IBBT

D1.1.4: Report on the final optimized video encoder prototype

IBBT-IMEC – 23 November, 2009

IBBT vzw, Interdisciplinair Instituut voor Breedbandtechnologie
Gaston Crommenlaan 8 (bus 102), 9050 Gent, België • E info@ibbt.be • T +32 9 331 48 00 • F +32 9 331 48 05 • www.ibbt.be



Revision History :

Version	Reviewer	Date
Creation	P. Rondao Alface	15 December 2008
V1	P. Rondao Alface	17 December 2008

Table of Contents

1. Introduction 4
2. Compression Pipeline Requirements 5
2.1. Description and Context 5
2.2. Requirements 5
2.3. Quality test 6
3. AVC/H.264 Encoder 8
3.1. AVC signal prediction and its impact on complexity 8
3.2. Video content description and AVC prediction 11
4. AVC intra encoding: state-of-the-art 14
intra mode decision 14
Introducing parallelism 15
Use of a GPU as co-processor 15
5. Optimization of the AVC Baseline Profile Intra encoder 16
5.1. Strategy 16
5.1.1. Slice parallelism 17
5.1.2. Macroblock-level parallelism 17
5.1.3. Vector-based parallelism 18
5.1.4. Algorithmic optimization 18
5.2. CPU “lossless” optimized implementation 19
5.3. CPU-GPU “lossy” optimized implementation 21
6. Conclusion 24
7. Bibliography 24

1. Introduction

IBBT URBAN deliverable D111 presented a benchmark of different state-of-the-art video codecs in the context of urban scenes digital surveying. The conclusions of this report have shown that given the (spatial and temporal) resolution of the video streams to be coded as well as the very nature of their content (motion characteristics, views, textures…) AVC/H.264 Baseline Profile Intra has been selected as the best solution for several reasons. The first one is linked to the decoding stage. The processing pipeline using the compressed bitstreams as input for the urban 3d reconstruction need a very fast random access to frames. In that case, accessing a frame inside a group of pictures (GOPs) with classical motion compensation would need to decode all previous frames in that GOP and a subsequent suboptimal system. Intra coding enables per-frame decoding access. The second reason is linked to the encoding stage and to the trade-off between complexity and rate-distortion performance.

Deliverable D112 provided then a functional implementation of the compression pipeline with debayering and AVC coding with no specific requirement on speed. The output quality after debayering and compression has been tested by GeoAutomation and satisfies the requirements of the 3d reconstruction accuracy.

Deliverable D113 was intended to be a prototype but we already delivered a report describing this prototype and the optimization techniques used in order to achieve the expected results. This report, deliverable D114, will then be very similar to deliverable D113.

This report is organised as follows. First the requirements of the compression pipeline are presented in Section 2. Section 3 is devoted to the AVC encoder structure, video content influences on performance. An overview of the state-of-the-art optimization techniques for AVC encoding is given Section 4. Section 5 is devoted to the description of our optimization strategy as well as to the experimental results. The deliverable finally concludes with a short summary of the achieved work.

2. Compression Pipeline Requirements

2.1. Description and Context

In WP1, 3D data is extracted for the masses of buildings in cities, with an emphasis on the precision. The goal is to allow cartographers at SPC to determine the 3D positions of the different objects needed by FGIA, and this within the prescribed precisions. GeoAutomation has developed a solution through which these measurements can be largely brought into the office, from behind a desktop. GeoAutomation has developed a van with eight cameras, and the necessary processing pipeline to extract 3D coordinates for points clicked on in an image. The traditional method would be to measure the points with GPS and photogrammetry, as field work. Now, only sparsely distributed reference points are measured on the field, to geo-reference the 3D data obtained from the imagery. Field work therefore is reduced to a minimum. This said, the actual data capture as well as the processing can be made more efficient. This is the main goal of this deliverable.

In the current configuration, every camera is connected to a single computer. The images are dumped in raw format (bayered images) on the disk, at a rate of about 20MByte/sec, resulting in data sizes in the order of one Tera-byte or more for one recording session. This is too much to store on the GeoAutomation servers. That is why the images are compressed, after they are debayered. At the moment, the JPEG2000 format is used. However, the on-board computers have low-performance CPUs, unable to perform this debayering and compression in a reasonable time. As a result, all the data must be transferred to the servers where the processing is done. A substantial gain could be achieved by performing the compression on-board during or after the recording. GeoAutomation has already tested PGF which is another wavelet-based encoding tool, for the encoding on the van. This solution is not real-time but already improves performances.

The video streams to be compressed in URBAN are recorded from a van capturing eight to ten different views with the objective of producing an accurate 3d reconstruction of urban environments. The streams are composed of BGGR-bayered data with a spatial resolution of 1628x1236 and a temporal resolution of 12 frames per second.

2.2. Requirements

De-bayering and compression should be performed on the van in order to improve its storage capacity (as described in deliverable D121 and illustrated on Figure 1). Their implementation should then be real-time (i.e. 12 fps). The reconstructed images after decoding should also be of sufficient quality to be re-used in the processing on the servers of GeoAutomation.

Deliverable D111 determined that AVC Baseline Intra is a good candidate for the GeoAutomation system as it provides low-complexity encoding and low complexity decoding as well (with respect to random frame access at the processing servers side). The reconstructed image quality has also been considered as satisfactory.

Quantified Requirements	Target
Compression pipeline speed (debayering+encoding)	12 fps
Reconstruction accuracy	Errors < 15cm
Envisaged Platform	Low-end (dual-core) PCs

Table 1 Requirements

The quality requirement is that for a given set of 20,000 images per view from compressed images, the final 3d reconstruction accuracy should be such that 3d point-wise errors should be inferior to 15cm as illustrated on
Table 1. There is some freedom for the platform to be used, but currently low-end PCs are used in the van.

