Optimization of the motion estimation for parallel embedded systems in the context of new video standards

Fabrice Urban, Olivier Déforges, Jean François Nezan

Optimization of the motion estimation for parallel embedded systems in the context of new video standards

Fabrice Urban[a], Olivier Déforges[b] and Jean-Francois Nezan[b]

[a]Institut National des Sciences Appliquées de Rennes (France)
[b]Technicolor (France)

ABSTRACT

The efficiency of video compression methods mainly depends on the motion compensation stage, and the design of efficient motion estimation techniques is still an important issue. An highly accurate motion estimation can significantly reduce the bit-rate, but involves a high computational complexity. This is particularly true for new generations of video compression standards, MPEG AVC and HEVC, which involves techniques such as different reference frames, sub-pixel estimation, variable block sizes. In this context, the design of fast motion estimation solutions is necessary, and can concerned two linked aspects: a high quality algorithm and its efficient implementation. This paper summarizes our main contributions in this domain. In particular, we first present the HME (Hierarchical Motion Estimation) technique. It is based on a multi-level refinement process where the motion estimation vectors are first estimated on a sub-sampled image. The multi-levels decomposition provides robust predictions and is particularly suited for variable block sizes motion estimations. The HME method has been integrated in a AVC encoder, and we propose a parallel implementation of this technique, with the motion estimation at pixel level performed by a DSP processor, and the sub-pixel refinement realized in an FPGA. The second technique that we present is called HDS for Hierarchical Diamond Search. It combines the multi-level refinement of HME, with a fast search at pixel-accuracy inspired by the EPZS method. This paper also presents its parallel implementation onto a multi-DSP platform and the its use in the HEVC context.

Keywords: motion estimation, video compression, real-time

1. INTRODUCTION

With the current communication systems and the improvement of video compression, video broadcasting is more and more widespread. Video compression is now an important feature of 3G cell phones, personal digital assistants, and other battery-powered devices. This kind of devices are very often based on Digital-Signal Processors (DSP) to optimize the performance-consumption ratio but have limited hardware resources. Video codecs are also needed in base stations for inline transcoding or in Real-time H.264 HD video encoding solutions. Here again DSP are widely chosen in multiple components and/or multiple cores hardware platforms.

Even though the area of video compression has existed for many decades, programming a coding algorithm is still a challenging problem. The actual bottleneck is to provide compressed video in real-time to communication systems. Real time encoders have to cope with timing constraints and HD video formats. All those constraints have to be solved while keeping a good tradeoff between visual quality and compression rates. The investigation and understanding of the foundations of video compression is therefore more important than ever. In this context, Motion Estimation (ME) is known to be a key operation.

A highly accurate ME can significantly reduce the bit-rate of a video stream, but involves a high computational complexity. The high performance of H.264 and HEVC is mainly due to improved motion compensation modes such as variable block-size motion compensation, multiple reference pictures and Fractional-accuracy Motion Estimation (FME).[1] However the introduction of numerous modes raises the complexity of the codec and makes real-time H.264 compression challenging, especially for high-definition video. Integer Motion Estimation (IME) has been widely studied in the past few years. Fast algorithms have been developed to reduce the computational burden with limited quality loss. The goal of this paper is to study ME algorithms and their use in the H264 standard in terms of both quality and complexity. It is an evolution of previous work[2] in that we introduce new implementation performance results and the reuse of the ME algorithm in the HEVC context. The reuse of the ME algorithms in the context of H264 or HEVC lead to modify the parameters of the algorithms, validating the

choice of a programmable multicore DSP system. An implementation of fast variable block-size is also presented and new application perspectives are proposed.

2. MOTION ESTIMATION TECHNIQUES

Motion estimation goal is to find relative motion between two images in order to eliminate temporal redundancy. For video compression where the picture is usually divided into blocks, Block Matching Algorithms (BMA) are most widely preferred. The basic hypothesis are non-deformable objects having an apparent translation in the image plane. One motion vector can then be estimated for each block.

2.1 Introduction to Block Matching Algorithms

BMA consists in searching for each $M \times N$ block of the current picture a match in a reference picture. A distance measure is computed between the current block and some candidates. This measure is most often the Sum of Absolute Differences (SAD) for its implementation simplicity.

The simplest BMA is the full search where every candidate within a search window of magnitude p pixels is considered. It is very computationally intensive. As the required processing power is too high, a lot of fast algorithms had been proposed using essentially three optimization techniques.

The first one computes a full SAD the least often as possible.[3,4] It is actually possible to eliminate rapidly bad candidates before the full SAD is computed. These algorithms reduce drastically the number of calculations, however they use a lot of test operations which make the implementation difficult to optimize and they do not consider memory bandwidth. Thus even if the computation time is statistically improved, the worst case might lead to a worse result than full search, which is unacceptable in a real-time context.

The second one reduces the candidate set by choosing a most likely search direction as soon as possible, under the assumption that the error (SAD) surface is monotonic. As this is not always verified, some algorithms get trapped in local minima. Chen and Al[5] suggest to search in one direction at a time, whereas logarithmic search proposed by Jain and Jain[6] and three step search from Koga and Linuma[7] begin by a coarse estimation then refine the result. In[8] motion is estimated recursively. At each step the SAD for some candidates in a diamond pattern around current position are computed. The motion is recursively refined following the decreasing SAD direction.

The third technique takes video sequence contents into account: motion fields present some continuities (spatially and temporally), so it is possible to predict the movement of a block from the neighboring blocks and previous images. A set of predictors is then available (from the causal neighborhood). Each of them is then evaluated (by calculating the SAD with the current block) and a local search is performed around the best one (which minimizes the SAD) to refine the movement. A lot of algorithms using this technique have been developed.[9–13] They differ by their predictor sets and their local search patterns. HME (Hierarchical Motion Estimator)[14] introduces a coarse to fine picture definition decomposition to add reliable hierarchical predictors.

EPZS[10] and HME are particularly interesting for a DSP implementation. They are the two finally selected techniques from which a third one called HDS is derived. They are more precisely detailed in the following.

2.2 EPZS

EPZS (Enhanced Predictive Zonal Search) algorithm is an improvement of PMVFAST[11] algorithm thanks to new predictors. The prediction step is consequently more accurate and the local search (Fig. 1-left) is thus reduced. Coarse refinement with a large diamond pattern in PMVFAST is unnecessary. The best predictor is directly refined using a small diamond or square pattern (Fig. 1-right). The improvement of prediction step reduces execution time. The refinement step consists in a recursive diamond search: the current best vector (initialized with the prediction step) is compared to its neighbors according to the test pattern and the the best vector becomes the new search center.

