each of which is composed of four pixels in figure 24. For each segment, high frequency activity measure is applied. For example, the activity measure \( a0 = (2 \times (p3 - p6) - 5 \times (p4-p5) + 4) \gg 3 \) is extracted in the center area. The same measure is used to extract high frequency activities in the left and right sides. The blocky effect is ignored either when PQUANT is smaller than the center measure or when the activity measure of the center area is the smallest one. Otherwise, filtering is performed. The filtering operation is nothing but a correction on \( p4 \) and \( p5 \) values with \( p4=p4-d \) and \( p5=p5+d \), where \( d \) is a signed clipped value of the value \( 5 \times ((\text{sign}(a0) \times a3)-a0)/8 \) [2]. On the horizontal direction, filtering operation is performed on the 3rd pixels pair shown figure 24.

H.264 uses an in-loop adaptive deblocking filter [43] that selects the value of the threshold according to the quantizer step used and checks for discontinuities in the adjacent blocks. It then determines if the edge should be left untouched or should be filtered.

To avoid drift errors by the deblinking filters, we turn off (on encoder side) the deblinking filter in VC-1 and use in H.264 in order to reduce blocking artifacts in the end result. For P/B frames where pixel information is not available, the algorithm described in [44, 45] can be used, but this is outside of the scope of our paper.
3.6 Intensity Compensation

For Main and Advanced profiles in VC-1, intensity compensation (IC) can be used. For video sequences with global illumination changes (fading), motion compensation algorithm may not find a good predictor and the entire frame may end up being coded as an intra frame. To avoid, this VC-1 detects global illumination changes prior to motion compensation. If detected, the encoder computes two illumination parameters, scale ($S_V$) and offset ($O_V$), which are then used to calculate the luminance and chrominance look up tables used to remap the reference frame pixels. H.264 Main and Extended profiles
implement a weighted prediction (WP) tool that allows the encoder to specify the use of a scaling \((S_H)\) and an offset \((O_H)\) when performing motion compensation. It provides a significant benefit in performance in special cases such as coding fades. By comparing the equations for \(S_V\) and \(O_V\) in VC-1 [1] (*) to the equations for \(S_H\) and \(O_H\) in H.264 [2] using explicit mode weighted prediction, we conclude that by setting the log weight denominator rounding factor \(\text{LogWD}\) to 6 we can reuse \(S_V = S_H\) and \(O_V = O_H\).

* From VC-1 Intensity Compensation algorithm:

LUMSCALE and LUMSHIFT:

\[
\text{iScale} = \text{LUMSCALE} + 32 \\
\text{if} \ (\text{LUMSHIFT} > 31) \\
\quad \text{iShift} = \text{LUMSHIFT} \times 64 - 64 \times 64 \\
\text{else} \\
\quad \text{iShift} = \text{LUMSHIFT} \times 64 \\
\text{// for rounding} \\
\quad (\text{iScale} \times \text{i} + \text{iShift} + 32) >> 6 \text{ // rounding by } 64
\]

From H.264 Weighted Sample Prediction Process:

If \((\text{logWD} \geq 1)\)

\[
\text{prefPart}[x,y] = \text{Clip1}((\text{prefPartL0}[x,y] \times \text{w}_0 + 2^{\logWD-1}) >> \text{logWD}) + \text{O}_0
\]

else

\[
\text{prefPart}[x,y] = \text{Clip1}(\text{predPartL0}[x,y] \times \text{w}_0 + \text{O}_0)
\]

Then

\[
\text{iScale} = \text{w}_0 \\
\text{iShift} = \text{O}_0
\]
3.7 High Profile Features

3.7.1 Adaptive Size Transform for H.264 High Profile

VC-1 uses an adaptive block size integer transform (8x8, 8x4, 4x8 and 4x4) implemented in 16 bits [1]. H.264 High profile uses adaptive size (4x4, 8x8) transform implemented in 16-bit arithmetic [2]. Although we can assume that VC-1 uses the R-D model to select which block transform to use, we cannot map the block size directly from VC-1 to H.264; it is because VC-1 uses 4x8 and 8x4 block transforms that do not exist in H.264. To map these two block sizes, we need some measures of the homogeneity of the blocks. To estimate the block homogeneity, we only use the six coefficients next to the DC coefficient following the zig-zag scan pattern used by the transform (figure 25) since we know that coefficients closer to the DC carry more information. The flow chart in figure 26 explains the process.

![Figure 25. Transform size mapping](image-url)

\[ SAD = \sum_{i=1}^{6} |X_i - Y_i| \]
3.7.2 Interlacing

In interlaced video, each frame contains data from two different time instants, where all the even lines are from one time instant and all the odd lines are from another time instant. In VC-1 there are two options, Field Picture Coding (LPC) and Frame Picture Coding (FPC) (figure 27). In LPC the two fields that make up the frame are separately coded. In FPC both fields in an interlaced frame are coded jointly. Each MB contains samples from two time instances. For intra MB the encoder has the option to reorder the luminance portion according to fields to try to increase the spatial correlation prior to transform coding (figures 28 and 29).
Figure 27. Interlace picture coding mode supported by VC-1 4:2:0 luma and chroma temporal and vertical sample positions (where from left to right is shown a top field, bottom field, top field, and bottom field).

Figure 28. VC-1 field picture reference.
Figure 29. VC-1 luminance macroblock structure for frame picture.

An inter MB may be coded in two ways, one without regards to the field structure; second, the fields in the MB are compensated separately using 2 field motion vectors (MVs) or 4 field MVs. After motion compensation (MC), the residuals can be re-ordered in the same way as the intra MB coding prior to transform. In H.264 there are three options for interlaced frames. First, each field may be coded separately (PAFF). Second, the two fields may be coded together as a frame. And third, when the frame is selected as frame coded, then each pair of vertically adjacent MBs can again be coded together or as two separated field MBs using adaptive frame-field (MBAFF) mode (figure 30). When MBAFF is used, instead of splitting up a 16x16 MB into two 16x8 blocks, H.264 considers a super MB (32x16) consisting of two vertically adjacent MBs. The super MB is then separated into two 16x16 MBs, one for each field, and coded separately. The decision of which mode to use is made based on R-D criteria, which makes it very computationally complex. Several articles propose speed-ups for this decision without significant reduction of the video quality [46]. In our case we re-use some information from the decoded VC-1 bit stream to speedup this decision. We map by default field MB
in VC-1 to field MB in H.264. To improve the mapping, two measures are being used. First, the number of decoded neighboring (if MB has enough neighbors) MBs that are field/frame coded is determined; if the majority of the neighbors of the MB are field coded, then code the actual MB as field, else code it as progressive or interlaced according to the next criterion using MVs, the process is described in figure 31. The neighboring MBs criterion measure is used since there is a strong correlation between neighbors’ MB field/frame code and the selection made by H.264 (figure 32). Second, since MBs with large motion vectors are the ones that benefit the most from field coding, we measure motion as the magnitude of MV (with coordinates $X$ and $Y$) in the MB as in Equation (2). The algorithm in figure 31 is repeated for the two MBs of an H.264 MBAFF pair. If both of them result in field MBs, the pair is coded in field mode, else as frame MBs.

If the MV magnitude is above a threshold ($th_{MV}$), then the MB is mapped to field mode, else the MB is mapped to frame mode. Since I-frames do not contain motion information, we only use the neighbor majority rule on them to decide field/frame mode.
Figure 30. H.264 PAFF and MBAFF field coding

Figure 31. Flowchart for interlaced video transcoding.
Figure 32. Distribution of the number of neighboring field-encoded MBs when the “brute force” H.264 method chooses frame or field mode for P-slice (sequence “Football”)
Chapter 4

Transform Domain Transcoding

VC-1 does not use intra prediction for I-frames but H.264 does, transform domain transcoding presents a challenge for I-frames. There are some works that perform intra transcoding in the transform domain for MPEG-2 to H.264 [47]. Our preliminary experiments showed that transform domain did not work well for intra frames in VC-1 to H.264 transcoding, we could not calculate a good distortion that would produce similar results as rate-distortion model mainly because the VC-1 and H.264 transforms are “less” related than DCT to H.264 and the error was not negligible. Therefore VC-1 to H.264 transform domain transcoding is only applied for P/B frames, except that intra blocks and MBs in P/B frames will be transcoded using the pixel domain transcoder described in Section 3. Transform domain coefficient conversion is not lossless, since some of the operations are non-linear. In this proposed method we also provide a compensation method in chapter 4.3 to recover part of the error introduced. VC-1 uses an adaptive block size integer transform (8x8, 8x4, 4x8, and 4x4) implemented in 16 bits as aforementioned in chapter 2. The transform kernel for 8x8 and 4x4 block size transforms and the formulas to obtain transform sizes 8x4 and 4x8 are represented in figure 33. The use of 8x8 size transform captures and preserves trends and periodic structures, while the
use of 4x4 size transform produces less ringing artifacts for areas with discontinuities. Intra frames and intra MB/blocks use 8x8 transforms. H.264 uses a fixed size 4x4 transform implemented in 16-bit arithmetic, and an 8x8 transform can also be used for the High profile. The 4x4 and 8x8 transform kernels used are shown in figure 34. The notations used in the following section are as follows: $r_i$ is the decoded residuals, $M_T$ is the transformed coefficients matrix of $M$, $M^T$ is the transpose of $M$, * is matrix multiplication, • is component wise multiplication, $1_N$ is the identity matrix of dimension $N$, and $\otimes$ denotes Kronecker Product.

$V_8=$

\[
\begin{bmatrix}
12 & 12 & 12 & 12 & 12 & 12 & 12 & 12 \\
16 & 15 & 9 & 4 & -4 & -9 & -15 & -16 \\
16 & 6 & -6 & -16 & -16 & -6 & 6 & 16 \\
15 & -4 & -16 & -9 & 9 & 16 & 4 & -15 \\
12 & -12 & -12 & 12 & 12 & -12 & -12 & 12 \\
9 & -16 & 4 & 15 & -15 & -4 & 16 & -9 \\
6 & -16 & 16 & -6 & -6 & 16 & -16 & 6 \\
4 & -9 & 15 & -16 & 16 & -15 & 9 & -4
\end{bmatrix}
\]

$V_4=$

\[
\begin{bmatrix}
17 & 17 & 17 & 17 \\
22 & 10 & -10 & -22 \\
17 & -17 & -17 & 17 \\
10 & -22 & 22 & -10
\end{bmatrix}
\]

Transform of 8x8, 8x4, 4x8, and 4x4 blocks:

\[
EMxN=(Dx \cdot VM + 4) >> 3
\]

\[
RMxN=(VTN \cdot EMxN + CN1M + 64) >> 7
\]

Where: $M,N = \{4,8\}$

\[
C8=(0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1) \ T
\]

Dx are inverse quantized transform coefficients (input)

RMxN are the inverse transformed blocks (output); values limited to -512 and 511.

**Figure 33.** 8-point and 4-point inverse transform in VC-1
\[ H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \]

\[ H_6 = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} \]

**Figure 34. H.264 transform matrices**

### 4.1 Coefficient Conversion

#### 4.1.1 VC-1 to H.264 Transform

The objective of this section is to find a transform matrix \( S \) for \( S_T \) transform domain (figure 4). \( S \in \mathbb{Z}^{M \times M} \) performs the inverse VC-1 transform and the H.264 forward transform in one step, as shown in figure 5. The general equation for this transform is as follows.

\[ Vout = S*Vin*S_T. \]  

(3)

Since there are four different transform sizes used in VC-1, we need to calculate four \( S \) transforms.

1) When VC-1 uses a 4x4 size transform it should be transformed to a one 4x4 block in H.264.

From H.264 [2] we have

\[ Vout = H_4*V_{7in}*H_4^T. \]  

(4)
From VC-1 standard [1] we have

\[ V_{\text{in}} = (V_4^T \cdot V_{\text{in}} \cdot V_4) \cdot N_4, \]  
(5)

where the normalization matrix \( N_4 = C_4 \cdot C_4^T \) and \( C_4 = \begin{pmatrix} 8/289 & 8/292 & 8/289 & 8/292 \end{pmatrix} \). Substituting (5) in (4), we have

\[ V_{\text{out}} = H_4 \cdot ((V_4^T \cdot V_{\text{in}} \cdot V_4) \cdot N_4) \cdot H_4^T. \]  
(6)

Since (\( \cdot \)) component wise multiplication is associative we have that

\[ V_{\text{out}} = (H_4 \cdot V_4^T \cdot V_{\text{in}} \cdot V_4 \cdot H_4^T) \cdot N_4. \]  
(7)

Therefore

\[ S_{4 \times 4} = V_4 \cdot H_4^T = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & c \\ 0 & 0 & a & 0 \\ 0 & -c & 0 & b \end{bmatrix} \]  
(8)

where \( a = 68 \), \( b = 108 \), and \( c = 4 \).

2) When VC-1 uses an 8x8 size transform it should be transformed to four 4x4 transform blocks (\( V_{\text{out}_1} \) to \( V_{\text{out}_4} \)) in regular H.264. The process is described as follows.