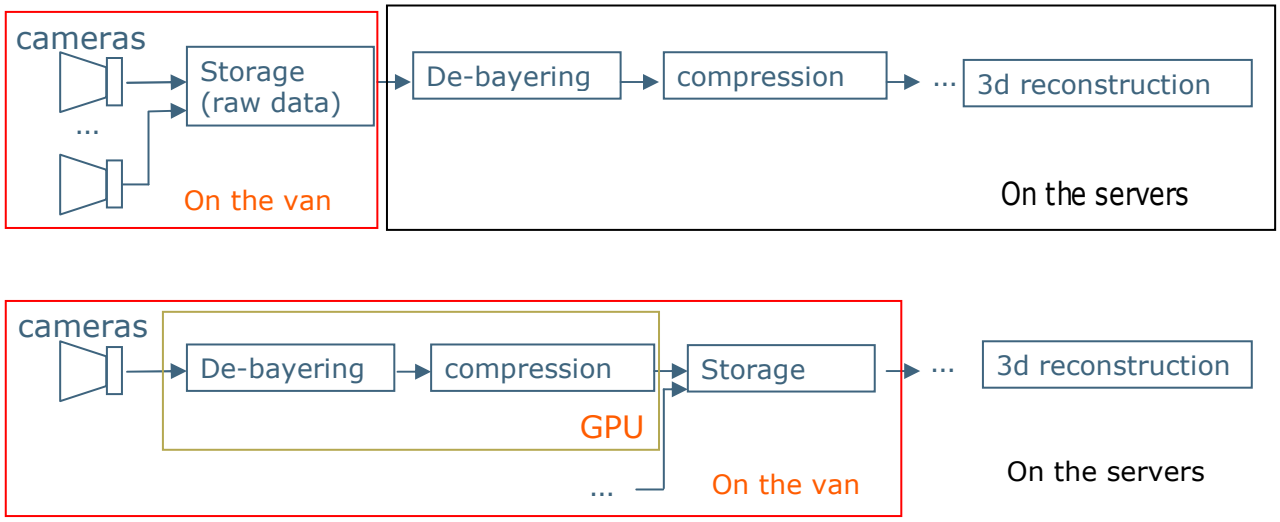


Figure 1: System composed of the pipeline on the van and the pipeline on the servers of GeoAutomation. Above: previous solution with raw image storage on the van. Below: on-board de-bayering and compression on GPU performed on the van.

2.3. Quality test

The quality test is performed by GeoAutomation on 8 views with 20,000 images. The comparison is performed between JPEG2000 and AVC intra coding (deliverable D112).

The selected quality for JPEG 2000 images is defined by their sizes: 295KB. For that experiment and a QP of 24, AVC bitstreams have a size of 350KB per image on average of all views. This is more than JPEG2000. However it can be mapped on the van on the contrary of JPEG2000 which is more complex.

The reconstruction is based on feature detection and matching. During SaM, feature tracks are extracted in consecutive images. Only points that are consistently reconstructed in 3D become part of a feature track.

JPEG2000: 101000 feature tracks
AVC: 92000 feature tracks

Reprojection error

After the SaM, ground control points (GCPs) are used to georeference the images. This is done in a global optimisation process (bundle-adjustment) which minimizes the reprojection error and the error on the GCPs. The mean reprojection error of all feature tracks is expressed in pixels.

JPEG2000: 1.126
AVC: 1.141

2D and 3D error

In the bundle adjustment, only 50% of the available GCPs are used, the other 50% are taken as test cases: the reconstruction of these points from the images should be as close as possible to the measured coordinates. Because the GCPs are usually measured using GPS for which the uncertainty in Z is known to be worse than in XY, we compute both the 2D (XY) and the 3D (XYZ) error. The mean error for all unused GCPs.

JPEG2000: 2D = 0.0597138 m; 3D = 0.113334 m
AVC: 2D = 0.0696471 m; 3D = 0.120172 m

The results for JPEG2000 and AVC are quite similar. The AVC scores are a bit worse but still within the accuracy needed.

3. AVC/H.264 Encoder

3.1. AVC signal prediction and its impact on complexity

In this Section, we give a short overview of an AVC/H.264 intra baseline profile encoder. The focus here is put on the complexity and data (content) dependency. For a more general introduction to the AVC scheme, the reader is referred to deliverable D111 and reference [1].

Figure 2 represents the AVC/H.264 video encoder scheme. The encoder is composed of some main functional components: the prediction, the residual (i.e. original signal minus the predicted signal) transform and quantization (transform), the entropy coding (CAVLC) and the rescaling and inverse transform (inverse transform).

- *The complexity of the prediction* is mainly due to its block-matching strategy using motion estimation or intra prediction (extrapolation). The error metric used to compute the optimal prediction or block-match is the sum of absolute differences (SAD). The number of tested positions and therefore, the number of SAD computations increases with the desired prediction quality. In most high quality AVC encoders, this is the bottleneck when the prediction is accurate.

- *The transform* is composed of two different steps. First the integer DCT which is called for 4x4 residual blocks and 8x8 residual chroma blocks. The output coefficients are then quantized with the selected input quantization parameter (QP). If the prediction is very accurate, the residual signal is very sparse and the transform will output a few coefficients which are likely to be removed after quantization. If the prediction fails, then a large number of quantized coefficients will be produced even for low quality (large QP). The number of output quantized coefficients is then increasing with the accuracy needed and the prediction inaccuracy.

- *The entropy coding* is a very complex tool which increases with the number of input quantized coefficients and parameters [1]. If the prediction is not accurate and constantly leads to too many coefficients, the encoding bottleneck might be shifted to the entropy coding. There are currently no studies in the literature providing a mathematical model for this complexity increase as it is very data-dependent.

- *The inverse transform* is similar to the transform and its complexity is reduced if many quantized coefficients are zero.

From these simple observations, the message is that the bottleneck is the prediction and that it should be optimized. However, we also know that if the prediction fails, the overall complexity is increased. There is therefore a trade-off to achieve so that the gain in complexity at one side is not lost in terms of encoding performance (quality and bitrate) as well as in terms of overall complexity.

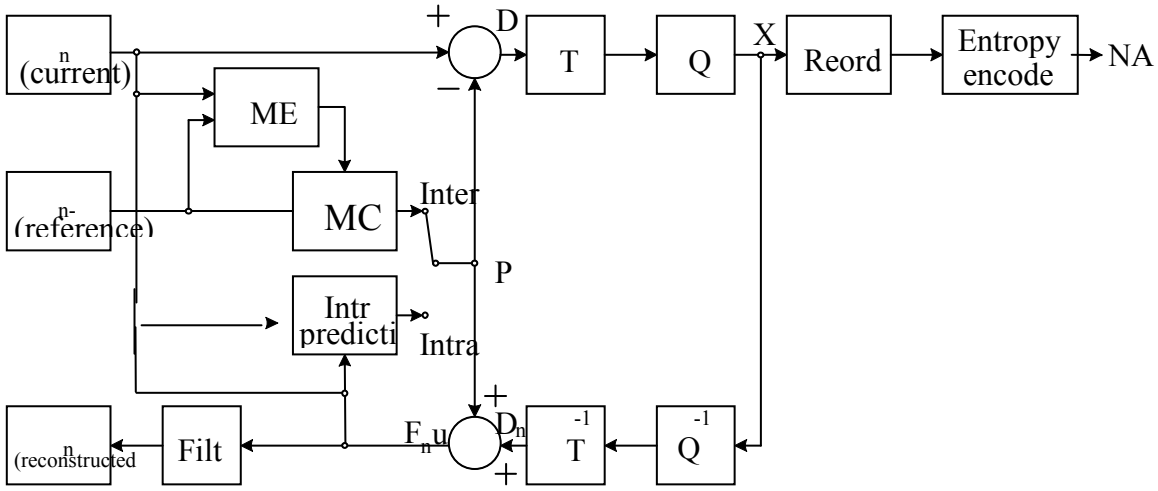


Figure 2 AVC encoder scheme

Following the conclusions of D111, the prediction to be used for the AVC encoding is intra prediction. This kind of prediction for a given macroblock (MB) uses the information already coded from previous neighboring MBs in order to extrapolate it using FIR filters [1]. Three different options are offered: 16x16 block prediction, 4x4 block prediction and 8x8 block prediction (the latter is used for chroma prediction) as depicted in figures 3 and 4. Four intra 16x16 modes and nine intra 4x4 modes can be used. These modes correspond to different extrapolation directions.

