A Fast Block Motion Estimation Algorithm Using Gray Code Kernels

Yair Moshe, Hagit Hel-Or
Dept. of Computer Science, University of Haifa
31905 Haifa, Israel

Abstract *– Motion estimation plays an important role in modern video coders. In such coders, motion is estimated using a block matching algorithm that estimates the amount of motion on a block-by-block basis. A full search technique for finding the best matching blocks delivers good accuracy but is usually not practical due to its high computational complexity. In this paper, a novel fast block-based motion estimation algorithm is proposed. This algorithm uses an efficient projection framework which bounds the distance between a template block and candidate blocks. Fast projection is performed with a family of highly efficient filter kernels – the Gray Code Kernels – using only 2 operations per pixel for each filter kernel. The projection framework is combined with a rejection scheme which allows rapid rejection of candidate blocks that are distant from the template block. Experiments show that the proposed algorithm significantly outperforms popular fast motion estimation algorithms, such as three-step search and diamond search. In addition, the tradeoff between computational complexity and quality of results could be easily controlled in the proposed algorithm, thus it enables adaptivity to image content.*

Keywords *– fast motion estimation, block matching, video compression, video coding, Gray Code Kernels (GCK).*

I. INTRODUCTION

Video compression has become an essential part of modern multimedia systems since it enables significant bit rate reduction of the video signal for transmission or storage. A video coder compacts a digital video sequence by decreasing video redundancies. Most effective is the removal of temporal redundancies which are significant due to the fact that the difference between consecutive frames of video is small. In recent years, several video compression standards have been proposed for various applications such as MPEG-1/2/4 [1][2], H.261/3/4 [3][4]. Although differing in many details, these standards maintain the same general hybrid DPCM/transform coding structure. In this structure, temporal redundancy is exploited using block-based motion estimation and compensation.

Block-based motion estimation and compensation compensates for movement of rectangular nonoverlapping regions called macroblocks. Two different types of macroblocks are defined – Intra and Inter. Motion estimation and compensation is performed on Inter macroblocks only. For each template Inter block in the current frame, the encoder performs the following procedure: Step 1: Search an area (search window) in the reference frame to find the 'best' matching block from amongst all candidate blocks in the search area. A popular matching criterion is the MSE (Mean Squared Error) between a candidate block and a template block, which measures the energy of the difference between the two blocks. Step 2: The chosen candidate block is subtracted from the template block to form a residual block. Step 3: The residual block is encoded and transmitted with a motion vector representing the displacement between the template block and chosen block. The decoder later uses the residual block and the motion vector to reconstruct the original template block [5].

A brute force technique for performing Step 1 is called *full search*. It is performed by comparing all candidate blocks in the search window with the template block. Full search motion estimation consumes 60%-80% of a typical video encoder's computation time. It delivers good accuracy in motion estimation but incurs high computation complexity and is not suitable for most real-time applications. Obtaining near-optimal block matching results is essential for achieving high coding efficiency. Thus a fast and accurate block matching algorithm is a critical part of every practical video coder with significant impact on coding efficiency.

Many suboptimal fast motion estimation algorithms have been proposed in the literature. Some of these algorithms reduce search complexity by limiting the size of the search window and the number of candidate blocks under the assumption that the matching error monotonically increases with the distance from the search position defined by the optimal motion vector. This assumption is not always valid and the process may converge to a local minimum on the error surface rather than to the global minimum as in the full search algorithm [11]. Examples for such algorithms are the three-step search [6], 2-D logarithmic search [7], cross search [8], diamond search [9], etc.

Another approach is to speed up the calculation of matching error for each candidate block. In [10], this approach is implemented by subsampling the pixels in the template and candidate blocks. This technique, as well, does not guarantee finding the optimal match with minimum matching error. The technique may be combined with a method to limit the number of search positions.

A different approach for fast motion estimation is to use some matching criteria to rule out search positions while ensuring the global minimum matching error is still attained. One such algorithm uses the block sum pyramid [11]. In this method, a sum pyramid structure is constructed for each block. Successive elimination is then performed hierarchically from the top level to the bottom level of the pyramid. Many search positions are determined as suboptimal and can be excluded from being further consideration in the motion vector search. Thus, search complexity is reduced. An improvement of this algorithm based on a winner-update strategy is presented in [12].

Orthogonal transforms have also been shown to be useful for matching. However, only very few algorithms using the fast Walsh-Hadamard Transform (WHT) for block motion estimation, have been proposed in the literature. Two examples of such algorithms are presented in [13] and [14].