An early stopping criterion already present in PMVFAST speeds-up the operation by avoiding unnecessary computations. The process is stopped as soon as the result is “good enough”, i.e. SAD is lower than an adaptive

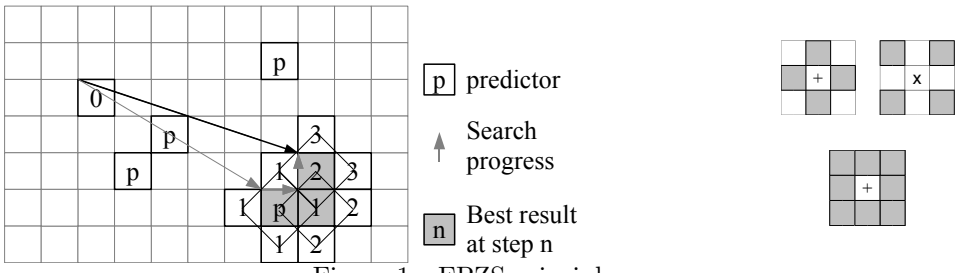


Figure 1: EPZS principles

threshold. The execution time is thus low but highly depends on the video sequence. This motion estimator has a low computation load and thus is a good candidate for a fast software implementation of motion estimation. EPZS is used in software video encoders such as XviD and JM H.264 reference software.

In our EPZS implementation, the early stop criterion has not been implemented in order to get a constant execution time. As a consequence the quality is slightly increased.

2.3 HME

The Hierarchical Motion Estimator (HME)[14] is based on a multi-level refinement process where the motion vectors are first coarsely estimated on a sub-sampled picture. The algorithm starts by building a pyramid of pictures (Fig. 2). Level 0 is the full-resolution picture, the level $n+1$ is level n low-pass filtered and sub-sampled picture.

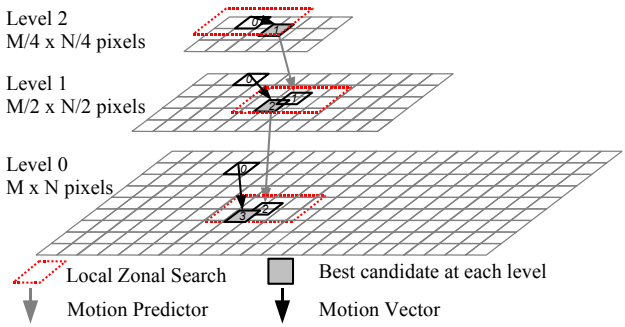


Figure 2: Pyramid of pictures in HME

A sub-sampled motion field is firstly estimated on the low resolution picture (highest level), then the motion field is successively refined. The block size through the pyramid is constant so that global motions are detected on the coarsest levels and refinement is achieved when resolution is increased. At each resolution level, a predictive motion estimation is performed, as for EPZS, with the difference that reliable hierarchical predictors are provided from lower resolution level.

The refinement step is a reduced full search around the best predictor. In addition to the motion estimation operations, HME implementation takes into account the computation of the sub-sampled pictures pyramid. Each level is a sub-sampled picture of the lower level's to which a 3-tap Gaussian low-pass filter is applied.

The local search and predictive mechanism in EPZS and HME naturally provide a low entropy homogeneous motion field. This is an advantage for video compression.

2.4 HDS algorithm

The two methods precedently presented have each their advantages. EPZS is very fast thanks to its recursive diamond search window but large motion vectors can not be found due to its limited search window. HME search window is not limited but its local full search requires more processing power. To combine the interest of both and to keep their advantages, we propose here a new algorithm based on a HME and EPZS combination: HDS (Hierarchical Diamond Search)is a recursive diamond search applied to a multi-level decomposition.

Zonal search	Predictors
EPZS: 1 resolution level	
8-neighbors pattern Diamond search	1 temporal
No early stop	4 spatial
HME: 4 resolution levels	
full search +/- 16 at lowest resolution	5 hierarchical
reduced full search (+/- 3)	4 spatial
HDS: 4 resolution levels	
full search +/- 16 at lowest resolution	5 hierarchical
8-neighbors pattern Diamond search	4 spatial

Table 1: Implementation details

The multi-level decomposition provides robust prediction. The reduced resolution levels on the top of the pyramid (Fig. 2) allow the detection of large motion with only a reduced search window. In the coarsest resolution level, a reduced full search search is performed to catch very large motion. Because the image size is reduced, the impact on computation cost is negligible. As an example, in high definition, a four-level pyramid with a search range of +/-16 in the coarsest level can catch small objects with motion as large as 128 pixels of amplitude. The multi-resolution approach provides robust hierarchical predictors.

At each level the motion is estimated block per block, and for each block, the motion is first predicted from hierarchical and spatial predictors. The first kind of predictors provides large motion detection abilities whereas the second one provides accurate information on the neighborhood movements and favors a homogeneous motion field and fast convergence of the algorithm. The different predictors are evaluated on a SAD basis and the best one is selected for a refinement step.

The refinement step consists in recursively trying a small displacement around the current best motion vector. This local search is initialized with the results of the predictive step. Then, at each iteration, a displacement of one pixel in every direction (eight neighbors) is analyzed. The best position is chosen as the new search center of the recursive process. If the best position is already at the center of the pattern, the search stops. To ensure limited calculation, the number of iterations can be bounded to a maximum value.

The hierarchical approach together with fast diamond search ensure both a robust predictive step and a fast processing. The resulting motion field is reliable and close to the physical motion.

3. REAL-TIME IMPLEMENTATION ON DSP

Real-time motion estimation implementation for MPEG-4 H.264 AVC[15] high definition video encoding is challenging. With tools such as variable block size, quarter-sample accuracy and multiple reference pictures, motion estimation needs high computation power and memory bandwidth. The detailed implementations of the three algorithms are given in table 1. Several SD (576p: 720x576) and HD (720p: 1280x720) video sequences have been used to test HDS performances and compare them to HME and EPZS ones. The content of sequences varies from high and complex movement (Formula1, Football) to small motion (RaidMaroc, Horses).

Results are presented with two distinct criteria: execution time and motion estimation quality. Quality is more important in high end solutions such as video broadcasting whereas for low cost solutions, execution time (or algorithm complexity) must be kept low. In order to provide consistent results, motion estimators have been implemented and optimized onto a Texas Instrument TMSC6416 DSP at 1Ghz. Digital Signal Processors (DSP) offer high flexibility and reduced implementation time compared to a full hardware solution. Numerous peripherals can be embedded such as a DMA controller, an Ethernet controller, a viterbi decoder, a cache controller, etc. The programming is done in “C” language which allows various algorithms to be tested. Because of their low cost and high performance, DSPs are popular for the prototyping of multi-processor signal processing applications. For quality comparisons, the motion estimators have been implemented in the JM H.264 video

encoder. In this section only 8x8 inter-frame coding mode is allowed on P frames to eliminate the influence of a decision algorithm and intensively stress the motion estimator. Others tools will be discussed later on.

