EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

Adapting hierarchical bidirectional inter prediction on a GPU-based platform for 2D and 3D H.264 video coding

Rafael Rodríguez-Sánchez[1*], José Luis Martínez[1], Jan De Cock[3], Gerardo Fernández–Escribano[1], Bart Pieters[3], José L. Sánchez[1], José M. Claver[2] and Rik Van de Walle[3]

Abstract

The H.264/AVC video coding standard introduces some improved tools in order to increase compression efficiency. Moreover, the multi-view extension of H.264/AVC, called H.264/MVC, adopts many of them. Among the new features, variable block-size motion estimation is one which contributes to high coding efficiency. Furthermore, it defines a different prediction structure that includes hierarchical bidirectional pictures, outperforming traditional Group of Pictures patterns in both scenarios: single-view and multi-view. However, these video coding techniques have high computational complexity. Several techniques have been proposed in the literature over the last few years which are aimed at accelerating the inter prediction process, but there are no works focusing on bidirectional prediction or hierarchical prediction. In this article, with the emergence of many-core processors or accelerators, a step forward is taken towards an implementation of an H.264/AVC and H.264/MVC inter prediction algorithm on a graphics processing unit. The results show a negligible rate distortion drop with a time reduction of up to 98% for the complete H.264/AVC encoder.

1 Introduction

Nowadays, graphics card production is booming due to the increase in demand from the video-game and game console business. The chips on these cards are normally called Graphics Processing Units (GPUs) and have emerged as co-processing units to assist Central Processing Units (CPUs). CPUs and GPUs have different instruction set architectures, forming what it is known as a heterogeneous computing platform. Although GPUs were initially designed for graphic applications, now they can be used to accelerate various numerical and signal processing applications [1,2], among others. GPUs consist of hundreds of highly decoupled processing cores that are able to achieve immense parallel computing performance. With high memory bus widths, high speed memory chips, and high processor clock speeds, graphics cards deliver specifications never seen before. GPUs are made up of individual stream processors, each running at 1.X GHz. The stream processors can be grouped together to perform Single Instruction Multiple Data (SIMD) operations that are suitable for arithmetic intensive applications. In the particular case of NVIDIA, a powerful GPU architecture called the Compute Unified Device Architecture (CUDA) [3] has been developed. The main feature of these devices is a large number of processing elements integrated into a single chip at the expense of a significant reduction in cache memory. Each core executes the same instruction at the same clock cycle but on different data. GPUs also have an external DRAM memory which can be classified depending on its access mode.

Video coding is today the main technology behind a wide range of applications such as video conferencing, video streaming, and High Definition TeleVision (HDTV), among others [4,5]. By efficiently exploiting temporal and spatial redundancy in video content, the latest video coding standards, such as H.264/MPEG-4 Advanced Video Coding (AVC) [6] and High Efficiency Video Coding (HEVC) [7], offer high quality and efficient video compression for a wide range of bit-rates and resolutions [8]. It is true that in the future the new standard called HEVC

*Correspondence: rrsanchez@dsi.uclm.es
[1] Instituto de Investigación en Informática de Albacete, Universidad de Castilla-La Mancha, Avenida de España s/n, 02071 Albacete, Spain
Full list of author information is available at the end of the article

[7] will replace its predecessor, namely H.264/AVC; but at present most architectures and video coding solutions are implemented using H.264/AVC. Moreover, some years ago, H.264/AVC established an extension for 2D (single view) to 3D (*n*-views) called H.264/MVC in order to support the sensation of 3D. Both the H.264/AVC standard and its extension, MVC, adopt many improved coding tools such as multiple reference frames, weighted prediction, a de-blocking filter, variable block-size and quarter-pixel precision for Motion Compensation (MC).

Until now, the highest complexity procedure has also been referred to in the literature as inter prediction, and is based on a variable block-size Motion Estimation (ME) with block sizes ranging from 16 x 16 to 4 x 4, with many options available between these. All these improved tools allow an optimum performance to be achieved, but at the expense of an increase in the computational complexity of the encoder. Moreover, this complexity increases as the number of views increases.

At this point, the approach presented here performs inter prediction with both full-pixel and sub-pixel accuracy for *P* (forward prediction) and *B* (bi-directional prediction) frames on the GPU. In fact, the algorithm called hierarchical *B* prediction [9] is implemented on the GPU for High Definition video sequences. As far as the authors know, this is the only work in the literature which deals with *B* frame inter prediction on a heterogeneous computing platform for both cases: single-view and multi-views. Furthermore, the complexity can be even higher when using hierarchical predictions because the distance between reference frames can be greater inside the Group of Pictures (GOP) with respect to traditional GOP patterns [9], such as IBP or IBBP. In inter prediction, the ME process is carried out many times per MB partition. ME performs the same Sum of Absolute Differences (SAD) operations over a large amount of data (over the search area). This procedure fits well in the SIMD execution processing model. One of the major challenges for our proposed algorithm is how to remove or mitigate the dependencies between MBs. The present approach is shown for both cases: 2D and 3D scenarios and the results show a noteworthy time reduction (TR) of up to 99% with only a negligible Rate-Distortion (RD) penalty. Moreover, the proposed algorithm outperforms one of the fastest ME algorithms included in the H.264/AVC JM [10] reference software, namely Unsymmetrical Multi-Hexagon Search (UMHexagonS) [11].

The rest of the article is organized as follows. Section 2 summarizes the technical background in the field of video coding and GPUs. Section 3 presents related proposals, and the implementation details of our approach and its performance evaluation are presented in Sections 4 and 5, respectively. Finally, the conclusions are set out in Section 6.

2 Background

2.1 H.264/AVC

H.264/AVC [6] emerged with the objective of creating a standard able to provide good video quality for the encoded video sequence, as well as offering a significant reduction in the bit-stream produced by the H.264/AVC encoder, when compared with previous standards, such as MPEG-2 [12].

H.264/AVC supports variable block-size for MC. The available modes are 16 x 16, 16 x 8, 8 x 16, and 8 x 8. If the 8 x 8 mode is selected, it can be further divided into 8 x 8, 8 x 4, 4 x 8, and 4 x 4 blocks. ME estimates each partition by using a block located in a previous frame (*P* and *B* slices), by using a block located in a future frame (*B* slices), or by using a combination (*B* slices). This latter mechanism, which uses a combination of previous and future frames, is known as bi-directional prediction.

Bi-directional prediction is carried out by obtaining the differences between an MB partition and two reference blocks, one located in a previous frame and the other one in a future frame. Traditional GOP patterns, such as IBP or IBBP, use the closest *I* or *P* frame in each direction as reference frames. However, in recent years new hierarchical GOP patterns have been introduced [8,9]. These new GOP patterns allow the encoding of *B* frames by using other *B*-frames as reference frames. Figure 1 shows a fully hierarchical GOP pattern using seven *B*-frames. The *B*-frames are divided into three levels and a frame belonging to level *n* is estimated using the

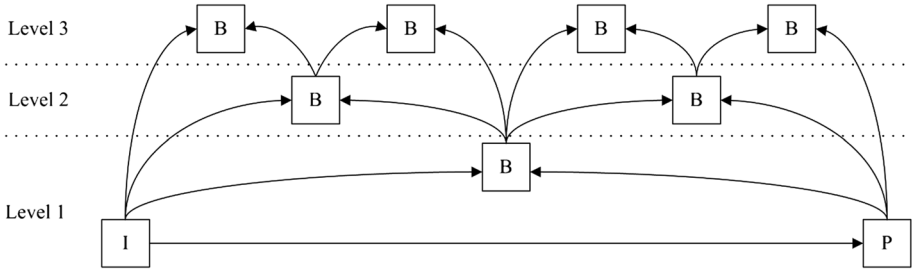

Figure 1 Configured GOP pattern.

closest frames in each direction belonging to level $n-1$ or below.