From H.264 standard [2],

\[ V_{\text{out}_1} = \begin{bmatrix} H_4 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} V_{\text{in}} \end{bmatrix} \cdot \begin{bmatrix} H_4^T & 0 \\ 0 & 0 \end{bmatrix}, \]

\[ V_{\text{out}_2} = \begin{bmatrix} H_4 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} V_{\text{in}} \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 \\ 0 & H_4^T \end{bmatrix}, \]

\[ V_{\text{out}_3} = \begin{bmatrix} 0 & 0 \\ 0 & H_4 \end{bmatrix} \cdot \begin{bmatrix} V_{\text{in}} \end{bmatrix} \cdot \begin{bmatrix} H_4^T & 0 \\ 0 & 0 \end{bmatrix}, \]

\[ V_{\text{out}_4} = \begin{bmatrix} 0 & 0 \\ 0 & H_4 \end{bmatrix} \cdot \begin{bmatrix} V_{\text{in}} \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 \\ 0 & H_4^T \end{bmatrix} \]  
(9)

From VC-1 Standard [1],
\[ V_{\text{in}} = (V_{8}^T \ast V_{\text{in}} \ast V_{8}) \cdot N_{8}, \]  
(10)

where normalization matrix \( N_{8} = C_{8} \ast C_{8}^T \) and \( C_{8} = (8/288 \ 8/289 \ 8/292 \ 8/289 \ 8/289 \ 8/289 \ 8/292 \ 8/289)^T \). Substituting (10) in (9) to obtain \( V_{\text{out}} \) (for \( i = 1 \) to \( 4 \)), and rewriting these \( V_{\text{out}} \) equations as a single equation \( V_{\text{out}} \) we get

\[ V_{\text{out}} = H_{4} \otimes 1_{2} \ast V_{8}^T \ast V_{\text{in}} \ast V_{8} \ast H_{4}^T \otimes 1_{2} \cdot N_{8}. \]  
(11)

Therefore

\[
S_{8 \times 8} = V_{8} \ast H_{4}^T \otimes 1_{2} = \begin{bmatrix}
  a & 0 & 0 & 0 & a & 0 & 0 & 0 \\
  b & c & -d & 0 & -b & c & d & 0 \\
  0 & e & 0 & f & 0 & -e & 0 & -f \\
  -g & h & i & 0 & g & h & -i & 0 \\
  0 & 0 & a & 0 & 0 & 0 & a & 0 \\
  j & -k & l & m & -j & -k & -l & m \\
  0 & -f & 0 & e & 0 & f & 0 & -e \\
  -n & p & -q & r & n & p & q & r
\end{bmatrix}\]  
(12)

where \( a = 48, b = 44, c = 30, d = 4, e = 76, f = 8, g = 14, h = 60, i = 26, j = 12, k = 32, l = 36, m = 34, n = 6, p = 16, q = 18, \) and \( r = 68 \). The same calculations can be used for the \( 8 \times 4 \) and \( 4 \times 8 \) transforms with the following results.

3) When \( 4 \times 8 \) block is used by VC-1, it should be transformed to two \( 4 \times 4 \) transform blocks in H.264 using the matrix

\[ V_{\text{out}} = H_{4} \ast V_{4}^T \ast V_{\text{in}} \ast V_{8} \ast H_{4}^T \otimes 1_{2} \cdot N_{4} = S_{4 \times 4} \ast V_{\text{in}} \ast S_{8 \times 8}. \]  
(13)

4) When \( 8 \times 4 \) block is used by VC-1 it should be transformed to two \( 4 \times 4 \) transform blocks in H.264 using the matrix

\[ V_{\text{out}} = H_{4} \otimes 1_{2} \ast V_{8}^T \ast V_{\text{in}} \ast V_{4} \ast H_{4}^T \otimes 1_{2} \cdot N_{8} = S_{8 \times 8} \ast V_{\text{in}} \ast S_{4 \times 4}. \]  
(14)

5) H.264 High profile uses adaptive size transforms (\( 4 \times 4 \) and \( 8 \times 8 \)). This research is focusing on Baseline/Main profiles with the transform size fixed to \( 4 \times 4 \). The transform \( S \) for the case where H.264 uses \( 8 \times 8 \) transforms can be obtained in a similar way as above and Equations (7), (11), (13) and (14) will become:
\[ Vout = H_8 * V_{T4} \otimes 1_2 * Vin * V_4 \otimes 1_2 * H^{T8} \cdot N_8. \quad (15) \]
\[ Vout = (H_8 * V_8^T * Vin * V_8 * H_8^T) \cdot N_8. \quad (16) \]
\[ Vout = H_8 * V_{T4} \otimes 1_2 * Vin * V_8 * H^{T8} \cdot N_8. \quad (17) \]
\[ Vout = H_8 * V_{T8} * Vin * V_4 \otimes 1_2 * H^{T8} \cdot N_8. \quad (18) \]

### 4.2 Fast Algorithm Implementation

The implementation of the transform matrix \(S\) (from Section 4.1) can be further studied to perform fast computation of the transform. The fundamental problem is as follows:

“Given an \(N \times N\) square matrix \(S\), construct an algorithm to evaluate the linear mapping \(x \mapsto S \cdot x\) with as few arithmetic operations as possible.” [48].

Most of the algorithms for fast transform computations have been found by manually determining the relations between the elements in the transform matrix. These relations are called the “symmetry”. Finding the symmetry has a complexity that is not lower than testing graph isomorphism, which is known to be NP-hard [48]. Since \(V_8\) and \(H_4\) are DCT-like and they were constructed conserving the previous symmetry, we can calculate the product of \(V_8 \cdot H^{T4}\) in 72 total arithmetic operations (32 adds and 40 multiplications).

In Case 1, it is easy to see that \(S_{4x4}\) is very close to a diagonal matrix. Therefore (7) can be simplified to:

\[ Vout = Vin \cdot \text{diag}(S_{4x4}) \cdot \text{diag}(S_{4x4})^T \cdot N_4. \quad (19) \]

Here, the impact of approximating \(S_{4x4}\) with a diagonal matrix is negligible [13]. In Case 2, from \(S_{8x8}\) (12) we can see that the matrix has even symmetry about the center column for all rows.
As suggested by (3), the 2D $S_T$ transform is separable. Therefore it can be achieved through 1D transforms, i.e., column transform followed by row transforms. Hence, we shall describe only the computation of the 1D Transform.

Let $in$ be an 8 point column vector, and a vector $U[i]$ be the 1D transform of $i$. The following steps provide a method to determine $U[i]$ efficiently from $i$, which is also described in figure 35 as a flow-graph. We need to calculate the following $m_i$ inter median values.

\[
\begin{align*}
m_1 &= a* in[1] \\
m_5 &= a* in[5] \\
m_7 &= m* in[6] + r* in[8] \\
\end{align*}
\]

And therefore:

\[
\begin{align*}
U[1] &= m_1 + m_2 \\
U[2] &= m_3 + m_4 \\
U[3] &= m_5 - m_6 \\
U[4] &= m_7 + m_8 \\
U[5] &= m_1 - m_2 \\
U[6] &= m_3 - m_4 \\
U[7] &= m_5 - m_6
\end{align*}
\]
Equations (20) and (21) can be implemented in 20 multiplications and 20 additions. The number of computations can be further reduced; however, it would require more intermediate results to be stored. This could cost more than the extra arithmetic operations needed.

The number of operations needed to perform the transforms in cascade transcoder in figure 3 is 120 (8x7 adds and 8x8 multiplications) arithmetic operations to implement the VC-1 inverse transform followed by four H.264 forward transforms (4x3 adds and 4 multiplications since multiply by one is not considered as multiplication) that take extra 64 arithmetic operations. This gives a total operation number of 184. Without using the fast implementation proposed in (21) the number of operations (to implement transform) needed in our case is 72, this reduces the number of operations to about 39% of the original. Since we only need 40 for the fast transform domain transcoding, we reduce the number of operations to about 22% of the original.
Another way of reducing the number of operations needed for transform domain transcoding is to use VC-1/H.264 transform implementations with lower complexity than matrix multiply [61].

4.3 Drift Error

Recent video standards use prediction to compress video sequences. If an error occurs in the frame used to predict the following frames, this error propagates, which becomes a drift error. Drift errors in transcoding have been studied in [10, 11] and can be expressed as:

\[ d = d_q + d_r \]  \hspace{1cm} (22)
The drift error is decomposed in two parts. The first term $d_q$ represents the error in the reference picture used for MC, mainly caused by requantization. This error is general for all transcoders and has been studied thoroughly [10, 11]. It does represent the difference between the actual reference frame used for prediction in the VC-1 decoder and the frame used by the H.264 encoder. It is compensated as described in [10, 11] as

$$d_q = Ms \ast (Y_{n-1}^1 - Y_{n-1}^2)$$

where $Ms$ is the transform domain MC in H.264, $Y_{n}^1$ is the H.264 input signal and $Y_{n}^2$ is the H.264 reconstruction signal.

The second term $d_i$ is the result of performing motion compensation in the transform domain between standards with different interpolation filters for half and quarter pixel displacement. The goal of motion compensation in the transform domain is to find transform coefficients of the reference block from the transform coefficients of the four transform coefficients overlapped by the reference block. Figure 36 presents the case in which the MC is performed in the DCT transform domain, from a 8x8 transform size (i.e. MPEG-2) to a 4x4 transform size (i.e. H.264). Since VC-1 and H.264 use both integer transform that are DCT alike the concepts from performing MC in the transform domain can be extrapolated to the VC-1 to H.264 transform domain. The MC in the transform domain for the transcoding of VC-1 to H.264 is complicated by the fact that both standards use different interpolation methods for half and quarter pixel samples. Therefore the error $d_i$ is specific to the kind of video standards the transcoder is working on. In VC-1 MVs are specified in pixel displacements, with allowed sub-pixel accuracy of $\frac{1}{4}$, $\frac{1}{2}$ and $\frac{3}{4}$. It uses a 4-tap bicubic filter $F_{vc-1} = (-4 53 18 -3)/64$ for $\frac{1}{4}$ precision and $(-1 9 9 -1)/16$ for $\frac{1}{2}$ precision and also allows a bilinear filter (weighted average of two
neighboring pixels) for \( \frac{1}{2} \) precision [1]. H.264 uses a 6-tap filter \( F_{H.264} = (1, -5, 20, 20, -5, 10)/32 \) to generate the \( \frac{1}{2} \) pixel accuracy, whereas averaging the signal at integer and \( \frac{1}{2} \)-pixel positions generates the \( \frac{1}{4} \)-pixel accuracy positions [2]. Following the same kind of reasoning as in [10] we can estimate \( d_i \) for the two different kinds of transforms used by H.264 High Profile 4x4 and 8x8. Since main and advanced VC-1 profiles utilize in-loop filter, another factor for the drift error is the use of different deblocking filters by each standard (section 3.5).

\[
\begin{align*}
C_w &= \begin{bmatrix} 0 & 0 \\ I_w & 0 \end{bmatrix} \\
C_h &= \begin{bmatrix} 0 & I_h \\ 0 & 0 \end{bmatrix}
\end{align*}
\]

\[
DCT(B_i) = \sum_{i=1}^{4} DCT(C_h)DCT(A_i)DCT(C_w)
\]

**Figure 36.** MC in the DCT transform domain

### 4.3.1 Integer pixel displacement

If the reference MV is displaced by an integer number of pixels with respect to the reference block in both horizontal and vertical directions then MC in the HT domain is same as MC in the VC-1 domain, the only difference is the block size used by VC-1 and the block size used for H.264. In general, neither the vertical nor the horizontal MV is an
integer multiple of the block size; thus, the displaced block intersects several neighboring blocks $B_i$. The number of neighboring intersected blocks $i$ depends on the transform size in VC-1 (4x4, 4x8, 8x4 and 8x8) and the transform size used by H.264 (4x4 and 8x8 for high profile). Assuming by now that H.264 uses only main profile (transform size 4x4 only) we can derive the MC reference block $Br$ from the HT coefficients of blocks $B_1$ to $B_i$.

$$Br = \sum_{i=1}^{Z} V_i B_i H_i$$

(24)

where $Z$ is 4 for VC-1 transform size 8x8, 2 for VC-1 transform size 8x4 and 4x8, and 1 for 4x4 (figure 38), and $V_i$ and $H_i$ are the constant geometric transform matrices defined by the height and weight of each block generated by the intersection of $B_i$ with $B_r$.

The number of rows ($h$) and columns ($w$) that each block $B_i$ is intersected by $B_r$ defines which matrices $V_i$, $H_i$ should be applied for each $B_i$. The values of $V_i$, $H_i$ depend on the transform size used by VC-1, and therefore we have the following four cases (figure 37):

a.) VC-1 uses a 4x4 transform size and H.264 uses a 4x4 transform size:

No transform necessary the transform coefficients can be reused directly. ($V_i=1$, $H_i=1$)

b) VC-1 uses an 8x8 transform size and H.264 uses a 4x4 transform size:

$$V1 = \begin{bmatrix} 0 & 1_{h\times h} \\ 0 & 0 \end{bmatrix} = V2$$

(25)

$$V3 = \begin{bmatrix} 0 & 1_{4-h} \\ 0 & 0 \end{bmatrix} = V4$$

(26)

$$H1 = \begin{bmatrix} 0 & 0 \\ 1_{w\times w} & 0 \end{bmatrix} = H3$$

(27)
\[ H2 = \begin{bmatrix} 0 & 0 \\ 1_{4-w4-w} & 0 \end{bmatrix} = H4 \]  

where \( I \) is the identity matrix, and \( 1 \leq h, w \leq 4 \).

