Performance Analysis and Comparison of JM 15.1 and Intel IPP H.264 Encoder and Decoder

K.V.Suchethan Swaroop and K.R.Rao, *IEEE Fellow*
Department of Electrical Engineering, University of Texas at Arlington
Arlington, USA
e-mail: suchethanswaroop@gmail.com, rao@uta.edu

Abstract— Joint Model (JM) reference software is used for academic reference of H.264 and it was developed by JVT (Joint Video Team) of ISO/IEC MPEG & ITU-T VCEG (Video coding experts group). The Intel IPP (Integrated Performance Primitives) H.264 is a product of Intel which uses IPP libraries and SIMD instructions available on modern processors. The Intel IPP H.264 is multi-threaded and uses CPU optimized IPP routines. In this paper, JM 15.1 and Intel IPP H.264 codec are compared in terms of execution time and video quality of the output decoded sequence. The metrics used for comparison are SSIM (Structural Similarity Index Metric), PSNR (Peak-to-Peak Signal to Noise Ratio), MSE (Mean Square Error), motion estimation time, encoding time, decoding time and the compression ratio of the H.264 file size (encoded output). Intel IPP H.264 clearly emerges as the winner against JM 15.1 in all parameters except for the compression ratio of H.264 file size.

Keywords- H.264/AVC, Intel IPP H.264, JM 15.1, Performance comparison

I. INTRODUCTION

H.264 or AVC (Advanced Video Coding) is the latest digital video codec standard which has proven to be superior to earlier standards in terms of compression ratio, quality, bit rates and error resilience [1]. It was developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG) as the product of a collective partnership effort known as the Joint Video Team (JVT) [1]. H.264 delivers stunning quality at remarkably low data rates [7]. Ratified as part of the MPEG-4 standard (MPEG-4 Part 10), this ultra-efficient technology gives excellent results across a broad range of bandwidths, from 3G (3rd Generation) for mobile devices to iChat (Instant chatting application) for video conferencing to HD for broadcast and DVD [7].

II. PREDICTION OF INTER-CODED MACROBLOCKS IN P-SLICES IN H.264

Inter prediction includes both motion estimation (ME) and motion compensation (MC) processes to perform prediction. It generates a predicted version of a rectangular array of pixels, by choosing another similarly sized rectangular array of pixels from a previously decoded reference picture and translating the reference array to the position of the current rectangular array [4]. The translation from other positions of the array in the reference picture is specified with quarter pixel precision [4]. H.264 supports motion compensation block sizes ranging from 16x16 to 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4 sizes as shown in Figures 1 and 2. This method of partitioning macroblocks into motion compensated sub-blocks of varying size is known as tree structured motion compensation [6]

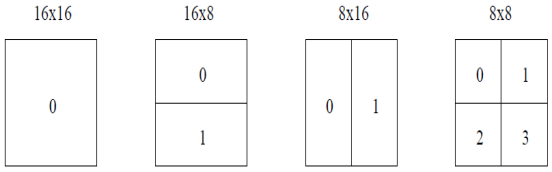

Figure 1: MB partitions: 16x16,16x8,8x16 and 8x8

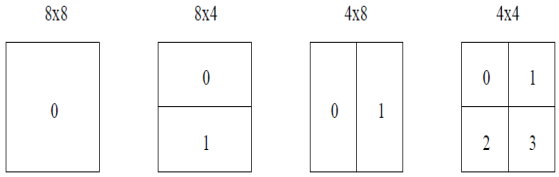

Figure 2: MB partitions: 8x8,8x4,4x8 and 4x4

For evaluating the motion vectors, each partition in an inter-coded macroblock (MB) is predicted from an area of the same size in a reference picture. The offset between the two areas (the motion vector) has ¼-pixel resolution (for the luminance component). The luminance and chrominance samples at sub pixel positions do not exist in the reference picture and so it is necessary to create them using interpolation from nearby image samples. If components of the motion vectors are integers, the relevant samples in the reference block actually exist. If one or both vectors components are fractional values, the prediction samples are generated by interpolation between adjacent samples in the reference frame. Sub-pixel motion compensation can provide significantly better compression performance than integer-pixel compensation, at the expense of increased complexity [6].The MB mode decision in Inter frames is computationally the most expensive process due to the use of the features such as variable block-size, motion estimation, and quarter-pixel motion compensation. The union of all mode evaluations, cost comparisons and exhaustive

search inside ME block causes a great amount of time spent by the encoder. Complex and exhaustive ME evaluation is the key to good performance achieved by H.264, but the cost is in the encoding time [6].