On the other hand, encoding one Motion Vector (MV) for each partition can increase the number of bits required to encode an image, especially if small partition sizes are chosen. However, it is known that MVs from neighbouring partitions are often highly correlated and each MV can be predicted using these neighbouring MVs. The MV Predictor (MVP) forming method depends on the availability of nearby MVs and on the partition size. Figure 2 shows an MB and its neighbouring MBs involved in the MVP calculation. If there is more than one partition in the neighbouring MBs (in Figure 2 the left and upper MBs are divided into more than one partition) the nearest partition to the top-left corner of the MB is selected in selected in order to calculate the MVP (see A and B partitions in Figure 2). The MVP is calculated as the median of the three selected partitions (A, B, and C partitions in Figure 2).

2.2 H.264/MVC

The H.264/MVC [13] coding standard, which was developed as an extension of H.264/AVC, has recently been finalized by the Joint Video Team (JVT). MVC mainly uses H.264/AVC while exploiting temporal and inter-view dependencies [14] which are based on Hierarchical Bidirectional Pictures prediction to exploit both temporal and inter-view correlations [15]. Moreover, MVC provides other techniques such as Disparity Estimation (DE), which is used in the process of inter-view coding.

Basically, variable block-size matching DE [14] is used to reduce the inter-view redundancies between frames. In this coding system, variable block-size DE is carried out using eight inter prediction modes (SKIP, Inter 16 x 16, Inter 16 x 8, Inter 8 x 16, Inter 8 x 8, Inter 8 x 4, Inter 4 x 8, and Inter 4 x 4). MVC determines which partitions deal with cost as best MB partition. This results in high encoder complexity. Therefore, it is necessary to develop a method that can reduce the execution time of MVC with minimal loss of image quality. The JM 17.2 [10] reference software includes support for MVC, implementing the Stereo High Profile.

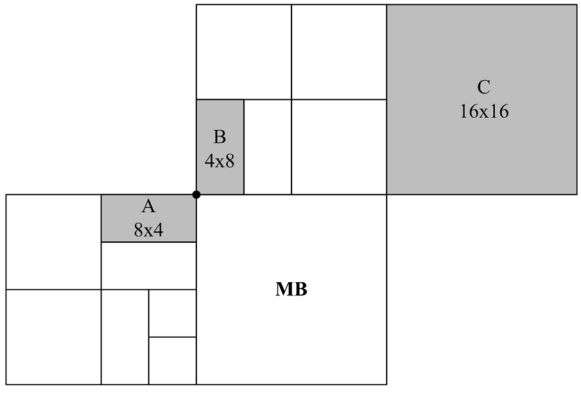

Figure 2 Current and neighbouring partitions.

2.3 Graphics processing units

In the past few years new heterogeneous architectures have been introduced in high performance computing [16]. Examples of this type of platforms are GPUs, Cell Broadband Engines (Cell BEs), and Field-Programmable Gate Arrays (FPGAs). Modern graphics cards include a many-core processor chip, which is known as a GPU. This processor chip is built following the SIMD programming model, and it is able to perform arbitrary, programmable operations on data sent to it.

Recently, there has been a marked increase in the performance and capabilities of GPUs, and they have attracted a lot of attention. GPUs are able to achieve up to three TFLOPS working in simple precision mode and up to 190 GB/s in memory transfer rate. In fact, over the last 10 years, the performance of GPUs has doubled every 6 months, whereas that of the CPU has doubled every 18 months.

CUDA [3] was introduced in 2006 by NVIDIA and consists of a general purpose parallel computing architecture that makes it possible to solve many complex computational problems by using the parallel engine on NVIDIA GPUs. CUDA is a C-based high-level programming language that was designed to maintain a low learning curve for programmers familiar with standard C. Although CUDA requires programmers to write special code for parallel processing, it does not require them to explicitly manage threads in the conventional sense, which greatly simplifies the programming model. CUDA development tools work alongside a conventional C/C++ compiler, so programmers can mix GPU code with general purpose code for the host CPU.

3 Related work

As far as the authors know, the work presented in this article is the first which deals with H.264/AVC bi-directional prediction using GPUs. In the literature, there are some articles that deal with inter-prediction in *P*-frames using GPUs, but no algorithms for bi-directional prediction have been presented. However, some proposals have been presented for hardware accelerators, but they focus on MC. Moreover, in recent years 3D video coding has attracted a lot of attention, but in the literature it is not possible to find many articles that focus on accelerating the inter prediction carried out in a 3D encoder.

The following paragraphs will provide an overview of the state of the art, focusing on previous works on *P*-frames, on proposals for *B*-frames using hardware accelerators, and on proposals aimed at accelerating 3D video coding.

Lee et al. [17] presented a multi-pass and frame parallel algorithm to accelerate H.264/AVC ME using a GPU. By using the multi-pass method they unroll and rearrange the multiple nested loops. The integer ME can be implemented with a two-pass process on the GPU. Moreover, fractional ME needs six passes for frame interpolation with a six-tap filter and MV refinement. They obtain a speed-up of between 6x and 56x, depending on the encoding conditions. Cheng and Hang [18] decomposed the H.264/AVC ME algorithm into five steps so as to achieve highly parallel computation with a low external memory transfer rate. Experimental results show that, with the assistance of a GPU, the processing time is 12 times shorter than when using only the CPU. However, the major failing of these two approaches is that they do not show RD performance; although the speed-up and TR are acceptable, they are only valid if they keep the RD as close as possible to the sequential approach. More recently, Cheung et al. [19] proposed a GPU implementation of the simplified UMHexagons (smpUMHexagonS) ME algorithm, which is a fast ME technique implemented in the H.264/AVC reference software. The authors divide the current frame into multiple tiles and they obtain a speed-up of up to 3x when compared with this fast algorithm. Additionally, they report significant bit-rate increments (12%) with a penalty in quality (0.4 dB) depending on the sequence and the tile length.

Tseng et al.[20] presented a FPGA prototype which supports variable block-size ME from multiple reference frames for both *P* and *B* slices, quarter-pixel accuracy, and weighted bi-directional prediction. More recently, Zheng et al. [21] presented a VLSI architecture of MC for multiple standards. The proposed design has a macroblock-level pipelined structure which consists of an MV Predictor, cache-based fetch, and a pixel interpolation unit. The proposed architecture exploits the parallelism in the MC algorithm to accelerate the processing speed and uses the dedicated design to optimize memory accesses.