As it can be seen, the 4x4 modes enable a finer grain prediction which can better deal with high frequency details than 16x16 modes which enable a lower prediction complexity for low frequency content regions.

From our experiments, it turns out that using intra prediction for high frequency content (vegetation, leaves…), intra modes have all to be checked (higher complexity) and that the residual signal can still be non-sparse leading to a higher complexity at the entropy coding side. The next Section gives more highlight on this aspect.

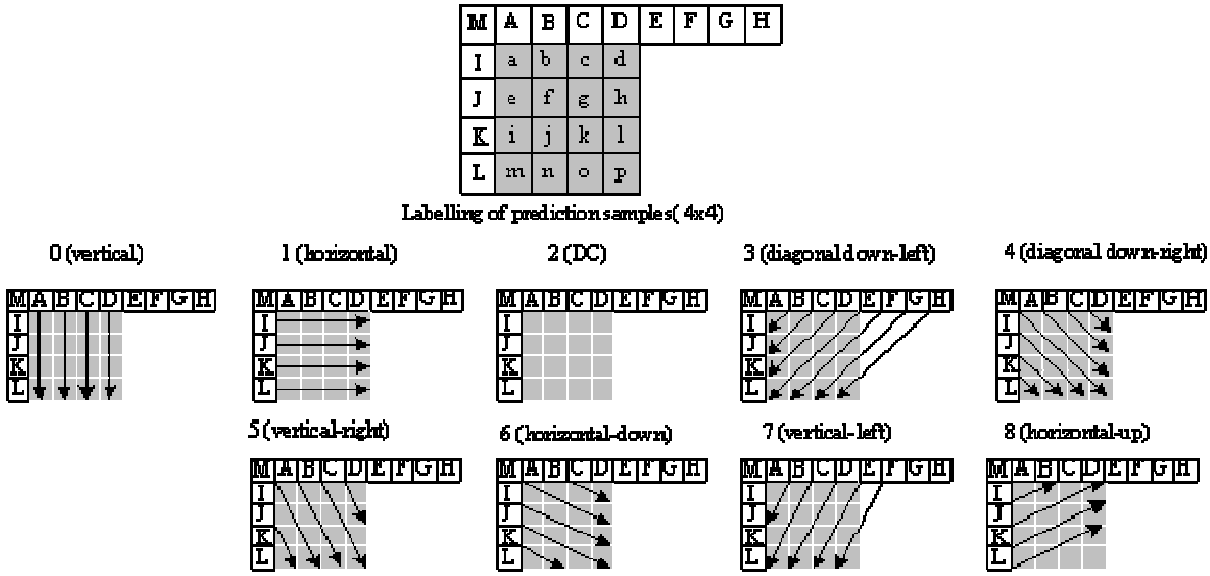


Figure 3 Intra 4x4 prediction modes

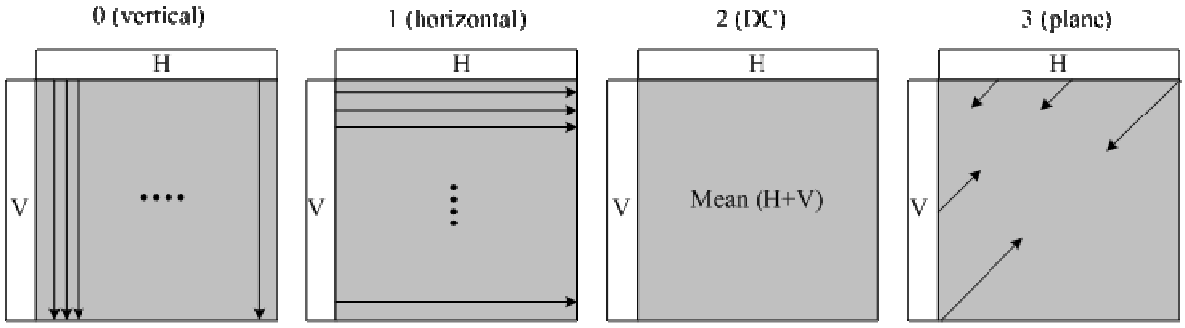


Figure 4 Intra 16x16 prediction modes

3.2. Video content description and AVC prediction

The video streams[1] to be compressed in URBAN are recorded from a van capturing eight to ten different views with the objective of producing an accurate 3d reconstruction of urban environments. The streams are composed of BGGR-bayered data with a spatial resolution of 1628x1236 and a temporal resolution of 12 frames per second. The impact of these features on the AVC prediction is different whether we consider motion estimation or intra prediction.

This combination of high spatial resolution with very low temporal resolution is not so usual in video coding applications and strongly limits the performance of motion estimation algorithms. Indeed, if the van drives at a speed of 50 km/h, then the camera displacement between two frames is of nearly one meter. In lateral views, depending of the distance between objects and the van, the motion between corresponding macroblocks between frames is very large and the theoretical motion vector has a prohibitive cost. For front and rear views, the zooming effect is difficult to predict at this low temporal resolution by motion estimation which is more intended for translational motion. For lateral views, the motion estimation works correctly and performs a lower (but still significant) intra mode selection. However, the increased complexity is not compensated by a gain in rate-distortion when compared to intra coding.



Figure 5 Intra prediction is dominant in motion estimation when the van is driving at moderate speed. Intra predicted macroblocks are represented in pink. Motion compensated macroblocks are represented in green.

[1] The interested reader can refer to D111 in order to see some snapshots of the input views to be compressed.

These observations are confirmed by our experiments, for lateral views, P slices contain a majority of intra predicted macroblocks when the van is driving (Figure 5 and Figure 6). The optimal motion vector field is also not smooth and does not allow for low complexity motion estimation even after specialization of it (e.g. adding a priori knowledge of the motion based on the camera orientation of the van).