III. JOINT MODEL (JM) REFERENCE SOFTWARE

The JM reference software has been developed by JVT. The JM 15.1 [13] has a configuration file through which the input parameters can be specified.

The command used to execute the encoder is

lencod –f encoder.cfg

The command used to execute the decoder is

ldecod –i testfile.264 –o testoutput.yuv

IV. INTEL IPP H.264

Intel® Integrated Performance Primitives (Intel® IPP) [2] is an extensive library of highly optimized software functions for digital media and data-processing applications. Intel IPP offers thousands of optimized functions covering frequently-used fundamental algorithms. Intel IPP functions are designed to deliver performance beyond what optimized compilers alone can deliver [2]. It also has a parameter file called h264.par wherein the input parameters can be given.

The command used to execute the encoder is
umc_video_enc_con.exe h264 h264.par testfile.h264

The command used to execute the decoder is
umc_h264_dec_con.exe -i testfile.h264 -o testoutput.yuv

V. INPUT PARAMETERS AND TEST SEQUENCES

A. *Input Parameters*

The parameters that are changed in the configuration file (JM) and the parameter file (Intel IPP) were frame width, frame height, frame rate, YUV format, number of frames to be encoded, profile, bit rate, Quantization parameter, Intra period and number of reference frames to be used.

B. *Test Sequences*

1. CIF (*Common Intermediate Format*) [5] is a video format used in video conferencing systems. CIF is part of the ITU H.261 videoconferencing standard [3]. It specifies a data rate of 30 frames per second (fps), with each frame containing 288 lines and 352 pixels per line. Hence it has a resolution of 352 x 288 [3]. Table 1 contains basic information of "football.yuv".

Frame rate (fps)	30
Width	352
Height	288
Size (MB)	13
Number of frames	90

Table 1: Basic information for CIF sequence "football.yuv" [5]

2. QCIF (*Quarter CIF*) [5] is related to CIF. Its resolution per frame is 144 lines, with 176 pixels per line. Multiplying 352 and 288 yields 101,376 total pixels in the CIF format, which is exactly four times the number of pixels contained in a QCIF frame (25,344). The "Quarter" terminology is meant to indicate that QCIF frames contain quarter as many pixels as the CIF frame and thus take up less bandwidth [3].Table 2 contains basic information about QCIF "foreman.yuv" sequence.

Frame rate (fps)	30
Width	176
Height	144
Size (MB)	11.138 MB
Number of frames	300

Table 2: Basic information for QCIF sequence "Foreman.yuv"

VI. VIDEO QUALITY ASSESSMENT

The metrics used to calculate the video quality of the decoded sequence are

- Structural similarity index metric (SSIM)
- Peak to peak signal to noise ratio (PSNR)
- Mean square error (MSE)

The MSE and its derivative PSNR are conventional metrics to compare any two images. MSE measures the difference between the original and distorted pixels. PSNR is a logarithmic representation of the inverse of this measure. Compared to other objective measures, PSNR is easy to compute and well understood by most researchers. However both MSE and PSNR do not correlate well with the subjective quality of the reconstructed images. The subtle differences between degradations of different intensities are not properly reflected using PSNR. The SSIM proved to be a metric that was closest to a human perception of the received video sequence. This method utilizes structural distortion as an estimate of perceived visual distortion, where as most other proposed approaches are error sensitivity based methods [8].

VII. EXPERIMENTAL RESULTS

1. The results obtained from JM 15.1 for "Foreman.yuv" QCIF sequence are tabulated as shown in Table 3.

Bit rates used (KB/Sec)	Motion estimation time (sec)	Total encoding time (sec)	Decoding time (sec)	H.264 file size
25	1396.664	1658.874	15.824	251 KB
50	1426.045	1829.835	18.384	501 KB
75	1442.001	1951.315	19.059	751 KB

100	1376.785	1941.312	17.805	1.001 MB
125	1398.593	2045.541	19.375	1.251 MB
150	1509.317	2280.951	19.307	1.501 MB

Table 3: Results obtained from JM 15.1 for "Foreman.yuv" QCIF sequence

The results obtained from Intel IPP for "Foreman.yuv" sequence are tabulated as shown in Table 4.

