

Estimação de Movimento com Uso de Sequências de Baixa Discrepância para Compressão de Vídeo

R.C. LINS¹, Centro de Informática, CIn, UFPE - Universidade Federal de Pernambuco, 50740-530, Recife, PE, Brasil.

E.A. LIMA², Escola Politécnica de Pernambuco, Universidade de Pernambuco, 50100-010, Recife, PE, Brasil.

S.B. MELO³, Centro de Informática, CIn, UFPE - Universidade Federal de Pernambuco, 50740-530, Recife, PE, Brasil.

Resumo. Em processamento de vídeo, um problema relevante é a estimação de movimento, visto que corresponde a aproximadamente 80% da complexidade computacional dos sistemas de codificação de vídeo. Parte desta complexidade diz respeito a medida de similaridade usada no casamento de blocos. Neste artigo, ao invés de usar uma medida que avalia todos os pixels dentro do bloco, usa-se uma que seleciona um subconjunto destes pixels induzida pelas sequências de baixa discrepância. O método proposto é aplicado ao algoritmo de busca em três passos (TSS) tradicionalmente usado como técnica de estimação de movimento em aplicações de vídeo a uma baixa taxa de transferência. Os resultados mostram uma redução da complexidade computacional do TSS em 90% com pouca degradação do quadro recuperado.

Palavras-chave. Estimação de movimento, compressão de vídeo, sequências de baixa discrepância.

1. Introdução

As técnicas de estimação de movimento, por *block-matching*, fornecem uma maneira eficiente para identificar e expressar o movimento e são comumente adotadas em muitos padrões de compressão de vídeo, como, por exemplo, MPEG-2, MPEG-4 e H.264/AVC [1]. A TV digital, a videoconferência e os sistemas multimídia são aplicações que fazem uso destas técnicas.

Na estimação de movimento, segmenta-se o quadro atual em um número fixo de blocos retangulares e considera-se que todos os pixels do bloco têm um modelo de movimento translacional. Os blocos do quadro atual são comparados com blocos do quadro anterior numa janela de busca específica, visando encontrar aqueles que

¹rcl@cin.ufpe.br

²eal@poli.br

³sbm@cin.ufpe.br

mais se assemelham com os blocos pesquisados. O desvio absoluto médio é normalmente utilizado como medida de similaridade, pois possui uma baixa complexidade quando comparada com outras medidas, tais como a soma de diferenças quadradas ou coeficiente de correlação cruzada [1].

O bloco com maior similaridade pode ser obtido pelo algoritmo de busca exaustiva (*Full Search* (FS)), no qual o bloco de referência é comparado com todos os blocos candidatos na janela de busca. Visto que o FS possui uma alta complexidade computacional, vários algoritmos rápidos têm sido desenvolvidos [3, 9]. Esses algoritmos podem ser classificados em dois grupos: a) algoritmos que otimizam o procedimento de busca, pela redução do número de blocos candidatos; b) algoritmos que reduzem a complexidade da medida de similaridade utilizada.

Tipicamente, a busca em três passos [2], a nova busca em três passos [10], a busca em quatro passos [12], a busca por diamante cruzado [8] e a busca por gradiente descendente são algoritmos que pertencem ao primeiro grupo. No que diz respeito ao segundo grupo, podem-se citar, a busca por eliminação sucessiva [11], a busca por distorção parcial [13] e a busca adaptativa [5], que reduzem o número total de operações realizadas no cálculo do desvio absoluto médio baseadas no comportamento típico dos movimentos em seqüências de vídeo.

Outras estratégias para obter a redução da complexidade computacional das técnicas para estimação de movimento, consistem em examinar, para cada bloco candidato, apenas parte dos pixels dentro do bloco ao custo de degradação da imagem recuperada o que conduz ao estudo de qual estratégia de amostragem de tal subconjunto de pixels é a mais adequada no sentido de promover uma cobertura do bloco como um todo, ser simples de ser gerada e fornecer resultados tão próximos quanto possível dos resultados obtidos examinando a totalidade dos pixels [6].

O uso de uma amostragem uniforme tem por principal desvantagem a quantidade excessiva, i.e., próxima da totalidade dos pixels, necessária a fornecer uma boa precisão dos vetores de movimento. Já a amostragem puramente aleatória pode, conforme explicado adiante, fornecer uma cobertura pouco densa da imagem. Uma maneira de combinar uma cobertura densa de um conjunto, por exemplo dos pixels de uma imagem bidimensional, com uma convergência rápida - demonstrada nos resultados deste trabalho - é utilizar uma seqüência de baixa discrepância que, essencialmente, é uma seqüência pseudo-aleatória classicamente utilizada como amostragem de pontos para estimação de valores de integrais múltiplas por métodos tipo Monte-Carlo [7]. Dentre as seqüências de baixa discrepância conhecidas, uma destaca-se por sua simplicidade de geração: A seqüência de Vander Corput Halton (VDH) [7].