Ding et al. [22] proposed a content-aware prediction algorithm for inter-view mode decisions. The proposed algorithm is able to save unnecessary computational load by exploiting the correlation between the different views in MVC. The MB coding modes and their corresponding MVs may be predicted by using the DE and the coding information of neighbouring views. Therefore, ME computational complexity can be greatly reduced since some MBs may be early identified as SKIP, INTRA or DE modes and it is not necessary to perform the ME. Experimental results show a TR of nearly 98% for the ME time, with a quality loss of up to 0.06 dB. Also, Huo et al. [23] presented a scalable prediction structure for MVC in which inter-view prediction may be disabled if the inter-view redundancy can be almost eliminated by temporal and intra prediction. In this way, the time employed for DE may be saved by reducing encoder complexity. The authors use a hierarchical GOP pattern and propose not to carry out the DE in one or more of the highest temporal layers of the hierarchical GOP pattern, since they observed that the percentage of temporal predictions increases with the increment in the temporal layer index. Experimental results report a TR of 30.60% and a bit rate increment of 1.55 on average, when not performing the DE in the highest level in a GOP pattern with five levels (GOP 16).

Shen et al. [24] proposed a fast DE and ME algorithm based on the correlation between the prediction/mode size and on motion homogeneity. MBs with homogeneous motion usually select temporal prediction with large block sizes, and MBs with complex motion usually select inter-view prediction or temporal prediction with small block sizes. The proposal uses the spatial properties of the motion field, which is generated by the corresponding MVs of the 4 x 4 partitions. On average, experimental results show a TR of 63% and a bit rate increment of up to 2%. In 2010, textDeng et al. [25] proposed a fast ME and DE algorithm, which uses the correlation between temporal and inter-view reference frames. First, the algorithm obtains a predictor by taking into account the correlation of motion (ME) and disparity (DE) vectors of neighbouring MBs. Then, an iterative search algorithm is run to find the optimal motion and disparity vectors. The iterative search algorithm is executed using small window sizes of 5 x 5, which is sufficient to maintain coding efficiency. The algorithm is only implemented for the 16 x 16 block size, so it does not support variable block-size ME and DE. Simulation results show that the overall average encoding TR is 86%, whit a bit rate increment of up to 6%.

Finally, Lu and Hang [26] presented the first GPU-based ME/DE algorithm for MVC. The algorithm is implemented using integer precision (no sub-pixel ME or DE is carried out using the GPU). The algorithm is based on the PHODS method proposed by Chen et al. [27] and is a fast ME algorithm designed to reduce the number of sequence steps and search area points. However, it must be adapted to have a regular control flow and a fixed number of instructions for each iterative process. In this way, each iterative process can be independently executed by an independent GPU thread. Experimental results show that the proposed algorithm can obtain a speed-up of 9–20x for integer ME and DE algorithms when compared with the reference EPZS search algorithm implemented in JMVC, while displaying a coding quality loss of up to 0.026 dB.

4 Proposed algorithm

In this section, the proposed GPU-based inter prediction algorithm for hierarchical *B*-frames is presented. The proposed algorithm is divided into two main parts: forward

and backward prediction, and bi-directional prediction. The algorithm implemented for forward and backward prediction is based on the one previously developed for *P*-frames [28], but there are some differences in the MVP calculation. The proposed algorithms are integrated in the well-known H.264/AVC JM reference software encoder, version 17.2 [10].

In what follows, we describe the main parts of our proposed inter prediction algorithm for B frames.

4.1 Forward and backward prediction

This part is divided into two main parts: full-pixel prediction and sub-pixel prediction. Reference H.264/AVC JM full-pixel prediction and sub-pixel prediction are carried out sequentially for all partitions, for each MB in a frame. Our main idea is to obtain all the motion information concurrently at the beginning of coding each frame.

Full-pixel prediction is implemented using two GPU kernels. The first one obtains the encoding costs for all 4 x 4 partitions, within the search area, for all MBs in a frame. Then, by reusing the 4 x 4 encoding costs previously obtained, it calculates the encoding costs for all other partitions. Figure 3 shows how to calculate them. In order to obtain the motion information for the eight 4 x 8 and for the eight 8 x 4 sub-partitions, it is only necessary to add two 4 x 4 SAD costs for each of them, e.g. by adding #0 and #2 SAD costs from the 4 x 4 sub-partition, the #0 SAD cost for the 8 x 4 sub-partition is obtained (see shaded boxes), and so on. Finally, this kernel performs a reduction in the generated encoding costs for all partitions by a factor of 256, which is the number of positions checked by any of the GPU multi-processors. Usually, there are more than 256 candidate positions within the search area. Therefore, an independent kernel performs the final reduction of the generated encoding costs in order to obtain the best MV for all partitions of all MBs in a frame.

Before starting with sub-pixel prediction, the frames must be up-sampled to half-pixel accuracy by means of a 6-tap filter, and then they are up-sampled to quarter-pixel accuracy by means of a bilinear filter. The up-sampling is carried out on the GPU because the frames are transferred to the GPU memory with full-pixel accuracy. It is faster to generate the frames with quarter-pixel accuracy than to transfer them. Three GPU kernels are needed since there are dependencies in the up-sampling process.

Sub-pixel prediction is carried out in two steps: half-pixel refinement and quarter-pixel refinement. The MV obtained by full-pixel prediction becomes the center point for half-pixel prediction, and the MV obtained by half-pixel prediction becomes the center point for quarter-pixel prediction. The algorithm is similar to the one used for full-pixel prediction. The MBs are divided into sixteen 4 x 4 blocks, the motion information for all partitions is obtained and finally a reduction is performed in order to obtain the best MV.

4.2 MVP for forward and backward prediction

In the algorithm implemented for encoding *P*-frames, only previous frames are used as reference frames (backward prediction). The MVP is calculated using the MV of the 16 x 16 partition of the MB located in the same position, but in the previously encoded *P*-frame. Note that, in a video sequence which only uses *I*- and *P*-frames, the encoding order is the same as the temporal order.

However, when coding a video sequence using *B*-frames, the frames are encoded out of order, since future frames are required to encode a *B*-frame. Therefore, the MVs of reference frames (in future and/or

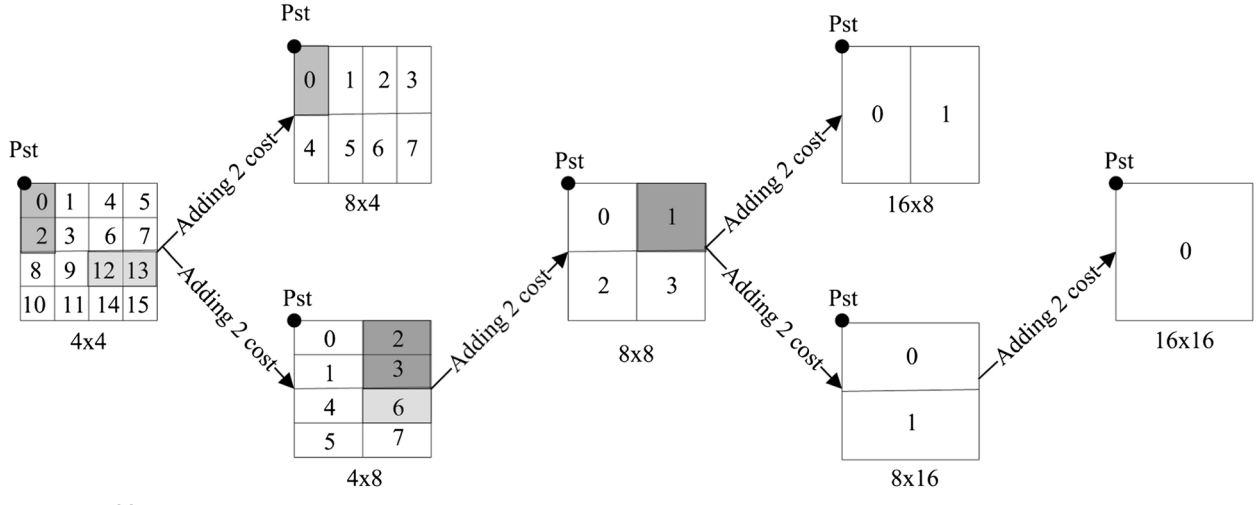

Figure 3 Building SAD costs.

previous frames) are available and can be used in order to calculate the MVPs.