Bit rates used KB/sec	Motion estimation time (sec)	Total Encoding time (sec)	Decoding time (sec)	H.264 file size
25	2.51	3.41	0.1257	294 KB
50	2.58	3.31	0.1519	562 KB
75	3.07	3.93	0.1061	824 KB
100	2.88	3.74	0.1232	1.082MB
125	2.94	3.77	0.1386	1.334 MB
150	3.25	4.13	0.1577	1.583 MB

Table 4: Results obtained from Intel IPP for "Foreman.yuv" QCIF sequence

From Tables 3 and 4, it is evident that the encoding time, motion estimation (ME) time and decoding time of Intel IPP H.264 is more than 480 times faster than JM 15.1.

A. *SSIM and PSNR*: The values of SSIM and PSNR obtained for JM 15.1 and Intel IPP H.264 at different bit rates are tabulated in Table 5.

Bit rate (KB/sec)	JM (SSIM)	INTEL IPP (SSIM)	JM (PSNR in dB)	Intel IPP (PSNR in dB)
25	0.9609	0.96297	37.18359	37.46751
50	0.98146	0.98186	41.13141	41.28917
75	0.98789	0.98876	43.17939	43.7612
100	0.99063	0.99221	44.37394	45.63609
125	0.99227	0.99419	45.26677	47.11219
150	0.9934	0.99546	45.95815	48.34548

Table 5: Values of SSIM and PSNR achieved by JM and Intel IPP for "foreman.yuv" QCIF sequence

Figures 3 and 4 show the plots of SSIM and PSNR achieved by JM and Intel IPP for different bit rates. It is evident from these graphs that Intel IPP achieves a slightly greater SSIM and PSNR than JM 15.1.

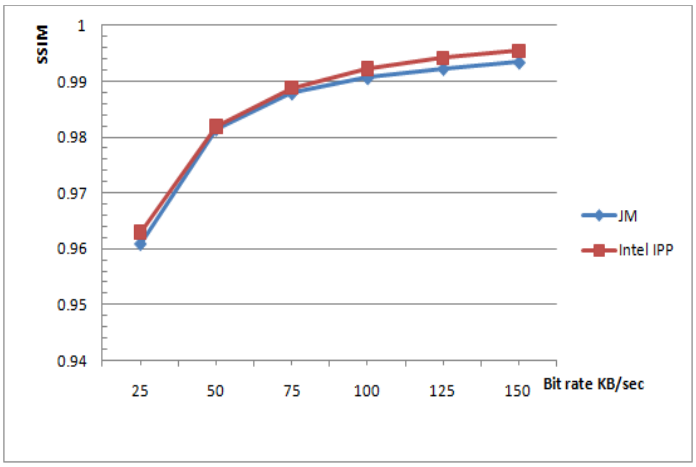

Figure 3: Comparison of SSIM achieved by JM and Intel IPP

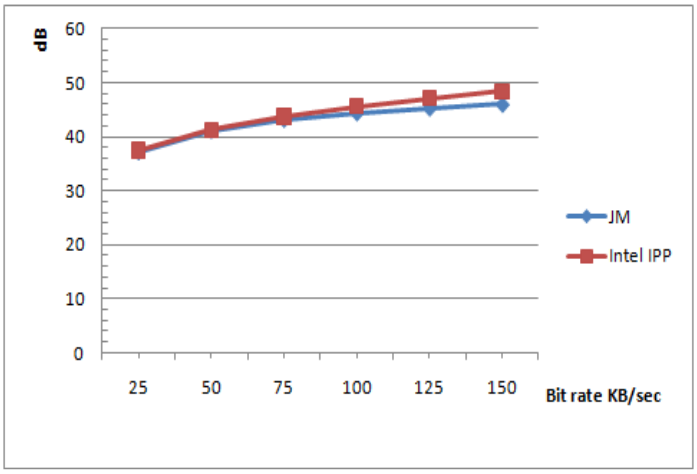

Figure 4: Comparison of PSNR achieved by JM and Intel IPP

B. *MSE and compression ratio (CR):*The values of MSE and compression ratios obtained for JM 15.1 and Intel IPP H.264 at different bit rates are tabulated in Table 6. Compression ratio has been calculated as the ratio between the original sequence file size to H.264 file size.