3.1 DSP optimizations

To get the best performance out of these processors, platform-independent optimizations such as loop unrolling and loop interchanging are usually done. Memory access can also be very time-consuming. For video processing and especially high-definition, internal memories are too small, which involves inherent external data accesses. Without an enhancement mechanism this causes performance to drop up to 2 orders of magnitude. Some DSPs integrate a cache controller that provides an automatic way to significantly reduce performance loss but it involves the user to ensure memory consistency in a multi-component context. In previous work, we developed an automatic cache management tool that uses the integrated cache controller of the device.[16] Using a prototyping methodology, inter-processor communications are automatically generated, external memory accesses are enhanced and data consistency is ensured by the tool. HD image and video processing applications implementation on a DSP is made fast and reliable with very limited memory constraints. The user can thus focus on application specific optimizations.

HME and EPZS have been implemented and optimized for TI C64x DSP. Loops have been optimized using SIMD vectorization and loop unrolling. Compilation process have been optimized with specific key-words like "#pragma", "restrict", "inline" and "const", and memory accesses have been enhanced using cache, on-chip memory and EDMA (Enhanced Direct Memory Access peripheral) transfers. Execution times have been reduced by a factor of 5 compared to the original code with only straightforward compilation optimizations. Results will be detailed in the following section.

To compute the multi-resolution image pyramid, a separable filter has been implemented. The chosen 3-tap low-pass filter is optimized together with the sub-sampling in order to compute only needed samples (every other pixel horizontally and vertically) and reduce constraints on memory bandwidth. Furthermore, memory access is also optimized for high definition resolution where data is located in external memory. In this case, computations are performed concurrently with memory accesses using the on-chip EDMA.

H.264 FME feature allows the use of quarter-sample precision motion vectors. The compression efficiency is highly improved at the cost of a higher computational complexity. The picture definition is increased by interpolating successively at half-sample accuracy with a 6-tap filter and at quarter-pel accuracy with a linear filter. The quarter-pel precision can be achieved by different means; the first one is to search directly in the interpolated picture with a search window 4 times as large horizontally and vertically, involving a 16:1 data and bandwidth increase ratio. The second one consists in 2 steps:[17] the motion estimation is firstly performed at pixel accuracy then refined at sub-pixel accuracy. This last method allows to compute sub-pixel samples "on the fly" meaning that half and quarter pixel positions are interpolated only when needed. Implementations can thus take advantage of high bandwidth local memories such as caches. To reduce further computations for FME, the motion vector is also refined first to half-pixel accuracy (8 neighbors), then to quarter-pixel accuracy starting from the half-pixel location (8 neighbors). The sub-pixel search has an amplitude of $\frac{3}{4}$-pixel in each direction. This two-step technique gives one sub-pixel position out of 49 (7×7) with only 16 ($8 + 8$) search points. Quarter-pixel samples are interpolated after the half-pixel best position is known hence only necessary values are computed.

3.2 Execution times

Fig. 3 gives the execution times for three implemented motion estimators on SD (720x576) and HD (1280x720) progressive image sequences. For each algorithm, pixel and quarter-pixel accuracy versions have been considered. Motion Estimation is performed onto 8x8 blocks. Regardless the resolution and accuracy, EPZS Motion estimator is faster than HME and HDS. It can be partly explained by the computation of the multi-resolution pyramid and the motion estimation of lower levels. In addition, HME also has more predictors and a zonal search inducing more computations.

EPZS and HDS implementations at quarter-sample accuracy reach respectively more than 30 and 25 frames per second on a DSP for high definition video. For a HME implementation at 30 frames per seconds, the processing power of at least 2 DSPs is needed, for example in a two-stage pipeline composed of hierarchical levels

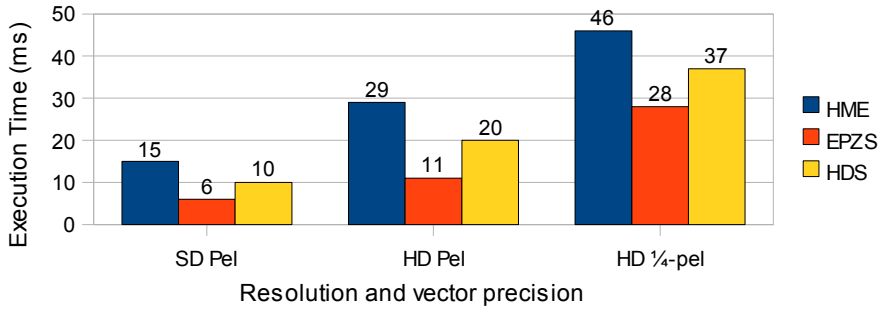
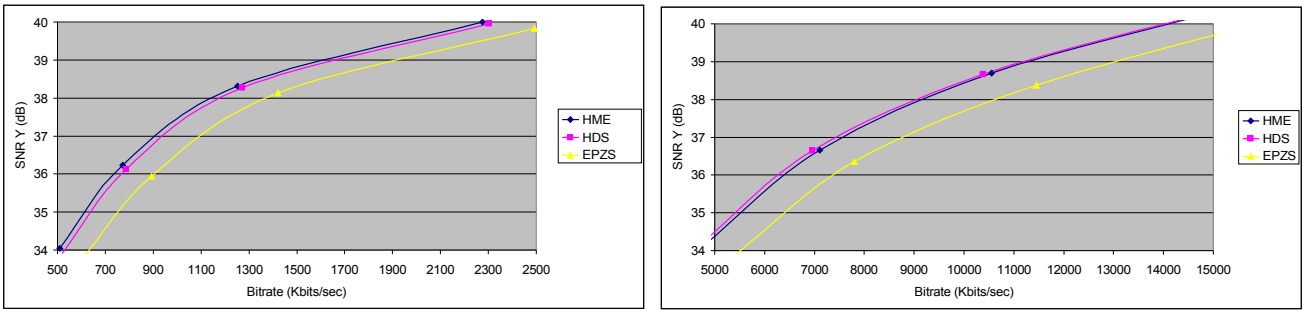


Figure 3: Execution time comparison of the motion estimators



(a) Formula1 720x576 (b) Football 1280x720

Figure 4: Rate/distortion curves for SD and HD sequences

in the first stage and one full resolution level in the second. The full search executed at a given level in HME leads to more calculations than HDS Diamond Search. Both HME and HDS hierarchical algorithms are more complex than EPZS.

3.3 Motion estimation quality

In order to evaluate the quality of motion vector fields for each technique, each motion estimator has been integrated in an H.264 encoding software. For the comparison purpose, P and I frame are allowed, with one I frame every 25 frames. In P frames only 8x8 inter-frame mode is activated to stress the motion estimator and compare only motion field quality, with no decision interfering. Rate control has also been deactivated so that rate/distortion curves based on PSNR reflect the ability of the motion estimator to find a good match.