The block motion estimation problem is a variant of the pattern matching problem that involves finding a particular pattern in an image. In [15][16] a novel pattern matching technique using Walsh-Hadamard (WH) projection kernels is presented. The suggested approach uses an efficient projection scheme which bounds the distance between a pattern and an image window using very few operations on average. The projection framework is combined with a rejection scheme which allows rapid rejection of image windows that are distant from the pattern. In [17][18] a family of efficient filter kernels – the Gray Code Kernels (GCK) – is introduced. Filtering an image with a sequence of Gray Code Kernels is highly efficient. Advantageously, this family includes the WH kernels amongst others. In this paper, a novel fast block motion estimation technique for video compression is presented based on the fast pattern matching techniques developed in [15][16][17][18], hence is denoted FME-GCK.

The paper is organized as follows: Fast pattern matching algorithms using WH projection kernels and GCK are first described in sections II and III, respectively. The proposed fast bock motion estimation algorithm is presented in section IV. Complexity analysis and results are given in section V. Finally, conclusions are drawn in section VI.

II. FAST PATTERN MATCHING USING WALSH-HADAMARD PROJECTION KERNELS

Finding a given pattern in an image can be performed naively by scanning the entire image and evaluating the similarity between the pattern and a local 2D window about each pixel. Assume a 2D $k \times k$ pattern, $\mathbf{p}(x,y)$, is to be matched within an image $\mathbf{I}(x,y)$ of size $n \times n$. For each pixel location (x,y) in the image, the Euclidean distance may be calculated:

$$d_E^2(\mathbf{I}_{x,y},\mathbf{p}) = \sum_{\{i,j\}=0}^{k-1} \left(\mathbf{I}(x+i,y+j)-\mathbf{p}(i,j)\right)^2 \tag{1}$$

where $\mathbf{I}_{x,y}$ denotes a local window of $\mathbf{I}$ at coordinates (x,y). In the context of motion estimation, this procedure is equivalent to full search block matching of a template block $\mathbf{p}(x,y)$ to a set of candidate blocks in a search window of size $n \times n$ with the MSE criterion.

Referring to the pattern $\mathbf{p}$ and window $\mathbf{w}$ as vectors in $\Re^{k^2}$, $\mathbf{d}=\mathbf{p}-\mathbf{w}$ is the difference vector between $\mathbf{p}$ and $\mathbf{w}$. The Euclidean distance can then be rewritten in vectorial form:

$$d_E(\mathbf{p},\mathbf{w}) = \|\mathbf{d}\| = \sqrt{\mathbf{d}^T\mathbf{d}} \tag{2}$$

Now assume that $\mathbf{p}$ and $\mathbf{w}$ are not given but only the values of their projection onto a vector $\mathbf{u}$ (Fig. 1). Let

$$\mathbf{b} = \mathbf{u}^T\mathbf{d} = \mathbf{u}^T\mathbf{p} - \mathbf{u}^T\mathbf{w} \tag{3}$$

be the projected distance value. Since the Euclidean distance is a norm, it follows from the Cauchy-Schwartz inequality that a lower bound on the actual Euclidean distance can be inferred from the projection values:

$$d_E^2(\mathbf{p},\mathbf{w}) \geq \mathbf{b}^2 / \|\mathbf{u}\|^2 \tag{4}$$

If a collection of projection vectors are given $\mathbf{u}_1 \ldots \mathbf{u}_m$ along with the corresponding projected distance values $\mathbf{b}_i = \mathbf{u}_i^T\mathbf{d}$, the lower bound on the distance can then be tightened:

$$d_E^2(\mathbf{p},\mathbf{w}) \geq \mathbf{b}^T(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{b} \tag{5}$$

where $\mathbf{U}=[\mathbf{u}_1 \ldots \mathbf{u}_m]$ and $\mathbf{b}=(\mathbf{b}_1 \ldots \mathbf{b}_m)^T$ so that $\mathbf{U}^T\mathbf{d}=\mathbf{b}$.

Note that, if the projection vectors are orthonormal, the lower bound reduces to $\mathbf{b}^T\mathbf{b}$. As the number of projection vectors increases, the lower bound on the distance $d_E(\mathbf{p},\mathbf{w})$ becomes tighter. In the extreme case when the rank of $\mathbf{U}$ equals k^2, the lower bound reaches the Euclidean distance.