Desta forma, neste artigo propõe-se uma técnica para estimar o movimento baseada em seqüências de baixa discrepância, que reduz a complexidade da medida de similaridade usada no casamento de blocos, selecionando um subconjunto pixels do bloco por meio da seqüência VDH a qual, conforme demonstrado nesta pesquisa, apresenta uma boa velocidade de convergência no cálculo da métrica utilizada na predição de movimento e precisão nos resultados quando comparada com métodos a busca exaustiva [7, 4].

2. Estimação de Movimento por Block-Matching

As técnicas para estimação de movimento por block-matching identificam os blocos de referência no quadro atual (k) com os blocos de melhor casamento no quadro anterior ($k - 1$) dentro de uma janela (ou área) de busca, como mostra a figura 1. A translação entre esses blocos especifica o vetor de movimento. O deslocamento máximo de um bloco dentro da janela de busca é expresso por dm . Pode-se usar como medida de similaridade, entre os pixels do bloco de referência e os pixels do bloco da janela de busca, o desvio médio absoluto expresso por

$$MAD(i, j) = \frac{1}{MN} \sum_{m=1}^M \sum_{n=1}^N |s_k(m, n) - s_{k-1}(m + i, n + j)|, \quad (2.1)$$

onde $-dm \leq i, j \leq dm$, $M \times N$ é o tamanho do bloco, e $s_k(m, n)$ é o valor do pixel na posição (m, n) . A fim de encontrar o bloco com menor MAD , vários procedimentos de busca foram e continuam sendo propostos [1]. A seguir, apresentam-se dois destes procedimentos.

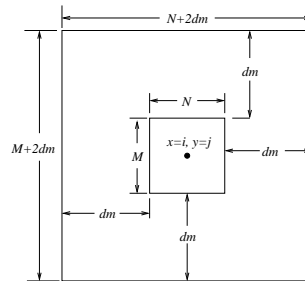


Figura 1: Geometria do bloco $M \times N$ e da janela de busca.

2.1. Procedimento de Busca Exaustiva

No algoritmo FS, um bloco de tamanho $M \times N$ do quadro atual (chamado bloco de referência) é comparado com todos os blocos dentro da janela de busca (chamados blocos candidatos) no quadro anterior. A equação (2.1) é calculada para todas as $(2dm + 1)^2$ posições de blocos candidatos dentro da janela de busca, e o bloco com menor MAD (distorção mínima) é usado para predição, ou seja, a posição desse bloco dentro da janela de busca corresponde ao vetor de movimento. Como o FS possui uma alta complexidade computacional, métodos rápidos para estimar o movimento são utilizados.

2.2. Procedimento de Busca Três Passos

O TSS (*Three Step Search*) [2] é um método simples e efetivo tradicionalmente usado para estimar o movimento. Em cada passo são estabelecidos 9 pontos, em que um deles é o central e os outros 8 são distribuições vizinhas a este central. O vetor de

movimento é encontrado quando o mínimo MAD for descoberto a uma distância de um pixel. A Figura 2 ilustra o algoritmo para $dm = 6$. Neste exemplo, o ponto $(i + 3, j + 3)$ é obtido como uma primeira aproximação do vetor de movimento, usando o critério de casamento em (2.1). No segundo passo, os oito pontos de pesquisa têm o espaçamento reduzido em torno da primeira aproximação, e o ponto $(i + 5, j + 3)$ é obtido. O terceiro passo fornece o vetor de movimento final, que é o ponto $(i + 6, j + 2)$.

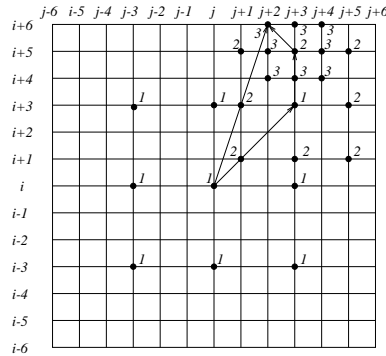


Figura 2: Um caminho de busca do TSS.