The noise in motion vector directions (anisotropy) can be explained by the fact that the optimal reference macroblock is out of the search area when the predicted motion vector does not correctly follow the motion flow perceived from the van. A perfect motion vector predictor would lead to only skipped predicted macroblocks in the case of a purely translational motion with no acceleration. Using a priori information on the camera orientation enables defining such a predictor as well as an oriented fast search. However, the presence of many intra predicted macroblocks due to the prohibitive cost of motion vectors in some areas of the frame (typically objects at a longer distance from the camera) does not allow for a smooth prediction vector field. This is mainly due to the coarse temporal resolution that implies a large displacement field between two consecutive frames.

The oriented fast search we used is based on the diamond search strategy but with an anisotropic mask: Only samples of the diamond that are in the a priori direction angle are first checked. If the resulting cost is not good enough (a QP-dependent threshold is defined), the remaining directions are checked (in case of other moving objects or the best reference macroblock being out of the search area).

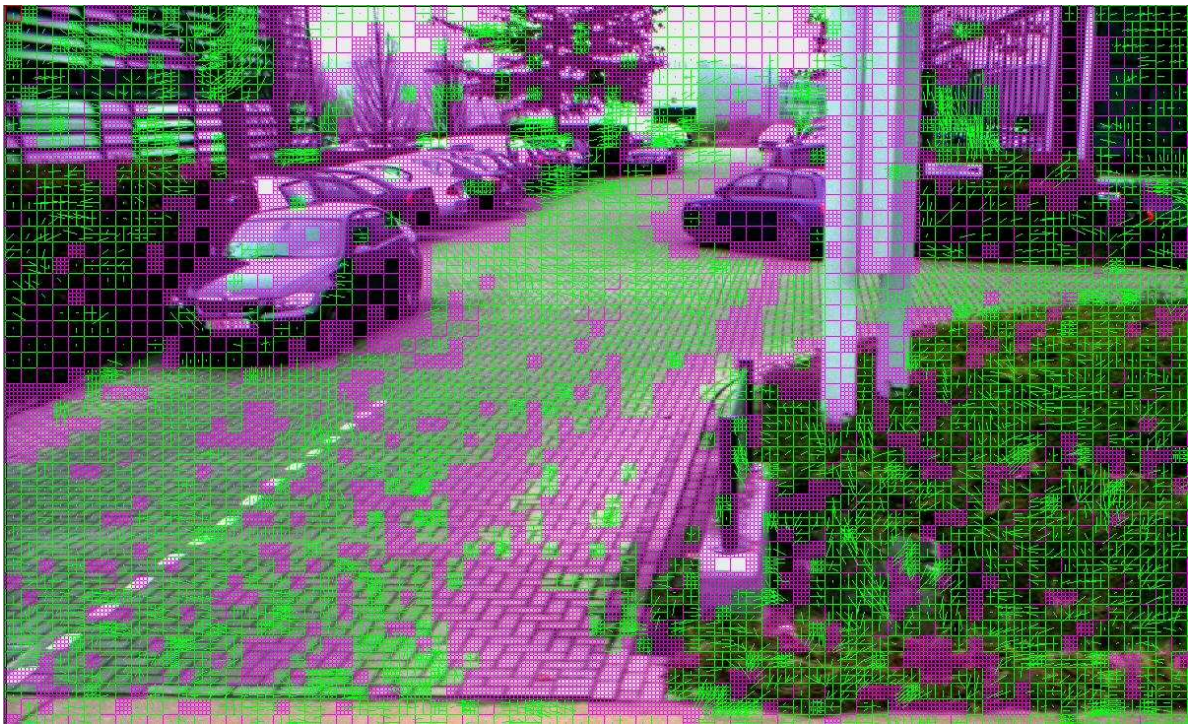


Figure 6 Motion Vector field lacks of regularity. Even with *a priori* knowledge on the motion of the van, motion vectors have unpredictable diirections (almost no skipped macroblocks). Motion vectors are represented by green lines.Intra preidicted macroblocks are represented in pink.

These problems have as a matter of consequence that AVC intra-only gives a better complexity/rate-distortion trade-off than AVC with motion estimation. The encoding with un-optimized intra prediction[2] is between two and three times faster than encoding with optimized motion estimation as already demonstrated in D111.

Furthermore, in the case of real-time encoding, it is interesting to have a predictable complexity at the encoding side. In this context, intra coding is also more interesting as the number of tested modes can be chosen as constant while for motion estimation, intra coding can be called in case the prediction is not good enough.

Analyzing further Figures 5 and 6, we can also see that intra blocks are distributed between intra 16x16 modes and intra 4x4 modes (finer pink grid). The choice made by the encoder follows the intuition that smooth regions can be well predicted by an almost constant 16x16 block and more complex areas by a mosaic of finer 4x4 modes.

In conclusion, the features of the content (high frequency…) will have an impact on the quality of the (intra) prediction but likely not on its complexity. However, if 4x4 and 16x16 modes fail at producing a sparse enough residual, the entropy encoding complexity and the final bitrate will be increased.

[2] Code used for the prototype delivered in D112

4. AVC intra encoding: state-of-the-art

Here we only discuss intra prediction and the typical block type intra 4x4 and intra 16x16. Intra prediction for the chroma signal uses similar techniques as those for luma Intra-16x16 predictions. H.264/AVC uses rate-distortion optimization (RDO) [2] technique to obtain the best result maximizing visual quality and minimizing bitrates. To choose the best macroblock mode, H.264 encoder calculates the RDcost (Rate distortion cost) of every possible mode and chooses the mode having the minimum value. Therefore, the computational complexity is extremely increased compared with previous standards, so it makes H.264/AVC difficult for applications with low computational capability, such as mobile devices.

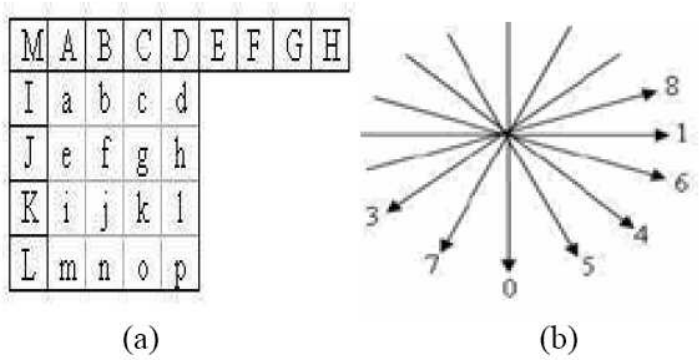


Figure 7 Intra mode directions (from [7])

intra mode decision

To reduce this complexity, few approaches have been proposed for fast intra prediction. Pan et al. [3, 4], proposed to select the possible mode based on the directions of the local edge information. Jongho Kim and Jechang Jeong [5] exploit the directional masks and mode information of neighboring blocks to chose the probable modes. Jun Sung Park and Hyo Jung Song [6] proposed the idea that direction of a bigger block is similar to that of smaller blocks, the effects of fast mode decision is then reduced.
Elyousfi et al [7], present a fast intra prediction algorithm using a gradient prediction function and a quadratic prediction function to improve the encoding speed without much sacrifice at RDO performance. Homogeneous areas are predicted by the gradient prediction function while heterogeneous areas are better handled with a quadratic prediction function.