An iterative scheme for calculating the lower bound is also possible; given an additional projection vector $\mathbf{u}_{m+1}$ and projection value $\mathbf{b}_{m+1}$, the previously computed lower bound can be updated without recalculating the inverse of the entire system $(\mathbf{U}^T\mathbf{U})^{-1}$ (see [15][16] for details).

Many calculations can be spared if the vectors are chosen according to the following two necessary requirements:

- The projection vectors should be highly probable of being parallel to the vector $\mathbf{d}=\mathbf{p}-\mathbf{w}$.
- Projections of image windows onto the projection vectors should be fast to compute.

One set of projection vectors shown in [15][16] to satisfy the above two requirements are the WH basis vectors. These vectors capture a large portion of the pattern-window distance with very few projections on average.

The elements of the WH (non-normalized) basis vectors are orthogonal and contain only binary values (±1). Thus, computation of the transform requires only integer additions and subtractions. The WHT of an image window of size $k \times k$ (with k a power of 2) is obtained by projecting the window onto k^2 WH basis vectors. In our case, it is required to project each $k \times k$ window of the $n \times n$ image onto the vectors. This results is a highly overcomplete image

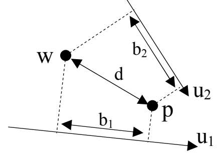

Fig. 1. Projection of $\mathbf{p}-\mathbf{w}$ onto projection vector $\mathbf{u}_i$ produces lower bound on distance $\|\mathbf{d}\| = \|\mathbf{p}-\mathbf{w}\|$

representation.

The projection vectors associated with the 2D WHT of order $k = 8$ are shown in Fig. 2. Each basis vector is of size 8×8, where white represents the value +1 and black represents the value -1. In Fig. 2, the basis vectors are displayed in order of increasing sequency (the number of sign changes along rows and columns of the basis vector). A 'snake' ordering of these vectors is shown by the overlaid arrow. This ordering is induced by the algorithm discussed in section IV and, although not exactly according to sequency, captures the increase in spatial frequency.

As discussed above, a second critical requirement of the projection vectors is the speed and efficiency of computation. An efficient method for calculating the projections of all image windows onto a sequence of WH vectors is discussed in section III.

III. THE GRAY CODE KERNELS

In [17][18] a family of filter kernels – the *Gray-Code Kernels (GCK)* – is introduced. Filtering an image with a sequence of GCK is highly efficient and requires only 2 operations per pixel for each filter kernel, independent of the size or dimension of the kernel. This family of kernels includes the WH kernels among others, thus it enables very efficient projection onto the WH basis vectors.

Consider first the 1D case where signal and kernels are one-dimensional vectors. Denote by $\mathbf{V}_s^{(k)}$ a set of 1D filter kernels expanded recursively from an initial seed vector $\mathbf{s}$ as follows:

$$\mathbf{V}_s^{(0)} = \mathbf{s}$$

$$\mathbf{V}_s^{(k)} = \left\{ \left[\mathbf{v}_s^{(k-1)} \quad \alpha_k \mathbf{v}_s^{(k-1)} \right] \right\} \quad s.t. \quad \mathbf{v}_s^{(k-1)} \in \mathbf{V}_s^{(k-1)}, \qquad (6)$$
$$\alpha_k \in \{+1, -1\}$$

where $\alpha_k \mathbf{v}$ indicates the multiplication of kernel $\mathbf{v}$ by the value α_k and [ ] denotes concatenation.

The set of kernels and the recursive definition can be visualized as a binary tree of depth k. An example is shown in Fig. 3 for $k = 3$. The nodes of the binary tree at level i represent the kernels of $\mathbf{V}_s^{(i)}$. The leaves of the tree represent the 8 kernels of $\mathbf{V}_s^{(3)}$. The branches are marked with the values of α used to create the kernels (where +/- indicates +1/-1).

Denote $|s| = t$ the length of s. It is easily shown that $\mathbf{V}_s^{(k)}$ is an orthogonal set of 2^k kernels of length $2^k t$. Furthermore, given an orthogonal set of seed vectors $\mathbf{s}_1, \ldots \mathbf{s}_n$, it can be shown that the union set $\mathbf{V}_{s_1}^{(k)} \cup \ldots \cup \mathbf{V}_{s_n}^{(k)}$ is orthogonal with $2^k n$ vectors of length $2^k t$. If $n = t$ the set forms a basis. Fig. 3 also demonstrates the fact that the values, $\alpha_1 \ldots \alpha_k$ along the tree branches, uniquely define a kernel in $\mathbf{V}_s^{(k)}$.

