Low Discrepancy Sequences Applied in Block Matching Motion Estimation Algorithms

Robson Lins*, Emerson Lima† and Silvio Melo*
* Center of Informatics, Federal University of Pernambuco, Recife, Brazil
Email: rcl@cin.ufpe.br, sbm@cin.ufpe.br
† University of Pernambuco
Email: eal@poli.br

***Abstract*—This work presents a fast method for motion estimation by using low discrepancy sequences. The basic idea in this proposal is one based on the reduction of the computational effort involved in the matching of a given block with the reference block. Instead of using a metric that employs all corresponding pixels in both blocks, it uses one that selects a subset of pixels for which the coordinates are determined by the so called low discrepancy sequences. The proposed method is applied to the New Three Step Search (NTSS), typically used as a technique for motion estimation in video applications. The results show a reduction in computational complexity of about 80% with regard to the NTSS, with little degradation in the recovered frame. The block matching of this present method can couple nicely with virtually any motion estimation technique, requiring very little change.**

***Keywords*-motion estimation; low-discrepancy sequences; video coding**

I. Introduction

Block matching Motion Estimation (ME) provides a simple and elegant way to identify and express movement and, because of that, is widely adopted by video coding standards such as MPEG-2, MPEG-4 and H.264/AVC [1]. In the motion estimation, a video frame is divided up into non-overlapping blocks of equal size and the best matching block in a reference frame is found within a predefined search window. In general, this is performed by minimizing a block distortion measure such as Mean Absolute Difference (MAD) between this pair of blocks. The most straightforward technique is the Full Search (FS), which exhaustively evaluates all possible candidate blocks within the search window. However, the computational complexity of FS is considered too high and can consume up to 80% of the total computation of the video encoding process [2]–[8].

Many computationally efficient search algorithms for ME have been proposed to reduce the computational effort and while maintaining high-coding efficiency. These methods employ different search point patterns to search for the best matching block. The Three-Step Search (TSS) [9], the New Three-Step Search (NTSS) [10], the Four-Step Search (4SS) [11] and the Block-Based Gradient Descent Search (BBGDS) [12] attempt to employ square-shaped pattern of different sizes to search for the best-matching block within the search window. The Unsymmetrical-cross Multi-Hexagon grid Search (UMHexagonS) [13] consists of four steps: unsymmetrical cross search, uneven multi-hexagon grid search, extended hexagon based search and a small diamond search. The UMHexagonS algorithm effectively reduces the number of candidate blocks within a searching window, and has been adopted in H.264/AVC JM reference software.

For any strategy of used search, part of the computational complexity of ME concerns the adopted block distortion measure. In this paper, we propose a method that reduce the computational effort of block distortion measure used in ME techniques. Instead of using a metric that employs all corresponding pixels in both blocks, it uses one that selects a subset of pixels for which the coordinates are determined by the so called *low discrepancy sequences*. The advantages of the proposed method are: (i) points coverage in the block can be adjusted, (ii) the low discrepancy sequence needs to be generated only once for a given block, (iii) can be applied to any block distortion measure, therefore to all ME techniques.

II. Motion Estimation

Given a block of size $M \times N$ and a search window of size $(2dm+1) \times (2dm+1)$, where dm is the maximum displacement, in pixels, for the vertical and horizontal coordinates of the block. The translation between these blocks is called a motion vector. To find the best match between the blocks one can use the MAD, defined by.

$$MAD(i,j) = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} |s_k(m,n) - s_{k-1}(m+i, n+j)|, \quad (1)$$

where $s_k(m,n)$ represents the gray level of the reference block, $s_{k-1}(m+i, n+j)$ is the gray level of the block in previous frame, (m,n) gives the coordinates of the upper left corner of the reference block in current frame and (i,j) is the offset in the search window.

The goal is to find a motion vector (u,v) associated to the minimum $MAD(i,j)$, where i and $j \in [-dm, dm]$. In FS, the equation (1) is calculated for all $(2dm+1)^2$ positions of candidate blocks in the search window, and the block with the lowest MAD (minimum distortion) is used for prediction, i.e., the position of this block within the search window corresponds to the motion vector. As was mentioned before different methods have been propodes to attack the

computational high demand of FS method. The following three traditional techniques for ME are described.