The reference block \( B_r \) can be then computed as follows:

\[ HT(B_r) = HT\left( \frac{4}{i=1} V_i B_i H_i \right) = \frac{4}{i=1} HT(V_i)HT(B_i)HT(H_i) \]  

(29)

c) VC-1 uses a 4x8 transform size and H.264 uses a 4x4 transform size:

\[ H1 = \begin{bmatrix} 0 & 0 \\ 1_{wxw} & 0 \end{bmatrix} = H3 \]  

(30)

\[ H2 = \begin{bmatrix} 0 & 0 \\ 1_{4-wx4-w} & 0 \end{bmatrix} = H4 \]  

(31)

The reference block \( B_r \) can be then computed as follows:

\[ HT(B_r) = HT\left( \frac{2}{i=1} B_i H_i \right) = \frac{2}{i=1} HT(B_i)HT(H_i) \]  

(32)

d) VC-1 uses an 8x4 transform size and H.264 uses a 4x4 transform size:

\[ V1 = \begin{bmatrix} 0 & 1_{hxh} \\ 0 & 0 \end{bmatrix} = V2 \]  

(33)

\[ V3 = \begin{bmatrix} 0 & 1_{4-h} \\ 0 & 0 \end{bmatrix} = V4 \]  

(34)

The reference block \( B_r \) can be then computed as follows:

\[ HT(B_r) = HT\left( \frac{2}{i=1} V_i B_i \right) = \frac{2}{i=1} HT(V_i)HT(B_i) \]  

(35)

H.264 uses 4x4 or 8x8 as transform size in the high profile. The same kind of calculations described above will be needed but since the transform size of H.264 could be 8x8, we have the following cases:
a) VC-1 uses an 8x8 transform size and H.264 uses an 8x8 transform size:

No transform necessary the transform coefficients can be reused directly. \((V_i=1, H_i=1)\).

b) VC-1 uses a 4x4 transform size and H.264 uses an 8x8 transform size:

\[
B_i = \begin{bmatrix} B_1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & B_2 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ B_3 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & B_4 \end{bmatrix}
\]  

(36)

And now the case is same as case a) and \(B_r = B_i\).

c) VC-1 uses a 4x8 transform size and H.264 uses an 8x8 transform size:

\[
B_i = \begin{bmatrix} B_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ B_2 \end{bmatrix}
\]  

(37)

And now the case is same as case a) and \(B_r = B_i\).

d) VC-1 uses an 8x4 transform size and H.264 uses an 8x8 transform size:

\[
B_i = \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix}
\]  

(38)

And now the case is same as case a) and \(B_r = B_i\).
Figure 37. Motion compensation in the HT domain for integer displacement

4.3.2 ½ Pixel Accuracy

Figure 38. Predicted block for ½ pixel accuracy for VC-1 transform size=8x8
Since H.264 uses a 6-tap filter, it needs a block of \((5+N)\times(5+M)\) reference pixels to calculate the \(\frac{1}{2}\) accuracy pixels (figure 38). Therefore, the half pixel accuracy operation for the 4x4 block shown in figure 39 can be interpreted as computing the coefficients of the target \((5+N)\times(5+N)\) H.264 transformed motion compensated block \(B_r\) from the coefficients of its \((5+N)\) neighboring blocks.

\[
B_{(5+N)\times(5+M)} = \sum_{i=1}^{N+5} (V_i((5+N)\times 4))B_i(H_i((5+M)\times 4)).
\]  

(39)

with:

\[
V_1 = V_2 = V_3 = \begin{bmatrix} 0 & I_h \\ 0 & 0 \end{bmatrix}
\]

(40)

\[
V_4 = V_5 = V_6 = \begin{bmatrix} 0 \\ I_4 \\ 0 \end{bmatrix}
\]

(41)

\[
V_7 = V_8 = V_9 = \begin{bmatrix} 0 & 0 \\ I_{5-h} & 0 \end{bmatrix}
\]

(42)

\[
H_1 = H_4 = H_7 = \begin{bmatrix} 0 & 0 \\ I_w & 0 \end{bmatrix}
\]

(43)

\[
H_2 = H_5 = H_8 = \begin{bmatrix} 0 & I_4 & 0 \end{bmatrix}
\]

(44)

\[
H_3 = H_6 = H_9 = \begin{bmatrix} 0 & I_{5-w} \\ 0 & 0 \end{bmatrix}
\]

(45)

and \(N, M = 4\) or 8 (depending of the VC-1 transform size), \(I_Y\) are identity matrices with dimensions \(Y\times Y\) and \(1 \leq h, w \leq 4\).

To get the 4x4 block of \(\frac{1}{2}\) pixel accuracy, we have the MC in the transform domain \((B_{4,4}^T)\) from the coefficients of the neighboring \((5+N)\) blocks \(B_i\) as in [10], as
\[ B_{4x4}^{H.264} = F_{H.264} \ast (B_{(5+N)(5+M)}) \ast F_{H.264}^T. \] (46)

where \( N, M = 4 \text{ or } 8 \). \( F_{H.264} \) is a \((4xN)\) matrix which results from padding with zeros for the filter \( 1/32(1-5 20 20 -5 1) \) to get the first \( 1xN \) rows and shifting by one for the following rows. The predicted \( 4x4 \) block \( B_{4x4}^{H.264} \) is computed as in [10].

For VC-1 we can make the same previous reasoning,

\[ B_{4x4}^{VC-1} = F_{VC-1} \ast B_{(5+N)(5+M)} \ast F_{VC-1}^T. \] (47)

where \( F_{VC-1} \) is a \((4xN)\) matrix which results from padding with zeros for the filter \( 1/16 (-1 9 9 -1) \), or \( 1/2 (1 1) \) if filter used is bilinear, to get the first \( 1xN \) rows and shifting by one for the following rows. The predicted \( 4x4 \) block \( B_{4x4}^{VC-1} \) is computed as in [10].

### 4.3.3 \( \frac{1}{4} \) Pixel Accuracy

The calculations performed for the \( \frac{1}{2} \) pixel MC in the transform domain can be applied for calculation the drift error in \( \frac{1}{4} \) pixel accuracy but by using different filters. To avoid repeating we do not present here the detailed calculations that should be same as the ones described in section 4.3.2 but with:

\( F_{VC-1} = (-4 53 18 -3)/64 \) for \( \frac{1}{4} \) precision

\( F_{H.264} = (1,-5, 20, 20,-5, 1)/32 \) to generate the \( \frac{1}{2} \) pixel accuracy, whereas averaging the signal at integer and \( \frac{1}{2} \)-pixel positions generates the \( \frac{1}{4} \)-pixel accuracy positions.

### 4.3.4 Error Drift Amount

The error drift amount is given as

\[ d_i = B_{4x4}^{VC-1} - B_{4x4}^{H.264}. \] (48)

The architecture that we implemented to compensate for the drift error amount is represented in figure 3 and can be summarized as follows. The VC-1 encoded stream is
decoded to transform coefficients level first. These coefficients are transformed using the $S_T$ transform described in Section 4.1.1 then quantized and run-length coded to produce the H.264 encoded output. This algorithm is very computationally efficient but has an important disadvantage -- drift error. To compensate the drift error we calculate the drift due to different interpolation kernels in the transform domain by pre-computing the matrices in Equations (46) and (47).

4.4 High Profile Features

4.4.1 Adaptive Size Transform

For the pixel domain implementation in the mapping, the flow chart in figure 26 is enough. But for the transform domain implementation, we need to have the transform domain transcoding $S_T$ (i.e., figure 5) matrices.

Case 1: VC-1 uses 8x8 transform and maps to 8x8 transform in H.264. Eq (16) is used.

Case 2: VC-1 uses 4x4 transform and maps to 4x4 transform in H.264. Eq (7) is used.

Case 3: VC-1 uses 4x8 transforms and then measures the homogeneity of the block $H$, using the technique described above.

$$\begin{align*}
\text{If } SAD \geq th_{adapTS} & \quad \text{TransformSize}_{H.264} = 4 \times 4 \quad \text{Eq.}(13) \text{ is used} \\
\text{Else} & \quad \text{TransformSize}_{H.264} = 8 \times 8 \quad \text{Eq.}(17) \text{ is used}
\end{align*}$$

Case 4: VC-1 uses 8x4 transforms and then measures the homogeneity of the block $H$.

$$\begin{align*}
\text{If } SAD \geq th_{adapTS} & \quad \text{TransformSize}_{H.264} = 4 \times 4 \quad \text{Eq.}(14) \text{ is used} \\
\text{Else} & \quad \text{TransformSize}_{H.264} = 8 \times 8 \quad \text{Eq.}(18) \text{ is used}
\end{align*}$$

where $th_{adapTS}$ denotes a threshold obtained by experimentation.

4.4.2 Interlacing
For pixel domain transcoding once the mapping is done (figure 32) the algorithm finish, but for transform domain we need to obtain the equations for transcoding. If there is no change in the frame/field mode selected by VC-1, Equations (7), (11), (13)-(19) remain the same. But if the mode selected is changed, we need some extra transformations. The decision mode changes the ordering of the coefficients if a frame MB in VC-1 is separated into two different fields as shown in figures 27 and 29. In such a case, however, the H.264 coefficients are not separated. For frame-to-frame or field-to-field transcoding, coefficients are not separated and hence no reordering is needed. When reordering happens, we need to permutate back the input coefficients. We permutate back the rows (1,5,2,6,3,7,4,8) using permutation matrix $P$ to transform Equations (7), (11), (13)-(18).

For example, (16) becomes:

$$V_{out} = (H_8 * P^T * V_{in} * V_8 * H_8^T) \bullet N_8.$$  \hspace{1cm} (49)

Here

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$  

When frame/field mode changes, MVs are scaled in the vertical direction to prepare them for the H.264 entropy encoder stage. Integer pixel accuracy MC in the transform domain is performed as described in Section 4.3.1
Chapter 5

Results for Pixel and Transform Domain Transcoder

In this section we present the experimental evaluation of the proposed pixel domain and transform domain transcoders and compare the quality of the compressed video with the reference transcoder (the reference transcoder has full VC-1 decoding followed by full H.264 encoding without any mode mapping or MV reuse). The H.264/AVC reference software used was JM 13.2 [49] and SMPTE 421M April 2006 for VC-1. The experiments were conducted with six video sequences: “Logo” 128x96, “Foreman” 176x144 and 352x288, “Claire” 176x144, “Walk” 176x144, and “Parkrun” 1280x720, with no interlacing; “Football” and “Train” 720x480 with interlacing. The sequences can be found at [62]. The sequences were encoded at 30 fps. To run these experiments a Dell Inspiron 300m running at 1.2 GHz and 256 MB memory was used. The GOP used was IPPPP… for all sequences using only I/P frames, and IBPBP… for the ones using I/P/B frames. The rate control was set off on all sequences. Thresholds’ values used are given in Table III. A description of the encoder options for VC-1 and H.264 can be found in Tables IV and V respectively. Table V shows that we used RDO and Full Search in H.264 because they provide better results. In our work we are mainly concerned with
reducing the time complexity of transcoding while maintaining the visual quality closely without increasing the bit-rate significantly. In order to maintain the original VC-1 encoded bit-rate, the QPs for the H.264 encoder were selected by using the method described in section 7.2. To simplify the performance evaluation we have used the Bjontegaard method described in [50] to compare the R-D curves to produce Tables VII and VIII.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{th}_{\text{intra}}$</td>
<td>1 for QP&lt;16; 50 otherwise.</td>
</tr>
<tr>
<td>$\text{th}_{\text{intraHP}}$</td>
<td>31</td>
</tr>
<tr>
<td>$\text{th}_{\text{intraPred}}$</td>
<td>1</td>
</tr>
<tr>
<td>$\text{th}_{\text{inter}}$</td>
<td>20</td>
</tr>
<tr>
<td>$\text{th}_{\text{MV}}$</td>
<td>8</td>
</tr>
<tr>
<td>$f$</td>
<td>sequence resolution/QCIF resolution</td>
</tr>
<tr>
<td>$\text{th}_{\text{adapTS}}$</td>
<td>15</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Num frames</td>
<td>30</td>
</tr>
<tr>
<td>Frame Rate</td>
<td>30</td>
</tr>
<tr>
<td>Profile</td>
<td>Main</td>
</tr>
<tr>
<td></td>
<td>Advanced</td>
</tr>
<tr>
<td>Level</td>
<td>L2</td>
</tr>
<tr>
<td>Intra Period</td>
<td>16</td>
</tr>
<tr>
<td>Interlace</td>
<td>No (when not being used, yes when used)</td>
</tr>
<tr>
<td>Quality</td>
<td>6,12,20,25, 27, 29</td>
</tr>
<tr>
<td>Quantizer</td>
<td>Uniform</td>
</tr>
<tr>
<td>ChromaFormat</td>
<td>420</td>
</tr>
<tr>
<td>LoopFilter</td>
<td>0</td>
</tr>
<tr>
<td>OverlappedTransform</td>
<td>0</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>FramesToBeEncoded</td>
<td>30</td>
</tr>
<tr>
<td>FrameRate</td>
<td>30</td>
</tr>
<tr>
<td>ProfileIDC</td>
<td>66, 77 for B Frames and 100 for interlace/TS= 8x8</td>
</tr>
<tr>
<td>LevelIDC</td>
<td>30</td>
</tr>
<tr>
<td>IntraPeriod</td>
<td>16</td>
</tr>
<tr>
<td>QPISlice</td>
<td>16,20,24,28 for table VII/VIII; 16,18,20,24,28,31, 35, and 38 for Fig.9</td>
</tr>
<tr>
<td>QPBPslice</td>
<td>16,20,24,28 for Tables VII &amp; VIII; 16,18,20,24,28,31,35, and 38 for Fig.9; plus also 45 for train and football.</td>
</tr>
<tr>
<td>FrameSkip</td>
<td>0 when not using B-frames, 1 when using B-frames</td>
</tr>
<tr>
<td>UseHadamard</td>
<td>1</td>
</tr>
<tr>
<td>SearchRange</td>
<td>16</td>
</tr>
<tr>
<td>NumberReferenceFrames</td>
<td>2</td>
</tr>
<tr>
<td>NumberBFrames</td>
<td>1 (when B frames used)</td>
</tr>
<tr>
<td>QPBSlice</td>
<td>always 2 QP units more than for I Frames</td>
</tr>
<tr>
<td>DirectModeType</td>
<td>1</td>
</tr>
<tr>
<td>BLList0References</td>
<td>1</td>
</tr>
<tr>
<td>BLList1References</td>
<td>1</td>
</tr>
<tr>
<td>BLReferencePictures</td>
<td>0</td>
</tr>
<tr>
<td>PicInterlace</td>
<td>0 if no interlace, 2 if interlace (Adaptive frame/field coding)</td>
</tr>
<tr>
<td>MbInterlace</td>
<td>0 when no interlace use, 2 when using interlace</td>
</tr>
<tr>
<td>RDOptimization</td>
<td>1: RD-on (high Complexity mode)</td>
</tr>
<tr>
<td>RateControlEnable</td>
<td>0 : Disable</td>
</tr>
<tr>
<td>SymbolMode</td>
<td>CAVLC</td>
</tr>
<tr>
<td>MbInterlace</td>
<td>0 (3 when using interlace)</td>
</tr>
<tr>
<td>WeightedPrediction</td>
<td>1</td>
</tr>
<tr>
<td>LoopFilterDisable</td>
<td>0</td>
</tr>
<tr>
<td>PicInterlace</td>
<td>0 (2 when using interlace)</td>
</tr>
<tr>
<td>RDOptimization</td>
<td>1</td>
</tr>
</tbody>
</table>
5.1 Results