A few sequences have been encoded to evaluate the compression performances of h.264 encoding with the motion estimators. SD formula1 and HD football are high motion sequences with traveling and many moving objects. These sequences highlight the matching ability of motion estimators. SD RaidMaroc and HD horses are slow motion sequences with few moving objects in favor of low entropy motion fields. All these sequences are common content and must be well handled by the encoder. Fig. 4 shows the rate/distortion curves corresponding to the first 200 pictures of two sequences: SD fomula1 and HD football. For each sequence, the quality (mean PSNR) is plotted against the mean bit-rate. Fig. 5 sums up results for several SD and HD video sequences. It represents the data-rate increase reached at a given quality (constant PSNR). It is expressed using the difference with the data-rate reached with the encoding software based on HME motion estimation solution.

These results show that EPZS algorithm lead to a bit-rate increase of almost 20% for high motion sequences, which is unacceptable for high-end solutions. With results comparable to HME (less than 4% bit-rate increase on the worst case and 1.5% decrease on the best case), HDS appears to be a good trade-off between encoding performances and processing time. This illustrates the ability of HDS to find an appropriate vector in case of high motion sequences and a low entropy motion field in case of slow motion. Hierarchical algorithms have a more robust prediction step whereas EPZS has a limited search range and relies on temporal prediction. Therefore, HME and HDS perform a lot better than EPZS at scene cuts.

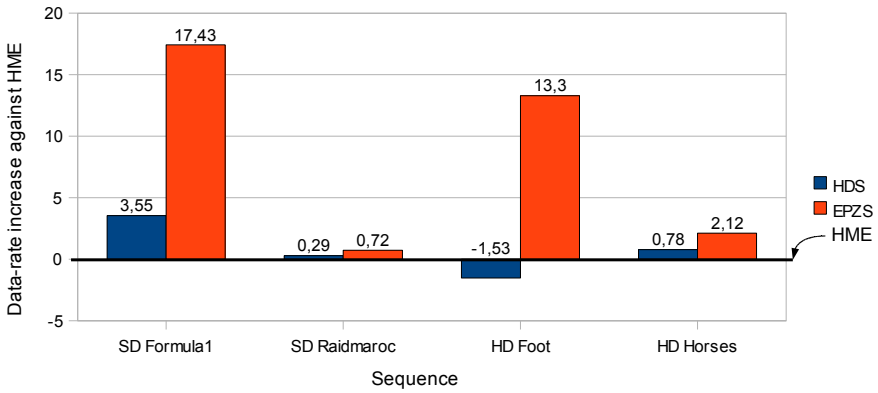


Figure 5: Encoding results for SD (576p) and HD (720p) video

The reduced range of the motion vectors well handled by EPZS is a limitation in high motion sequences. Moreover, the displacement between a frame to encode and a reference frame will be increased when using B pictures, and even more in case of hierarchical GOP structure because the motion vector amplitude increases. Therefore, EPZS-based techniques need to be associated to a vector-tracing technique[18] to reach efficient vector prediction. It may involve the estimation of motion fields not used by the video encoder but by the motion estimator prediction step only, thus increasing processing load.

Motion estimation performances of HME and HDS are comparable, and outperform EPZS algorithm. Performances improvement of HDS over EPZS is worth its slight increase in computational complexity. HDS motion estimation algorithm is therefore chosen for the rest of this paper.

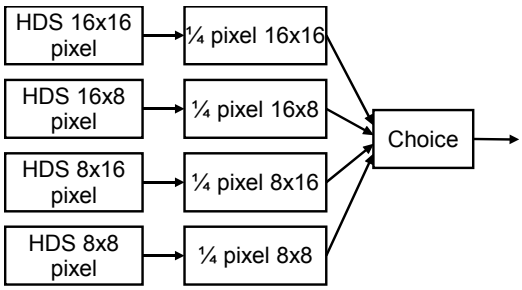
3.4 Variable block-size and quarter-pixel

On one hand, variable block-size and quarter-pixel motion compensation bring effective compression gain among the various coding tools of AVC.[19] On the other hand, the computational complexity of the motion estimation operation is consequently highly increased, making real-time implementation of this operation challenging for high definition video. For motion compensation in the H.264 standard, 16×16 pixel macro-blocks can be divided in 16×8, 8×16 or 8×8 partitions. The last one can be further split in 8×4, 4×8 or 4×4 blocks. Choosing small blocks improves motion compensation but increases the coding cost as more motion vectors need to be transmitted. For high definition video, it has been shown that block-size smaller then 8×8 brings little compression improvement considering the complexity it brings. This section describes a variable motion estimator handling 16×16 to 8×8 block sizes.

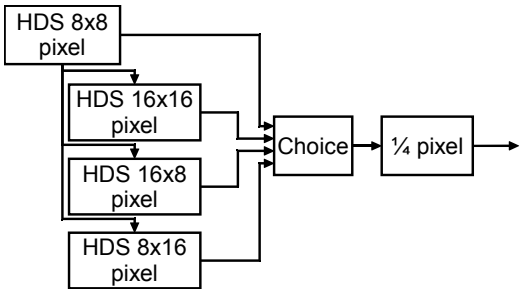
The straightforward implementation of the variable block-size motion estimator leads to the solution presented in Fig. 6-a: the scheme for one block-size is repeated for every block-size. The main drawback is the increased amount of computation and memory bandwidth.

The first optimization is to start motion estimation including multi-resolution levels with one block-size at integer-pixel accuracy. The results serve then as accurate prediction for other block sizes and hierarchical level motion fields are reused without re-computation. A good initial block-size must be not too big to catch small objects motion and not too small to be less sensitive to noise. Moreover a block size closer to the deduced other sizes (4×4 to 16×16) statistically improves prediction accuracy. Therefore, we chose 8×8 size as a first motion estimation step.

Secondly, fractional-pixel refinement implies heavy interpolation operations and matching evaluation. In variable block-size, every output vector must be refined to quarter pixel accuracy to avoid compression performances drop. To reduce computation complexity, we choose to select the best motion compensation partition after

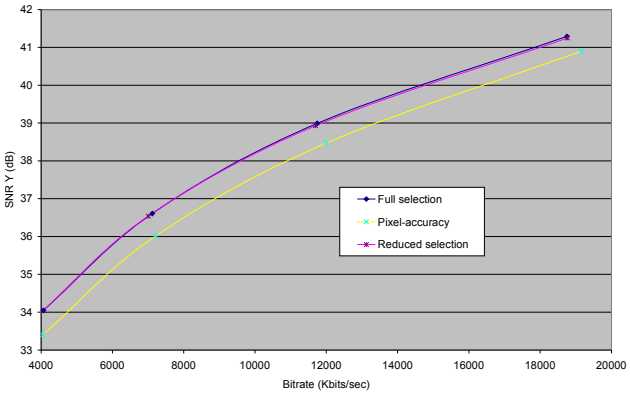


(a) Full selection



(b) Reduced selection

Figure 6: Variable block-size implementation



(a) Quality

Figure 7: HDS variable block size mode on HD sequence Football

pixel-accuracy results in order to refine only one block-size per macro-block and thus avoid unnecessary calculation (Fig. 6-b). Therefore it drastically reduces computation burden with limited impact if any on compression performances.