A. The Three-Step Search

The Three-Step Search (TSS) [10] reduces the number of candidate blocks within a searching window. In each step nine points are established, forming a 3 by 3 grid, as shown in Fig. 1 (filled circles). The motion vector is determined when the minimum MAD is found at a distance of one pixel. TSS became the most popular one for low bit-rate video application (including videophone and videoconferencing), owing this to its simplicity and effectiveness. However, TSS uses a uniformly allocated search pattern in its first step, which is not very efficient to catch small motions appearing in stationary or quasi-stationary blocks. To remedy this problem, the new three step search technique has been proposed.

B. The New Three-Step Search

The New Three-Step Search (NTSS) differs from TSS by (1) assuming a center-biased checking point pattern in its first step and (2) incorporating a halfway-stop technique for stationary or quasi-stationary blocks. The steps are given below:

1) In the first step, in addition to the original checking points used in TSS, eight extra points are added, which are the eight neighbors of the search window center, as shown in Fig. 1.
2) A halfway-stop technique is used for stationary and quasi-stationary block in order to fast identify and then estimate the motions for these blocks:
   a) If the minimum MAD in the first step occurs at the search window center, stop the search. (This is called the first-step-stop.)
   b) If the minimum MAD point in the first step is one of the eight neighbors of the window center, the search in the second step will be performed only for eight neighboring points of the minimum[1] and then stop the search. (This is called the second-step-stop.)

C. The Four-Step Search

For maximum motion displacements of ± 7, the Four-Step Search (4SS) algorithm utilizes a center-biased search pattern with nine checking points on a 5×5 window in the first step instead of a 9×9 window in the 3SS. The center of the search window is then shifted to the point with minimum MAD. The search window size of the next two steps depends on the location of the minimum MAD points. If the minimum MAD point is found at the center of the search window, the search will go to the final step (Step 4) with 3 x 3 search window. Otherwise, the search window size is maintained in 5×5 for step 2 or step 3. In the final step, the search window is reduced to 3×3 and the search stops at this small search window. The 4SS algorithm is summarized as follows [11]:

[1]Notice that some of these eight points have already been checked in the first step.

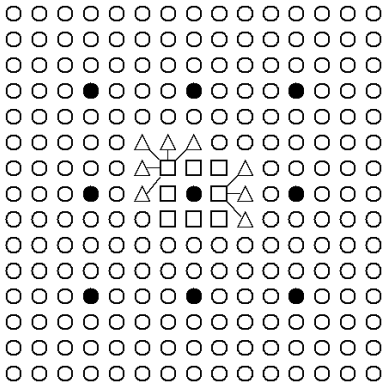

Fig. 1. Filled circles are the checking points in the first step of TSS, square are the 8 extra points added in the first step of NTSS, and triangles explain how the second step is performed if the minimum MAD in the first step is at one the 8 neighbors of the window center.

1) A minimum MAD point is found from a nine-checking points pattern on a 5×5 window located at the center of the 15×15 searching area as shown in Fig. 2(a). If the minimum MAD point is found at the center of the search window, go to Step 4; otherwise go to Step 2.
2) The search window size is maintained in 5×5. However, the search pattern will depend on the position of the previous minimum MAD point.
   a) If the previous minimum MAD point is located at the corner of the previous search window, five additional checking points are used, as shown in Fig. 2(b).
   b) If the previous minimum MAD point is located at the middle of a horizontal or vertical axis of the previous search window, then three additional checking points are used as shown in Fig. 2(c).
   If the minimum MAD point is found at the center of the search window, go to Step 4; otherwise go to Step 3.
3) The searching pattern strategy is the same as Step 2, but at the and it will go to Step 4.
4) The search window is reduced to 3×3 as shown in Fig. 2(d) and the direction of the overall motion vector is considered as the minimum MAD point among these nine searching points.

In the next section, the low discrepancy sequences are defined and the relationship with the block matching for ME is then exploited. This implies a faster evaluation of the MAD and therefore an acceleration of block matching motion estimation algorithms.