5.1.1 Pixel Domain Results

5.1.1.1 I-Frames Results

The intra MB mode (with block size 4x4, 8x8 or 16x16) was determined based on the variance of the DC coefficients in the input VC-1 MBs. The first column (after the sequence column) of Table VI shows the percentages of MBs that were correctly mapped. Note also that the definition of “correct” MB used in this paper indicates the MBs whose transcoding prediction fully matched with what H.264 encoder would choose. This is not a necessary condition to obtain a good PSNR, but is only to obtain the optimal mapping assuming H.264 encoder takes an R-D optimized decision. The high percentages in intra MBs show that I frames have almost the same quality with our pixel domain transcoding compared with those of using cascading method. For example, the sequence foreman (qcif) coded at QP=30 only using I-frames resulted in PSNR=36.39 when cascading is used and PSNR=36.36 when pixel domain transcoder is used at same bit rate.
<table>
<thead>
<tr>
<th>Sequence</th>
<th>% Correct I-Frame MBs</th>
<th>% Correct P-Frame MBs</th>
<th>% Correct B-Frame MBs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logo</td>
<td>100</td>
<td>87</td>
<td>60</td>
</tr>
<tr>
<td>Foreman qcif</td>
<td>91</td>
<td>49</td>
<td>72</td>
</tr>
<tr>
<td>Claire qcif</td>
<td>73</td>
<td>52</td>
<td>90</td>
</tr>
<tr>
<td>Walk qcif</td>
<td>93</td>
<td>62</td>
<td>47</td>
</tr>
<tr>
<td>Foreman cif</td>
<td>91</td>
<td>48</td>
<td>70</td>
</tr>
<tr>
<td>Park Run 720p</td>
<td>98</td>
<td>64</td>
<td>72</td>
</tr>
</tbody>
</table>

5.1.1.2 P-Frames Results

Figure 39 shows the % of correctly predicted MB (MB size used for MC) in different sequences for every frame. The effect of the use of mixed MB mode mapping can be seen in frames 4 and 7 for Claire in which VC-1 predicts intra MB while H.264 uses inter MB. A bad % of correctly predicted MB does not imply that the resulting quality would be bad.

Inter MB mapping is based on the sizes of the VC-1 transform used as described in Chapter 3. The second column of Table VI shows the total percentages of P-frames inter MBs that were correctly predicted (block size used for MC as discussed in section 3.2 by the transcoder for the whole video sequence. For sequences with less movement (Logo) the percentage of correctly predicted MBs is higher (due to more skipped MBs, where skipped MBs are correctly predicted by our algorithm). Sequence Logo is a very low bitrate sequence. Small variations produce a big difference in the bitrate especially at lower QPs. Sequence Walk contains a scene change and fairly static (in number of
skipped MBs) with initial frames, so we can see a good improvement in performance by the transcoder. Figure 40 (obtained using different QPs) and Table VII present PSNR/bitrate results; one can see that the performances of our algorithm follow very closely with those of the cascade transcoder. Table VII shows that the performances of our pixel domain transcoder are slightly worse in some cases; this disadvantage is greatly compensated by the reduction in computational complexity (Table X).

In figure 41 we see the result of the PSNR vs. frame we can see that the difference between reference cascade and our proposed implementation is negligible.

Figure 42 shows the result of a subjective quality evaluation of the image using cascade transcoding versus the image transcoded using our proposed algorithm; the visual differences are hardly noticeable.

![Graph showing percentage of correctly predicted MB](image)

**Figure 39.** Percentage of correctly predicted MB in different sequences for every frame.

The effect of the use of mixed MB mode mapping, described in 3.1.2 can be seen in frames 4 and 7 for Claire in which VC-1 predicts intra MB while H.264 user inter MB.
### Table VII
PSNR (dB)/Bitrate (kbits/sec) for I-P-Frames with respect to Cascade (Baseline)

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Δ PSNR</th>
<th>Δ Bitrate%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logo (128×96)</td>
<td>-1.70</td>
<td>0.07</td>
</tr>
<tr>
<td>Foreman qcif</td>
<td>-0.72</td>
<td>0.05</td>
</tr>
<tr>
<td>Claire qcif</td>
<td>-0.24</td>
<td>0.01</td>
</tr>
<tr>
<td>Walk qcif</td>
<td>-0.02</td>
<td>0.001</td>
</tr>
<tr>
<td>Foreman cif</td>
<td>-0.21</td>
<td>0.01</td>
</tr>
<tr>
<td>Park Run hd</td>
<td>-0.15</td>
<td>-0.07</td>
</tr>
<tr>
<td>Football sd*</td>
<td>-0.96</td>
<td>0.62</td>
</tr>
<tr>
<td>Train sd*</td>
<td>-1.10</td>
<td>0.40</td>
</tr>
</tbody>
</table>

### Table VIII
PSNR (dB)/Bitrate (kbits/sec) for I-P-B-Frames with respect to Cascade (Main)

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Δ PSNR</th>
<th>Δ Bitrate%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logo (128×96)</td>
<td>-1.99</td>
<td>0.98</td>
</tr>
<tr>
<td>Foreman qcif</td>
<td>-0.45</td>
<td>0.36</td>
</tr>
<tr>
<td>Claire qcif</td>
<td>-0.20</td>
<td>1.44</td>
</tr>
<tr>
<td>Walk qcif</td>
<td>-0.14</td>
<td>0.09</td>
</tr>
<tr>
<td>Foreman cif</td>
<td>-0.13</td>
<td>0.13</td>
</tr>
<tr>
<td>Park Run hd</td>
<td>-0.15</td>
<td>0.01</td>
</tr>
<tr>
<td>Football sd*</td>
<td>-0.42</td>
<td>0.29</td>
</tr>
<tr>
<td>Train sd*</td>
<td>-1.91</td>
<td>1.04</td>
</tr>
</tbody>
</table>

* High profile tools used in these SD sequences
5.1.1.3 B- Frames Results

Table VI last column shows the total percentages of MBs that are correctly predicted by the transcoder for B-frames. The percentages are higher than the percentages for P-frames in general because H.264 selects direct mode for the majority of MBs in B-frames and this type of MBs are easily predicted with our algorithm. Table VIII also shows that our pixel domain transcoder follows very closely the performances of the reference (cascade) domain in the I-P-B case.

5.1.1.4 Transform Size in H.264 High Profile and Interlaced Results

Table VII last two rows present the results for ΔPSNR and ΔBitrate% for high profile and interlace results. We can see that the proposed algorithm for transcoding is still stable when using high profile transform size in H.264 and interlace. In figure 40 last two graphs (“Football” and “Train” 720×480 with interlacing and coded with high profile tools) we can see how the transform domain implementation compares to the reference cascade. The proposed algorithm maintains the PSNR when compared to the reference model (cascade) while reducing the complexity (Table X). We can see that PSNR differences are negligible. ΔPSNR = 0.06 dB (for Train) and 0.04 dB (for Football) comparing the cascaded and our transform domain transcoding with drift compensation. The bit-rate slightly increases with a ΔBitrate = 0.96 % (Train) and ΔBitrate == 1.10 % (Football) but these disadvantages are compensated by a much better complexity performance as described in Table X.

5.1.2 Transform Domain Results

In this section we present the implementation of the proposed transform domain model (using $S_T$ as in figure 5) and compare with the quality of the video compressed using the
reference model of figure 3. We transcode I frames using the pixel domain transcoder described in Chapter 3 and P/B frames using the model proposed in Chapter 4. For P frames the following equations summarize the prediction.

\[
\text{Inv. VC-1} \rightarrow r_{vc-1} + MC_{vc-1} = \text{pixels}, \quad (50)
\]

\[
\text{Fwd. H.264} \rightarrow \text{pixels} - MC_{H.264} = r_{H.264}, \quad (51)
\]

where \( r \) represents the residuals and \( MC \) is the motion compensated pixels. If we assume that motion vectors are re-used during the transcoding process the error for assuming \( MC_{vc-1} = MC_{H.264} \), or what is the same \( (r_{vc-1} - r_{H.264}) \), is reasonably small. Under this assumption we can apply the transform matrix \( S \) for the different transform sizes, with no motion estimation (ME) needed. The problem with this model is that it is not drift error free; to avoid drift error accumulation we add the drift error equations (23) and (48).

From the R-D curves in figure 40 we see that the actual PSNRs for the pixel transcoded version are almost the same as the reference cascade transcoder. There are some small anomalies on the graphs; on the Logo sequence (figure 40(a)) the pixel domain transcoder has poor performance at low bit-rate values (high QP). This is because in the sequence there are some initial frames that are all black (with some variations on color) containing mostly skipped MBs and any errors on mapping skipped MB (especially that the sequence has a small size) could result in a noticeable bitrate increase. Also the sequence Foreman cif (figure 40(b)) has a slightly poorer performance in the transform domain due to small variations when calculating the drift errors that are more visible in lower quality sequences. The subjective quality of the video sequence was also evaluated. It is noted that there is a small loss of visual quality especially in areas that require more details such as the facial regions around the eye of Foreman; this is due to
errors in the MV prediction, since we reused MVs from the VC-1 decoder. The transform
domain transcoder without compensation for drift error has on average a 1 dB drop in
quality with respect to the reference transcoder. However, our transform domain
transcoder with drift error compensation added has a slight degradation compared to the
reference cascade transcoder (hence if this slight degradation is a concern, one can choose
pixel domain transcoding over transform domain in those cases). Table IX present
PSNR/bitrate results.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>∆PSNR</th>
<th>∆Bitrate%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logo (128×96)</td>
<td>-2.08</td>
<td>0.06</td>
</tr>
<tr>
<td>Foreman qcif</td>
<td>-1.60</td>
<td>1.24</td>
</tr>
<tr>
<td>Claire qcif</td>
<td>-1.81</td>
<td>1.50</td>
</tr>
<tr>
<td>Walk qcif</td>
<td>-1.59</td>
<td>0.82</td>
</tr>
<tr>
<td>Foreman cif</td>
<td>-1.73</td>
<td>1.54</td>
</tr>
<tr>
<td>Park Run hd</td>
<td>-1.21</td>
<td>0.76</td>
</tr>
<tr>
<td>Football sd*</td>
<td>-0.97</td>
<td>0.60</td>
</tr>
<tr>
<td>Train sd*</td>
<td>-0.54</td>
<td>0.20</td>
</tr>
</tbody>
</table>

* High profile tools used in these SD sequences

5.1.3 Complexity

Table X shows the time complexity reduction (in percentage, total execution time) to
encode the different sequences by our proposed transcoding algorithm compared to using
cascade reference transcoder. We can see from the table and the figure that by using the proposed
transcoding algorithms we reduce the time used to encode a sequence by around 60%
without a significant loss in PSNR. The cascaded transcoder has to perform the two most computationally intensive operations in H.264, which are ME and R-D optimization. According to [51] these two operations increase the complexity of the encoder by about 60% and 40%, respectively, using one reference frame. These percentages are only an approximation obtained for Main profile. In our algorithm we determine the transform size (only 11 extra operations needed per block) and the interlace type (only 4 extra operations needed per block) without having to perform R-D optimization. For the transform domain transcoder, none of these operations are needed and therefore the complexity is further reduced. However, there is the problem of drift error. To compensate for drift error, some operations can be pre-computed and stored in memory with \(4*(64 + 120)\) (since we have to perform H.264 transform and VC-1 transform four times) operations per block. However, this increase in the number of operations is small compared to the decrease in complexity from not having to perform ME and R-D `optimizations. The proposed algorithm thus reduces the complexity of the transcoder without a significant loss in visual quality.
(a) Logo

(b) Foreman q cif
Figure 40. R-D performances for the proposed and reference cascade transcoders.
(a) Logo (same bitrate 37 kbps)

(b) Foreman (same bitrate 200 kbps)
Figure 41. PSNR with/without transcoding
<table>
<thead>
<tr>
<th>Sequence</th>
<th>% Reduction Cascade vs. Pixel domain</th>
<th>% Reduction Cascade vs. Transform domain</th>
<th>% Reduction Cascade vs. Transform+drift comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logo qcif</td>
<td>62</td>
<td>68</td>
<td>62</td>
</tr>
<tr>
<td>Foreman qcif</td>
<td>60</td>
<td>73</td>
<td>70</td>
</tr>
<tr>
<td>Claire qcif</td>
<td>60</td>
<td>75</td>
<td>70</td>
</tr>
<tr>
<td>Walk qcif</td>
<td>59</td>
<td>70</td>
<td>65</td>
</tr>
<tr>
<td>Foreman cif</td>
<td>51</td>
<td>66</td>
<td>63</td>
</tr>
<tr>
<td>Park Run hd</td>
<td>52</td>
<td>77</td>
<td>73</td>
</tr>
<tr>
<td>Football sd*</td>
<td>50</td>
<td>76</td>
<td>68</td>
</tr>
<tr>
<td>Train sd*</td>
<td>52</td>
<td>76</td>
<td>68</td>
</tr>
</tbody>
</table>

* High profile tools enabled for these SD sequences
Figure 42. The fifth frame of the video sequence using H264 to encode it (first column) and using the transcoding algorithm described in this thesis (second column)
5.2 Summary of Results

Two algorithms for transcoding VC-1 to H.264 are presented. First, is a pixel domain transcoder based on exploiting the variable transform size used in VC-1 to determine MB coding mode in H.264. The results show that this approach works reasonably well by reducing the complexity by about 60% with negligible drop in PSNR. Second, is a transform domain transcoder that uses the previous algorithm for I frames and matrix multiplication techniques to convert transform coefficients in VC-1 to transform coefficients in H.264 without having to calculate inverse transforms. The results show that this approach reduces the complexity but the PSNR drops by around 1 dB; to improve the PSNR value we add drift error compensation which increases the complexity slightly.

