

# A NEW ALGORITHM FOR FAST FULL SEARCH BLOCK MOTION ESTIMATION BASED ON NUMBER THEORETIC TRANSFORMS

TUUKKA TOIVONEN, JANNE HEIKKILÄ, OLLI SILVÉN

*Machine Vision Group*

*Infotech Oulu and Department of Electrical Engineering*

*P. O. Box 4500, FIN-90014 University of Oulu, Finland*

*{tuukkat,jth,olli}@ee.oulu.fi*

A new fast full search algorithm for block motion estimation is presented, which is based on convolution theorem and number theoretic transforms. It can be used with common video coding standards such as H.263 and MPEG. The algorithm applies the sum of squared differences (SSD) criterion, and the encoded video quality is equivalent or even better than what is achieved with conventional methods, but the algorithm has low theoretical complexity. The algorithm is implemented for H.263 software video encoder, and a great reduction in execution time is achieved.

## 1. Introduction

Most video coding standards, such as MPEG, use block motion estimation for removing temporal redundancy<sup>1</sup>. Each frame in a video sequence is divided into blocks, typically  $16 \times 16$  picture elements (pixels). A motion vector is estimated for each block by comparing the current block  $\mathbf{B}$  to candidate blocks  $\mathbf{C}$  in the search area  $\mathbf{S}$  in the previous frame. A typical motion vector range is  $\pm 16$  pixels, and thus the search area is  $48 \times 48$  pixels.

The exhaustive search algorithm (ESA) compares the current block to all of the candidate blocks, and selects the motion vector corresponding to the candidate block, which yields the best criterion function value. Typical criteria used are sum of absolute differences (SAD) or sum of squared differences (SSD). The latter can be expressed as follows:

$$\text{SSD}(c_y, c_x) = \sum_{y=0}^{B_h-1} \sum_{x=0}^{B_w-1} [\mathbf{B}(y, x) - \mathbf{S}(c_y + y, c_x + x)]^2 \quad (1)$$

where  $(c_y, c_x)$  is the candidate motion vector and the block size is  $B_h \times B_w$  pixels. However, in practical applications ESA is often unacceptably slow and faster methods are needed.

There are many fast algorithms for block motion estimation, for example the Three Step Search (TSS)<sup>2</sup>. Most of the algorithms, including TSS, reduce the computation by testing only the most promising motion vector candidates, and ignoring the rest. This degrades the estimation result, because these algorithms assume that the error criterion is unimodal, which is rarely true. Thus alternatives have been examined.

There is a class of fast full search algorithms, that test all candidate vectors without degrading the result. Most of these compute the SAD (or SSD) lower bound for the current vector, compare the best SAD found so far to the bound, and reject the candidate vector, if the lower bound is greater (worse). Examples of algorithms like this are Partial Distortion Elimination (PDE)<sup>2</sup>, Successive Elimination Algorithm (SEA)<sup>2</sup>, Multilevel SEA (MSEA, or BSPA)<sup>3</sup>, Winner-Update Strategy<sup>4</sup>, and Category-Based Block Motion Estimation Algorithm (CBME)<sup>5</sup>. However, the problem with these algorithms is the unpredictable amount of computation. If the video sequence is noisy, or there is a large amount of motion, these algorithms reject only small part of the candidate motion vectors and require more computation.

Another approach for fast full search motion estimation is to compute the correlation between the current block and the search area. The correlation can be used for evaluating the SSD criterion effortlessly, as follows:

$$\begin{aligned} \text{SSD}(c_y, c_x) = & -2 \underbrace{\sum_{y=0}^{B_h-1} \sum_{x=0}^{B_w-1} \mathbf{B}_t(y, x) \mathbf{S}(c_y + y, c_x + x)}_{\text{correlation } r(c_y, c_x)} \\ & + \underbrace{\sum_{y=0}^{B_h-1} \sum_{x=0}^{B_w-1} \mathbf{B}(y, x)^2}_{\|\mathbf{B}\|_2^2} + \underbrace{\sum_{y=0}^{B_h-1} \sum_{x=0}^{B_w-1} \mathbf{S}(c_y + y, c_x + x)^2}_{\|\mathbf{C}(c_y, c_x)\|_2^2} \end{aligned} \quad (2)$$

There exist fast algorithms for computing differentially the block  $L_2$ -norms<sup>6</sup>, and the correlation can be computed with a fast convolution algorithm. One approach is to consider the correlation as FIR filtering, and to decompose the filter<sup>6</sup>. However, this is useful only when a fast multiply-accumulator (MAC) unit is available.

A straightforward way to compute the correlation is to use the fast Fourier transform (FFT): the candidate block and the search area are transformed, the transformed blocks are multiplied together, and the result is inverse transformed<sup>7</sup>. However, Fourier transform requires complex domain, which is cumbersome with real domain data. Also the value ranges

are large, which suggests using floating point arithmetic. On the other hand, floating point arithmetic is not very suitable for application specific integrated circuits (ASICs) since it requires a large silicon area. The advantage of correlation-based motion estimation algorithms is that they have extremely regular data flow and deterministic execution time, unlike most other fast motion estimation algorithms, in particular any other fast full search type algorithm.

This paper presents a new algorithm for computing the correlation via number theoretic transforms (NTT) for motion estimation purposes. The search area is assumed to be  $48 \times 48$  pixels, and the current block  $16 \times 16$  pixels. Other sizes can be also used, but those are not investigated in the paper.