The MVP for backward prediction is calculated using the MV of the 16 x 16 partition of the MB located in the same position, but in the temporally previous frame. The MVP for forward prediction is calculated using the MV of the 16 x 16 partition of the MB located in the same position, but in the temporally future frame. Remember that the temporal order is different from the encoding order.

4.3 MVP for inter-view prediction

When coding a 3D video sequence, some frames are temporally predicted (ME) and some frames are inter-view predicted (DE). The algorithm for both mechanisms is quite similar but there is a difference. The reference frame is located in either the same view or in a different view.

The inter-view prediction tries to remove the redundancies between two or more cameras which are recording the same scene, while the temporal prediction tries to remove the redundancies inherent in a video stream. Therefore, it is not a good idea to calculate the MVP of an inter-view predicted frame using the motion information of a temporally predicted frame and vice versa.

The MVP for inter-view prediction is calculated using the MV of the 16 x 16 partition of the MB located in the same position, but in the temporally previous inter-view coded frame. Otherwise, an inaccurate MVP would be calculated, thus affecting coding efficiency, since an inaccurate MVP means that the optimal (or nearly optimal) matching block cannot be found.

4.4 MVP for bi-directional prediction

Bi-directional prediction is carried out in two steps. In the first step, a block is estimated in the previous frame using MVP_1 (in what follows we will refer to this block as the opposite block), a search area region is estimated in the future frame using MVP_2 and the bi-directional prediction is performed by searching (ME) in the future frame (see Algorithm 1, line 3). In the second step, a search area region is estimated in the previous frame using MVP_1, the opposite block is estimated in the future frame using MVP_2 and the bi-directional prediction is performed by iterating in the previous frame (see Algorithm 1, line 4). The opposite block dimensions depend on the MB partition size used. Figure 4 shows a graphical description of the first step for a given partition. In the description of the algorithm we focus on the first step, locating the search area in the future frame. The algorithm for the second step is identical, but interchanging the locations of the opposite block and search area.

Algorithm 1 Proposed bi-prediction algorithm

1 Backward prediction (MVP_1 calculation)
2 Forward prediction (MVP_2 calculation)
3 Bi-prediction (Step 1)
  3.1 For each thread block configured - Kernel 1
    3.1.1 Transfer current block from GPU texture memory to shared memory
    3.1.2 Transfer opposite block from GPU texture memory to shared memory (MVP_1)
    3.1.3 Transfer search area from GPU texture memory to shared memory (MVP_2)
    3.1.4 For each thread within the thread block
      3.1.4.1 Calculate motion information (SAD cost)
      3.1.4.2 Reduction (1 MV per MB-partition within the thread block)
  3.2 Final reduction - Kernel 2 (1 MV per MB-partition)
4 Bi-prediction (Step 2)
  4.1 For each thread block configured - Kernel 3
    4.1.1 Transfer current block from GPU texture memory to shared memory
    4.1.2 Transfer opposite block from GPU texture memory to shared memory (MVP_2)
    4.1.3 Transfer search area from GPU texture memory to shared memory (MVP_1)
    4.1.4 For each thread within the thread block
      4.1.4.1 Calculate motion information (SAD cost)
      4.1.4.2 Reduction (1 MV per MB-partition within the thread block)
  4.2 Final reduction - Kernel 4 (1 MV per MB-partition)

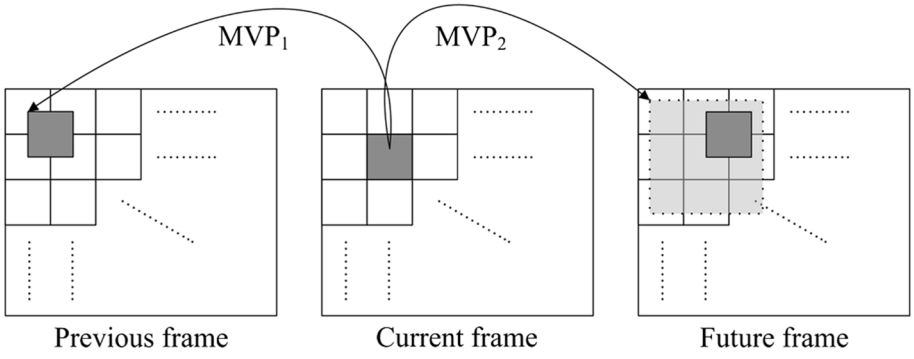

Figure 4 Bi-directional predictions.

The first task in order to carry out the bi-directional prediction is to calculate the MVPs for both the previous frame (MVP_1) and for the future frame (MVP_2) (see Algorithm 1, lines 1 and 2). As MVP_1 for a given partition we propose to use the MV obtained with the backward prediction for that partition, and as MVP_2 for a certain partition, the MV obtained with the forward prediction for that partition.

The opposite block is located and estimated using its MVP (MVP_1 if it is located in the previous frame or MVP_2 if it is located in the future frame). The search is located in the same position as the current MB partition, but in the reference frame, and it is estimated using its MVP. Note that a common search area is established for all MB partitions, but all MB partitions have their own MVP.

The H.264/AVC JM reference software uses different MVPs from the ones used in this proposal. The opposite block for a specific partition is estimated and located using the MV obtained with the backward or the forward prediction for that partition, depending on which frame it is located in (MVP_1 if it is located in the previous frame or MVP_2 if it is located in the future frame). The search area is located and estimated using the motion information of neighbouring MBs, which are not available in a parallel execution. Note that each MB partition has its own independent search area.

The MVPs are used jointly with the cost metric to calculate the encoding cost of all available MB partitions. Equation (1) shows how to calculate this [29]:

$$\text{Cost} = \text{SAD}_{\text{cost}} + \lambda * R_{\text{MV}_1} + \lambda * R_{\text{MV}_2} \quad (1)$$

where SAD_{cost} is the metric used to calculate the differences, λ is an encoder parameter which depends on the Quantization Parameters (QPs) used, R_{MV_1} is the number of bits required to encode MV_1 minus MVP_1, and R_{MV_2} is the number of bits required to encode MV_2 minus MVP_2.

However, there is a problem in our proposed MVP calculation, as forward and backward predictions must be completely performed before starting the bi-directional prediction, and with the H.264/AVC JM encoding order this does not occur. The encoding order implemented in the H.264/AVC JM reference software is: backward prediction, bi-directional prediction (first step), forward prediction and bi-directional prediction (second step). Our solution is to modify the encoding order implemented in the H.264/AVC JM reference software: firstly backward and forward predictions are performed, and then the bi-directional predictions are performed (see Algorithm 1, lines 1, 2, 3, and 4).

4.5 Bi-directional prediction

As occurs with forward and backward prediction, the reference bi-directional prediction is sequentially carried out for all possible partitions configured in the configuration file, for each MB in a frame, whereas our proposed algorithm is executed at the beginning of coding each B frame and obtains all the motion information in parallel.