The two techniques, pixel and transform domain transcoding, can be jointly used at frame level. Pixel domain transcoding techniques should be used for I frames, while transform domain transcoding could be used for P and B frames. Since P frames are used as references the transcoder would typically produce better quality results if P frames are transcoded by pixel domain method. However, P frames can also be transcoded by transform domain method depending on the computational requirement of the system. On the other hand, B frames are most suitable for transform domain transcoding since they do not propagate errors. The combination of the two techniques (pixel for I frames and transform for P and B frames) can optimize transcoder systems in terms of both computational complexity and visual quality of the results (figure 43).
Figure 43. Proposed transcoder architecture. Focus is on reducing complexity while maintaining the PSNR/bitrate performance using pixel and transform domain transcoder.
Chapter 6

Resolution Conversion

6.1 Downsampling

For P/B frames the DCT decimation scheme with the 8×8 block size has proven to achieve significantly better visual quality compared to the schemes using pixel domain filtering followed by down sampling [19, 20]. VC-1 [1] and H.264 [2] use transforms which are DCT-like, so most of the already researched algorithms can be used. Algorithms for arbitrary size scaling use DCT matrices of several sizes but in our case we only have H.264 transforms for sizes 4×4 and 8×8, therefore transcoding VC-1 to H.264 can only be done with scaling by a factor of 2 in each dimension.

![Diagram](image)

**Figure 44.** Down sampling an 8x8 block in VC-1 to a 4x4 block in H.264 in the transform domain
Our approach uses the transcoder described in Chapters 3 and 4 but modified to perform down sampling and is described as follows (figure 44). Given a VC-1 8x8 block of transform coefficients \( B \) we obtain the four left corner 4x4 coefficients \( \hat{b} \) in the pixel domain:

\[
\hat{b}_{4x4} = V_4^{-1} \cdot [I_4 \quad 0_4] \cdot B_{8x8} \cdot [I_4 \quad 0_4]^T \cdot V_4^{-1,T}
\]

(52)

where \( V_n^{-1} \) and \( V_n^{-1,T} \) are the \( n \times n \) VC-1 inverse and inverse transposed matrices. \( I_n \) and \( 0_n \) are the \( n \times n \) identity and zero matrices. The down sampled \( \hat{b} \) is then transformed to H.264 (\( B_{4x4}^{down} \)):

\[
B_{4x4}^{down} = (H_4 \cdot \hat{b}_{4x4} \cdot H_4^T) \cdot N_4
\]

(53)

where \( H_n \) and \( H_n^T \) are the \( n \times n \) forward and transposed H.264 transform matrices. \( N \) is VC-1 normalization vector whose value can be found in [1] and • is a component wise multiplication.

For I frames, down sampling are done in the pixel domain by first applying a low pass filter followed by down sampling by a factor of 2 each dimension. Figure 45 shows the effect of downsampling by four an original image.

![Figure 45](image)

**Figure 45.** Result of using downsampling by four on an image
6.2 Upsampling

Figure 46. Up sampling an 8x8 block in VC-1 to a 4x4 block in H.264 in the transform domain

We use the transcoder described in Chapters 3 and 4 modified to perform up sampling. For the P/B frames we can perform up sampling in the transform domain by zero insertion followed by low pass filtering (figure 46). Given a VC-1 4x4 block of transform coefficients $B$, to up sample by zero padding to get the 8x8 block of pixels $\hat{b}$:

$$
\hat{b}_{8x8} = V_{8}^{-1} \times \begin{bmatrix} B_{4x4} & 0 \\ 0 & 0 \end{bmatrix} \times V_{8}^{-1T}.
$$

(54)

The up-sampled $\hat{b}$ is then transformed to H.264 to obtain an 8x8-transform coefficient block in H.264 ($B_{8x8}^{up}$):

$$
B_{8x8}^{up} = (H_{8} \times \hat{b}_{8x8} \times H_{8}^{T}) \times N_{8}
$$

(55)

For I frames we perform up sampling in the pixel domain by first taking an inverse VC-1 transform followed by zero insertion followed by filtering.

As we can see in figures 45 and 47 the quality of the original image is greatly reduced when resolution change is used, therefore we apply our SR quality enhancement
with irregular sampling method of section 6.3 to improve the reconstructed video quality in transcoding.

![Image](image.png)

**Figure 47.** Result of using upsampling by four on an image

### 6.3 Quality Enhancement using Irregular Sampling

#### 6.3.1 Introduction to Super Resolution

Super-resolution (SR) is the task of estimating high-resolution (HR) frames from a set of low-resolution (LR) observations. In video sequences these observations are acquired by a camera imaging the scene over a period of time, the critical requirement is that the observations contain different but related views of the scene. Usually the term SR applies to a signal processing approach to image reconstruction, other approaches like reducing pixel size depends on the current image sensor technology but this approach is usually more costly and is not compatible with existing sensors [35]. The problem of video resolution enhancement is currently of great importance due to the emergence of high definition (HD) displays, a good technical overview of SR can be found in [35, 36]. One application for SR is the display of SDTV bitstreams on HD devices or to display low quality videos generated by mobile devices in high quality displays, increasing the spatial
resolution of the original bitstream is thus necessary (e.g. up sampling), which implies that a higher resolution frame has to be obtained by up sampling from its low resolution version. Another important application of SR is the synthetic zooming of region of interest for surveillance, medical, and satellite imaging. The addition of high frequency information in such cases is a well-known ill-posed problem as it increases the noise. This high frequency additional information may be obtained from a group of several shifted versions of the low resolution image, a collection of optimal estimated filters selected for specific image content, or from a training set that combines low and high resolution image pairs [35, 37]. Many video SR algorithms borrow from still image SR solutions, but video processing presents several complications, different objects move at different speed in the scene, movement of objects can result in parts of the scene being cover/uncovered and the need for faster processing. [53] examined the LR image acquisition process as a whole, the method assumes affine motion in the sequence and considers the point spread function to be known. [37] proposed an example based image magnification method using a large database to add high frequency information. For scalable video coding and distributed video coding some frames (key frames) are sent at a HR, [54] uses this key frames to super-resolve the LR non-key frames, the motion estimation process in performed using blocks of band-pass versions of the frames, rather than low pass ones. [55] uses an adaptive framework where the target frame is divided into adaptive-sized blocks, which are classified into categories by their features, and then different conventional SR algorithms are applied to the blocks in different categories, a de-block process is also applied to reduce the block edge effects. In [56] a solution is presented where a relatively simple block-processing linear minimum mean squared error
(LMMSE) spatial-domain interpolation is used; the solution is similar to previous approaches for still-image scenarios but with modifications to improve performance and tackle some complications that arise specifically for the case of video. In [52], an iterative and temporally recursive technique is used to improve the resolution of a video sequence. In this dissertation, a novel Super-Resolution (SR) algorithm based on Irregular Sampling (IS) is presented. The high resolution (HR) frame is obtained as an interpolation of one or more previous frames; the resulting interpolated frame has samples non-uniformly spaced in the areas where movement happened. To reconstruct the irregular sampled frame we use well known irregular sampled algorithms (modified to perform in 2D space). There are several articles that explain irregular sampling theory in depth [21, 22], and in particular how to use of IS for static image quality enhancement [23-25]. To the authors knowledge IS has not been used for video quality enhancement.

Figure 48. Super resolution example

As part of the proposed SR algorithm we need motion vectors (MVs) that describe the trajectory of objects in the scene. Since block based motion estimation
proposed by video compression standards [1, 2] are not suitable for our proposed algorithm since often are unreliable and do not represent the actual motion. Finding true MV is beyond the scope of this work, any true-MV algorithm will work with our proposed SR method, for prove of concept we use the true MV algorithm described in [38] since is very efficient and hardware friendly.

The basic idea of super-resolution can be found in figure 48, SR combines information from a set of successive low-resolution (LR) frames of the same scene to generate a relatively high resolution (HR) image by extracting additional information from successive low resolution frames with sub-pixel displacements.

6.3.2 Irregular Sampling Theory

Sampling functions can be categorized into regular and irregular ones. Sampling on regular grids, based on Nyquist sampling theorem, is efficient and simple but some amount of aliasing is unavoidable. Intuitively, sampling with regular spacing is wasteful where no variation in the signal exits. The irregular sampling problem is concerned with the problem of recovering a band-limited signal x[n] with bandwidth M from a sequence of samples which may be taken in an irregular way. Assigning lower sampling density to regions of low variation leads to reductions in the number of sample points. Since, in many problems, the number of samples is large, efficient reconstruction algorithms are required. While irregular sampling in one dimension is well developed both in theory and its numerical exploitation [22, 23], there exit only few efficient reconstruction methods for the two dimensional problem [24, 25]. Gröchening [21] analyzed the irregular sampling problem. The main lemma can be summarized as follows [25]: Let \( A \) be a bounded operator on a Banach space \( B \) such that \( \| I - A \|'_\ast < 1 \) where \( \| \cdot \|'_\ast \) denotes the
operator norm on $B$ and $I$ is the identity operator. Then $A$ is invertible on $B$ and

$$A^{-1} = \sum_{n=0}^{\infty} (I - A)^n.$$ Moreover, every $f \in B$ can be reconstructed by the iteration

$$\begin{align*}
\phi_0 &= Af \\
\phi_{n+1} &= \phi_n - A\phi_n \\
f &= \sum_{n=0}^{\infty} \phi_n
\end{align*}$$

with convergence in $B$. The operator $A$ which includes the sampling and aperture functions must be bounded with $\|I - A\| < 1$. Gröchening showed that if $f$ is band-limited on a Banach space and the sampling is $\delta$-dense with $\delta^* \omega < \ln(2)$, where $\omega$ represents the highest frequencies present in $f$, then $f$ can be reconstructed from its samples using the previous iteration.

For 1-D signals the irregular sampling theory is briefly explained as follows. Let $s$ be a real-valued one-dimensional signal, band-limited in frequency. This means that $s$ has spectrum in some interval $[-\Omega_s, +\Omega_s]$, it is possible to reconstruct the original signal $s$ from samples taken in a non-uniform (irregular) way if the maximal distance between two consecutive sampling times does not exceed the Nyquist period $T_s = \frac{\pi}{\Omega_s}$. Most irregular reconstruction algorithms are iterative in nature [22]. Starting from some initial guesses, typically based on the given sampling values, further approximations of $s$ are obtained step by step, using the available (assumed) knowledge about $\Omega_s$. This is the case of the Allebach [23] algorithm, which is made of 3 steps. Step 1 consists of the interpolation of the sampling values. The interpolated signal contains many high frequencies outside of $[-\Omega_s, +\Omega_s]$. In step 2 the interpolated signal is low-pass filtered.
with a cutoff frequency slightly greater than $\Omega_s$. Step 3 is the recursive reconstruction of the error, if significant, so that we can again recover a certain portion of the remaining signal by repeating the first two steps. More discussions can be found in [21-25].

6.3.3 Irregular Sampling for Video Super-Resolution

In this section we use a modification to the irregular sampling algorithm to obtain an improved video quality.

Frame $N$ is an interpolation of frames $t0$ and $t1$, were $t0$ is the current frame with regular samples at red (circular) grid points, and $t1$ is the frame at $t1=t0+1$ regularly sampled at the blue (squared) grid points. $MV1$ represents the true motion vector from $t0$ to $t1$. Frame $N$ contains irregular samples for objects with motion.

**Figure 49.** Video quality enhancement using irregular sampling.

The basic idea for video quality enhancement with super-resolution can be shown in figure 49 and is based on motion compensation interpolation (MCI); the novel idea used
in this article is that we use irregular sampling techniques to reconstruct the interpolated frame. The proposed algorithm works as follows.

Step 1: Obtain true motion vectors \((MV_1)\) between frames current \(f_{\text{curr}}\) and previous \(f_{\text{prev}}\).

To obtain the true motion vectors of the video sequence we use the 3-D recursive method described in [38].