Among Intra prediction modes, papers [8, 9] have also proposed optimizations for dedicated hardware. It is not clear whether these optimizations can be useful in the context of a PC.

Main existing algorithmic optimizations concern the prediction mode selection issue. In the functional block "Choose Intra prediction", many features can be used in order to lower the complexity of the commonly used "brute force" where all modes are tested and compared. Features can be related to the MB itself: mean, activity (variance of the MB pixels), edge presence and directions, or to the neighboring MB: their prediction mode for example in the same frame or in previous frames.

Introducing parallelism

Some papers propose to accelerate AVC encoding with slice or frame parallelism as well as SIMD SSE optimizations such as [10] and [11]. Authors achieve significant speed-ups but only consider motion estimation rather than intra coding. All conclusions are then not valid in our case such as the fact that using OpenMP [11], task parallelism can lead to better results because of frame and slices dependencies which do not exist in our case (intra prediction). Intel specific SSE, vector-based SIMD optimizations can be applied (if of course the platform contains an Intel CPU).

Use of a GPU as co-processor

In this case, parallelism is defined at MB level in order to exploit the fact that modern GPUs using shading language [12] or CUDA [13] can launch thousands of threads in parallel in order to execute the same task. Both papers only target the prediction on the GPU as the transform, CAVLC and inverse transform induce too many dependencies. The approach in [13] is intended for full search motion estimation and does not take into account motion vector predictors: this lowers the prediction quality. No numbers on the communication between CPU and GPU are given. Motion vector predictors and communication are better handled in [12], however it also considers fast and full search motion estimation; no intra prediction is implemented.

5. Optimization of the AVC Baseline Profile Intra encoder

5.1. Strategy

The strategy used to optimize the AVC code is based on three different hierarchical levels of parallelization. It is supported by the data hierarchy present in AVC encoding as illustrated on Figure 8. Frames in a sequence are organized in GOPs (for intra coding, the GOP size can be of one frame only). Frames are further decomposed into slices which are completely independent from each other. Slices are then composed of MBs which can be further partitioned into sub-blocks ranging from 4x4 to 16x16. Therefore H.264 encode process can be divided into multiple threads via data domain decomposition or via functional decomposition naturally.

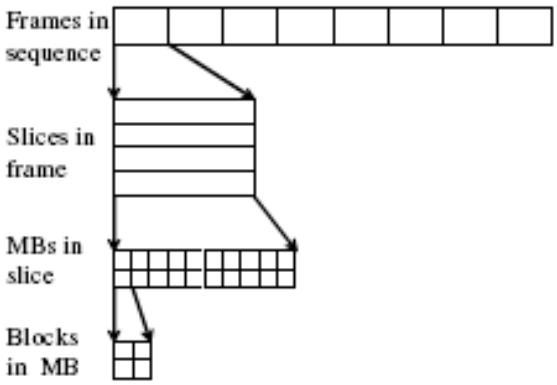


Figure 8 Data hierarchy in AVC/H.264

The most important features to be observed when designing a parallel solution are the following:

- *Scalability*: In the data-domain decomposition, to increase the number of threads, we can decrease the size of the processing unit of each thread. Because of the hierarchical structure in frames, slices, MBs, and blocks of the H.264 encoder, there are many choices to select the size of the processing unit. Thus, it seems easy to achieve a good scalability. In functional decomposition, each thread has a different function. In order to increase the number of threads, we must select to partition a function into two or more threads. It is a difficult task when the function is unbreakable.

- *Load balance*: In the data domain decomposition, each thread processes the same operation on different data block that has the same dimension. In theory (without cache misses or other non-deterministic factors), all threads should have the same process time. On the other hand, it is difficult to achieve good load balance among functions, as the execution time of each function is determined by the

algorithm. Furthermore, how to functionally decompose the video encoder with good load balance highly depends on algorithms. As the standard keeps improving, the algorithms will change over time.

In the following, we only performed data-level parallelism as it is the most optimal choice for intra coding. The entropy encoding however, could benefit from task-level decomposition if some quality loss is allowed, which is not the case for URBAN.

5.1.1. Slice parallelism

In intra coding, there is no inter-frame dependency such as for motion estimation. For the sake of random access, frames are coded independently. Slices being then independent, they can be processed in parallel as tasks where their MBs are processed sequentially. This is called *slice parallelism*. This can be achieved using openMP standard API for parallel computing for C [11].

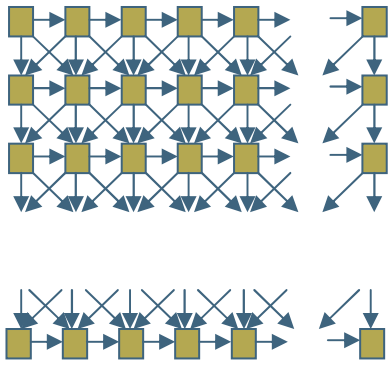


Figure 9 MB level dependencies for intra prediction

5.1.2. Macroblock-level parallelism

Some tasks at a MB level can be processed in parallel as well for a given slice. It is the case of motion estimation prediction (if we neglect the motion vector prediction which is a dependency for a MB to its 3 previous neighbors and which can be acceptable in some cases such as full search). It is less the case of intra coding where dependencies are created between all MBs and their 4 previous neighbors depending on the selected intra mode direction (Figure 9). In this context, some parallelism can still be introduced by separating the intra mode selection by the intra prediction itself. Indeed one can compute the intra mode prediction based on the original frame rather than the MB per MB reconstructed frame. All MBs can be processed at the same time. However, this is suboptimal as there is drift between the decoder and encoder: the decoder will not use the optimal intra prediction mode for its local reconstructed data. However, in our case, we have the requirement of high quality reconstruction and we can assume the difference between original and reconstructed data is negligible. Our experiments show that on the average of the data processed,

the PSNR loss is smaller than 0.3 dB. This strategy is more suitable for a GPU as coprocessors that computes intra modes in parallel and sends back the best modes to the CPU which sequentially reproduces the prediction using that mode without testing other modes.