However, in contrast with the forward and backward predictions, in the bi-directional predictions it is not possible to calculate the encoding cost of higher MB partitions using the ones obtained for the smallest MB partitions (Figure 3). Different MVPs are used to locate the opposite blocks, so the opposite blocks of each MB partition can be located in different positions.

Bi-directional full-pixel prediction is performed using two GPU kernels (see Algorithm 1, lines 3.1 and 3.2, or lines 4.1 and 4.2). The first GPU kernel uses 256 threads per thread block in order to calculate the motion information for 256 contiguous positions for all MB partitions configured (see Algorithm 1, lines 3.1.4 or 4.1.4). This first kernel is carried out in as many iterations as needed. Thus, two 16 x 16 memory matrices are defined in multiprocessor shared memory. One of them is filled in with the current MB to be coded (in what follows we will refer to this block as the current block) (see Algorithm 1, lines 3.1.1 or 4.1.1). The other one is either filled in with the 16 x 16 opposite block, or with the two 16 x 8 opposite blocks, or with the two 8 x 16 opposite blocks, and so on

(see Algorithm 1, lines 3.1.2 or 4.1.2). The algorithm is executed using the complete 16 x 16 blocks. Therefore, if the three highest partitions are configured in the configuration file, three iterations are needed. Both the current block and the opposite block are originally located in GPU texture and all available threads cooperate to fill in the allocated multiprocessor shared memory.

The search area is also transferred to multiprocessor shared memory, but it is not necessary to copy the complete search area (see Algorithm 1, lines 3.1.3 or 4.1.3). Only the search area required for 256 contiguous positions is copied. The required search area for 256 positions depends on the search range, e.g. if a search range of 32 is used a search area of 79 x 19 positions must be allocated and copied. Figure 5 shows this search area allocation. The 15 extra positions allocated on the right and at the bottom of the allocated memory are necessary to move the 16 x 16 blocks over all possible positions. In the figure the current block (shadow block) is located in the bottom-right position of the allocated search area. The search area for the current block is originally located in GPU texture memory and all available threads cooperate to fill in the allocated multiprocessor shared memory.

On each kernel iteration, each GPU thread obtains the SAD costs for the sixteen 4 x 4 blocks into which both the current block and the opposite block can be divided for a specific position. If the opposite block was filled in with a 16 x 16 partition the sixteen SAD costs are added in order to obtain the motion information of the 16 x 16 partition; if the opposite block was filled in with two 16 x 8 partitions, eight SAD costs are added in order to obtain the motion information of the upper 16 x 8 partition, and eight SAD costs are added in order to obtain the motion information of the lower 16 x 8 partition; and so on (see Algorithm 1, lines 3.1.4.1 or 4.1.4.1). The SAD costs for the 4 x 4 blocks are calculated using Equation (2) [29].

$$\mathrm{SAD} = \sum_{i=0}^{3}\sum_{j=0}^{3} |C_{ij} - ((O_{ij} + R_{ij} + 1) >> 1)|, \qquad (2)$$

where C is a subset of the current block, O is a subset of the opposite block and R is a portion of the search area.

At this point, each thread block has the encoding cost of 256 positions for all available partitions, and applying a binary reduction per MB partition obtains the best MV of the 256 positions for each MB partition (see Algorithm 1, lines 3.1.4.2 or 4.1.4.2). The binary reduction is the same as previously developed for *P* frames. Usually, there are more than 256 candidate positions within the search area. Therefore, an independent kernel performs the final reduction of the generated encoding costs (see Algorithm 1, lines 3.2 or 4.2).

In order to carry out the bi-directional sub-pixel prediction it is not necessary to up-sample the reference frames, because they have been previously up-sampled before applying the forward and backward predictions.

The algorithm developed for bi-directional sub-pixel prediction is quite similar to the one developed for full-pixel prediction. However, there are some differences: the algorithm is performed in two GPU kernels, one for half-pixel refinement and one for quarter-pixel refinement; the metric used is Hadamard SAD instead of SAD, since it gives better results but is more complex; the search area is not located in multiprocessor shared memory, because it is not common for all MB partitions, the starting point for half-pixel refinement is the best MV for full-pixel prediction and the starting point for quarter-pixel refinement is the best MV for half-pixel refinement; the number of search area positions checked by the kernels is eight, since there are eight half-pixels surrounding each full-pixel, and there are eight quarter-pixels surrounding each half-pixel.

5 Experimental results

The proposed algorithm has been implemented in the H.264/AVC JM version 17.2 reference software encoder [10]. The H.264/AVC profiles used for testing are the Main Profile and the Stereo High Profile, and therefore the parameters used are those included in these profiles. The proposal presented in this article is independent of the profile used; different profiles which support *B* frames may be used. The Main profile was selected because is intended for broadcasting applications which is a potential application of the proposal presented in this article, and the Stereo High profile was selected because is the

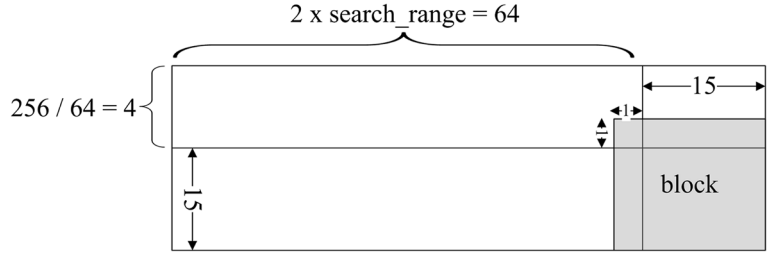

Figure 5 Search area located in multiprocessor shared memory.

one which provides support for stereoscopic videos. Only some parameters are changed in the configuration file:

- The tests are carried out using full HD sequences (1080p, 1920 x 1080 pixels).
- The sequences are rated at 50 frames per second (50 Hz) when using the Main Profile, and each view is rated at 25 frames per second (25 Hz) when using the Stereo High Profile.
- The number of reference frames is set to 4 for P-frames, and to 2 in both directions (forward and backward) for B-frames.
- The number of B frames inserted between each I or P-frame is set to 7 and the intra period is 32. Therefore, in each GOP there are 28 B-frames, 3 P-frames and 1 I-frame.
- The GOP pattern used is full hierarchy and is depicted in Figure 1. Note that in the figure only the reference for the closest reference frames is depicted, but as mentioned above, multiple reference frames are used.
- The QP for P and I frames is varied among 28, 32, 36, and 40, according to [30-32]; the QP for B-frames depends on the hierarchical level on which the B-frame is located (see Figure 1) and is incremented by 1 per hierarchical level in reference with the one configured for P and I-frames.
- The search range is set to 32, which means 409,6 candidate positions inside the search area.
- RD-optimization is disabled to support low-delay/complexity.

In order to make a proper comparison, an unmodified H.264/AVC JM 17.2 reference software encoder implementation was run on the same machine as the proposed algorithm, with the same configuration and with no calls to the GPU.

5.1 System

To evaluate the performance of the proposed algorithm, the following development environment was used: the host machine used was an Intel Core i7 running at 2.80 GHz with 6 GB of DDR3 memory. The GPU used was an NVIDIA GTX480 with an NVIDIA driver and CUDA support (260.19). The operating system for this scheme was Linux Ubuntu 10.4 x 64 with GCC 4.4. Table 1 shows the main GPU features.