The sequence $\alpha = \alpha_1, \ldots \alpha_k \quad \alpha_i \in \{+1, -1\}$ that uniquely defines a kernel $\mathbf{v} \in \mathbf{V}_s^{(k)}$ is called the *α-index* of $\mathbf{v}$. Two kernels $\mathbf{v}_i, \mathbf{v}_k \in \mathbf{V}_s^{(k)}$ are defined to be *α-related* if and only if the hamming distance of their α-index is one. Without loss of generality, let the α-indices of two α-related kernels be $(\alpha_1 \ldots \alpha_{r-1}, -1, \ldots \alpha_k)$ and $(\alpha_1 \ldots \alpha_{r-1}, +1, \ldots \alpha_k)$. We denote the corresponding kernels as $\mathbf{v}_+$ and $\mathbf{v}_-$ respectively. Since $\alpha_1 \ldots \alpha_{r-1}$ uniquely define a kernel in $\mathbf{V}_s^{(r-1)}$, two α-related kernels always share the same prefix vector of length $2^{r-1} t$. The arrows of Fig. 3 indicate examples of α-related kernels in the binary tree of depth $k = 3$.

Of special interest are sequences of kernels that are consecutively α-related. An ordered set of kernels $\mathbf{v}_0 \ldots \mathbf{v}_n \in \mathbf{V}_s^{(k)}$ that are consecutively α-related form a sequence of GCK. The sequence is called a *Gray Code Sequence (GCS)*. The kernels at the leaves of the tree in Fig. 4 in a left to right scan, are consecutively α-related, and form a GCS. Note, however that this sequence is not unique and that there are many possible ways of reordering the kernels to form a GCS.

The main idea presented in [17][18] relies on the fact that two α-related kernels share a special relationship. Given two α-related kernels $\mathbf{v}_+, \mathbf{v}_- \in \mathbf{V}_s^{(k)}$ their sum $\mathbf{v}_p$ and their difference $\mathbf{v}_m$ are defined as follows:

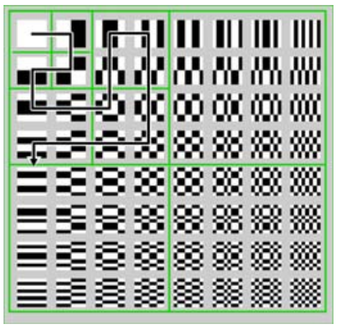

Fig. 2. The projection vectors of the WHT of order $n = 8$ ordered with increasing spatial frequency. White represents the value 1 and black represents the value -1. A 'snake' ordering of these basis vectors is shown by the overlaid arrow.

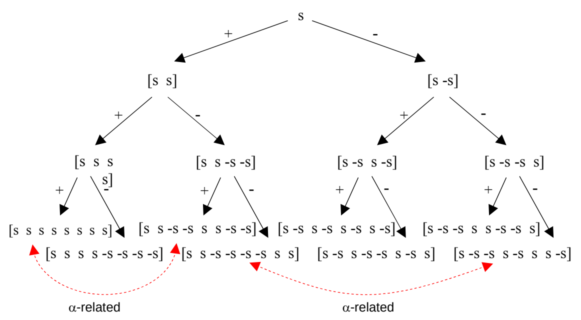

Fig. 3. The set of kernels and the recursive definition can be visualized as a binary tree. In this example the tree is of depth $k = 3$ and creates $2^3 = 8$ kernels of length 8. Arrows indicate pairs of kernels that are α-related.

$$\begin{aligned}\mathbf{v}_p &= \mathbf{v}_+ + \mathbf{v}_- \\ \mathbf{v}_m &= \mathbf{v}_+ - \mathbf{v}_-\end{aligned} \tag{7}$$

In [17][18] it is proven that the following relation holds:

$$[\mathbf{0}_\Delta \quad \mathbf{v}_p] = [\mathbf{v}_m \quad \mathbf{0}_\Delta] \tag{8}$$

where Δ is the length of the common prefix and $\mathbf{0}_\Delta$ denotes a vector with Δ zeros. For simplicity of explanation, we now expand $\mathbf{v} \in \mathbf{V}_s^{(k)}$ to an infinite sequence such that $\mathbf{v}(i) = 0$ for $i < 0$ and for $i > 2^k t$. Using this convention, the relation (8) can be rewritten in a new notation:

$$\mathbf{v}_p(i - \Delta) = \mathbf{v}_m(i) \tag{9}$$

and this gives rise to the following Corollary:

$$\begin{aligned}\mathbf{v}_+(i) &= +\mathbf{v}_+(i-\Delta) + \mathbf{v}_-(i) + \mathbf{v}_-(i-\Delta) \\ \mathbf{v}_-(i) &= -\mathbf{v}_-(i-\Delta) + \mathbf{v}_+(i) - \mathbf{v}_+(i-\Delta)\end{aligned} \tag{10}$$