Fig. 7 shows the typical result of this computational reduction on compression performances. Variable block-size is activated on the encoder. For the reduced selection variable block-size algorithm, the block size is chosen at the motion estimator, otherwise we let the encoder decide. The rate/distortion curve shows clearly the improvement of quarter-pixel refinement for variable block-size on compression performances. The impact of the reduced selection algorithm appears to be very limited and negligible.

Fig. 8 is a comparison of execution times on the TI DSP. Thanks to the reduced selection quarter-pixel refinement, the motion estimation process is accelerated by a factor of two compared to the full selection implementation without modifying the hardware. The overhead of sub-pixel refinement is thus minimized compared to the full sub-pixel refinement.

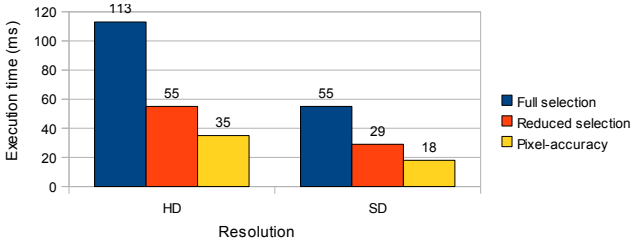


Figure 8: HDS Variable block-size implementation comparison

The reduced selection solution drastically lowers the computational complexity of variable block-size and quarter pixel motion estimation for video compression with very low impact on compression performances if any. For one reference frame, this solution reaches 30 frames per second on one DSP in standard definition. For high definition, a multi-component solution could reach real time. More precisely, the interpolation operations needed for sub-pixel ME could benefit from the high level of parallelism offered by an FPGA.

4. MULTI-COMPONENT HETEROGENEOUS ME

Integer motion estimation algorithms perform well on software based processors such as DSPs. Fast algorithms take advantage of branching and random memory access abilities. Fractional-pixel accuracy, however, requires huge computation power and memory bandwidth due to interpolation filters and distortion evaluation. These operations take advantage of the high parallelism of VLSI implementation.

A heterogeneous multi-component motion estimator has been prototyped on a platform from comprising A TI C6416 DSP at 1GHz and a Virtex-2 Pro FPGA. The inter-connection bus between the DSP and the FPGA is 32-bit wide and is clocked at 133 MHz. The DSP handles IME and allows algorithmic enhancements to reduce sub-pixel refinement complexity, whereas the Virtex 2 Pro FPGA runs as a quarter-pixel motion refinement coprocessor. The design takes into account both reduced inter-component communication bandwidth and the limited resources of the FPGA.

4.1 H.264 Motion estimation algorithm

IME algorithms have been presented in section 2. The EPZS algorithm is used in the following because the lack of the hierarchical level allows faster real-time ME for 720p video at 60 frames per second. Note that the HDS algorithm could also be implemented with a pipeline of 2 DSPs: one for the hierarchical levels, and one for the full resolution level with FME.

FME highly contributes to enhance coding efficiency of the MPEG-4/AVC H.264 standard. However, it involves a high computational complexity that is magnified by variable block-size. Computational and memory constraints are thus greatly increased. The quarter-pixel accuracy luminance picture is interpolated with two successive filtering operations. Half-pixel samples are interpolated first using a 6-tap separable FIR filter. Once half-pixel samples are available, quarter-pixel samples are computed using linear interpolation. Improving motion vector accuracy for a block-based motion estimator obviously increases complexity. Firstly the density of candidate motion vectors is increased. To limit the number of search points a two-step approach is generally preferred: the motion is estimated at integer pixel accuracy and then refined to quarter-pixel with a limited search range (usually]-1;1[pixel) around the integer-accuracy best match. Secondly the reference image must be enlarged, involving the use of interpolation filters and introducing higher memory constraints. In order to reduce these constraints, already very high for IME,[20] on-the-fly data interpolation is used : sub-pixel samples are computed blockwise, when needed. This restricts the 16:1 data increase ratio to temporary buffers. Hardware and software implementations can thus take advantage of high bandwidth local memories such as caches.

On the Texas Instruments C6416 DSP, the motion vector is refined to fractional-pixel accuracy in two steps: the motion vector is first refined to half-pixel accuracy (8 neighbors), then to quarter-pixel accuracy starting from the half-pixel location (8 neighbors). The sub-pixel search has an amplitude of $\frac{3}{4}$-pixel in each direction. The two-step technique gives one sub-pixel position out of 49 (7×7) with only 16 ($8+8$) search points. Quarter-pixel samples are interpolated after the half-pixel best position is known hence only necessary values are computed. Table 2 gives detailed execution times per 8x8 blocks. IME is performed in 900 ns. FME requires a 1200 ns overhead using an H.264 filters.

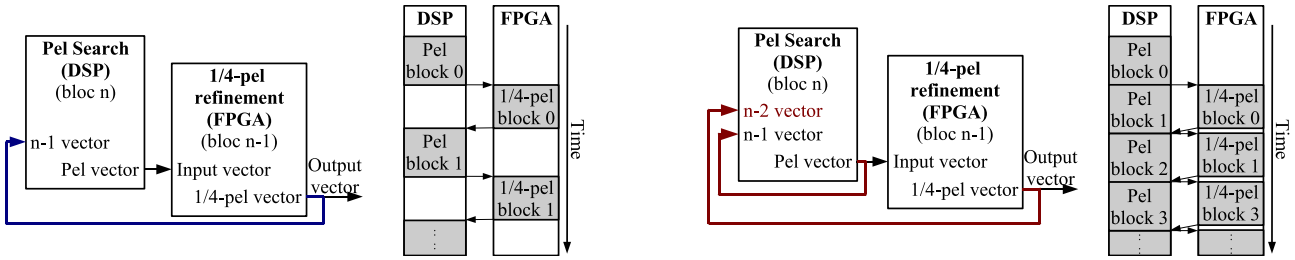
filter type	H.264
IME	900 ns (77 fps for 720p)
FME	1200 ns
Total IME+FME	2100 ns (33 fps for 720p)

Table 2: DSP implementation timings for a 8x8 block

The sub-pixel refinement operation is the most time-consuming operation in motion estimation on a DSP. One single DSP has not enough computation power to handle real-time quarter-pixel motion estimation of high-definition video at 60 frames per second. The next section presents a sub-pixel refinement coprocessor to lower the computational burden on the DSP.