Table 1 NVIDIA GTX480 features

Characteristic	Value
Compute capability	2.0
Global memory	1.5 GB
Number of multiprocessors	15
Number of cores	480
Constant memory	64 KB
Shared memory per block	48 KB
Registers per block	32,768
Max. active threads per multiprocessor	1,536
Clock rate	1.40 GHz

5.2 Metrics

The following metrics were used to evaluate the proposal:

5.2.1 RD function

In the definition of the RD function, the PSNR is the distortion for a given bit rate. The averaged global PSNR is based on Equation (3). The Luminance PSNR is multiplied by four, since the YUV input files are in the format 4:2:0.

$$\mathrm{PSNR} = \frac{4 * \mathrm{PSNR}_Y + \mathrm{PSNR}_U + \mathrm{PSNR}_V}{6} \tag{3}$$

5.2.2 TR and speed-up

This is to evaluate the time saved by the proposed algorithm. TR is based on Equation (4) and Speed-up is based on Equation (5).

$$\mathrm{TR}(\%) = \frac{T_{\mathrm{JM}} - T_{\mathrm{FI}}}{T_{\mathrm{JM}}} * 100 \tag{4}$$

$$\text{Speed-up} = \frac{T_{\mathrm{JM}}}{T_{\mathrm{FI}}} \tag{5}$$

where T_{JM} denotes the coding time used by the H.264/AVC JM 17.2 reference software, and T_{FI} is the time taken by JM using the algorithm proposed in this article. T_{FI} also includes all the computational costs for the operations needed in order to prepare the information required by our proposal, e.g. transferences of data from host memory to device memory, and vice versa.

Note that no overlapping mechanism in communications is implemented; first of all because the memory transfers on average spend less than 2% of the total GPU execution time; but also because the JM reference encoder allocates the reference frames in the lists of references just before executing the ME algorithm and, as a consequence, our proposal is executed just after allocating the reference frames. Additionally, when using an overlapping mechanism, the GPU memory allocations have a large initialization overhead which must be amortized, but due to the proximity of the memory transfers and the GPU execution, the execution time is not reduced. On the other hand, the memory transfers from GPU memory are used to synchronize the CPU and GPU executions.

5.2.3 ΔPSNR and Δ bit-rate

The detailed procedures for calculating bit-rate and PSNR differences can be found in the work by Bjøntegaard [30],

Table 2 Timing results of the proposed encoder for 1080p sequences

		ME algorithm				Full encoder			
		Full search		UMHexagonS		Full search		UMHexagonS	
Sequence	#Frames	TR %	Speed-up	TR %	Speed-up	TR %	Speed-up	TR %	Speed-up
Crowd	500	98.94	94.62	78.26	4.60	97.94	48.64	65.70	2.92
Ducks	500	99.09	110.31	78.87	4.73	98.22	56.20	66.25	2.96
Intotree	500	98.90	91.29	73.47	3.77	97.90	47.54	60.35	2.52
Oldtown	500	98.53	68.08	69.67	3.30	97.18	35.49	55.76	2.26
ParkJoy	500	99.10	111.11	80.28	5.07	98.22	56.21	68.04	3.13
Average		98.91	92.12	76.11	4.19	97.89	47.46	63.22	2.72

and make use of Bjøntegaard and Sullivana's [31] common test conditions. These procedures have been recommended by the JVT Test Model Ad Hoc Group [32].

5.3 2D scenario (main profile)

Table 2 shows the timing results of our proposed H.264/AVC encoder when coding five full HD video sequences. The results are divided into two main parts: the timing results focusing exclusively on the algorithm presented in this article (*ME algorithm* column) and the timing results focusing on the complete H.264/AVC encoder (*Full encoder* column). Moreover, the proposed algorithm is tested against two search algorithms implemented by the JM reference encoder, and therefore the results are further divided depending on the reference algorithm used. These reference search algorithms are full search and UMHexagonS search [11].

The results show that the proposed algorithm outperforms both search algorithms. The proposed algorithm obtains a speed-up of over 92x (TR of 98.91%) on average, which means a speed-up of over 47x (TR of 97.89%) for the complete H.264/AVC encoder, when compared with full search. The proposed algorithm obtains a speed-up of over 4x (TR of 76.11%) on average, which means a speed-up of over 2.7x (TR of 63.22%) for the complete H.264/AVC encoder, when compared with UMHexagonS search.

Table 3 shows the Δbit-rate and ΔPSNR results of our proposed H.264/AVC encoder when coding five full HD video sequences. As in the previous table, the proposed algorithm is tested against the two above-mentioned search algorithms implemented by the JM reference encoder, and therefore the results are divided depending on the reference algorithm used. The proposed algorithm obtains an acceptable RD degradation (the RD degradation is negligible if the computational savings are taken into account) when compared with full search, and it surpasses the encoding efficiency obtained by UMHexagonS search. The proposed algorithm obtains, on average, a bit-rate increment of 3.16% and a PSNR loss of 0.074 dB when compared with full search. On the other hand, the proposed algorithm obtains, on average, a bit-rate decrement of 6.75% and a PSNR gain of 0.170 dB when compared with UMHexagonS search.

Figure 6 shows the RD graphic results for the reference and the proposed approach, for different sequences in 1080p format, from a value of 28 to 40 for QP, comparing them with the ones obtained by the full search and the UMHexagonS algorithms. Due to space limitations only four sequences are shown. The PSNR vs. bit rate obtained with the proposed encoder, based on our algorithm, deviates slightly from the results obtained when applying the sequential reference encoders; the curve obtained by the proposed encoder is located in the middle of the curves obtained by the other search algorithms. The curves of the proposed encoder show that the proposed encoder obtains better PSNR results for a given bit-rate than the curves obtained by the UMHexagonS algorithm (it improves upon the results of UMHexagonS). The curves of the proposed encoder show that the

Table 3 RD results of the proposed encoder 1080p sequences

		Full search		UMHexagonS	
Sequence	#Frames	Δ bit-rate (%)	Δ PSNR (dB)	Δ bit-rate (%)	Δ PSNR (dB)
Crowd	500	3.08	−0.090	−6.63	0.203
Ducks	500	0.42	−0.011	−1.80	0.046
Intotree	500	6.01	−0.119	−13.45	0.297
Oldtown	500	3.76	−0.076	−5.73	0.111
ParkJoy	500	2.53	−0.076	−6.12	0.195
Average		3.16	−0.074	−6.75	0.170

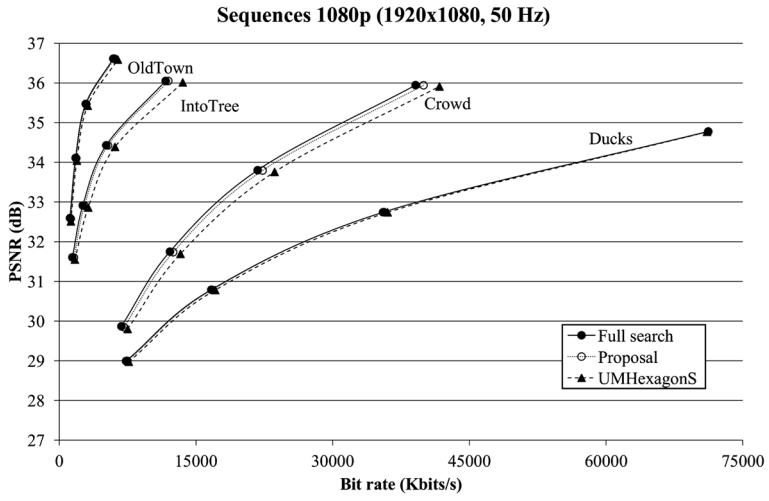

Figure 6 RD graphic results. 1080p sequences.

proposed encoder obtains slightly worse PSNR results for a given bit-rate than the curves obtained by the full search algorithm.