5.1.3.Vector-based parallelism

This kind of parallelization is well-known in the Computer Science research area, for Intel processors it consists in SSE optimizations (see reference [11]) where data words are concatenated into larger words so that arithmetic operations can be processed in parallel on each sub-word (SIMD). Intel provides a library of low-level building blocks for many multimedia applications (IPP [14]) which are optimized using SSE-alike methodology. These optimizations also come with compiler optimizations in order to allow loop unrolling, better branch prediction etc.

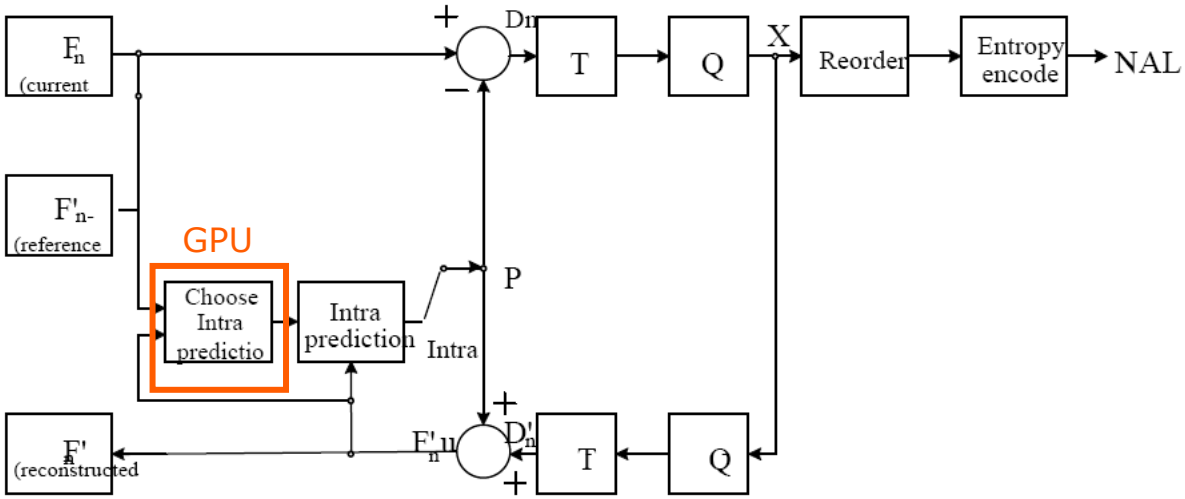


Figure 10 Intra mode selection separated from the intra prediction

5.1.4. Algorithmic optimization

The last option for optimization is algorithmic specific tuning. Some gains can be achieved by the fact that the encoder is now specific to AVC intra coding. Many branches needed for a general encoder have been deleted in order to simplify and speed-up the code. The deblocking filter has also been disabled as it is not needed[3] at the encoder side for intra prediction. The decoder will of course perform the deblocking filter at its side without any loss in quality. Other algorithmic tuning optimizations are possible for intra 4x4 fast mode selection. These methods are similar to the ones presented in the state-of-the-art section. They are based on

[3] The deblocking filter is performed in the end of the reconstruction of a frame when it will be used as a reference frame for motion estimation.

mode selection statistics and content features such as edges. The possible speed-ups are not impressive and the final quality is always lower than the full intra search. This kind of optimization better suits the case where intra prediction is called during motion estimation when the prediction is not good enough.

5.2. CPU “lossless” optimized implementation

We consider here first the results of the encoder when using a CPU (slice and vector-based parallelism) without GPU as co-processor (no MB-level parallelism). It must be noted that the PSNR and bitrate are the same between the different versions of the optimization for CPU. Only the encoding time varies. This is no longer the case for the use of a GPU.

Table 2 illustrates the kind of speed-up that can be reached from a C implementation to a vector-based optimized implementation. The most spectacular one is the SAD computation used to compare intra modes in order to find the best prediction.

Module	**Speed-up**
SAD	3.5x
Hadamard transform	1.6x
Integer transform and quantization	1.3x
CAVLC	1.4x

Table 2 Specific modules speed-up using IPP and SSE optimizations

The complexity of the AVC intra encoder proposed in deliverable D112 is illustrated on Figure 11 on a per-module basis. As explained before, the choice of the QP impacts the number of non-zero quantized coefficients to be entropy encoded by the CAVLC. From a QP of 24 to 20, CAVLC takes 37% to 55% of the total encoding time.

Using only vector-based optimizations, the complexity balance changes as it can be seen on Figure 12. The CAVLC speed-up is much more significant than the one for transform where little can be done to parallelize it.

Interestingly, CAVLC is again the bottleneck after openMP-based slice parallelism. In this case, we however question the capacity of the profiling tool to correctly handle the openMP thread management. We prefer then to refer to the total encoding times given in Table 3 Comparative performance for the AVC implementation measured on Intel Core 2 Duo CPU, 2.20GHz, 2GB RAM

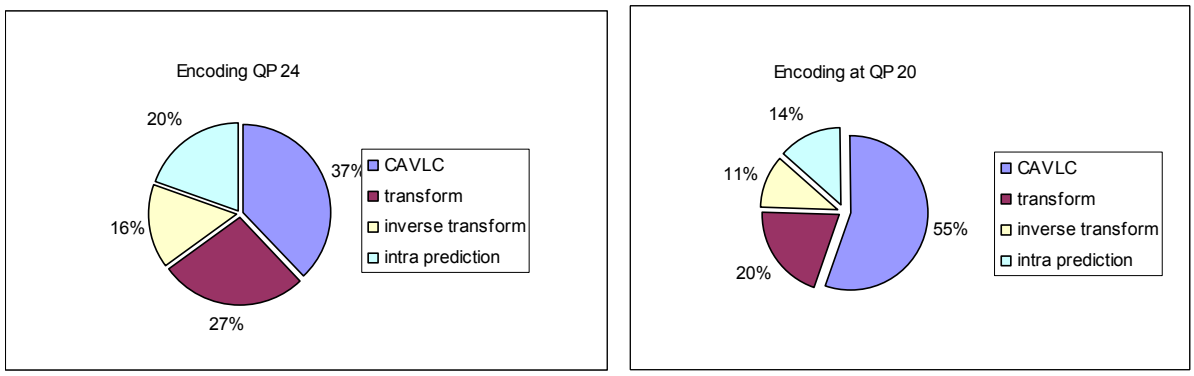


Figure 11 AVC Intra encoder (no optimizations) complexity depends on the QP. On the left, lower quality encoding with a QP of 24. On the right, higher quality encoding with a QP of 24.