4.2 FPGA as a sub-pixel refinement coprocessor

The IME is based on a predictive algorithm (cf Section 4.1). Previously estimated motion vectors are necessary to predict the current motion. This causes data dependencies between IME and FME which result in inevitable sequential processing. To take advantage of the parallel multi-component architecture, we propose to modify data dependencies (Fig 9): the motion vector of the left block is input in the IME stage at integer accuracy instead of quarter-pixel accuracy, and the result of the fractional-pixel accuracy refinement is one block delayed. As a result the motion estimation architecture is a 2-stage pipeline. The first stage is the IME on DSP and the second stage is the FME on FPGA.



(a) Sequential scheduling (b) Modified dependencies: parallel scheduling

Figure 9: Block-level pipeline implementation

Despite the small search window in the sub-pixel refinement step, it is the most time-consuming part of motion estimation. The sub-pixel refinement operation is very computationally intensive. Moreover, computation of interpolated samples and matching distortion involves only simple operators such as shifters and adders. It could therefore benefit from the high parallelism of VLSI implementation. To meet the DSP external bus constraints, data dependencies have been reduced to lower communication peak bandwidth.

4.2.1 Overall VLSI architecture design

To benefit from the high degree of parallelism of VLSI implementation, the algorithm must be regular. A two-step approach requires either to save half-pixel samples for subsequent quarter-pixel interpolation, or to recompute them.[21,22] Thus, in order to meet real-time constraints, interpolation filters must be oversized along with memory bandwidth. Therefore a quarter-pixel accuracy full search approach is adopted with a search range of $\frac{3}{4}$ pixels in each direction. 48 candidates ($(2 \times 3 + 1)^2$-1) are evaluated around the IME best match.

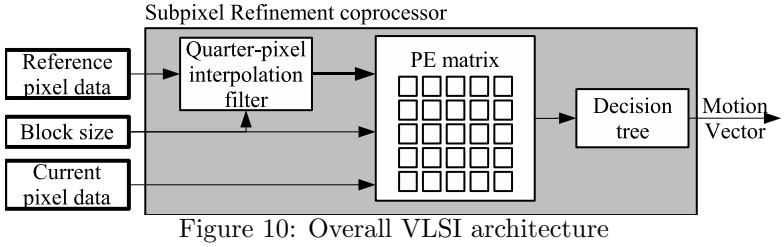


Figure 10: Overall VLSI architecture

The sub-pixel refinement coprocessor includes (fig 10) a quarter pixel interpolation module to generate sub-pixel samples on-the-fly. A Processor Element (PE) matrix computes a distortion measure for each candidate displacement. The distortion measure used is the Sum of Absolute Differences (SAD) because it needs far less hardware resource than an SATD. The best candidate (i.e. with the smallest SAD) is finally selected in the decision tree. Block-size parameters can be changed without any hardware modification in order to support multiple block-size motion estimation. The developed VLSI architecture is more precisely described in.[23]

4.2.2 Distortion computation matrix details

From the existing dedicated architectures for IME described in previous work,[20,24,25] designs using inter-level parallelism[20] (or type II[24]) are best-suited to line-scan input mode and best minimize hardware resources. In[24] the full search IME algorithm with a search range of $\pm p$ for a $M \times N$ block is expressed as 4 nested loops. Inter-level parallelism is obtained by unrolling search area loops in hardware. The distortion measure is computed simultaneously for every search point in $(2 \times p + 1)^2$ PEs. Consequently, the current pixel $x_1(k, l)$ is broadcast to all PEs and all reference pixels from $x_2(k - p, l - p)$ to $x_2(k + p, l + p)$ must be available and are propagated to the PEs through $(2p + 1)^2$ registers. In order to remove the registers needed to propagate reference data, we propose to invert the architecture. The sub-pixel refinement can thus be expressed by Alg. 1.

Algorithm 1 new FME nested loops

for $u = 1..M + 1$ (block height)
 for $v = 1..N + 1$ (block width)
  for $\Delta i = 0..1$ (integer vertical search range)
   for $\Delta j = 0..1$ (integer horizontal search range)
    for $\Delta g = -(a - 1)..0$ (fractional v. search range)
     for $\Delta h = -(a - 1)..0$ (fractional h. search range)
      $SAD(\Delta j, \Delta i) += |x_1(u - \Delta i, v - \Delta j)$
            $-x_2(au + \Delta g, av + \Delta h)|$
     end Δh
    end Δg
   end Δj
  end Δi
 end v
end u

PE Matrix

with $-(a - 1) \leq a\Delta i + \Delta g \leq (a - 1)$; $-(a - 1) \leq a\Delta j + \Delta h \leq (a - 1)$ and $1 \leq u - \Delta i \leq M$ and $1 \leq v - \Delta j \leq N$
loops Δg and Δh model fractional-pixel accuracy. Δi and Δj are integer-pixel displacements.

Loops Δi, Δj, Δg and Δh are unrolled in hardware (e.g. for quarter-pixel accuracy this results in a 7×7 PE matrix). Inequalities $1 \leq u - \Delta i \leq m$ and $1 \leq v - \Delta j \leq n$ are ensured in hardware by propagating an "*enable*" signal to the PEs along with current block data. Consequently, not all the results are available simultaneously. The first a^2 cost results are available after $M(N + 1)$ cycles, and subsequent results after appropriate delays. Reference data is broadcast to the PEs and the current block is propagated. The modification has little impact on IME, but presents several advantages when transposed to FME, beginning by the reduction of propagation registers. Indeed images x_1 and x_2 do not have the same scale in FME. Data density is thus higher for the search window than for the current block.

The proposed computation matrix architecture is adaptive to the block size and matches the interpolation filter design in order to reduce hardware requirements. A high degree of parallelism is achieved with very low external communication bandwidth. The design supports variable block-size with no hardware modification and full utilization of hardware resources.

4.3 Timing results

Several configurations of the motion estimator have been benchmarked for 720p high-definition video sequences (1280x720). Execution times are given in the table 3 for IME only on DSP, FME only on FPGA, and the complete heterogeneous quarter-pixel accuracy motion estimator (DSP+FPGA).

Sub-pixel refinement can be performed on the FPGA in only 842 and 1925 ns and is realized in parallel thus with no overhead on IME. Practical results present an execution time of 1250 and 4600 ns for 8x8 and 16x16. The small overhead is due to the prototyping platform constraints: for each block, both current block and reference window must be transferred by the DSP to input buffers on the FPGA. This increases the refinement operation time which has to take account of data transfers.