Embora o TSS tenha sido proposto para aplicações de vídeo a uma baixa taxa de transferência, em que a janela de busca é relativamente pequena ($dm = 6$ ou 7), o mesmo pode estendido para casos com $dm > 7$. Dessa forma, n passos são necessários para determinar o vetor de movimento. A figura 3 ilustra o algoritmo.

```

dmin ← valormax; vmin ← (0,0); k ← dm/2;
enquanto k >= 1
  para i ← -k ate k, passo de k
    para j ← -k ate k, passo de k
      d ← MAD(i, j);
      se d < dmin
        dmin ← d; w ← (i, j);
    fim
  fim
  fim
  vmin ← vmin + w; k ← k - 1;
fim

```

Figura 3: Algoritmo Busca em Três Passos.

3. Sequências de Baixa Discrepância

Métodos de Monte Carlo são uma classe de algoritmos que usam números pseudo-aleatórios em tarefas que envolvem o processamento numérico de, frequentemente,

uma grande quantidade de dados [14]. Esses métodos são aplicados em vários problemas numéricos como, por exemplo, na estimativa de integrais iteradas, bem como apresentam grandes vantagens sobre as abordagens clássicas de partição de intervalos tais como métodos de Simpson e semelhantes.

Dentre as sequências pseudo-aleatórias, as chamadas sequências de baixa discrepância tem sido alvo de muita pesquisa dada a sua aparente superioridade em critérios de cobertura uniforme de espaços e velocidade de convergência das estimativas integrais. Entende-se por *Discrepância* uma medida de uniformidade de uma dada sequência, formalmente [7]:

Definição 3.1 (Discrepância). *Seja $\omega = (x_1, x_2, \dots, x_n, x_{n+1}, \dots)$ sequência infinita de números reais no intervalo $[0, 1]$. Seja então $I \subseteq [0, 1]$ um sub-intervalo do intervalo $[0, 1]$. Seja $A(I; n)$ a quantidade de pontos da sub-sequência x_1, x_2, \dots, x_n pertencentes a I , i.e, $A(I; n) = |I \cap \{x_1, x_2, \dots, x_n\}|$. Em uma sequência uniforme, a quantidade de pontos $A(I; n)$ seria proporcional a medida de I , i.e, $\frac{A([\alpha, \beta]; n)}{n} = \beta - \alpha$ considerando $I = [\alpha, \beta]$. A medida do desvio*

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

é chamada de Discrepância dos primeiros n pontos da sequência ω .

O cálculo da medida D_n possui uma alta complexidade. Normalmente, utiliza-se uma medida associada mais simples chamada *Discrepância-estrela* expressa por

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

As medidas D_n e D_n^* estão relacionadas pela desigualdade $D_n^* \leq D_n \leq 2D_n^*$. Considerando sequências no intervalo $[0, 1]$, se a discrepância da sequência for zero, a cobertura da sequência é completamente uniforme e se for próxima de 1 a sequência será pouco uniforme deixando partes consideráveis do intervalo sem pontos amostrados. Algumas sequências podem ser de baixa discrepância, isto é, de discrepância próxima a zero, quando consideradas como um todo e ter discrepância alta durante etapas anteriores de construção. As sequências de Vander Corput-Halton (VDH), possuem a propriedade de terem baixa discrepância em todas as etapas de sua construção [7, 4].

Definição 3.2 (Sequência de Van der Corput-Halton). *Seja $b > 1$ número inteiro positivo. A sequência $VDH_b = (x_1, x_2, \dots)$ na qual cada termo x_i é expresso por $x_i = \sum_{j=0}^s \frac{a_j}{b^{j+1}}$, em que $\sum_{j=0}^s a_j b^j$ é a expansão na base b do número $i - 1$, é chamada de sequência de Van der Corput-Halton na base b .*

Note-se que, os primeiros pontos da sequência VDH na base 2 são: $0, \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \dots$. Pode-se demonstrar que $\lim_{n \rightarrow \infty} D_n(VDH_b) = 0$. De fato, pode-se mostrar que $D_n(VDH_b) \leq \frac{\ln(n+1)}{n}$. Sendo assim, dentre as sequências de baixa discrepância, a sequência VDH é uma das mais conhecidas.