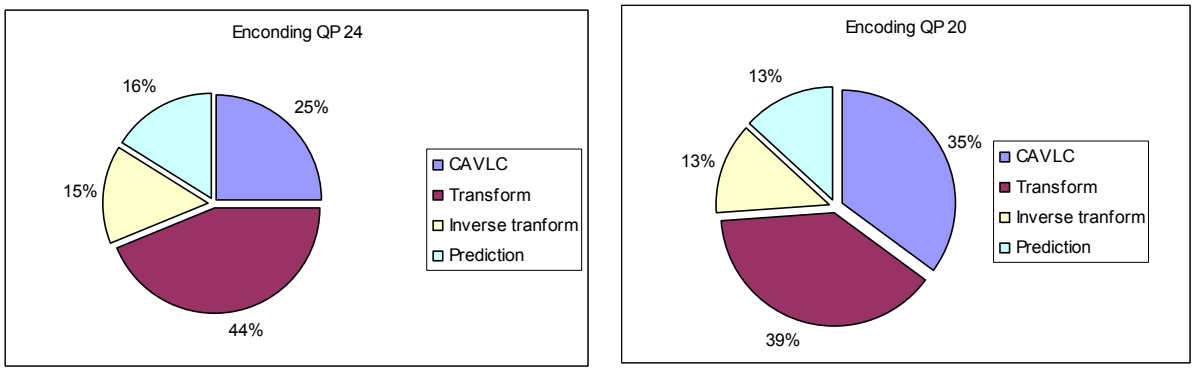


Figure 12 AVC encoder complexity per module after vector-based parallelism

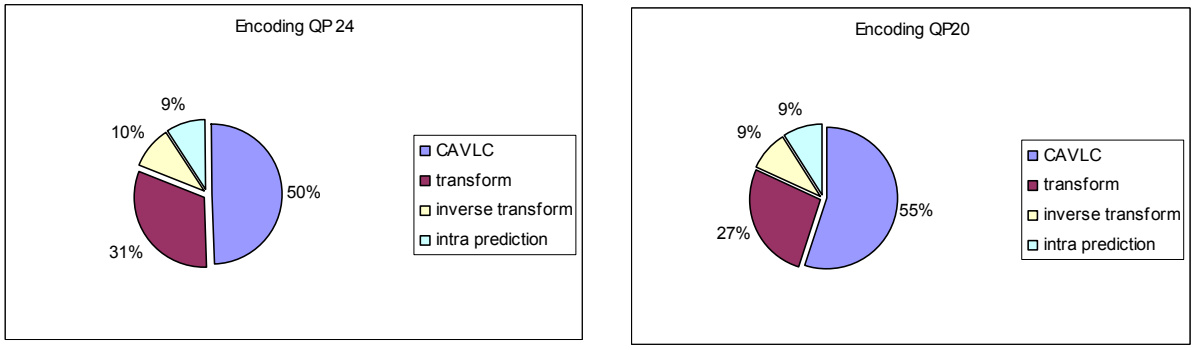


Figure 13 AVC encoder complexity per module after vector-based parallelism and slice parallelism

The results represented on that table are based on the average for the eight views for 2000 images each. There were some variations between frames base on their content. For very smooth images near walls or tunnels, the encoding speed can reach a speed-up of 5-6 and result in a very small bitrate. For very complex images, the worst case was 12.13 an 13.5 fps for QP20 and QP 24 respectively.

This timing also takes into account the debayering optimized with vector-based parallelism: 215 fps instead of 40 fps as already presented in D121.

	QP 20		QP 24	
	Speed (fps)	Time (msec)	Speed (fps)	Time (msec)
AVC Intra encoding with no optimization	4.29	233	4.84	207
AVC Intra encoding with vector-based optimization	6.45	155	7.75	129
AVC Intra encoding with vector-based and slice parallelism	13.7	73	15.3	66

Table 3 Comparative performance for the AVC implementation measured on Intel Core 2 Duo CPU, 2.20GHz, 2GB RAM, 32-bit operating system

Some tests have been run on Intel Core I7 CPU, 2.67GHz, 3GB RAM, 64-bit operating system enabling 4 dual cores. The speed-up is not linear but 38 fps have been reached using the same conditions are in Table 3. This could enable the van to process three cameras for a single I7 pc. It must also be said that allowing more slices for enabling more multi-core processors working concurrently implies a little quality loss of 0.1% in PSNR on average (same conditions as above).

5.3. CPU-GPU “lossy” optimized implementation

This implementation is the same as the CPU one except that the intra mode selection is processed in parallel by the GPU. As explained before (section 5.1.2), in order to enable the best speed-up at the GPU side, some dependencies have to be broken between macroblocks.

The solution proposed is to use the original image instead of the reconstructed frame for the intra mode selection. This causes a drift with the decoder and a small loss in quality. It however enables MB-level parallelism and also reduces the amount of memory to be managed in the GPU.

The communication between the CPU and the GPU is as follows. The CPU sends a full original frame to the GPU. The GPU uses it as self-reference for the intra mode selection. The best mode numbers are then sent back to the CPU which can use them for the intra prediction. The fact that we perform the debayering on the GPU is important in order to reduce the data communication between host and device as the bandwidth is limited. CUDA enables to concurrently process different tasks on CPU and GPU and to launch kernels on the GPU while transmitting information between host and device based on a message passing signal. This helps to hide a bit the latency.

The way the modes are selected on the GPU can be done either by full search or by edge-based decisions. The latter technique is however not the best choice as divergent branches between threads cause their serialization and loss in speed. It however suits well for CPU fast intra prediction. As edges can be efficiently computed on the GPU, we still describe this

technique as it could be efficiently merged with some further processing if the application conditions enable it.

The Intra prediction modes are represented on Figures 3 and 4. It can be seen that 16x16 modes can only reconstruct very uniform data at macroblock level. These modes are well suited for constant macroblocks or macroblocks with a vertical, horizontal or diagonal (45 degrees) edge that separates more or less uniform regions. On the contrary the 9 possible 4x4 modes enable many more possible representations with more directions and enable a finer granularity in pixel variations at macroblock level.

Computing these macroblock features or storing information from the neighborhood in memory has a non-negligible cost that has to be carefully analyzed. The less complex features are the mean value and activity, but they only can help in deciding whether 16x16 prediction or 4x4 prediction is the most interesting. Indeed a low activity will be present in uniform macroblocks that are very well represented by 16 x 16 modes. A higher activity will indicate that intra 4 x 4 is a better choice. However, these features do not enable to take a fast decision inside the different 16 x 16 modes or 4 x 4 modes.

A correlation can be found between large edges present in the macroblock of 4 x 4 subblocks. With this information, some papers propose to choose the direction of the mode to be aligned on the direction of the edge. This however asks for a convolution of the macroblock with an edge-detector filter (e.g. Sobel) of which the complexity is significantly higher than for the activity or mean. Furthermore, testing the four 16x16 modes can be less complex than a good edge detector. In order to avoid extra computations due to the borders computation overheads, it is more interesting to compute edges at the frame level than for each macroblock or sub-block separately.