Block size	8x8	16x16
DSP pel	900 ns	4000 ns
FPGA $\frac{1}{4}$ pel refinement	842 ns	1925 ns
Total (Pel + $\frac{1}{4}$ pel)	1250 ns	4600 ns
DSP + FPGA	**(720p frame: 18 ms)**	**(16.5 ms)**

Table 3: Execution times for 8x8 and 16x16 blocks

The computation time of a 1/4-pixel high-definition motion field with only one DSP is around 30 ms for both 8x8 and 16x16 block-sizes. The addition of an FPGA at 133 MHz and an appropriate operation schedule significantly decrease running times, achieving 55 and 60 frames per second for 8x8 and 16x16 block-size respectively. These figures take data transfers between the DSP and the FPGA into account. The coprocessor design leads to low data bandwidth which is appropriate for the prototyping platform. Moreover, the programmability of the DSP results in implementation flexibility which allows complexity reduction for variable block-size and FME. The reduced selection quarter-pixel refinement algorithm presented above can be used with this architecture.

5. EXTENSION TO HEVC

In this article motion estimation has been studied for H.264 AVC compression. Nevertheless, the work can be reused for other standards, such as HEVC (High Efficiency Video Coding),[26] the newest video compression standard, currently under its final standardization stage, developed by a Joint Collaborative Team of ISO/IEC MPEG and ITU-T VCEG (JCT-VC). The Final Draft International Standard is expected to be delivered in January 2013. An overview is available in.[27] The overall requirements of HEVC is to improve the compression efficiency by a factor of at least two compared to the H.264/AVC compression standard. HEVC has been designed to target Ultra High Definition (UHD) with higher frame rates compared to H.264/AVC. Real-time constraints are thus strengthened compared to the current generation of video coding standards. In this context, multi-resolution ME such as HDS is well suited to UHD where large displacements are to be expected.

5.1 Inter coding tools in HEVC

Variable block size and sub-pixel ME are still important tools in HEVC, with even more modes and more complex interpolation filters. Coding Tree Blocks (CTB) (or LCU - Largest Coding Unit) can contain 1 or more Coding Units (CU) (Fig. 11) composed of one or more Prediction Unit (PU) which can be sub-partitioned into 4 square or 2 rectangular partitions. In the same way as macro-blocks in H.264/AVC, each PU partition is built with unidirectional or bi-prediction motion compensation, using $\frac{1}{4}$ (luma) or 1/8 (chroma) pel precision MV.[26] The range of variable block size is thus enlarged compared to H.264, up to 64×64 block size, with square, rectangular and asymmetrical size. The smallest PU size, 4×4, have recently been removed, but there remain a large choice of different prediction sizes to evaluate. A reduced selection quarter-pixel refinement algorithm and the heterogeneous architecture can be adapted to the context of HEVC. The HDS algorithm is especially adapted to these two coding features.

5.2 High level parallelism tools

If advances in hardware architectures might overcome the constraints imposed by the new coding modes of HEVC, coping with UHD and higher frame-rate is still challenging. To this aim, HEVC is designed with implementation considerations in mind. The standard defines tools to allow computation parallelism at a high level. Slices, Tiles and Wavefront Parallel Processing allow data-dependency reduction for parallel processing at frame level. The architectures described above can thus be multiplied to increase the computation power and overcome real-time constraints.

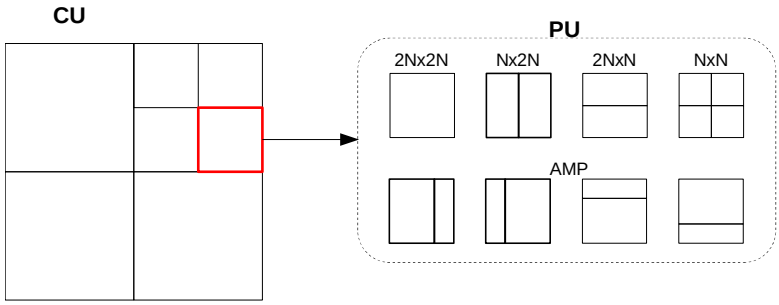


Figure 11: CTB, CU and PU partitioning

As in H.264/AVC it is possible to divide a frame into slices and tiles. Slices are groups of CTBs in scan order. Slice can be used both for network packetization and for parallel processing. However, a severe penalty on rate distortion performance is incurred when using slices, due to the breaking of all dependencies at their boundaries and to the slice header size, a set of parameters that has to be transmitted at the beginning of each slice. A frame can also be partitioned into a number of independent tiles, rectangular groups of CTBs. Tile boundaries are vertical and horizontal and extend across the whole picture. Tiles are processed in raster scan order, and the MTBs inside each tile are also processed in raster scan order. All dependencies are broken at tile boundaries, so there can be no pixel, motion vector or context prediction across them. The entropy coding engine is reset at the start of each tile. Only the deblocking filter is applied across tiles, in order to limit visual artifacts. Consequently, tiles can be encoded and decoded by independent cores working in parallel, and only the deblocking stage requires cross-tile communications. This comes at the expense of moderate rate-distortion loss.

With these two solution, memory constraints for each core are reduced because of the broken data-dependencies. Motion continuity is also broken at slices/tiles frontiers because prediction is lost. This can induce visual artifacts. Hierarchical motion estimators can use MV from lower resolution levels to improve motion coherency around the frontiers. Motion estimation for hierarchical level must then have access to the whole reference pictures and be processed with the lowest data dependency loss as possible.

Parallel encoding and decoding without dependency loss is proposed by Wavefront Parallel Processing (WPP). It consists of resetting the CABAC probabilities of the first CTB in each line (of a frame, slice or tile) with the probabilities obtained after processing the second CTB of the line above. All other interblock dependencies are maintained. Thus, parallel encoding and decoding is possible with little compression efficiency degradation. It is then possible to use N cores , with $1 \leq N \leq CTB\,rows$, each core computing 1 row out of N. This tool allows motion estimation with full dependencies, as in the sequential mode, but requires communicating motion vectors between cores for prediction.

6. CONCLUSION

A state of the art of motion estimation technique has been drawn. HME and EPZS techniques present good quality and reduced computational complexity. They have been prototyped on a 1Ghz TI C64x DSP and integrated in an H.264 video encoder. Their performances concerning both their execution time and result quality have been compared. EPZS can reach 30 frames per second for HD definition at quarter-pixel accuracy and up to 77 frames per second at pel accuracy. It is a good candidate for a low cost video encoder. HME needs more computational power, but is a good candidate for a high-end video encoder. Compression gain brought by quarter-pel accuracy is worth its computationally expensive implementation. HME regularity would make its implementation interesting onto highly parallel hardware implementations (FPGA, ASIC) whereas HDS appears to be more interesting for software implementations because of data-access pattern. HDS is a good compromise between motion estimation quality and computation complexity.