Bit rate KB/sec	JM (MSE)	INTEL IPP (MSE)	JM (CR)	Intel IPP (CR)
25	12.43717	11.6501	44.37	37.92
50	5.01117	4.83241	22.23	19.82
75	3.1271	2.73502	14.83	13.52
100	2.37513	1.77612	11.12	10.3
125	1.93376	1.26433	8.9	8.35
150	1.64917	0.95177	7.42	7.04

Table 6: Values of MSE and compression ratios achieved by JM and Intel IPP for foreman sequence

Figures 5 and 6 show the plots of MSE and compression ratios achieved by JM and Intel IPP. From Figure 5, it is evident that

the MSE achieved by Intel IPP is slightly lower than JM 15.1 at all bitrates. From Figure 6, it is evident that the compression ratio achieved by JM 15.1 is better than Intel IPP H.264 at all bitrates.

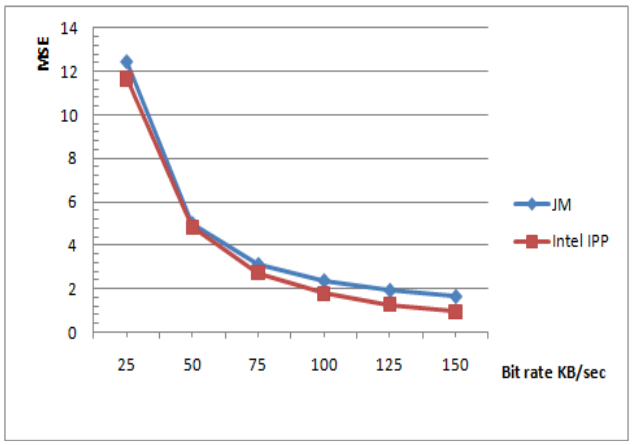

Figure 5: Comparison of MSE achieved by JM and Intel IPP

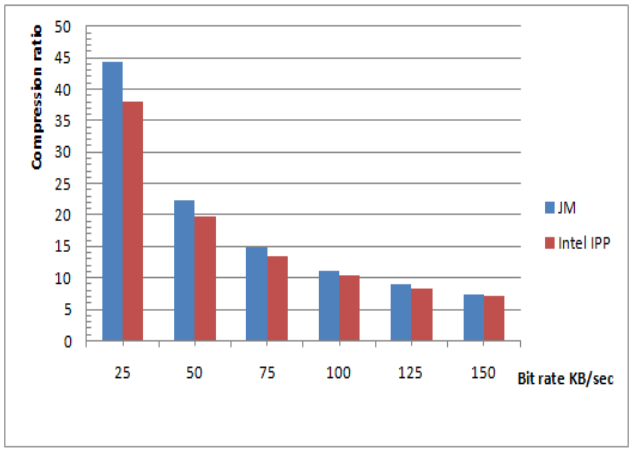

Figure 6: Compression ratios of JM 15.1 and Intel IPP

2. The results obtained from JM 15.1 for "Football.yuv" CIF sequence is tabulated in Table 7.

Bit rate (KB/sec)	ME time (sec)	Encoding time (sec)	Decoding time (sec)	H.264 File size (KB)
25	1867.852	1977.511	7.46	115
50	1895.832	2011.676	8.643	151
75	1870.535	1994.897	8.793	225
100	1787.468	1914.402	10.001	300
125	1762.494	1894.901	9.81	375
150	1752.587	1889.357	10.397	450

Table 7: Results obtained from JM for football CIF sequence.

The results obtained from Intel IPP H.264 for "Football.yuv" CIF sequence are tabulated in Table 8.

Bit rate (KB/sec)	ME time (sec)	Encoding time (sec)	Decoding time (sec)	H.264 File size (KB)
25	1.47	1.77	0.0483	88
50	1.96	2.27	0.0567	186
75	2.25	2.58	0.0642	280
100	2.38	2.71	0.0629	372
125	2.47	2.93	0.0628	465
150	2.53	2.86	0.0898	554

Table 8: Results obtained from Intel IPP for football sequence.

From Tables 7 and 8, it is evident that Intel IPP H.264 is more than 660 times faster than JM 15.1 in terms of encoding time, ME time and decoding time.

A. *SSIM and PSNR:* The values of SSIM and PSNR obtained for JM 15.1 and Intel IPP H.264 at different bit rates are tabulated in Table 9.