Step 2: Interpolation. Create an interpolated frame \(f_N\) with \(f_{\text{curr}}\) and the data of \(f_{\text{prev}}\) motion compensated (MC) with the MVs obtained in step 1. Frame \(N\) will have irregular sample patterns in objects that have motion.

\[
f_N = \text{Interpl} (f_{\text{curr}}, \text{MC}(f_{\text{prev}})). \quad (56)
\]

Step 3: Irregular Sampling. Use irregular sampling (section 6.3.2), but modify it to 2-D, to reconstruct \(f_N\) at a higher resolution. We use Allebach [23] algorithm with Voronoi interpolation (because it is simple to implement and has very good PSNR performance, although time complexity is slightly worse than other algorithms [24]) modified as follows.

a) Interpolation. Voronoi (nearest neighbor) interpolation is used. When two or more points are at the same distance then the mean is used.

b) Reconstruction filtering. We use the Sinc2D function \(\text{Sinc2D}(x,y)\) and the Blackman2D window \(W(t)\) to improve the frequency response when truncating its support. The magnitude response of the filter \(S'\) is shown in figures 50 and 51.

\[
\text{Sinc2D}(x,y) = \text{Sinc}(x) \ast \text{Sinc}(y) \quad (57)
\]

where \(\text{Sinc}(t) = \frac{\sin(\pi t)}{\pi t}\), \(x\) and \(y\) are the coordinates of the image.
\[ W(n) = 0.42 - 0.5 \cos\left(\frac{2\pi n}{w}\right) + 0.08 \cos\left(4\pi \frac{n}{w}\right) \]  \hspace{1cm} (58)

where \(0 \leq n \leq w\), \(n\) is the sample number, \(w = N-1\), and \(N\) is the supported size of the filter. The result of the reconstructed filter is \(S'(x,y)\):

\[ S'(x, y) = w(x)w(y)Sinc2D(x, y) \]  \hspace{1cm} (59)

![Figure 50. Magnitude response N=2*w-1=19 (2 sides)](image-url)
6.3.4 Use of Different Interpolation Techniques

In [24] a summary of different interpolation techniques for use in irregular sampling is presented. According to [24] a method based in adaptive weights, block Toeplitz matrix and conjugate gradient method (ABC) requires less floating point operations (flops) to reach a certain precision than the Voronoi interpolation, since Voronoi interpolation at each iteration step has to establish a nearest neighborhood tessellation of the sampled image. The rate of convergence is better for the first interactions when using Voronoi; the reason for this is that Voronoi expends much computational effort in computing a good first approximation. We implemented Voronoi interpolation for the simplicity of it, but
for real-time application the ABC interpolation has lower computational complexity and could be used.

6.3.5 Number of Frames Used for Interpolation

Irregular sampling reconstruction works well when there are no large gaps in the sampling. If an image is sampled near the Nyquist density or if there are large gaps in the sampling set, we need to interpolate more than one frame to create frame $f_N$. There is no easy way of knowing in advance if the true MVs would create a “big” gap when used to interpolate for frame $f_N$. Therefore, if using one frame enhances the resulting quality by less than 1 dB we would then use a second frame to improve interpolation.

6.3.6 Hardware Feasibility Study

Figures 52 and 53 present a high-level description of how the hardware to perform IS could work. True MVs are extracted from current reconstructed luma MBs and at least one reference frame. The reference frame pixels are pulled-back on an over-sampled grid to its motion-compensated position. The use of Voronoi interpolation filters up the gaps on unknown over-sampled points. Then a low pass filter (LPF) is used to remove high frequency artifacts. This may destruct known samples at the irregular sampled points; to recover them the process is repeated on each iteration on the errors generated on the known samples. The final improved image is generated as the cumulative sum of the resulting images at the iterations. The number of iterations is controlled by the magnitude of the resulting errors up to a maximum given by the hardware cycle budget. The algorithm can be easily implemented in hardware attractive block-based manner by including the overlapping neighborhoods using steps like Voronoi interpolation and LPF. Block-based motion estimation and compensation requires accesses to the whole frame.
data or at the very least to a neighborhood as large as the maximum expected motion vector (MV), increases the difficulty to implement.

![Data flow diagram](image1)

**Figure 52.** Data flow diagram

![Hardware block diagram](image2)

**Figure 53.** Hardware block diagram

### 6.4 Results

We tested, as proof of concept, the algorithm in section 6.3.3 on two simple video sequences “BeforeMan” and “Car” that contain two frames each of size 1920×2560. The second frame was displaced by one pixel to the right and 4 pixels down from the first frame. The frames were then down-sampled by four and enhanced with a conventional low pass filter and reconstructed with the irregular sampling algorithm described in
section 6.3. In addition, a standard video sequence was used to evaluate the performance of our down/up sampling with IS transcoder proposed in section 6.3, and also to compare the performance to other video super-resolution algorithms. The first eight frames of the “Mobile” video sequence 352×288 at 30 frames/sec were used. Tables IV and V show our encoder configurations (with [1] and [2] with JM 13.0).

6.4.1 Subjective Quality Evaluation
Figures 54-56 compare an enlarged portion of the frames (since picture quality is difficult to appreciate on printed paper a web page containing these results and some other experiments is provided at http://webpages.scu.edu/ftp/mpantoja/). The frames filtered using irregular sampling based in Allebach algorithm as proposed in this thesis (figures 54-56 e) have almost the same subjective quality as the original images (figures 54-56 a). Compared to the down-sampled images (figures 54-56 c) the quality-enhanced images (figures 54-56 e) preserve the high frequency components better, as can be seen on the neckline and zipper of “BeforeMan” and on the letters shown in “Car”.

6.4.2 PSNR Performance
The results of the PSNR values are shown in Table XI which summarizes the PSNR results of applying the down and up samplings and the quality enhancement to the video sequences “BeforeMan” and “Mobile”. Our enhancement improves the video quality by about 1-2 dB on the average as shown in Table XI, which is significant.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>BeforeMan</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Down-sampled sequence</td>
<td>29.16</td>
<td>17.58</td>
</tr>
<tr>
<td>Up-sampled sequence</td>
<td>32.67</td>
<td>19.13</td>
</tr>
<tr>
<td>Down-sampled + Quality enhancement</td>
<td>32.77</td>
<td>18.42</td>
</tr>
<tr>
<td>Up-sampled + Quality enhancement</td>
<td>34.79</td>
<td>20.32</td>
</tr>
</tbody>
</table>

6.4.3 Time Complexity Evaluation

The time complexity of the algorithm was also analyzed. Table XII reports the computational time complexity per each main block of the quality enhancement algorithm. From [24] SA-Voronoi takes $2.5 \times 10^5$ flops to reach an error of less than $10^{-4}$ for an image of $(512 \times 750)$; from [38] the true MV search would take 10 operations/pel. It is clear from Table 4 that the most time consuming operation is to find the true MVs. This step is a required step to perform a reasonable quality enhancement (QE) anyway even if a different QE algorithm is used. As proven in [24], irregular sampling is an efficient method to perform QE. The number of operations per pixel cannot be estimated for the transcoder since we used VC-1 and H.264 reference software which are implemented in C. Comparing the time to perform transcoding with and without IS enhancement, we obtained an increase in computational time by about a factor of 2.5.
### Table XII.
**Computational time complexity.**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>BeforeMan</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME for True MVs</td>
<td>10 op/pixel</td>
<td>10 op/pixel</td>
</tr>
<tr>
<td>IS Voro.-Alleb.</td>
<td>0.65 op/pixel*</td>
<td>0.65 op/pixel*</td>
</tr>
<tr>
<td>IS ABC</td>
<td>0.13 op/pixel**</td>
<td>0.13 op/pixel**</td>
</tr>
</tbody>
</table>

(*) $2.5 \times 10^5$ and (**) $0.5 \times 10^5$ flops to reach an MSE $= 10^{-4}$

#### 6.4.4 Comparison with Other SR Algorithms

We choose to compare our algorithm with the super-resolution algorithm described in [52] that uses motion compensation and a multiple input algorithm (MIA), and to bilinear interpolation. Since the PSNR results reported in [52] are difficult to replicate, we actually compare the reported improvement in PSNR over bilinear interpolation (figures 54-56 d). MIA showed approximately a 1 dB improvement over using bilinear interpolation. Our proposed IS algorithm also showed an improvement of approximately 1 dB over bilinear interpolation. Table XIII shows the PSNR performances for the different algorithms.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mobile PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bilinear</td>
<td>18.98</td>
</tr>
<tr>
<td>[12]</td>
<td>1(*)</td>
</tr>
<tr>
<td>Proposed-Allebach</td>
<td>1(*)</td>
</tr>
</tbody>
</table>

(*) Improvement over bilinear interpolation
e) Down by 4 and QE with IS

**Figure 54.** Video quality enhancement using irregular sampling for “BeforeMan”
Figure 55. Video quality enhancement using irregular sampling for “Car”
6.5 Summary

Results show that the resolution conversion transcoder on the DCT domain has better performance than on the pixel domain. But still there is an important drop in the video quality, to further improve the resulting quality we implemented also a novel quality enhancement algorithm based on super-resolution method with irregular sampling. The proposed solution is not intended for any specific application, but we have specifically tested the algorithm in a transcoding application. Experiments show that our method improves video quality significantly.

Figure 56. Video quality enhancement using irregular sampling for “Train”
Chapter 7

Rate Control Transcoding

7.1 Rate Control in VC-1 and H.264

For encoders without rate control video can be compressed with a constant quantization parameter (QP), the output video frames may have similar quality but the bit-rate fluctuates depending on the different content complexity of each frame. These fluctuations are sometimes unacceptable and most of the time constraints are imposed by the encoder buffer size and network bandwidth, which make the uses of rate control algorithm a necessity. Take, for macroblock (MB) level rate control, for example, to keep a desired bit-rate we would need to determine the number of bits at the MB level and dynamically vary the QP based upon estimates of the source complexity, so that each MB gets an appropriate allocation of bits.

7.1.1 VC-1 Rate Control

VC-1 [1] has not adopted any specific rate control algorithm, as part of the standard, although the reference encoder provides rate control capabilities. The rate control in the reference decoder depends on the settings of the Hypothetical Reference Decoder (HRD). If the number of HRD leaky buckets is zero then the sample encoder uses a fixed
quantizer setting for all pictures. This quantizer value is entered by the user in the configuration file and its value ranges from 1 (best quality) to 31 (poor quality). If at least one HRD leaky bucket is defined in the configuration structure then the sample encoder will attempt to meet the requirements of all the HRD leaky buckets. If the choice of quantizer causes a buffer overflow, then the sample encoder replaces the picture by a skipped frame.

VC-1 allows the use of two quantizers, a regular uniform quantizer where quantization intervals are the same which performs well at high bit rates, and a uniform quantizer with a “dead-zone” where all intervals are the same except for the one surrounding zero which leads to substantial bit savings at low bit rates. The encoder typically uses dead-zone quantizers at larger step sizes, and uses uniform quantizer at lower step sizes.

VC-1 encoder calculates a number of variables to implement the rate control described above; the variables can be used to accelerate the rate control algorithm used by H.264. The variables are, at frame layer: the VC-1 encoder buffer size \( W_{VC-1} \), the boolean variable \( S_{K(VC-1,n)} \) that determines if frame \( n \) is skipped, and the frame-layer target bits for current frame \( B_{vc-1} \). For MB-layer: the variable calculated is the \( QP_{VC-1} \) selected. To improve the performance of rate control for transcoding a scene change estimator is also determined. Section 7.2 describes how these parameters are used.

7.1.2 H.264 Rate Control

H.264 also uses HDR to enable cost-effective encoder implementations. The algorithm for rate control adopted by H.264 is described in [26]. It is more complex than [27, 28] since quantization parameters are specified in both rate control (RC) and rate distortion
optimization (RDO) and the statistics for the current frame are not available until RDO is performed; but RDO needs the value of QP, which is calculated by RC. The dilemma is solved by defining a basic unit in which all MBs have the same QP and by calculating the complexity of the current basic unit. The mean absolute difference (MAD) (or another similar complexity estimate) can be estimated by a linear prediction model using the MAD of the basic unit from the collocated position of the previous frame. The coefficients of the prediction model are updated after coding each basic unit. To compute the target bits for the current frame a fluid traffic model is used and to determine the quantization step \( QS \), the following second order equation can be used [57]:

\[
\text{Texture}_{-}\text{Bits} = c_1 \cdot \frac{\text{MAD}}{QS} + c_2 \cdot \frac{(\text{MAD})^2}{QS^2}
\] (60)

where \( c_1 \) and \( c_2 \) are constant value coefficients that may be estimated empirically and dynamically updated; \( QP \) is related to \( QS \) as discussed in [2]. Note that \( QS \) doubles in size for every increment of 6 in \( QP \). H.264 uses a scalar quantizer. The quantization step size is chosen by \( QP \), which supports 52 different quantization parameters.

Once the QP is calculated H.264 performs RDO to select the MB type to be used. The coding mode for each block is determined by using the Lagrangian cost function described in [51]. The Lagrangian mode decision for a MB, \( S_k \), proceeds by minimizing the following cost function (61).

\[
D(S_k, I_k \mid QP) + \lambda_{\text{MODE}} R(S_k, I_k \mid QP)
\] (61)

where \( I_k \) is the set of MB types, \( D \) is the distortion, \( R \) is the rate, and \( \lambda_{\text{MODE}} = 0.85 \times 2^{(QP-12)/3} \) typically.

7.2 Low Complexity
7.2.1 Frame Layer Rate Control

VC-1 I, P, and B frames are mapped to H.264 I, P, and B frames, respectively. To improve the performance of our transcoding algorithm we also detect scene changes. When a scene change happens, the estimates become erroneous and the performance of the rate control algorithm suffers. VC-1 has a frame type, BI, which is a B-frame where all macroblocks are intra-coded, but the frame cannot be used as a reference. This type of frame is mostly used when a scene change happens in the video sequence. In our transcoder rate control we use it as a prediction for a scene change. Therefore if the frame in VC-1 is a BI frame we map all macroblocks in this frame as intra in H.264 and reuse the $Q_{P_{VC-1}}$, as described in section 7.2.2. In cases where a very low target bit-rate is required and frame skip is necessary, to avoid drift error we only skip B frames (figure 57).