For variable block size motion estimation the scheme used for 8×8 blocks can be duplicated for each block size. The overhead due to the hierarchical levels processing of HDS is then reduced because it can be done only once for 8×8 block size and serve as hierarchical predictors for all the other block size searches. The complexity

of fractional-pixel accuracy motion estimation is studied and the limitations of DSP processing power for the sub-pixel motion vector refinement is highlighted. Then a scalable and flexible low-complexity VLSI architecture is designed for a sub-pixel refinement coprocessor. A flexible hybrid solution for FME is proposed based on a DSP for IME and an FME coprocessor on FPGA. Data dependencies have been studied to provide a SW/HW distribution and scheduling with parallel computing. The results presented in this paper shows that Motion Estimation for a H.264 coder using full features can be realized in real time for HD video. A 30 frames per second quarter-sample precision motion estimator for HD H.264 video encoding can be prototyped onto an heterogeneous DSP/FPGA platform.

The programmability of the DSP combined with the flexibility of the coprocessor design allow various implementation trade-offs as well as adaptability to existing and future standards. The presented solutions are adaptable and well suited for the next generation video encoding standard HEVC. H.264 SVC standard provides scalability features to manage, store and distribute video content towards multiple kinds of terminals and over different access technologies. A single SVC bitstream is used instead of one AVC bitstream per terminal, saving the global available bandwidth. A scalable extension of HEVC is under study in the JVT consortium. The hierarchical structure of the HDS is well adapted to scalable video compression algorithms.

REFERENCES

1. Iain E.G.Richardson, [*H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*], John Wiley and Sons (2003).
2. Urban, F., Poullaouec, R., Nezan, J. F., and Déforges, O., "Real-time multi-dsp motion estimator for mpeg-4 avc/h.264 high definition video," in [*International Conference on Signals and Electronic Systems*], (Sept. 2006).
3. Li W., S. E., "Successive elimination algorithm for motion estimation," *IEEE Transactions on Image Processing* **4**, 107–110 (1995).
4. Y-S Chen, Y-P Hung, and C-S Fuh, "Fast Block Matching Algorithm Based on the Winner-Update Strategy," in [*IEEE Transactions on Image Processing*], **10** (August 2001).
5. M.J. Chen, L.G. Chen, T.D. Chiueh, "One-dimensional full search motion estimation algorithm for video coding," *IEE Transactions on Circuits and Systems for Video Technology* **4**, 504–509 (1994).
6. Jain, J. R. and Jain, A. K., "Displacement measurement and its application in interframe coding," *IEEE Transactions on Communications* **COM-29(12)**, 1799–1808 (1981).
7. T.Koga, K.Linuma, A.Hirano, Y.Iijima and T.Ishiguro, "Motion compensated interframe coding for video conferencing," *Proc. of National Telecommunication Conference* **NTC81**, G5.3.1–G5.3.5 (1981).
8. Tham, J., Ranganath, S., Ranganath, M., and Kassim, A., "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Transactions on circuits and systemes for video technology* **8, NO. 4** (Aug 1998).
9. Hosur, P. and Ma, K., "Motion Vector Field Adaptive Fast Motion Estimation," *Second International Conference on Information, Communications and Signal Processing (ICICS '99)* (1999).
10. Alexis Michael Tourapis, "Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation," *proceedings of Visual Communications and Image Processing* , 1069–79 (2002).
11. Tourapis, A. M., Au, O. C., and Liou, M. L., "Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) ," in [*Proceedings of Visual Communications and Image Processing 2001 (VCIP'01)*], (2001).
12. Z. Chen, P. Zhou, and Y. He, "Fast motion estimation for JVT." JVT-G016.doc (March 2003).
13. K. Virk, N. Khan, S. Masud, F. Nasim, S. Idris, "Low Complexity Recursive Search Based Motion Estimation Algorithm for Video Coding Applications," in [*Proceedings of 13th European Signal Processing Conference*], (2005).
14. B. Chupeau, P. Robert, M. Pecot, P. Guillotel, "Multiscale motion estimation," *Workshop on Advanced Matching in Vision and Artificial Intelligence* **Munich** (5th, 6th June 1990).
15. Joint Video Team of ITU-T and ISO/IEC 14496-10, "Draft of version 4 of H.264/AVC," tech. rep. (Nov 2004).
16. Urban, F., Raulet, M., Nezan, J. F., and Déforges, O., "Automatic DSP cache memory management and fast prototyping for multiprocessor image applications," *14th EUSIPCO* (Sept 2006).

17. W. IL Choi, B. Jeon, J. Jeong, "Fast motion estimation with modified diamond search for variable motion block sizes," in [*International Conference on Image Processing*], **2**, 371–4 (Sept. 2003).
18. Mattavelli, M. and Zoia, G., "Vector-Tracing Algorithms for Motion Estimation in Large Search Windows," *IEEE Transactions on circuit and systems for video technology* **10**, 1426–1437 (December 2000).
19. Sullivan, G. and Wiegand, T., "Video Compression - From Concepts to the H.264/AVC Standard," *Proceedings of the IEEE* **93**, 18–31 (Jan 2055).
20. Chen, C.-Y., Chien, S.-Y., Huang, Y.-W., Chen, T.-C., Wang, T.-C., and Chen, L.-G., "Analysis and architecture design of variable block-size motion estimation for H.264/AVC," *IEEE Transactions on Circuits and Systems I* **53**, 578 – 593 (March 2006).
21. Chen, T.-C., Huang, Y.-W., and Chen, L.-G., "Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC," *ICASSP* **5**, 9–12 (2004).
22. Yang, C., Goto, S., and Ikenaga, T., "High performance vlsi architecture of fractional motion estimation in h.264 for hdtv," in [*IEEE International Symposium on Circuits and Systems*], 2605–2608 (May 2006).
23. Urban, F., Poullaouec, R., Nezan, J.-F., and Deforges, O., "H.264 fractional motion estimation refinement: a real-time and low complexity hardware solution for hd sequences," in [*15th EUSIPCO*], 836–840 (Sept 2007).
24. Vos, L. and Stegherr, M., "Parameterizable VLSI architectures for the full-search block-matching algorithm," *IEEE Transactions on Circuits and Systems* **36 issue 10**, 1309–1316 (1989).
25. Yeo, H. and Hu, Y. H., "A novel modular systolic array architecture for full-search blockmatching motion estimation," *ICASSP* **5**, 3303–3306 (May 1995).
26. JCTVC-I1003, "High Efficiency Video Coding (HEVC) text specification draft 6," tech. rep., San Jose (Feb 2012).
27. P.Bordes, G.Clare, F.Henry, M.Raulet, and J.Viéron, "An overview of the emerging HEVC standard," in [*International Symposium on Signal, Image, Video and Communications, ISIVC*], (July 2012).