Let $\mathbf{b}_+$ and $\mathbf{b}_-$ be the signals resulting from convolving a signal $\mathbf{x}$ with filter kernels $\mathbf{v}_+$ and $\mathbf{v}_-$ respectively:

$$\begin{aligned}\mathbf{b}_+(i) &= \sum_j \mathbf{x}(j)\mathbf{v}_+(i-j) \\ \mathbf{b}_-(i) &= \sum_j \mathbf{x}(j)\mathbf{v}_-(i-j)\end{aligned} \tag{11}$$

Then, by linearity of the convolution operation and corollary (10) we have the following:

$$\begin{aligned}\mathbf{b}_+(i) &= +\mathbf{b}_+(i-\Delta) + \mathbf{b}_-(i) + \mathbf{b}_-(i-\Delta) \\ \mathbf{b}_-(i) &= -\mathbf{b}_+(i-\Delta) + \mathbf{b}_+(i) - \mathbf{b}_+(i-\Delta)\end{aligned} \tag{12}$$

This forms the basis of an efficient scheme for convolving a signal with a set of GCKs. Given the result of convolving the signal with the filter kernel $\mathbf{v}_-$ ($\mathbf{v}_+$), convolving with the filter kernel $\mathbf{v}_+$ ($\mathbf{v}_-$) requires only 2 operations per pixel independent of the kernel size.

Considering Definition (6), and setting the prefix string to $\mathbf{s} = [1]$, we obtain that $\mathbf{V}_s^{(k)}$ is the WH basis set of order 2^k. A binary tree can be designed such that its leaves are the WH kernels ordered in dyadically increasing sequency and they form a GCS (i.e. are consecutively α-related). An example for $k = 2$ is shown in Fig. 4 where every two consecutive kernels are α-related. Thus, by ordering the WH kernels to form a GCS and given the result of filtering an image with the first WH kernel, filtering with the other kernels can be performed using only 2 operations per pixel per kernel

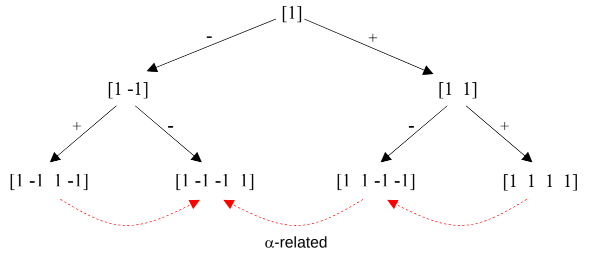

Fig. 4. Using initial vector s = [1] and depth $k = 2$ a binary tree creates the WH basis set of order 4. Consecutive kernels are α-related, as shown by the arrows.

easily generalized to two dimensions due to the separability of the WHT [17][18].