Bit rate KB/sec	JM (SSIM)	Intel IPP (SSIM)	JM (PSNR in dB)	Intel IPP (PSNR in dB)
25	0.78131	0.76785	29.33657	29.15126
50	0.82761	0.8471	30.98771	32.14291
75	0.8734	0.8876	33.01121	34.08814
100	0.90238	0.91482	34.73708	35.63852
125	0.92068	0.93117	36.00283	36.81034
150	0.93351	0.94339	37.02637	37.82698

Table 9: Values of SSIM and PSNR achieved by JM and Intel IPP for football CIF sequence

From Table 9, it is clear that Intel IPP achieves better SSIM and PSNR than JM 15.1 at every bit rate except at 25 KB/sec. Figure 7 and Figure 8 show the plot of SSIM and PSNR achieved by JM and Intel IPP for different bit rates.

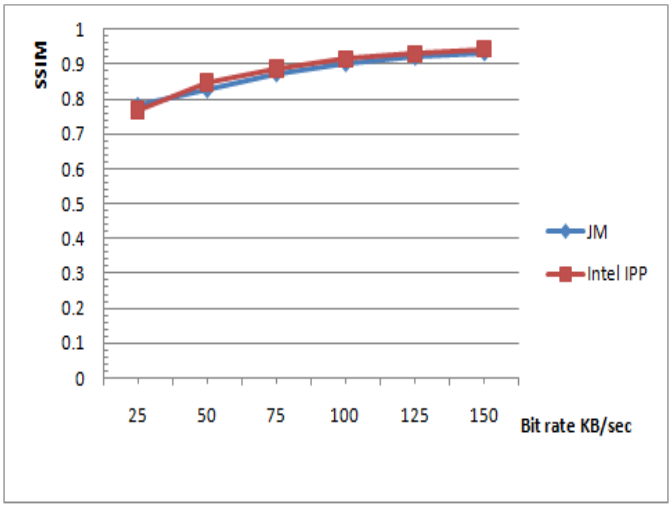

Figure 7: SSIM achieved by JM and Intel IPP

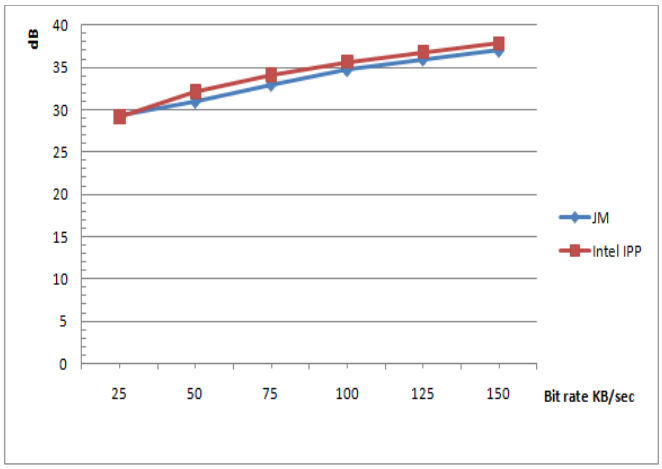

Figure 8: PSNR achieved by JM and Intel IPP for "football.yuv" CIF sequence

B. MSE and Compression ratio: The values of MSE and Compression ratios obtained for JM and Intel IPP at different bit rates are tabulated in Table 10 and the corresponding graphs are plotted in Figures 9 and 10.

bit rate (KB/sec)	JM (MSE)	Intel IPP (MSE)	JM (CR)	Intel IPP (CR)
25	75.7571	79.05947	116.21	151.87
50	51.79758	39.69989	88.5	71.85
75	32.50572	25.36683	59.4	47.73
100	21.84603	17.75123	44.55	35.92
125	16.32292	13.55333	35.64	28.74
150	12.89565	10.72463	29.7	24.12

Table 10: MSE and Compression ratios achieved by Intel IPP and JM 15.1 for football CIF sequence

From Table 10, it is clear that the MSE achieved by Intel IPP is less than JM in all cases except at 25 KB/sec. It is also evident that the compression ratio achieved by JM is better than Intel IPP in all cases except at 25 KB/sec.