3.1. Sequências de Baixa Discrepância e Block-Matching

Sejam X e Y blocos de tamanho $M \times N$ do quadro atual e da janela de busca do quadro anterior, respectivamente. Seja $P = \{p_1, p_2, \dots, p_k\}$ um subconjunto de $\{1, 2, \dots, M\} \times \{1, 2, \dots, N\}$, i.e, um subconjunto do conjunto possível de coordenadas dos blocos X e Y . Definimos o desvio absoluto médio *induzido* por P como

$$MAD_P(X, Y) = \frac{1}{k} \sum_{l=1}^k |X(p_l) - Y(p_l)|. \quad (3.1)$$

Note que se $P = \{1, 2, \dots, M\} \times \{1, 2, \dots, N\}$, então a equação (3.1) corresponde à equação (2.1) para posições (m, n) e (i, j) fixadas. Como, essencialmente, somatórios e integrações são operações correlatas, o uso de técnicas de Monte Carlo para avaliação dos somatórios tal como é feito no caso da integração é sugerido neste trabalho. Em particular, o uso de sequências de baixa discrepância na avaliação rápida da métrica MAD.

Para utilizar sequências de baixa discrepância no cálculo da métrica MAD, usando o conceito de métrica induzida, deve-se transformar a sequência VDH_b de números reais no intervalo $[0, 1)$ em uma sequência P de inteiros contida no conjunto⁴ $1, 2, \dots, M \times N$.

Uma abordagem consiste em tomar, como conjunto P a ordenação do conjunto $\{1, 2, \dots, M \times N\}$ induzida pela ordenação de VDH_b , i.e, $p_i = j$ se, e somente se, x_i é o j -ésimo elemento na lista ordenada dos elementos de VDH_b . Por exemplo, a sequência $A = (0.8, 0.2, 0.7, 0.4)$ induz, por este mecanismo, a sequência $B = (4, 1, 3, 2)$ significando que na listagem ordenada de A , a citar $(0.2, 0.4, 0.7, 0.8)$ os elementos originais de A ocupam, respectivamente, a quarta, primeira, terceira e segunda posições. Por esta abordagem, problemas de repetições de elementos são evitados e os resultados de convergência são mantidos.

3.2. Sequências de Baixa Discrepância no Algoritmo TSS

Considere k pontos na sequência VDH de dimensão 2 nas bases 2 e 3, isto é, a sequência $(P_l)_{l=1 \dots k} = ((a_l, b_l))_{l=1 \dots k}$ em que a_1, a_2, \dots, a_k é a sequência VDH na base 2 e b_1, b_2, \dots, b_k é a sequência VDH na base 3. Normalizando as sequências a_1, a_2, \dots, a_k para o intervalo $[1, M]$ e b_1, b_2, \dots, b_k para o intervalo $[1, N]$ obtemos uma varredura de um bloco num quadro por uma sequência denotada P_l . Ressaltamos o fato que, independentemente da abordagem de transformação e da complexidade na geração da sequência, para cada tamanho de bloco, apenas uma sequência precisa ser gerada e pode ser utilizada a partir daí para qualquer estimativa de MAD entre dois blocos como função do número de pontos amostrados na sequência. Assim, no algoritmo TSS (ver Figura 3), a equação (2.1) pode ser substituída pela equação (3.1). Isto implica numa rápida avaliação da métrica MAD, bem como aceleração do TSS original.

⁴de fato, a sequência P na definição de métrica induzida é subconjunto de $\{1, 2, \dots, M\} \times \{1, 2, \dots, N\}$. Utilizando a transformação padrão de listar linha a linha os elementos de uma matriz obtendo um vetor representativo da matriz, basta considerar P subconjunto de $1, 2, \dots, M \times N$

4. Resultados

As simulações foram feitas com o uso dos componentes de luminância para os primeiros 80 (oitenta) quadros das sequências *Claire*, *Football*, *Akiyo*, *Coastguard*, *Mobile* e *Container*. O tamanho utilizado dos quadros foi 352×288 quantizados a 8 bits por pixel. Estipulou-se o deslocamento máximo em ± 7 pixels tanto nas direções horizontais quanto verticais para blocos de tamanho 16×16 .