IV. FAST BLOCK MOTION ESTIMATION

The proposed fast block matching algorithm, FME-GCK, will now be described. Assume the video sequence is composed of images $\mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2, \ldots$ of size $n_1 \times n_2$, macroblocks are of size $k \times k$, and search windows of size $n \times n$. Also assume a set of m WH basis vectors $\{\mathbf{v}_i\}_{i=0}^{m-1}$ that form a GCS is given. Denote by $\mathbf{b}_i^{(j)}$ the projection values of macroblocks of image $\mathbf{I}_j$ onto WH basis vector v_i. Denote by $\mathbf{p}_{x,y}^{(j)}$ or $\mathbf{w}_{x,y}^{(j)}$ a macroblock of image $\mathbf{I}_j$ at coordinates (x, y), and denote by $sw(\mathbf{p})$ the search window around macroblock $\mathbf{p}$.

```
For each image $I_j$
1)  Project $\mathbf{I}_j$ onto $\{\mathbf{v}_i\}_{i=0}^{m-1}$ to obtain $\{\mathbf{b}_i^{(j)}\}_{i=0}^{m-1}$.
2)  For each Inter macroblock $\mathbf{p}_{x1,y1}^{(j)}$
    2.1) For each macroblock $\mathbf{w}_{x2,y2}^{(j-1)} \in sw(\mathbf{p}_{x1,y1}^{(j)})$
        2.1.1) Calculate lower bound on distance between $\mathbf{p}_{x1,y1}^{(j)}$ and $\mathbf{w}_{x2,y2}^{(j-1)}$ using $\{\mathbf{b}_i^{(j)}\}_{i=0}^{m-1}$ and $\{\mathbf{b}_i^{(j-1)}\}_{i=0}^{m-1}$.
    2.2) Calculate the MSE between $\mathbf{p}_{x1,y1}^{(j)}$ and the $q$ macroblocks $\mathbf{w}_{x2,y2}^{(j-1)}$ with the smallest distance lower bound from $\mathbf{p}_{x1,y1}^{(j)}$.
    2.3) From the $q$ macroblocks, select $\mathbf{w}_{x2,y2}^{(j-1)}$ with the smallest Euclidean distance from $\mathbf{p}_{x1,y1}^{(j)}$ as the best matched macroblock.
```

The FME-GCK algorithm will now be discussed in details. In order to perform efficient GCK calculations, the set of WH basis vectors must be consecutively α-related, thus forming a GCS. Finding an optimal GCS is shown to be NP-Complete [17][18]. The sequence of kernels that was used with the FME-GCK is shown as overlaid arrow in Fig. 2. This 'snake' order forms a GCS and captures the increase in spatial frequency.

Step 1 of the algorithm is performed using GCK with only 2 operations per pixel for each WH kernel. An exception for this efficient calculation is the first kernel (DC component) that can be calculated using 4 operations per pixel as described in [19].

Notice that the GCK approach can not be used efficiently for projecting macroblocks on the image boundary (specifically at the top and left image boundaries). This limitation, although minor, might increase algorithm complexity substantially. In an experiment for a CIF

(352x288) video sequence, boundary macroblock projections were performed by direct filtering with WH basis vectors and nonboundary macroblock projections were performed using the FME-GCK. Boundary projections were found to require about 55% of the calculation time spent on all projections. A solution to this problem is to zero-pad the upper and left boundaries of the image by $\Delta + k - 1$ rows and $\Delta + k - 1$ columns respectively. This, naturally, also increases the size of the projection images $\left\{\mathbf{b}_i^{(j)}\right\}_{i=0}^{m-1}$. The upper Δ rows and left Δ columns of these projection images $\left\{\mathbf{b}_i^{(j)}\right\}_{i=0}^{m-1}$ are filled with zeros. This is correct since projecting a zero macroblock onto any kernel results in zero. For all other image pixels starting from the $\Delta+1$ row and column, projections are performed using the efficient GCK method. The proposed technique for fast boundary calculation is depicted in Fig. 5.

Step 2.1.1 is based on the projection framework described in section II. Although the WH basis vectors are not orthornormal, they are orthogonal. Therefore, the term $(U^T U)^{-1}$ in equation (5) can be ignored.

V. Complexity Analysis and Results

FME-GCK uses two parameters that affect the tradeoff between complexity (time and memory) and accuracy of resulting motion vectors. These parameters are m, the number of projections to perform for each image, and q, the number of candidate macroblocks for which the true MSE value is calculated (Step 2.2 in algorithm). Larger m produces more accurate results at the cost of higher time and memory complexity. Memory complexity is affected since m projections of image $\mathbf{I}_j$ and m projections of image $\mathbf{I}_{j-1}$ must be stored in memory, thus, memory complexity is approximately $2(m+1)n_1 n_2$. Note that if $m = k^2$ the algorithm results are guaranteed to be identical to that of the full search. Larger q also produces more accurate results at the cost of higher time complexity; it does not however, affect memory.

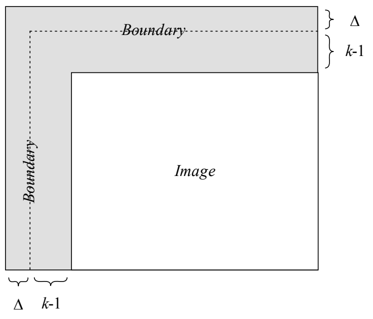

Fig. 5. Zero-padding each image with $\Delta+k-1$ rows and $\Delta+k-1$ columns enables rapid boundary calculation. The Δ upper rows and Δ left columns of each corresponding projection images are filled with zeros. GCK based computations start with the $\Delta+1$ row and $\Delta+1$ column.

of the three-step search algorithm, while reducing block matching operations to an average of 15.5 per block with $k = 16, n = 15$. Assuming 1 time unit for each operation of addition, subtraction, multiplication and minimum of two numbers, we obtain that performing a single MSE computation between two blocks requires $3k^2 - 1$ time units. Thus, performing a diamond search requires $15.5(3k^2 - 1)$ time units plus 14.5 time units (for calculating the minimum over all MSE values) per template macroblock. In the given configuration this sums to 11,903 time units per template macroblock.