## 2. Number Theoretic Transforms

Number theoretic transforms<sup>8</sup> are similar to the Fourier transform, but instead of complex field, they are computed in a finite field or ring. A NTT is defined as

$$X_k \equiv \sum_{n=0}^{N-1} x_n \omega^{kn} \pmod{q}, \quad k = 0, 1, \dots, N-1 \quad (3)$$

where  $x_n$  is the input data sequence,  $X_k$  is the transformed sequence,  $\omega$  is the transform kernel,  $N$  is the transform length and  $q$  is the modulus. NTTs possess the cyclic convolution property and share many of the other Fourier transform properties. However, the arithmetic is exact since only integer numbers are needed.

The major difficulty in NTTs is the inflexible relationship between  $q$ ,  $N$  and  $\omega$  especially for long transform lengths. Guidelines for finding suitable number triples are given in literature<sup>8</sup>. Luckily, in block-based motion estimation only short lengths are needed. In practice, a NTT is computed with a fast algorithm. Most fast Fourier transforms, such as radix-2 or Winograd Fourier Transform Algorithm (WFTA) can be modified for computing NTTs. Since the pixel values are between 0 and 255, the correlation maximum value may be almost  $2^{24}$ . The modulus  $q$  should be at least this to allow obtaining the correct result.

The most common and simplest FFT algorithm is the radix-2 algorithm. It can be used for motion estimation with a  $48 \times 48$  pixel search area by dividing it into four overlapping  $32 \times 32$  pixel quarter blocks (Fig. 1). The current block is zero-padded to  $32 \times 32$  pixels, and all blocks are transformed. The transformed current block is flipped and multiplied elementwise with each of the transformed quarters. The four results are

inverse transformed, and four  $16 \times 16$  element correlations are obtained. These correlations are stitched together into a  $32 \times 32$  element correlation, which is then used for motion estimation within the search area.

With this method, the motion estimation needs five transforms and four inverse transforms. If open loop motion estimation is acceptable, four transforms can be saved by storing the transformed current blocks into a buffer. These stored blocks can be combined with additions and subtractions in transform domain into full transformed quarter blocks which are used when the motion vectors for the next frame are estimated.

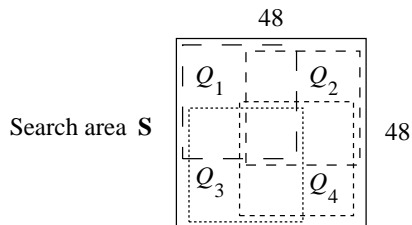


Figure 1. The four  $32 \times 32$  pixel quarters in the search area.

The Fermat number transform (FNT) is a special NTT which requires only multiplications by powers of 2. This is very efficient in an ASIC, since it can be implemented with simple bit shifts. The suitable FNT in our case is  $q = 2^{32} + 1$ ,  $\omega = 4$  and  $N = 32$ . However, the word length is unnecessary long 33 bits. Instead of the FNT, the modulus  $q = 2^{24} + 1$  was chosen, which allows presenting each constant multiplier as a linear combination of two terms, both a power of two. Each multiplication requires then only two bit shifts. The corresponding transform kernel is  $\omega = 2^{19} - 2^7$  (or  $2^{16} - 2^4$ ).

### 3. Simulation Results

For simulation purposes, the algorithm was implemented mostly in C and executed on the AMD Athlon 800 MHz processor. In the Table 1 the algorithmic operation counts (indexing is not accounted) and the time used are shown for the estimation of one motion vector.

Table 2 compares the NTT-based motion estimation algorithm to results reported in the literature for other fast full search algorithms. In the software implementation, the NTT algorithm do not appear to surpass other fast full search algorithms. However, it has other advantages, especially in an ASIC implementation.

Table 1. Simulation results.

Motion estimation algorithm	Additions	Multiplications	Time [ $\mu$ s]
ESA with SSD criterion	491071	246016	1407.5
Correlation via radix-2 NTT	45568	7328	234.5

Table 2. Fast full search algorithms.

Motion estimation algorithm	Savings
PDE	84–95 %
SEA	88 %
SEA with PDE	97 %
MSEA	95–98 %
Winner-Update Strategy	88–96 %
CBME	84 %
2-D FIR decomposition	77 %
Correlation via radix-2 NTT	83 %

#### 4. Conclusions

NTTs are a viable alternative for block motion estimation, especially when the minimum error and lowest bit rates are preferred. They seem to be best suited for real-time ASIC implementation due to congruent integer arithmetic, a very regular data flow, and deterministic execution time.

#### References

1. T. Toivonen, *Number Theoretic Transform -Based Block Motion Estimation*, Master's Thesis, University of Oulu, Finland (2002)
2. H. Wang and R. Mersereau, "Fast Algorithms for the Estimation of Motion Vectors," *IEEE Transactions on Image Processing* **8** (3), 435–438 (1999)
3. X. Q. Gao, C. J. Duanmu, and C. R. Zou, "Multilevel Successive Elimination Algorithm for Block Matching Motion Estimation," *IEEE Transactions on Image Processing* **9** (3), 501–504 (2000)
4. Y. Chen, Y. Hung, and C. Fuh, "Fast Block Matching Algorithm Based on the Winner-Update Strategy," *IEEE Transactions on Image Processing* **10** (8), 1212–1222 (2001)
5. H. A. Mahmoud and M. Bayoumi, "A Low Power Architecture for a New Efficient Block-Matching Motion Estimation Algorithm," *Proceedings of International Conference on Communication Technology* **2**, 1173–1179 (2000)
6. Y. Naito, T. Miyazaki, and I. Kuroda, "A Fast Full-Search Motion Estimation Method for Programmable Processors with a Multiply-Accumulator," *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 3221–3224 (1996)
7. C. Chen and J. F. Duluk, "System and Method for Cross Correlation with Application to Video Motion Vector Estimator," *U. S. Patent* **5,535,288**, (1996)
8. H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*, Springer-Verlag (1982)