III. Low-Discrepancy Sequences

Monte Carlo methods form a class of algorithms that use pseudo-random numbers in tasks that usually involve numerical processing of a large amount of data [14]. These methods have been applied to a great variety of numerical problems such as in the iterated integral estimate, and usually present

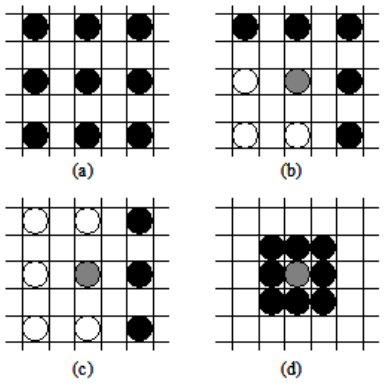

Fig. 2. Search patterns of the 4SS. (a) First step, (b) second/third step, (c) second/third step, and (d) fourth step.

great advantages over the traditional approaches of partition intervals such as Simpson's and the like.

Among the pseudo-random sequences, the so called low discrepancy sequences have been the object of major research, given its apparent superiority with respect to space uniform cover and the convergence performance in the integral estimates.

We call *discrepancy* the measure of uniformity of a given sequence[2]. More formally:

Definition 1 (Discrepancy)**.** *Let* $\omega = \{x_1, x_2, \cdots, x_n, x_{n+1}, \cdots\}$ *an infinite sequence of real numbers in the interval* $[0,1]$ *and let* $I \subseteq [0,1]$ *one sub-interval. We define* $A(I;n)$ *the amount of points of the subsequence* $x_1, x_2, \cdots, x_n$ *belonging to* I*, i.e,* $A(I;n) = |I \cap \{x_1, x_2, \cdots, x_n\}|$*. In a uniform sequence, the amount of points* $A(I;n)$ *is proportional to* I*'s measure, i.e,* $\frac{A([\alpha,\beta);n)}{n} = \beta - \alpha$ *considering that* $I = [\alpha, \beta)$*. The measure of the deviation*

$$D_n = D_n(\omega) = \sup_{0 \le \alpha < \beta \le 1} \left| \frac{A([\alpha,\beta);n)}{n} - (\beta - \alpha) \right|$$

is called Discrepancy *of the first* n *points of the sequence* ω*.*

The computation of D can be quite complicate. An associate measure of a somewhat simpler computation is the so called *star-discrepancy* expressed by:

$$D_n^* = D_n^*(\omega) = \sup_{0 < \beta \le 1} \left| \frac{A([0,\beta);n)}{n} - \beta \right|.$$

A well established result relates both measures through the inequality

$$D_n^* \le D_n \le 2D_n^*.$$

Considering sequences in the interval $[0,1]$, if the discrepancy of the sequence is zero, then the cover of the sequence is completely uniform and if it is near 1 then the sequence will be poorly uniform, leaving empty chunks in the interval. Some sequences may be regarded as a low discrepancy, i.e., near zero discrepancy, when considered as a whole, however they may present high discrepancy behavior during their early stages of construction. For instance, consider a total of 256 points for a given low discrepancy sequence, but it is possible that, if we take an intermediate phase of its construction, encompassing 64 points for example, this sequence may present a poorly uniform cover, which is a high discrepancy behavior. Other sequences possess the property of behaving like a low discrepancy sequence in all of its construction phases, such as Van der Corput-Halton's. In this paper, we will utilize Van der Corput-Halton's sequences [15]–[17], defined as follows:

Definition 2 (Van der Corput-Halton's Sequence (VDH))**.** *Let* $b > 1$ *a positive integer number. The sequence* $VDH_b = \{x_1, x_2, \cdots\}$ *for which the term* x_i *is defined by*

$$x_i = \sum_{j=0}^{s} \frac{a_j}{b^{j+1}},$$

where $\sum_{j=0}^{s} a_j b^j$ *is the expansion in the numerical base* b *of the number* $i-1$ *is called* Van der Corput-Halton's Sequence in the base b*.*

Note that the first points of the VDH sequence in base 2 are: $0, \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \cdots$. It can be shown that $lim_{n\to\infty} D_n(VDH_b) = 0$ and in fact it is not hard to show that $D_n(VDH_b) \le \frac{\ln(n+1)}{n}$, making the VDH sequence one of the lowest known discrepancy sequences.