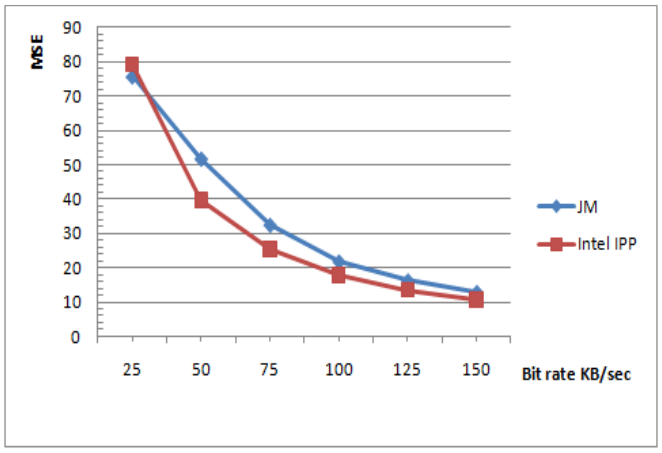

Figure 9: MSE achieved by JM and Intel IPP for "football.yuv" CIF sequence

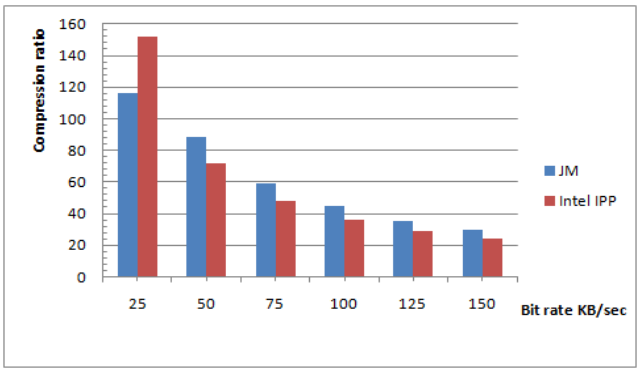

Figure 10: Compression ratio achieved by JM and Intel IPP

VIII. CONCLUSIONS

The motion estimation time, encoding time and decoding time are extremely less in Intel IPP H.264 compared to JM 15.1 and this is evident from the Tables 3,4,7 and 8. The quality of the video is also better than the JM output in terms of SSIM, PSNR and MSE in all the cases (except at bit rate of 25KB/sec CIF sequence). The only place where JM 15.1 scores over Intel IPP is the compression size of H.264 file. The maximum difference in size found was 104 KB (CIF sequence at bit rate of 150 KB/sec). The reason why Intel IPP is so fast is because it is multi-threaded and uses CPU-optimized IPP routines. It uses Intel IPP libraries to exploit benefits from SIMD (single instruction multiple data) instructions available on modern processors. In addition to this, it is also able to utilize multiple cores doing work in parallel. Hence, Intel IPP H.264 is highly optimized and emerges as the winner against JM 15.1 in all aspects measured except for compression size of H.264 file.

REFERENCES

[1] Soon-kak Kwon, A. Tamhankar and K.R. Rao "Overview of H.264 / MPEG-4 Part 10", J. Visual Communication and Image Representation, vol. 17, pp.186-216, April 2006.

[2] http://software.intel.com/en-us/ for Intel IPP software

[3] http://www.birds-eye.net/ for information about CIF and QCIF formats

[4] A.Puri, X.Chen and A. Luthra , **"** Video coding using H.264/MPEG-4 AVC compression standard**",** Science Direct. Signal processing: Image communication, vol.19, pp 793-849,Oct. 2004.

[5] Video test sequences (YUV 4:2:0): http://trace.eas.asu.edu/yuv

[6] P.Carrillo, H.Kalva, and T.Pin, "Low complexity H.264 video encoding", Applications of Digital Image Processing. Proc. of SPIE, vol. 7443, 74430A, Sept.2009.

[7] http://www.apple.com/quicktime/technologies for H.264 codec reference

[8] Z. Wang and A. C. Bovik, Modern Image Quality Assessment. Synthesis Lectures on Image, Video and Multimedia Processing. Morgan and Claypool, 2006.

[9] T. Wiegand and G. J. Sullivan, "The H.264 video coding standard", IEEE Signal Processing Magazine, vol. 24, pp. 148-153, March 2007.

[10] G.Sullivan and T.Wiegand, "Video compression – From concepts to the H.264/AVC standard," Proc. IEEE, vol.93, pp. 18-31, Jan.2005.

[11] D. Marpe, T. Wiegand and G. J. Sullivan, "The H.264/MPEG-4 AVC standard and its applications", IEEE Communications Magazine, vol. 44, pp. 134-143, Aug. 2006.

[12] http://iphome.hhi.de/suehring/tml/ for JM 15.1 software