5.4 3D scenario (stereo high profile)

Table 4 shows the timing results of our proposed H.264/AVC encoder when coding five 3D full HD video sequences. As in the evaluation section shown for the Main Profile, the results are divided into two main parts: the timing results focusing exclusively on the proposed algorithm (*ME algorithm* column) and the timing results focusing on the complete H.264/AVC encoder (*Full encoder* column). The proposed algorithm is tested against two search algorithms, and therefore the results are further divided depending on the reference algorithm used.

As expected, the proposed algorithm outperforms both search algorithms. However, the TRs and speed-ups obtained are lower than the ones obtained when using the Main Profile. This behaviour occurs because there is more redundancy in a 3D video sequence than in a 2D video sequence and the reference encoder consumes less time. The execution time of the proposed algorithm is almost constant (it is content independent), while the reference search algorithms are content dependent. Full search is implemented using an early-out termination which is able to skip some search area positions based on the cost obtained for previously checked positions, and the UMHexagonS algorithm carries out less algorithm iterations.

The proposed algorithm obtains a speed-up of over 44x (TR of 97.74%) on average, which means a speed-up of over 22x (TR of 95.53%) for the complete H.264/AVC encoder, when compared with full search. Also, the proposed algorithm obtains a speed-up of over 2.7x (TR of 63.32%) on average, which means a speed-up of nearly 2x (TR of 48.23%) for the complete H.264/AVC encoder, when compared with UMHexagonS search.

Table 5 shows the Δbit-rate and ΔPSNR results of our proposed H.264/AVC encoder when coding five 3D full HD video sequences. As in the previous table, the proposed algorithm is tested against the two above-mentioned search algorithms implemented by the JM reference encoder.

The RD analysis when coding 3D video sequences is similar to the one obtained when coding 2D video

Table 4 Timing results of the proposed encoder 3D 1080p sequences

		ME algorithm				Full encoder			
		Full search		UMHexagonS		Full search		UMHexagonS	
Sequence	#Frames	TR %	Speed-up	TR %	Speed-up	TR %	Speed-up	TR %	Speed-up
Beergarden	150	98.37	61.43	66.82	3.01	96.73	30.61	51.42	2.06
Cafe	200	96.97	33.01	60.25	2.52	94.09	16.92	45.33	1.83
CarPark	250	97.98	49.48	61.05	2.57	95.99	24.96	45.99	1.85
Hall	200	97.39	38.27	65.66	2.91	94.89	19.57	50.77	2.03
Street	250	97.97	49.16	62.80	2.69	95.97	24.79	47.61	1.91
Average		97.74	44.15	63.32	2.73	95.53	22.40	48.23	1.93

Table 5 RD results of the proposed encoder 3D 1080p sequences

		Full search		UMHexagonS	
Sequence	**#Frames**	**Δ bit-rate (%)**	**Δ PSNR (dB)**	**Δ bit-rate (%)**	**Δ PSNR (dB)**
Beergarden	150	2.21	−0.069	−10.21	0.354
Cafe	200	2.64	−0.058	−9.04	0.224
CarPark	250	−0.07	0.002	−5.63	0.149
Hall	200	6.96	−0.125	−9.71	0.182
Street	250	1.55	−0.033	−7.26	0.178
Average		2.66	−0.057	−8.37	0.217

sequences. The proposed algorithm obtains slightly worse results when compared with the full search algorithm and surpasses the results obtained by the UMHexagonS algorithm. The proposed algorithm obtains, on average, a bit-rate increment of 2.66% and a PSNR loss of 0.057 dB when compared with full search. On the other hand, the proposed algorithm obtains, on average, a bit-rate decrement of 8.37% and a PSNR gain of 0.217 dB when compared with UMHexagonS search.

Figure 7 shows the RD graphic results for the reference and the proposed approaches, for different 3D sequences in 1080p format, from a value of 28 to 40 for QP, comparing them with the ones obtained by the full search and the UMHexagonS algorithms. Due to space limitations only four sequences are shown. As can be seen from the figure, the PSNR vs. bit rate obtained with the proposed encoder, based on our algorithm, deviates slightly from the results obtained when applying the sequential reference encoders. The conclusions are the same as the ones obtained in the RD analysis made for 2D video sequences.

6 Conclusion

In this article an algorithm that concurrently performs the inter prediction carried out over *P* and *B*-frames is presented. The approach implements the hierarchical B frame prediction implemented in the H.264/AVC JM 17.2 reference software encoder, and it is tested using the Main and the Stereo High Profile, as well as, two of the most well-known search algorithms available in this reference software.

The proposed algorithm is based on an efficient parallel implementation of the inter prediction procedure for both *P* and *B*-frames involved in the ME and DE processes, and also includes a mechanism to obtain an accurate estimation of how movement takes place (MVP) depending on whether the MB is temporary or inter-view predicted. Exploiting current GPU computational capability gives us another way to accelerate inter prediction in traditional video codecs, with the aim of developing real time video encoders.

When using the Main Profile, the results show a noteworthy speed-up of over 47x with only a negligible RD drop compared with the reference encoder when using the full search algorithm, and a speed-up of 2.7x when using the UMHexagonS ME algorithm. The approach presented improves upon the coding efficiency obtained by the UMHexagonS ME algorithm.

When using the Stereo High Profile, the experimental results show a speed-up for the complete H.264/AVC encoder of over 22x when compared with the full search

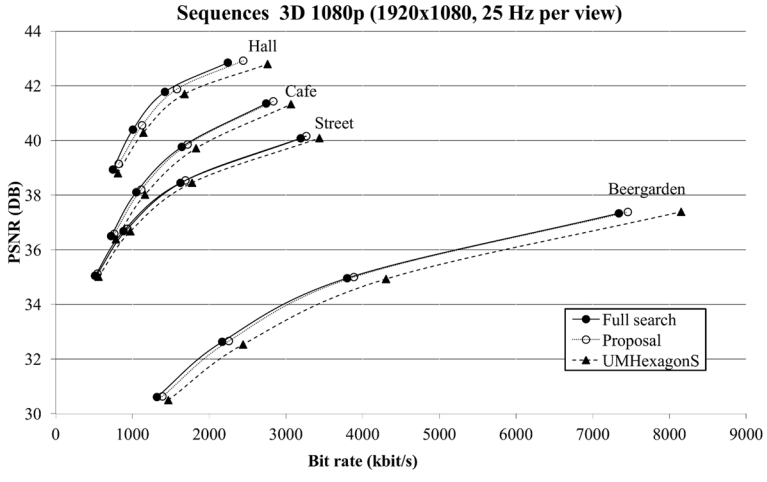

Figure 7 RD graphic results. 3D 1080p sequences.

algorithm, and a speed-up of nearly 2x when compared with the UMHexagonS algorithm. The proposed encoder slightly degrades the RD performance obtained when using the full search algorithm, but improves the RD performance obtained when using the UMHexagonS algorithm.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This work was supported by the Spanish MEC and MICINN funds, under the grants TIN2009-14475-C04-03 and TIN2012-38341-C04-04.