The relevance of this method depends on the platform chosen. Indeed edge detection can be done very efficiently in CUDA thanks to the availability of shared memory. This could be more efficient than computing the different possible modes which exhibit less processing regularity and more comparisons which are less suited for a GPU. Now on a CPU, there is an extra memory cost because the edge information has to be pre-computed and then stored to take the best mode decision later on. These different methods have thus to be measured and compared in order to assess their relevance in our context.

Using information from the neighboring macroblocks or subblocks can also be useful to take a fast decision in intra prediction modes. There exists some correlation between the modes in the spatial and temporal domain especially at high resolution (HD and high framerate). If this strategy is more robust to noise than edges, the method does not predict well modes in case of occlusions or of the arrival of new objects in the scene. Furthermore using the information of previous frames may result in a burden with respect to the memory usage. Using the spatial correlation in the same frame introduces then macroblock per macroblock dependencies. This leads to sequential processing which is less suitable for a GPU.

The results we have measured are presented on Table 4. It can be seen that a speed-up factor of 10 is possible even when measuring the output

communication. However, the communication of the original image from the CPU to the GPU reduces this speed-up by a factor of two.

We implemented the Sobel filter for edge detection and the measurements are provided in Table 5. It can be seen that the results are then less good because the edge detection and the branching are less suitable than the brute force intra mode selection. This method would better suit a multi-core cpu.

Measurement of the intra mode selection process	Time (ms)	Speed (fps)
CPU	0.13	7692
CUDA kernel only	0.0068	146324
CUDA with output communication	0.0069	146001
CUDA with in and out communication	0.0155	64516

Table 4 Speed-up and timing measurements in various configurations of the prediction mode selection process (CPU time is computed in the same conditions as the GPU: orginal image input, no dependencies)

Measurement of the edge-based intra mode selection	Time (ms)	Speed (fps)
CPU	0.0697	14342
CUDA kernel only	0.0166	60120
CUDA with in and out communication	0.0252	39682

Table 5 Speed-up and timing measurements in various configurations of the edge detector-based intra mode selection

	QP 20		QP 24	
	Speed (fps)	Time (msec)	Speed (fps)	Time (msec)
AVC Intra encoding with GPU mode selection	13.9	72	15.4	65
AVC Intra encoding with vector-based and slice parallelism	13.7	73	15.3	66

Table 6 Speed up using a GPU for intra mode selection

Using the first method based of brute-force intra mode selection, the final optimized encoder with quality loss is shown on Table 6. The conclusion is that in this case, the GPU is not necessary as it reduces even the quality for a very limited speed-up (1-2%). The GPU would be more suited for a motion estimation kernel. This is mostly due to the fact that after slice parallelism and vector-based parallelism, intra prediction mode selection is far from being the bottleneck of the application, while it was a more important module for the code delivered in D112. In our context a multi-core CPU is the best solution.

6. Conclusion

The real-time constraint of 12 fps without quality loss has been reached through optimizations enabling hierarchical data-parallelism. A multi-core PC can further accelerate the encoder if more slices are selected with a negligible loss in quality. The use of a GPU implies algorithmic changes that impact the quality and the final speed-up is very limited it should then be discarded in this context. The executables are available on demand.

7. Bibliography

[1] G. J. Sullivan, P. Topiwala, A. Luthra, "The H.264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions", SPIE Conf. on applications of digital image processing XXVII, vol. 5558, pp. 53-74, Aug. 2004.

[2] G. Sullivan and T. Wiegand, "Rate Distortion Optimization for Video Compression," IEEE Signal Processing Magazine, pp. 74-90, Nov' 98

[3] F. Pan, X. Lin, S. Rahardja, K.P. Lim, Z.G. Li, G.N. Feng, D.J. Wu, and S. Wu, "Fast mode decision for intra prediction," JVT-G013, 7th JVT Meeting, Pattaya, Thailand, Mar. 2003.

[4] Feng Pan, Xiao Lin, Susanto Rahardja, Keng Pang Lim, Z. G. Li, Dajun Wu, Si Wu, "Fast mode decision algorithm for intraprediction in H.264/AVC video coding", Circuits and Systems for Video Technology, IEEE Transactions on Volume 15, Issue 7, pp. 813-822, July 2005.

[5] J. Kim, J. Jeong, "Fast intra-mode decision in H.264 video coding using simple directional masks", VCIP 2005, of proceedings of SPIE Vol. 5960, pp.1071-1079.

[6] J. S. Park, and H. J. Song, "Selective Intra Prediction Mode Decision for H.264/AVC Encoders", Transactions on Engineering, Computing and Technology Volume 13 May 2006, pp.51-55.

[7] A. Elyousfi, A. Tamtaoui, E. Bouyakhf, "Fast Intra Prediction Algorithm for H.264/AVC Based on Quadratic and Gradient Model", International Journal of Computer Systems Science and Engineering Volume 4 Number 1, 2007.

[8] D. Wu, F. Pan, K. P. Lim, S. Wu, Z. G. Li, X. Lin, S. Rahardja, and C. C. Ko "Fast Intermode Decision in H.264/AVC Video Coding", IEEE Transactions On Circuits And Systems For Video Technology, Vol. 15, No. 6, July 2005

[9] W. S. Kim, D. S. Kim and K. W. Kim, "Complexity Reduction of Plane mode in Chroma Intra Prediction", presented at the 5th JVT meetig(JVTE050), Geneva, CH, 9-17 October, 2002.

[10] Rodriguez, A.; Gonzalez, A.; Malumbres, M.P., "Hierarchical Parallelization of an H.264/AVC Video Encoder," *Parallel Computing in*

Electrical Engineering, 2006. PAR ELEC 2006, pp.363-368, 13-17 Sept. 2006

[11] Yen-Kuang Chen; Tian, X.; Steven Ge; Girkar, M., "Towards efficient multi-level threading of H.264 encoder on Intel hyper-threading architectures," *Parallel and Distributed Processing Symposium, 2004. Proceedings.* 63-68, 26-30 April 2004

[12] Motion Estimation for H.264/AVC using Programmable Graphics Hardware Chi-Wang Ho; Au, O.C.; Gary Chan, S.-H.; Shu-Kei Yip; Hoi-Ming Wong; Multimedia and Expo, 2006, 9-12 July 2006 Page(s):2049 - 2052

[13] Wei-Nien Chen; Hsueh-Ming Hang; IEEE International Conference on Multimedia and Expo, June 23 2008-April 26 2008 Page(s):697 - 700

[14] Intel Integrated Performance Primitives 5.3, http://www.intel.com/cd/software/products/asmona/eng/302910.htm