A. Low-Discrepancy Sequences and Block-Matching

Let X and Y blocks of size $M \times N$ in the current frame and the search window of in the previous frame, respectively. Let $P = \{p_1, p_2, \cdots, p_k\}$ be the subset of $\{1, 2, \cdots, M\} \times \{1, 2, \cdots, N\}$, i.e, a subset of all possible coordinates of blocks X e Y. We define the mean absolute difference *induced* by P as:

$$MAD_P(X,Y) = \frac{1}{k} \sum_{l=1}^{k} |X(p_l) - Y(p_l)|. \qquad (2)$$

Notice that if $P = \{1, 2, \cdots, M\} \times \{1, 2, \cdots, N\}$ then Equation (2) corresponds to Equation (1) for fixed positions (m,n) and (i,j). Because summations and integrations are related operations, the use of Monte Carlo techniques for the evaluation of the summations, just like it is done with integrations, is suggested in this work. In particular, the use of low discrepancy sequences in the fast evaluation of the summations involved in the computation of the MAD, as will be seen briefly, presented very encouraging results.

In order to employ low discrepancy sequences making use of the concept of induced MAD, we need to transform the sequence VDH_b of real numbers in the interval $[0,1)$ into a sequence P of integers lying in the set[3] $\{1, 2, \cdots, m \times n\}$.

One approach consists of taking P as the ordering of $\{1, 2, \cdots, m \times n\}$ induced by the indexing of VDH_b, i.e.,

[2]The definitions and notation that follow are the same as in [15].

[3]in fact, the sequence P in the definition of induced metric is a subset of $\{1, 2, \cdots, m\} \times \{1, 2, \cdots, n\}$. By using the standard linearization of a matrix, it is enough to consider P as a subset of $\{1, 2, \cdots, m \times n\}$

$p_i = j$ if and only if x_i is the j-th element in the ordered list of VDH_b's elements. For instance, the sequence $A = \{0.8, 0.2, 0.7, 0.4\}$ induces, through this mechanism, the sequence $B = \{4, 1, 3, 2\}$, meaning that in the ordered listing of A its original elements hold respectively the fourth, first, third and second positions. Through this approach the repetitions are avoided but the convergence is preserved.

B. Low-Discrepancy Sequences Applied to the NTSS

Consider the sequence of k points of 2-dimensional VDH-low-discrepancy type in bases 2 and 3, that is, a sequence $\{P_l\}_{l=1\cdots k} = \{(a_l, b_l)\}_{l=1\cdots k}$ where $a_1, a_2, \cdots, a_k$ is the VDH sequence in the base 2 and $b_1, b_2, \cdots, b_k$ is the VDH sequence in the base 3. Normalizing the sequences $a_1, a_2, \cdots, a_k$ for the interval $[1, M]$ and $b_1, b_2, \cdots, b_k$ for the interval $[1, N]$ we obtain a scan of the whole block by a sequence denoted by P_l.

We emphasize that no matter which approach is used for the transformation, or how complex the sequence generation is, only one sequence needs to be generated for each block size. This sequence can then be used to produce any estimation of block distortion measure (MAD) as a function of the number of sampled points in the sequence. Therefore, the NTSS algorithm, Equation (1) can be replaced by Equation (2). This implies an acceleration the original NTSS, since only a percentage of the total points of the block is being used in the calculation of MAD. It should be clear that, whenever k varies from 1 to $m \times n$, in the scheme that avoids repetitions in the generation of P's points, MAD_P approaches and eventually equals the value of MAD. The proposed algorithm is shown in Fig. 3. The set V contains the 17 points as shown in Fig. 1. Note the use of Equation 2 as block distortion measure.

C. Computational Complexity

In the following, we compare the computation complexity of the TSS, NTSS, 4SS and our method (NTSS+VDH) concerning the maximum number of block matches. For a search window of size 15×15, the FS requires 225 block matches.

The TSS checks nine points in its first step, and then eight points in the two subsequent steps. Thus the TSS algorithm checks 25 points in the worst case.

The NTSS uses nine checking points of TSS plus eight center-biased points in its first step to favor blocks with small motion. In the worst case, the NTSS algorithm requires 33 block matches. Although NTSS uses more checking points in its first step as compared to TSS, the first-step-stop and second-step-stop can reduce computation significantly.

From the 4SS algorithm, we can find that the intermediate steps may be skipped and then jumped to the final step if at any time the minimum MAD point is located at the center of the search window. The worst case computational requirement of the 4SS is 27 block matches.