Diante destas seis sequências, realizaram-se simulações entre os algoritmos FS, TSS e TSS com sequências de baixa discrepância (TSS+VDH). Nos algoritmos FS e TSS, para cada posição dentro da área de busca, a totalidade dos 256 pontos de um bloco é utilizada no cálculo do MAD. No algoritmo proposto, dentro de cada bloco, para estimar os vetores de movimento foram usadas coberturas por 25, 50, 75 e 100 pontos na sequência VDH nas bases 2 e 3 normalizadas para a região $[1, 16] \times [1, 16]$, ou seja, os vetores de movimento foram estimados utilizando, respectivamente, em torno de 10%, 20%, 30% e 40% da totalidade dos pixels do bloco implicando em reduções da complexidade computacional do TSS em aproximadamente 90%, 80%, 70% e 60%. Os tempos médios típicos de CPU em cada simulação - sem contar o tempo de geração da sequência VDH - estão resumidos na tabela 1 para a sequência *Claire* (resultados análogos foram obtidos para as demais sequências) em um computador com processador *Intel I7* 64 bits com 2.8 GHz e 4 Gb de memória RAM, e mostram reduções compatíveis com os percentuais de redução da complexidade computacional obtidas pelo uso da sequência VDH. O tempo para geração da sequência é de aproximadamente 1 (um segundo) mas ressalta-se que a mesma somente precisa ser gerada uma única vez.

Tabela 1: Tempo de CPU gasto pelos diferentes algoritmos na estimação dos vetores de movimento da sequência *Claire*.

Algoritmo	Tempo de CPU (segundos)
Full Search	25.1
TSS	1.042
TSS + VDH (25 pontos)	0.156
TSS + VDH (50 pontos)	0.328
TSS + VDH (75 pontos)	0.452
TSS + VDH (100 pontos)	0.641

A figura 4 apresenta os valores de MSE entre o quadro original e o quadro estimado para todas as coberturas de pontos na sequência VDH. Observa-se que o desempenho do TSS+VDH é semelhante ao TSS tradicional, mas com a complexidade computacional reduzida. No caso em que foi utilizada uma cobertura de 100 pontos na sequência VDH, o resultados obtido pelo método é o que mais se aproxima do TSS. A redução da cobertura na sequência VDH implica numa menor aproximação, porém o comportamento ao longo do vídeo se mantém.

A tabela 2 mostra os valores médios em termos de MSE entre os quadros original e estimado pelos algoritmos FS, TSS e TSS+VDH. Note-se, para uma cobertura de

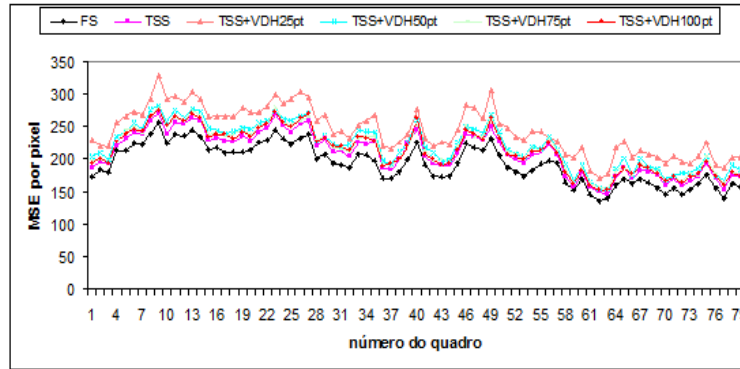


Figura 4: Valor de MSE entre o quadro original e o estimado para 80 quadros da sequência Football.

100 pontos na sequência VDH, a boa precisão do método proposto. A figura 5 mostra a eficiência do TSS+VDH, no que diz respeito à qualidade subjetiva do quadro reconstruído, para a sequência de vídeo *Claire*. As figuras 5(a) e 5(b) apresentam os quadros recuperados com o maior e o menor valor de MSE. A boa qualidade na compensação de movimento pode ser observada em cada caso.

Tabela 2: Médias de MSE para todas as sequências

Sequências	FS	TSS	MSE para sequências VDH			
			25 pontos	50 pontos	75 pontos	100 pontos
Claire	3.79	3.93	4.49	4.05	4.01	3.98
Football	192.56	209.36	243.50	221.44	216.64	214.42
Akiyo	5.27	5.60	6.43	5.76	5.69	5.68
Coastguard	102.21	119.10	142.81	126.09	121.66	121.14
Mobile	337.56	357.02	492.46	393.65	375.73	369.83
Container	12.37	12.38	13.08	12.47	12.41	12.40

5. Conclusões

Este artigo descreveu um novo método para estimação de movimento em vídeos digitais via sequências de baixa discrepância. Os resultados indicam que o método proposto neste trabalho promove uma significativa redução da quantidade de operações necessárias para estimar os vetores de movimento. Os vetores encontrados, mesmo com o uso de 10% do total de operações do TSS, são bem próximos dos vetores reais de movimento implicando em alta qualidade das imagens reconstruídas.