![Diagram](image-url)  
**Figure 57.** Frame Layer Rate Control
7.2.2 MB Layer Rate Control

The rate control for transcoding presented in this paper modifies the QP derived by the rate control in VC-1 to accelerate the computation in the RC/RDO algorithm in H.264. The modified QP is used for transcoding I/P/B frames. In VC-1, QP ($QP_{VC-1}$) has values that range from 1-31, in H.264 QP ($QP_{H.264}$) values range from 1-52. To find the relation between the QP in both standards the following experiment was performed with several video sequences:

1. Compress a video sequence using VC-1 for all QPs possible, no rate control turned on.
2. Compress the same video sequence as in step 1 using H.264 for all QPs possible with no rate control turned on.
3. Graph the bit-rate vs. QP for the two standards (figure 58) and obtain the relationship between $QP_{VC-1}$ to $QP_{H.264}$ using regression analysis (figure 59).

We performed steps 1-3 for different sequences to ensure repeatability of the results. Figures 58 and 59 show average results for the sequences used. From the experiments we conclude that:

a) VC-1 can only achieve the same bit-rates as H.264 if the QP for H.264 ranges from 10-29.

b) The polynomial that best adjusts the data (figure 59 $QP_{VC-1}$ vs. $QP_{H.264}$) is a second-degree polynomial obtained by using non-linear least squares regression.

$$QP_{H.264} = a \cdot QP_{VC-1}^2 + b \cdot QP_{VC-1} + c,$$

(62)
where $a=-0.02$, $b=1.10$, and $c=9.92$. The coefficient of correlation is $r=0.985$. Note that a linear equation has also been tried but the coefficient of correlation was very low and the resulting transcoded performance is affected.

**Figure 58.** QP vs. bit-rate for VC-1 and H.264

**Figure 59.** $QP_{VC-1}$ vs. $QP_{H.264}$ and second-degree polynomial

### 7.2.3 Low-Rate Control Tools

VC-1 implements as part of its standard some low-rate tools. To allow coding frames at
multiple resolutions, VC-1 scales down the $X$, $Y$, or both $X$ and $Y$ dimensions of each coded frame. The decoder is informed that these frames have been scaled down, and it up-scales the decoded image in $X$ and $Y$ before displaying them. By operating at a down-scaled level, we are able to effectively extend the range of quantization beyond the normal range (1-31) by a factor of $\sqrt{2}$ each time we down-scale any dimension by a factor of 2. For main and advanced profiles VC-1 allow to automatically use rate reduction when encoding video at very low bit rates, or to manually set the rate reduction parameters in the configuration file. To be able to map $QP_{VC-1}$ if rate reduction is used we follow the same principles described in previous section 7.2.2 (figures 60 and 61). We obtain the following equation:

$$QP_{H.264} = d \ast QP_{VC-1}^2 + e \ast QP_{VC-1} + f,$$

(63)

where $d=-0.02$, $e=0.99$, and $f=14.85$. The coefficient of correlation is $r=0.985$.

![Figure 60. QP vs. bit-rate for VC-1 (using low rate tool) and H.264](image)
Equations (62) and (63) were obtained using a wide range of video sequences but obviously, as with all equations found empirically, they cannot be generalized for all video sequences. But they are, as proven by the results presented in 7.4.1, a good approximation and at the very least can be used as a seed for the QP mapping.

7.2.4 Rate Control in Transcoding

Once we have the mappings to obtain $Q_{P_{H.264}}$ we estimate the value of block mode $\lambda_{\text{MODE}}$ in Equation (61). Cutting down the complexity in obtaining the values of $Q_{P_{H.264}}$ we have significantly reduced the complexity of the H.264 rate control algorithm performed in transcoding.

7.3 Medium Complexity Rate Control

By using the tools described in section 7.2 we observed that the target bit-rates can be maintained without a significant loss in PSNR only for $Q_{P_{H.264}}$ ranges of 10-29. To
increase the range we need to implement a more complex RC transcoder. There are several VC-1 variables that can be re-used to implement it.

7.3.1 Complexity Estimation

For the transform domain transcoder used for P and B frames, where there is no motion estimation (ME) performed (Chapter 4) (due to motion vectors re-use), it is difficult to obtain a value for MB complexity estimation (MAD) [34]. Therefore to estimate the complexity for the MB we use a different complexity estimator, the sum of absolute transform differences (SATD), calculated using the sum of the absolute values of the frequency transform of the residuals. Since we transcode from VC-1 to H.264 the SATD is calculated as:

\[ SATD = \sum_{i,j} |C(residual)_{i,j}|, \]

where \( VC \) is the VC-1 forward transform. SATD predicts visual quality more accurately than MAD or sum of absolute differences (SAD) from the standpoint of objective and subjective metrics. Also, SATD is more convenient for transform domain transcoding, which only partially decodes incoming P/B frames to the transform coefficient level.

For pixel domain transcoding for I frames, we use MAD as the incoming frames are decoded by pixel domain transcoding to the pixel level and because of its simplicity of calculation. Also the MAD in the pre-coded sequence follows very close the MAD of the transcoded sequence at MB and frame level [29].

7.3.2 Texture Bit Allocation

In H.264, bit allocation is typically performed per group of pictures (GOP), even though the standard does not specify a GOP structure (and neither does VC-1). GOPs are usually used to simplify rate control tasks and to allow predictable insertion of intra frames where
the application needs, and to perform bits per frame allocation. Bits per frame allocation in H.264 ($BitsF_{H.264}$) and per basic unit (BU) ($BitsBU_{H.264}$) can be dynamically adjusted based upon the value of VC-1 bits per frame ($BitsF_{VC-1}$).

With the value of the complexity estimation for VC-1 (section 7.3.1) and the value of texture bits assigned by the rate controller for VC-1, we substitute these two values into Equation (60) to obtain the value of QS; reducing the complexity for H.264 rate control greatly. As mentioned in section 7.2.4, once we have the value of $QP_{H.264}$ we can estimate the value of block mode ($\lambda_{MODE}$) in Equation (61).

To improve the performance of texture bit allocation we could also take into account the complexity of the frame. It is common to have fluctuations on actual texture bits with respect to the allocated texture bits. These fluctuations are usually due to the fact that QP changes need to be constrained or smoothed to avoid drastic changes in visual quality that would otherwise produce visible artifacts in the picture; a rate limiter is applied which typically limits changes in QP to no more than ±2 units. VC-1 measures texture complexity to allocate bits per frame. Information provided by VC-1 can alert ahead of time of especially complex regions of the picture which allows generating a corrected target for H.264 texture bits, at both frame and BU levels.

Providing $N$ frames of buffering between VC-1 and H.264 allows us to look into the future at the characteristics of the sequence on an $N$-frame sliding window ahead in time. A small $N$ is required for low latency applications and is an interesting problem in minimizing buffering memory requirements (for future study). For our experiments in section 7.4.2 we use $N = 1$. On the other hand, large $N$ allows us to get benefits similar to dual-pass off-line encoder, where a sequence is encoded once and the RC parameters
generated are used to re-adjust the second pass compensation for the excess or lack of bits with respect to overall sequence target.

![Flow chart of rate control algorithm](image)

**Figure 62.** Flow chart of rate control algorithm

### 7.4 Results

Figure 62 presents a flow chart of the proposed RC algorithm. The H.264/AVC reference software used in our experiments was JM13.1. The video sequences used for our tests
were “Foreman”, “Claire”, and “Walk” 176x144, and “Foreman” 352x288. The sequences were encoded at 30 fps. To run these experiments a Dell Inspiron 300m running at 1.2 GHz and 256 MB memory was used. The video sequence we used was IPPP.

7.4.1 Low Complexity Tools

In Table XIV we compare the results for two video sequences with a target bit-rate of 1,024 kbps. RC refers to cascade transcoding method with rate control turned on for both VC-1 and H.264; and ORC refers to our rate control algorithm. We see that there is no significant loss in PSNR in our case while the bit-rates are almost the same.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Actual bit-rate (kbps)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RC</td>
<td>ORC</td>
</tr>
<tr>
<td>Foreman qcif</td>
<td>1026.14</td>
<td>1024.56</td>
</tr>
<tr>
<td>Claire qcif</td>
<td>1024.65</td>
<td>1024.34</td>
</tr>
<tr>
<td>Walk qcif</td>
<td>1024.15</td>
<td>1024.28</td>
</tr>
<tr>
<td>Foreman cif</td>
<td>1024.03</td>
<td>1024.32</td>
</tr>
</tbody>
</table>

Table XV shows computational complexity results for the rate control transcoder, in terms of the time units used to encode the different sequences by using the cascaded VC-1 decoder - H.264 encoder transcoding structure and the time units used by our proposed transcoder model. By using the proposed transcoding we reduce the average time used to encode a sequence by around 55% without a significant loss in PSNR.
### Table XV

**Rate-Control Transcoding Time Units**

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Cascaded</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman qcif</td>
<td>19.4</td>
<td>9.2</td>
</tr>
<tr>
<td>Claire qcif</td>
<td>16.2</td>
<td>6.1</td>
</tr>
<tr>
<td>Walk qcif</td>
<td>17.3</td>
<td>8.1</td>
</tr>
<tr>
<td>Foreman cif</td>
<td>78.2</td>
<td>37.3</td>
</tr>
</tbody>
</table>

7.4.2 *Medium Complexity Tools*

In Table XVI, we demonstrate comparisons of the results for the video sequences for target bit-rates of 256 and 2,000 kbps. The comparisons are between the fully cascaded transcoder with rate control turned on for both VC-1 and H.264 and our rate control algorithm applied to the pixel/transform domain VC-1 to H.264 transcoder. From the table we see that there is no significant loss in PSNR for our cases while the bit-rates are almost the same.

Table XVII shows the computational complexity results for our medium complexity rate control transcoder as compared to that of the fully cascaded transcoder with rate control turned on. We can deduce from the table that we reduce the average time for transcoding by about 53% overall.
## Table XVI.

**TRANSCODING BIT-RATE AND PSNR RESULTS**

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Actual bit-rate (kbps)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fully cascaded (RC turned on)</td>
<td>Our rate control transcoder</td>
</tr>
<tr>
<td><strong>Target bit-rate = 256 kbps</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Foreman</td>
<td>256.61</td>
<td>256.5</td>
</tr>
<tr>
<td>Claire</td>
<td>250.51</td>
<td>250.72</td>
</tr>
<tr>
<td>Walk</td>
<td>256.12</td>
<td>256.34</td>
</tr>
<tr>
<td><strong>Target bit-rate = 2,000 kbps</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Foreman</td>
<td>1960.30</td>
<td>2000.21</td>
</tr>
<tr>
<td>Claire</td>
<td>1980.13</td>
<td>2000.34</td>
</tr>
<tr>
<td>Walk</td>
<td>2000.09</td>
<td>2000.17</td>
</tr>
</tbody>
</table>

## Table XVII.

**RATE CONTROL TRANSCODING TIME UNITS**

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Fully cascaded (RC turned on)</th>
<th>Our rate control transcoder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Target bit-rate = 256 kbps</td>
<td></td>
</tr>
<tr>
<td>Foreman</td>
<td>9.4</td>
<td>4.8</td>
</tr>
<tr>
<td>Claire</td>
<td>8.7</td>
<td>4.0</td>
</tr>
<tr>
<td>Walk</td>
<td>10.1</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td>Target bit-rate = 2,000 kbps</td>
<td></td>
</tr>
<tr>
<td>Foreman</td>
<td>48.12</td>
<td>23.1</td>
</tr>
<tr>
<td>Claire</td>
<td>40.22</td>
<td>18.3</td>
</tr>
<tr>
<td>Walk</td>
<td>51.21</td>
<td>27.1</td>
</tr>
</tbody>
</table>
Figure 63 presents the number of bits obtained by the proposed and reference cascade algorithm we can see that the proposed transcoder closely follows the optimal cascade transcoder.

Claire qcif

Foreman qcif
Figure 63. The number of bits obtained by the proposed and reference cascade algorithm

7.5 Summary

From the discussions and experimental results the low complexity rate control has a better computational complexity performance than that of the medium complexity rate control but can only be used for H.264 QP from 10 to 29. The medium complexity transcoder applies to a wider range of QPs but with less improvement in complexity performance. Video quality and bit-rates are maintained in both cases as compared to the fully cascaded transcoder with regular rate control turned on. We therefore propose to use a two-level approach - the low complexity transcoder for QP ranges within 10 to 29, and the medium complexity transcoder for all other QPs. Experimental results show that the proposed combination of low and medium complexities rate control for our transcoding is less complex than that for a full cascaded transcoder with regular rate control, and yet maintaining the target bit-rate and PSNR.
Chapter 8

Conclusion

This dissertation addresses an important problem of transcoding VC-1 to H.264 coding format. Both VC-1 and H.264 are hybrid video coding algorithms that exploit motion compensation and transform coding. Two algorithms for transcoding VC-1 to H.264 are presented. First is a pixel domain transcoder based on exploiting the variable transform size used in VC-1 to determine MB coding mode in H.264. The results show that this approach works reasonably well by reducing the complexity by about 60% with negligible drop in PSNR. Second is a transform domain transcoder that uses the previous algorithm for I frames and matrix multiplication techniques to convert transform coefficients in VC-1 to transform coefficients in H.264 without having to calculate inverse transforms. The results show that this approach reduces the complexity but the PSNR drops by around 1 dB; to improve the PSNR value we add drift error compensation which increases the complexity slightly.

The two techniques, pixel and transform domain transcoding, can be jointly used at frame level. Pixel domain transcoding techniques shall be used for I frames, while transform domain transcoding can be used for P and B frames. Since P frames are used as
references the transcoder would typically produce better quality results if P frames are transcoded by pixel domain method. However, P frames can also be transcoded by transform domain method depending on the computational requirement of the system. On the other hand, B frames are most suitable for transform domain transcoding since they do not propagate errors. The combination of the two techniques can optimize transcoder systems in terms of both computational complexity and visual quality of the results.