For a coverage of 50 points in the VDH sequence, the NTSS+VDH algorithm requires $33 \times 0.2 = 6.6$ block matches, since for each checking point only 20% of the pixels of the block were used to calculate the MAD. Thus, the worst case computational requirement of the NTSS+VDH is 80% smaller than the traditional NTSS. Note that this approach can also be applied to TSS or 4SS algorithms.

```
generate k points in the VDH sequence
dmin = maximum
vmin = (0,0)
find vmin in V with the lowest MAD_P
if (vmin==(0,0))
        return vmin
end
else if (abs(vmin)==1)
  (cx,cy) = vmin;
  //3 or 5 checking points
  //with center (cx,cy)
  if (vmin == (cx,cy))
        return vmin
  end
end
if (abs(vmin)==4)
  p = 2
  while (p >= 1)
        for i = -p to p, step p
          for j = -p to p, step de p
                d = MAD_P
                se d < dmin
                  dmin = d
                  w = (i,j)
                end
          end
        end
        vmin = vmin + w
        p = p - 1
  end
end
```

Fig. 3. The NTSS+VDH algorithm.

IV. Results and Discussion

The simulations were performed with the use of luminance components for the first 300 frames in the video sequences *Akiyo*, *Coastguard*, *Container*, *Foreman*, *Mobile* e *News*. The utilized frame size was 352×288 (CIF format) with an 8-bit-per-pixel quantization. It was established the maximum shift of ± 7 pixels in both horizontal and vertical directions for 16×16-size blocks.

For all six sequences, simulations were carried out between the algorithms FS, NTSS and our method (NTSS+VDH). In the FS and NTSS algorithms, for each point inside the search window, all the 256 pixels of a block are used in the MAD measure computation. In this proposed algorithm, for each block, in order to estimate the motion vectors, we used covers by 50, 75, 100 and 125 points in the VDH sequence in bases 2 and 3, normalized to the region $[1, 16] \times [1, 16]$, that is, the motion vectors were estimated by using respectively about 20%, 30%, 40% and 50% of the block's total amount of points, leading to reductions in computational effort of about 80%, 70%, 60% and 50% with respect to NTSS.

The Fig. 4 presents the MSE values between the original and the estimated frames for two covers (50 and 125) of points in the VDH sequence. Notice that the performance of the NTSS with low discrepancy sequence (NTSS+VDH) is

similar to the traditional NTSS, but with the computational effort severely reduced. In the case where 125 points are used, the result is the closest to the pure NTSS. The reduction in the amount of points in the VDH sequence naturaly means an inferior approximation, however the behavior along the video sequence is basically maintained. The Fig. 5, illustrate the method performance for video sequence *Foreman*.

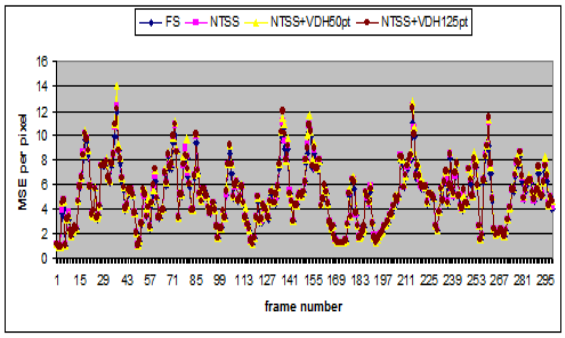

Fig. 4. MSE Value between the original and the estimated frame for the Akiyo video sequence's first 300 frames.

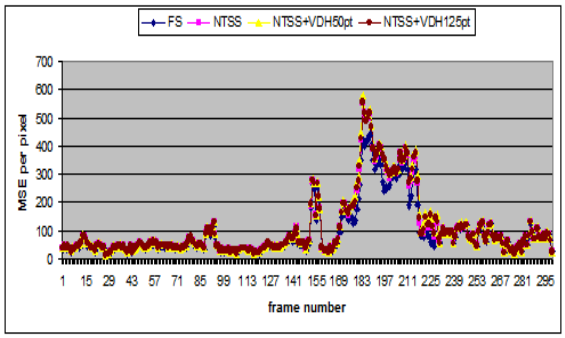

Fig. 5. MSE Value between the original and the estimated frame for the Foreman video sequence's first 300 frames.