Author details
[1]Instituto de Investigación en Informática de Albacete, Universidad de Castilla-La Mancha, Avenida de España s/n, 02071 Albacete, Spain. [2]Departamento de Informática, Universidad de Valencia, Avenida de la Universitat s/n, 46100 Burjassot, Valencia, Spain. [3]Multimedia Lab. Ghent University–IBBT, Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium.

Received: 15 November 2012 Accepted: 20 February 2013
Published: 02 April 2013

References

1. G Shen, GP Gao, S Li, HY Shum, YQ Zhang, Accelerate video decoding with generic GPU. IEEE Trans. Circ. Syst. Video Technol. **15**(5), 685–693 (2005)
2. J Krüger, R Westermann, Linear algebra operators for GPU implementation of numerical algorithms. ACM Trans. Graph. **22**(3), 908–916 (2003)
3. NVidia, NVidia CUDA C programming guide 4.2. NVidia 2012 http://developer.nvidia.com
4. T Wiegand, G Sullivan, G Bjontegaard, A Luthra, Overview of the H.264AVC video coding standard. IEEE Trans. Circ. Syst. Video Technol. **13**(7), 560–576 (2003)
5. G Sullivan, JR Ohm, A Ortega, E Delp, A Vetro, M Barni, dsp Forum-future of video coding and transmission. IEEE Signal Process. Mag. **23**(6), 76–82 (2006)
6. ISO/IEC, ISO/IEC 14496-10:2003, Information technology—Coding of audio–visual objects—Part 10: advanced video coding (2003)
7. JR Ohm, GJ Sullivan, H Schwarz, TK Tan, T Wiegand, Comparison of the coding efficiency of video coding standards-including high efficiency video coding (HEVC). IEEE Trans. Circ. Syst. Video Technol. **22**(12), 1669–1684 (2012)
8. T Wiegand, H Schwarz, A Joch, F Kossentini, G Sullivan, Rate-constrained coder control and comparison of video coding standards. IEEE Trans. Circ. Syst. Video Technol. **13**(7), 688–703 (2003)
9. H Schwarz, D Marpe, T Wiegand, in *2006 IEEE International Conference on Multimedia and Expo ICME '06*. Analysis of hierarchical B pictures and mcTF (Toronto Canada, 2006), pp. 1929–1932
10. Join collaborative Team on video coding, Reference software to committee draft, version 17.2 (2011). http://iphome.hhi.de/suehring/tml
11. CA Rahman, W Badawy, in *Proceedings of the Fifth International Workshop on System-on-Chip for Real-Time Applications IWSOC '05*. UMHexagonS algorithm based motion estimation architecture for H.264/AVC (IEEE Computer Society Washington DC, USA, 2005), pp. 207–210
12. ISO/IEC, ISO/IEC 14486-2 PDAM1: Infomation Technology-Generic Coding of Audio-Visual Objects-Part 2: Visual (1999)
13. ITU–T, ISO/IEC, ITU–T, ISO/IEC 14496-10:2007, Advanced Video Coding for Generic Audiovisual Services (MVC extension) (2009)
14. A Vetro, T Wiegand, G Sullivan, Overview of the stereo and multiview video coding extensions of the H.264/MPEG-4 AVC standard. Proc. IEEE. **99**(4), 626–642 (2011)
15. P Merkle, A Smolic, K Muller, T Wiegand, Efficient prediction structures for multiview video coding. IEEE Trans. Circ. Syst. Video Technol. **17**(11), 1461–1473 (2007)
16. WC Feng, D Manocha, High-performance computing using accelerators. Parallel Comput. **33**(10), 645–647 (2007)
17. CY Lee, YC Lin, CL Wu, CH Chang, YM Tsao, SY Chien, in *2007 IEEE International Conference on Multimedia and Expo CME '07*. Multi-pass and frame parallel algorithms of motion estimation in H.264/AVC for generic GPU (Beijing, China, 2007), pp. 1603–1606
18. WN Chen, HM Hang, in *2008 IEEE International Conference on Multimedia and Expo ICME '08*. H.264/AVC motion estimation implementation on compute unified device architecture (CUDA) (Hannover, Germany, 2008), pp. 697–700
19. NM Cheung, X Fan, O Au, MC Kung, Video coding on multicore graphics processors. IEEE Signal Process. Mag. **27**(2), 79–89 (2010)
20. HC Tseng, CR Chang, YL Lin, in *2005 IEEE Workshop on Signal Processing Systems Design and Implementation*. A hardware accelerator for H.264/AVC motion compensation (Athens, Greece, 2005), pp. 214–219
21. J Zheng, W Gao, D Wu, D Xie, A novel VLSI architecture of motion compensation for multiple standards. IEEE Trans. Consumer Electron. **54**(2), 687–694 (2008)
22. LF Ding, PK Tsung, SY Chien, WY Chen, LG Chen, Content—aware prediction algorithm with inter-view mode decision for multiview video coding. IEEE Trans. Mult. **10**(8), 1553–1564 (2008)
23. J Huo, Y Chang, M Li, Y Ma, in *IEEE International Symposium on Circuits and Systems (ISCAS '09)*. Scalable prediction structure for multiview video coding, (2009), pp. 2593–2596
24. L Shen, Z Liu, S Liu, Z Zhang, P An, Selective disparity estimation and variable size motion estimation based on motion homogeneity for multi-view coding. IEEE Trans. Broadcast. **55**(4), 761–766 (2009)
25. Z Deng, K Jia, YL Chan, CH Fu, WC Siu, Fast motion and disparity estimation for multiview video coding. Front. Comp. Sci, China. **4**, 571–579 (2010). http://dx.doi.org/10.1007/s11704-010-0061-z, doi:10.1007/s11704-010-0061-z
26. CT Lu, HM Hang, in *IEEE Visual Communications and Image Processing (VCIP '11)*. Multiview encoder parallelized fast search realization on NVIDIA CUDA (Tainan City, Taiwan, 2011), pp. 1–4
27. LG Chen, WT Chen, YS Jehng, TD Chiuch, An efficient parallel motion estimation algorithm for digital image processing. IEEE Trans. Circ. Syst. Video Technol. **1**(4), 378–385 (1991)
28. R Rodríguez-Sánchez, JL Martínez, G Fernández-Escribano, JL Sánchez, J Claver, H.264/AVC inter prediction on accelerator-based multi-core systems. Multimed. Tools Appl, 1–21 (2012)
29. IEG Richardson, *The H.264 Advanced Video Compression Standard*, 2nd edn. (Wiley, 2010)
30. G Bjontegaard, in *13th VCEG-M33 Meeting*. Calculation of average PSNR differences between RD-Curves (Austin, Texas, USA, 2001)
31. G Sullivan, G Bjontegaard, Recommended simulation common conditions for H.26L coding efficiency experiments on low-resolution progressive-scan source material. *ITU-T VCEG , Doc. VCEG-N81*. Santa Barbara, California, USA (2001)
32. JVT Test Model Ad Hoc Group, Evaluation Sheet for Motion Estimation. Draft version, 4 (2003)

doi:10.1186/1687-6180-2013-67
Cite this article as: Rodríguez-Sánchez *et al.*: Adapting hierarchical bidirectional inter prediction on a GPU-based platform for 2D and 3D H.264 video coding. *EURASIP Journal on Advances in Signal Processing* 2013 **2013**:67.