Performing the FME-GCK algorithm involves m projections of each image. Time complexity of this step is 2 time units per pixel for every projection with the exception of the first projection that requires 4 time units per pixel to calculate – a total of $2k^2(m+1)$ time units per template macroblock. Calculating the lower bound for candidate macroblocks requires another $3mn^2$ time units per template macroblock. Finding the q candidate macroblocks with the lowest lower bounds, if performed naively, requires less than qn^2 time units. Calculating the MSE for these candidate macroblocks and selecting the one with the minimal MSE requires $q(3k^2-1)+(q-1)$ time units. Thus, a total of $2k^2(m+1)+3mn^2+qn^2+3k^2q-1+\varepsilon$ time units are required per template macroblock. The ε is added due to two additional overheads: 1) The aforementioned additional boundary calculation and 2) The fact that both Inter and Intra macroblocks must be projected, in contradiction to zero calculations for Intra macroblocks incurred by the diamond search algorithm. A reasonable value for ε is one that adds about 10% to the total algorithm time complexity. This estimation takes into account the fact that finding the best q candidates could be performed using a more efficient nontrivial procedure.

In order to compare performance of our suggested approach we choose parameters q and m such that run times of the FME-GCK equals that of the diamond search. From the run times computed above we find that such selections are $m = 5$, $q = 3$ and $m = 4$, $q = 4$. Note that it has been verified by real-time code profiling that both selections are comparable in time complexity to the diamond search.

Table 1 compares the results of full search, diamond search, and the FME-GCK algorithm. The selection of parameters for the FME-GCK is such that its time complexity equals the time complexity of diamond search. Results are given in average MSE values between template blocks and 'best' found candidate blocks. It can clearly be seen that FME-GCK significantly outperforms diamond search in both configurations for all sequences in Table 1 except Akiyo. For this sequence the difference between the results is very small. We conclude that for the same amount of calculations, FME-

GCK significantly outperforms the diamond search on average.

Table 1. Performance of FME-GCK algorithm compared to full search and diamond search. The selection of parameters for the FME-GCK is such that its time complexity equals the time complexity of diamond search ($m = 5, q = 3$ and $m = 4, q = 4$). Macroblocks are of size 8×8, search window is of size 15×15, and GOP size is 15. Results are given in average MSE between template blocks and 'best' candidate blocks.

Sequence	Res.	FS	Diamond	GCK53	GCK44
Akiyo	CIF	456	502	552	553
Silent	CIF	621	699	670	674
Coastguard	QCIF	6,472	12,751	8,572	10,089
Carphone	QCIF	8,307	13,954	10,400	11,675
Mobile	CIF	69,991	133,359	90,981	95,522

An important property of the FME-GCK algorithm is that the tradeoff between complexity and accuracy of the results is controlled by the parameters m and q. The number of projections to perform for each image, m, could adapt to the video sequence characteristics with one frame of delay. The number of candidate macroblocks to perform MSE with for every template block, q, enables even more flexiblity since it could be different for every macroblock. Fig. 6 depicts the effect of different values of parameter m on FME-GCK performance with a constant $q = 3$. As expected, increasing the number of projections produces results which are closer to the optimal. This figure shows, that FME-GCK outperforms the diamond search starting from $m = 4$.

VI. CONCLUSION

In this paper a novel fast block motion estimation algorithm called FME-GCK has been presented. FME-GCK uses an efficient projection framework which bounds the distance between a template block and candidate blocks using highly efficient filter kernels. Candidate blocks that are distant from the template block are quickly rejected. This algorithm significantly outperforms diamond search and three-step search. In addition, this algorithm is guaranteed to converge to the optimal (full search) results and enables adaptivity of the block matching process based on image content and complexity limitations.