We present a solution for the transcoder being used for resolution conversion. The down and up sampling algorithm presents results that are according to expectations. But to further improve the resulting quality we also implemented a novel quality enhancement based on super-resolution method with irregular sampling is proposed. The proposed solution is not intended for any specific application (the quality enhancement algorithm can be used in any situation where the quality of the video needs to be enhanced), but we have specifically tested the algorithm in a transcoding application. The algorithm to perform quality enhancement is based on super-resolution method with irregular sampling. Experiments show that our method improves video quality significantly.

At last we present a solution for the case in which the transcoder is used for rate control. We focus on rate-control VC-1 to H.264 video transcoding in which video quality is obtained while maintaining a target bit rate for the transcoded output. The relationship between rate control in VC-1 and H.264 is studied and we use the rate control information obtained from VC-1 encoding to simplify the rate control algorithm in H.264. First, a low complexity algorithm which key idea is to find the relationship between the QPs in VC-1 to the QPs in H.264, a medium complexity algorithm which
key idea is to use MAD (for frames transcoded using pixel domain transcoder) and SATD (for frames using transform domain transcoder) in VC-1, to estimate the complexity of the MB in H.264, and also re-uses VC-1 texture bit allocation to predict H.264 texture map allocation. The low complexity transcoder has a better complexity performance but can only be use for QP from 10 to 29. Therefore we propose to use the low complexity transcoder is used for QP range of 10 to 29, and the medium complexity transcoder for all other QPs. Experimental results show that the proposed combination of low/medium rate control for transcoding is less complex than that for a full cascaded decoder while maintaining the target bit rate and PSNR at any QP value.

8.1 Summary of Techniques for Standard Independent Transcoding

8.1.1 Pixel Domain Transcoding

8.1.1.1 I- Frames

Different standards have different ways of compressing I-Frames. There are two main ways standards perform prediction for intra compression; prediction at the pixel level (H.264) and DC prediction at the transform coefficients (VC-1).

a) For standards that use prediction at the pixel level.

The best way to transcode to these standards is to estimate the complexity of the MB and with this information find the best block size used for prediction. The measures of the MB complexity can be diverse. For this work we used the variance of the four DC coefficients in the MB, but this measure can be changed to adapt to different standards.

b) For standards that use DC prediction only.

Since DC prediction is easy and computationally inexpensive it can be simple reversed if the transcoding needs to do so.
8.1.1.2 P-Frames

Compression in P-Frames is done in most standards by performing motion estimation (ME) and compensation. The main problem a transcoder is going to have to solve is the motion vector reuse. A general solution to solve this problem is to use the MV from the initial standard as a seed for the ME algorithm in the second standard, and to adapt the ME search window to the MB content (high or low movement). To estimate the MB content a good solution is to use the MV magnitude.

8.1.1.3 B-Frames

If the standard has B-frames, the main problem (at least the problem that is going to be different than a P-Frame) is to map direct MB. If the first standard does not use direct MB, then a good solution is to predict their occurrence by inspecting the MV of the neighboring MB and the prediction error information of the MB.

8.1.2 Transform Domain Transcoding

There are two main problems a transform domain transcoder has to solve.

a) Coefficient conversion

Most standards apply some kind of transform to the input (pixel) data; this transform is done mainly by applying a matrix (kernel) to the data. Through matrix multiplication we can easily find the new kernel needed to perform transcoding in one step. The steps explained in section 4.1 can be easily adapted to other transcoders by just changing the kernels for VC-1 and H.264 to the new standards to be transcoded.

b) Drift error analysis
Since drift error analysis is done at the transform domain level, to compensate for drift error we have to use techniques similar to performing MC on the transform domain as explained in section 4.3.

8.1.3 Resolution Conversion

It has been proven that the best way to perform resolution conversion is at the DCT coefficient level [19, 20]. Therefore a transcoder used for resolution conversion should implement a solution based in the DCT coefficients.

8.1.4 Rate Control

A rate control transcoder main function should be to find a relationship between the QP in both standards. This can be done if possible using the QP equations but if this is not possible it can be tried to find a empirical relationship between both standards as proposed in Chapter 7.2

8.1.5 Quality Enhancement

Until recently quality enhancement in video has been based in algorithms used in 2D images. A better solution for video quality enhancement is to use motion compensation interpolation (MCI) to get the extra information needed. In this dissertation we present a method based in MCI using irregular sampling. This method can be reused to enhance quality in video independently of the standard used.

8.2 Future of Transcoding

The success of consumer products requiring video access has been made possible partially by improvements on video compression and transmission. Ideally, there would be only one video compression standard in the market, but the truth is that right now there are several standards concurrently. Since some of these standards work better for
specific applications (VC-1 for video streaming, MPEG-2 and H.264 for broadcasting, H.263 for wireless) they are most probably going to be used for years to come, making the need for transcoding an important one. According to a study conducted by market research analyst International Data Corp. (IDC) [76] on November 2006, the development of multi-format transcoding is a critical challenge that needs to be addressed in order to drive the continued expansion of the video entertainment market. The main factors that will drive the increase in demand for transcoders are:

- Increase in demand for video on mobile phones
- Increase in demand for HD content
- Increase in demand of Internet video
- Increase in on-demand video content
- Competition between Pay-TV service providers
- Migration from MPEG-2 to H.264 in video broadcasting
- Entering new geographic regions and market segments offers growth prospects

Because video standards are rapidly changing, a hardwired device is not the best solution. There are several companies developing products that use DSP-based system-on-chip (SoC) for real-time, multi format, HD video transcoding.

Another solution that is being considered by MPEG is the development of Reconfigurable Video Coding standard (RVC). RVC aims to provide a framework allowing dynamic development, implementation and adoption of standardized video coding solutions. So far the specification of video coding standards has been done case by case providing textual and reference software specifications, but without sufficient
consideration for the problems of establishing an effective implementation flow process that, starting from the standard specification itself, provides efficient paths to software or hardware implementations. The goal of RVC Standard is to offer a more flexible use and faster path to innovation of MPEG standards in a way that is competitive in the current dynamic environment, and to provide a high level specification model for direct and efficient software and hardware synthesis [77].
REFERENCES


GLOSSARY

**AC coefficients** transform coefficients for which the frequency in one or both dimensions is nonzero.

**Basic Unit (BU)** could be a frame, slice, macro block etc, the “basic unit” determines for which distinct values the QP are calculated.

**Bitstream** An ordered series of bits that forms the coded representation of the data.

**Bitrate** The rate at which the coded bitstream is delivered to the input of the decoder.

**Block** 8x8 matrix of samples, or 64 transform coefficients.

**Byte** Sequence of 8 bits.

**CIF** Specifies a data rate of 30 frames per second (fps), with each frame containing 288 lines and 352 pixels per line.

**Codec** Stands for Coder/Decoder. A codec is a piece of hardware or software that compresses and decompresses digital audio and/or video.

**Chrominance** The portion of a video signal that specifies what color each portion of the picture is to be.

**Compression** Reduction in the number of bits used to represent an item of data.

**DC coefficient** Transform coefficient for which the frequency is zero in both directions.

**Decoder** A piece of hardware or software that is used to convert video or audio from the digital form used in transmission or storage into a form that can be viewed.

**Frame Rate** The rate at which frames are output from the decoding process.

**Group of Pictures (GOP)**. Specifies the order in which intra-frames and inter frames are arranged. The GOP is a group of successive pictures within an MPEG-coded video.
stream. Each MPEG-coded video stream consists of successive GOPs. From the MPEG pictures contained in it the visible frames are generated.

**H.261** ITU standard for video coding for videoconferencing. H.261 is a discrete cosine transform (DCT) based algorithm for video in the 64kb/s to 2mb/s range. All H.323 compliant video conferencing systems are required to support this codec.

**H.263** ITU standard for video coding within videoconferencing. H.263 offers better compression than H.261, particularly in the low bitrate range used by modems.

**H.264** The latest ITU standard for video compression. It is based on MPEG-4 and renders roughly equal video quality with H.263, but at half the bit rate (e.g. 256 Kbps instead of 512 Kbps for an H.263 stream. It was rectified in July 2003.

**Interlace** The property of frames where alternating lines of the frame represent different instances in time. In an interlace frame, one of the field is meant to be displayed first.

**Irregular Sampling (IS).** (See section 6.3)

**Luminance** The portion of a video signal that specifies how bright each portion of the picture is to be.

**Macroblock (MB)** The four 8x8 blocks of luma data and the two corresponding 8x8 blocks of chroma data coming from a 16x16 section of the luma component of the picture in 4:2:0 format.

**Mean Absolute Difference (MAD)** Is the average absolute deviation from the mean.

**Macro Block Adaptive Frame Filed (MBAFF)** In H.264 interlace mode, when the frame is selected as frame coded, then each pair of vertically adjacent MBs can again be coded together or as two separated field MBs using adaptive frame-field.
**Motion Compensation (MC)** The use of motion vectors to improve the efficiency of the prediction of sample values. The prediction uses motion vectors to provide offsets into the past and/or feature reference frames or reference fields containing previously decoded sample values that are used to form the prediction error.

**Motion Compensation Interpolation (MCI)** The use of frames at different times to create an intermediate frame.

**Motion Estimation (ME)** The process of estimating motion vectors

**Motion Vector (MV)** A two dimensional vector used for motion compensation that provides an offset from the coordinate position in the current picture or field to the coordinates in a reference frame or field.

**MPEG**

MPEG (Moving Picture Experts Group) is a series of ISO standards for digital video and audio, designed for different uses and data rates.

MPEG-1 - The initial MPEG standard, designed to encode full motion video so it could be played back off of a CD (150 kb/s).

MPEG-2 was a follow-on standard supporting higher data rates, and thus higher quality. MPEG-2 is the standard used in DVD video players, most digital satellite systems in North America, and in the new North American Digital TV system.

MPEG-3 was abandoned, as its planned functionality was included in MPEG-2.

MPEG-4 supports a wide variety of elements that can be transmitted separately and combined to form the video frame, such as a talking head in one stream and the background in another. That is, MPEG4 allows manipulation of objects within the video stream (addition, subtraction, object manipulation, etc.).
MPEG-7 is a developing standard for the description of multimedia objects. Not a video encoding format, it is a way to describe elements in a multimedia stream so that they can be accessed via database.

MPEG-21 defines a "Rights Expression Language" standard as means of sharing digital rights/permissions/restrictions for digital content from content creator to content consumer.

**NP-hard** A problem \( H \) is NP-hard if and only if there is an NP-complete problem \( L \) that is polynomial time Turing-reducible to \( H \), i.e. \( L \leq_T H \).

**Overlap Smoothing Transform (OLT)** The filtering operation that is conditionally performed across edges of two neighboring Intra blocks, after inverse transform and prior to the loop filter.

**Peak Signal to Noise Ration (PSNR)** is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale.

**QCIF** A standard related to CIF, QCIF (Quarter CIF) transfers one fourth the amount of data and is suitable for videoconferencing systems or telephone lines.

**Quantization** A process in which the continuous range of values of an input is divided into a non-overlapping sub range and a discrete, unique value is assigned to each sub range.

**Quantization Parameter (QP)** non-overlapping sub range of values used in quantization.
Rate Control (RC) rate control algorithm regulates the rate at which each frame enters the network. Its primary goal is to converge on a set of per-flow transmission rates that prevents congestion collapse from undelivered packets.

Rate Distortion Optimization (RDO) rate control algorithm regulates the rate at which each flow enters the network. Its primary goal is to converge on a set of per-flow transmission rates that prevents congestion collapse from undelivered packets.

Range Reduction A process of rescaling decoded pixel values in main profile.

SMPTE Society of Motion Picture and Television Engineers or SMPTE. International professional association, based in the United States of America, of engineers working in the motion imaging industries.

Sum of Absolute Transform Differences (SATD) is a widely used, extremely simple video quality metric used for block-matching in motion estimation for video compression. It works by taking the absolute value of the difference between each pixel in the original block and the corresponding pixel in the block being used for comparison. These differences are summed to create a simple metric of block similarity.

Super-resolution (SR) Techniques that in some way enhance the resolution of an imaging system. There are different views as to what is considered an SR-technique: some consider only techniques that break the diffraction-limit of systems, while others also consider techniques that merely break the limit of the digital imaging sensor as SR.

Transcoder A device that does transcoding.

Variable Length Coding (VLC) A reversible procedure for coding that assign a shorter code-word to symbols of higher probability and longer code-words to symbols of lower probability.
VC-1 is the informal name of the SMPTE 421M video codec standard initially developed by Microsoft. It was released on April 3, 2006 by SMPTE. It is now a supported standard for HD DVDs, Blu-ray Discs, and Windows Media Video 9.

**Weighted Prediction (WP)** allowing an encoder to specify the use of a scaling and offset when performing motion compensation, and providing a significant benefit in performance in special cases—such as fade-to-black, fade-in, and cross-fade transitions. This includes implicit weighted prediction for B-frames, and explicit weighted prediction for P-frames.

**Zigzag Scanning order** A specific sequential ordering of the transform coefficients from the lowest special frequency to the highest
PUBLICATIONS

Journals:


Conferences:


• M. Pantoja and N. Ling, “A Two-Level Rate Control Approach for Video Transcoding,” *Proc. IEEE Int. Conf. on Image Processing*, Cairo, Egypt Nov. 2009 (accepted for publication).
Maria Pantoja was born in Merida, Spain. She received her B.S. and M.S. degree in engineering from Universidad Politecnica de Valencia, Spain, in 1994, and a second M.S. in computer science from California State University of the East Bay, Hayward, CA, in 2004. She received her Ph.D. from the Department of Computer Engineering at Santa Clara University (SCU) in 2009. She has worked as a senior software engineer for Logical Automation, JDS Uniphase and Nuko. She received the Packard fellowship in 2008 and worked as a teaching assistant for the department of computer engineering at SCU. She is currently a full time lecturer at the computer engineering department at SCU. Her current research interests lie in the areas of video compression, video transcoding and reconfigurable video coding. She is a member of the IEEE.