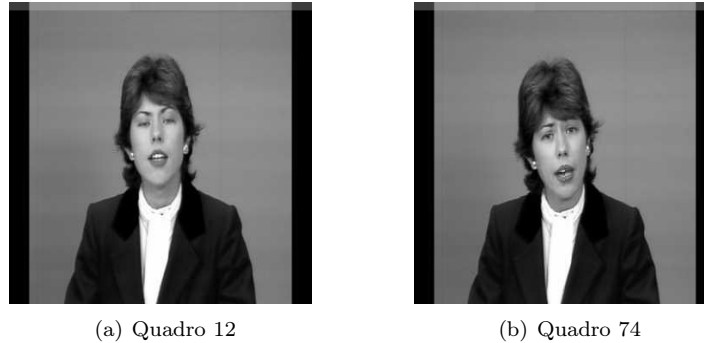


Figura 5: Quadros estimados da sequência Claire.

O tempo de CPU relativos a execução do algoritmo proposto são compatíveis com a redução da complexidade fornecida pelo menor número de operações realizadas.

Objetivos futuros de pesquisa incluem a utilização da técnica aqui proposta na escolha dos pontos dentro da área de busca em combinação ou não com a escolha dos pontos em cada bloco (conforme feito neste trabalho). Também, espera-se testar a eficiência desta técnica com uso de outras sequências de baixa discrepância e com a sequência VDH em outras bases.

Abstract. In video processing, a significant problem is the motion estimation, since it corresponds to approximately 80% of the computational effort in video coding systems. In part this complexity is due to the measure of similarity used in the block matching. In this article, instead of using a measure that evaluates every pixel within a block, it uses one that selects a subset of these pixels induced by a sequence of low discrepancy. The proposed method is applied to the Three Step Search (TSS), typically used as a technique for motion estimation in video applications video at a low transfer rate. The results show a reduction in computational complexity of about 90% with regard to the TSS, with little degradation in the recovered frame.

Referências

- [1] C. Caia, H. Zengb, S.K. Mitrac, Fast motion estimation for H.264, *Signal Processing: Image Communication*, **24**, No. 8 (2009), 630–636.
- [2] T. Koga et al., Motion-compensated interframe coding for video conferencing, In *Proceedings of NTC81*, p. C9.6.1–9.6.5, Los Angeles, November 1981.
- [3] Y-W. Chen et al., Content-aware fast motion estimation algorithm, *Journal of Visual Communication and Image Representation*, **19**, No. 4 (2008), 256–269.
- [4] J.H. Halton, On the efficiency of certain quasi-random sequences of points in evaluating multidimensional integrals, *Numer. Math.*, (1960) 84–90.

- [5] S. Jin, S-J. Park, J. Jeong, Adaptive fast full search algorithm using partitioned region and optimized search order, *IEEE Transactions on Consumer Electronics*, **53**, No. 4 (2009), 1703–1711.
- [6] X. Jing, L.P. Chau, Partial distortion search algorithm using predictive search area for fast full-search motion estimation, *IEEE Signal Processing Letters*, **14**, No. 11 (2007), 840–843.
- [7] L. Kuipers, H. Neiderreiter, “Uniform Distribution of Sequences”, Addison Wesley Publishing Company, Inc., New York, 1992.
- [8] C. W. Lam et al., A new cross-diamond search algorithm for fast block block matching motion estimation, *IEEE Int. Conf. Neural Networks and Signal Processing*, p. 1262–1265, 2003.
- [9] S. Lee, Fast motion estimation based on adaptive search range adjustment and matching error prediction, *IEEE Transactions on Consumer Electronics*, **55**, No. 2 (2009), 805–811.
- [10] R. Li, B. Zeng, L.M. Liou, A new three-step search algorithm for block motion estimation, *IEEE Transactions on Circuits Systems for Video Technology*, **4**, No. 4 (1994), 438–442.
- [11] W. Li, E. Salari, Successive elimination algorithm for motion estimation, *IEEE Transactions on Image Processing*, **4**, No. 1 (1995), 105–107.
- [12] L. Po, W. Ma, A novel four-step search algorithm for fast block motion estimation, *IEEE Transactions on Circuits and Systems for Video Technology*, **6** (1996), 313–317.
- [13] L.M. Po, C.K. Cheung, Normalized partial distortion search algorithm for block motion estimation, *IEEE Transactions on Circuits and Systems for Video Technology*, **10**, No. 3 (2000), 417–422.
- [14] O. Teytaud, When does quasi-random work? em “Proceedings of the 10th International Conference on Parallel Problem Solving from Nature: PPSN X”, p. 325–336, 2008.