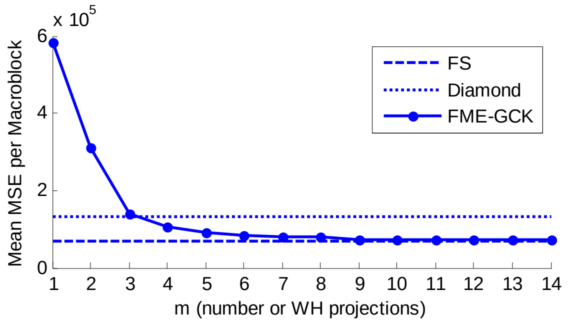

Fig. 6. FME-GCK algorithm performance for the sequence mobile CIF. Different values of m are shown while $q = 3$ is constant. Performance is measured in mean MSE between template macroblock and the 'best' found macroblock. Full search and diamond search results are shown as reference.

ACKNOWLEDGMENT

The authors would like to thank Prof. David Malah and Mr. Nimrod Peleg from the Signal and Image Processing Lab (SIPL), Dept. of EE., Technion - IIT, for their support and valuable advice.

REFERENCES

[1] ISO/IEC CD 13818-2 - ITU-T H.262 (MPEG-2 Video), "Information technology - Generic coding of moving pictures and associated audio information: Video, " 1995.

[2] ISO/IEC 14496-2 (MPEG-4 Video), "Information technology – Coding of audio visual objects," 1999.

[3] ITU-T Rec. H.263, "Video coding for low bit rate communication," 1998.

[4] ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, "Advanced video coding for generic audiovisual services," 2003.

[5] I. E. G. Richardson E. G. I., *H.264 and MPEG-4 Video Compression* - Video Coding for Next-generation Multimedia. England: Wiley, 2003.

[6] T. Koga, et al., "Motion-Compensated Interframe Coding for Video Conferencing," *Proc. NTC'81*, pp. G5.3.1-5, Dec.1981.

[7] J. R. Jain and A. K. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *IEEE Trans. Commun.*, COM-29, vol. 12, pp. 1799–1808, Dec. 1981.

[8] M. Ghanbari, "The Cross-Search Algorithm for Motion Estimation," *IEEE Trans. Commun.*, vol. 38, pp. 950–3, July 1990.

[9] J. Y. Tham, "A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion Estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8(4), pp. 369-377, Aug.1998.

[10] B. Liu and A. Zaccarin, "New Fast Algorithms for the Estimation of Block Motion Vectors," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 3(2), pp. 148–157, Apr. 1993.

[11] L. C. Hsing and C. L. Hwei, "A Fast Motion Estimation Algorithm Based on the Block Sum Pyramid," *IEEE Trans. Image Process.*, vol. 6 (11), pp. 1587-91, Nov. 1997.

[12] C. Y. Sheng, H. Y. Ping and F. C. Shann, "A Fast Block Matching Algorithm Based on the Winner-Update Strategy," *IEEE Trans. Image Process.*, vol. 10(8), pp. 1212-22, Aug. 2001.

[13] C. S. Young and C. S. Ik, "Hierarchical Motion Estimation in Hadamard Transform Domain," *Electronics Letters*, vol. 35(25), pp. 2187-8, Dec. 1999.

[14] M. Brünig and B. Mense, "A Fast Exhaustive Search Algorithm Using Orthogonal Transforms," *Proc. 7th Int. Workshop on Systems, Signals and Image Process. IWSSIP 2000*, pp.111-4, June 2000.

[15] Y. Hel-Or and H. Hel-Or, "Real-Time Pattern Matching Using Projection Kernels," *9th IEEE Int. Conf. Computer Vi*sion, pp. 1486-93, Oct. 2003.

[16] Y. Hel-Or and H. Hel-Or, "Real-Time Pattern Matching Using Projection Kernels," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1430-45, Sep. 2005.

[17] G. Ben-Artzi, H. Hel-Or and Y. Hel-Or, "Filtering with Gray Code Kernels," *Proc. of the 17th Int. Conf. Pattern Recognitio*n, vol. 1, pp. 556-9, Aug. 2004.

[18] G. Ben-Artzi, H. Hel-Or and Y. Hel-Or, "The Gray Code Filter Kernels," submitted to *IEEE Trans. Pattern Analysis and Machine Intelligence*.

[19] P. T. Simard, et al., "Boxlets: A Fast Convolution Algorithm for Neural Networks and Signal Processing," *Advances in Neural Information Processing Systems*, vol. 11, pp. 571-7, MIT Press, 1998.