Table I presents the MSE mean values between the original and estimated frames by the FS, NTSS and NTSS+VDH algorithms. Note, for a coverage of 125 points, the difference is very small. Fig. 6 shows the efficiency of the proposed method with respect to the visual quality of the recovered frame for the *Akiyo* video sequence. The Fig. 6(a) and Fig. 6(b) present the 300th recovered by NTSS and NTSS+VDH algorithms. It can be noticed the good quality in the motion compensation the NTSS+VDH when compared to the NTSS.

TABLE I
MSE MEAN VALUES FOR ALL SEQUENCES

Sequence	FS	NTSS	MSE mean for VDH			
			50	75	100	125
Akiyo	4.99	5.11	5.24	5.18	5.16	5.14
Coastguard	82.03	86.52	90.21	88.43	87.83	87.51
Container	13.28	13.52	13.65	13.58	13.56	13.54
Foreman	93.25	104.54	109.73	107.68	106.43	105.95
Mobile	301.43	305.48	318.50	312.26	310.01	308.58
News	29.24	31.98	33.96	33.14	32.72	32.52

(a) NTSS (b) NTSS+VDH

Fig. 6. The 300th estimated frames for the Akiyo sequence.

V. Conclusion

This paper presented a method that allows an acceleration in the techniques for block matching motion estimation using low-discrepancy sequences. To a certain block size, the sequence needs only be generated once and can be applied as a good estimate for the block distortion measure adopted, such as MAD, in motion estimation. The proposed method was applied to the NTSS algorithm, allowing a reduction of computational complexity by up to 80% with little degradation in the recovered frames.

Future directions in this research include the use of this technique in the choice of the points inside the search window, with or without a combination with the choice of the points inside each block. We also expect to test the efficiency of this technique with the use of other low discrepancy sequences and with the VDH sequence in other bases.

References

[1] T. Wiegand, G. J. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[2] C. Caia, H. Zengb, and S. K. Mitrac, "Fast motion estimation for H.264," *Signal Processing: Image Communication*, vol. 24, no. 8, pp. 630–636, 2009.

[3] C.-M. Kuo, Y.-H. Kuan, C.-H. Hsieh, and Y.-H. Lee, "A novel prediction-based directional asymmetric search algorithm for fast block-matching motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 6, pp. 893–899, 2009.

[4] S. Jin, S.-J. Park, and J. Jeong, "Adaptive fast full search algorithm using partitioned region and optimized search order," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 4, pp. 1703–1711, 2009.

[5] S. I. Shin, S. K. Lee, and J. S. Oha, "Fast partial difference elimination algorithm based on block matching error prediction," *Optical Engineering Letters*, vol. 46, no. 4, pp. 1–5, 2007.

[6] L. M. Po, K. H. Ng, K. W. Cheung, K. M. Wong, Y. M. S. Uddin, and C. W. Ting, "Novel directional gradient descent searches for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 8, pp. 1189–1195, 2009.

[7] C. W. Lam, L. M. Po, and C. H. Cheun, "A new cross-diamond search algorithm for fast block block matching motion estimation," *IEEE Int. Conf. Neural Networks and Signal Processing*, pp. 1262–1265, 2003.

[8] Y.-W. Chen, M.-H. Hsiao, H.-T. Chen, C.-Y. Liu, and S.-Y. Lee, "Content-aware fast motion estimation algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 256–269, 2008.

[9] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proceedings of NTC81*, Los Angeles, November 1981, pp. C9.6.1–9.6.5.

[10] R. Li, B. Zeng, and L. M. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, 1994.

[11] L. Po and W. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 313–317, June 1996.

[12] L. K. Liu and E. Peig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 8, pp. 419–423, 1996.

[13] Z. B. Chen, P. Zhou, and Y. He, *Fast integer pel and fractional pel motion estimation for JVT*, JVT-F017, 6th Meeting, Awaji, JP, pp. 5–12, 2002.

[14] O. Teytaud, "When does Quasi-random Work?" *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature: PPSN X*, pp. 325–336, 2008.

[15] L. Kuipers and H. Neiderreiter, *Uniform Distribution of Sequences*. New York: Addison Wesley Publishing Company, Inc., 1992.

[16] V. D. Corput, "Zahlentheoretische Abschaätzungen," *Math. Ann.*, pp. 53–79, 1921.

[17] J. H. Halton, "On the Efficiency of Certain Quasi-random Sequences of Points in Evaluating Multidimensional Integrals," *Numer. Math*, pp. 84–90, 1960.