MOTION VECTOR DETECTION ALGORITHM USING THE STEEPEST DESCENT METHOD EFFECTIVE FOR AVOIDING LOCAL SOLUTIONS

Yoshinori Konno † *and Tadashi Kasezawa* ††

† Hitachi Information & Control Solutions, Ltd.
Ibaraki, Japan
†† College of Engineering, Nihon University
Fukushima, Japan
E-mail: kasezawa@cs.ce.nihon-u.ac.jp

ABSTRACT

This paper presents a new algorithm that includes a mechanism to avoid local solutions in a motion vector detection method that uses the steepest descent method. Two different implementations of the algorithm are demonstrated using two major search methods for tree structures, depth first search and breadth first search. Furthermore, it is shown that by avoiding local solutions, both of these implementations are able to obtain smaller prediction errors compared to conventional motion vector detection methods using the steepest descent method, and are able to perform motion vector detection within an arbitrary upper limit on the number of computations. The effects that differences in the search order have on the effectiveness of avoiding local solutions are also presented.

Keywords: motion vector detection, block matching, steepest descent method

1. INTRODUCTION

In many video encoding schemes, a motion compensation prediction scheme [1], [2] is employed, and motion vectors need to be detected during the encoding process. A variety of motion vector detection methods have been proposed so far, which can be divided into two classes : (1) full search (FS) and (2) methods that aim to reduce the number of computations (such as three-step-search [3] and gradient descent search [4]).

Although FS is able to achieve smaller prediction errors, it requires a large number of computations as it involves an exhaustive search throughout the search domain. While the second class of methods do not perform an exhaustive search, they are susceptible to falling into a locally optimal solution. A variety of methods [5], [6] have been proposed recently for avoiding the local solution problem. These methods are generally equipped with a mechanism for setting the initial search point in the vicinity of the optimal solution in order to avoid local solutions.

In recent years, there has been a rapid increase in devices that have functions for sending or recording video, and the range of features demanded of motion vector detection is becoming more and more diverse. One of the most sought after features is the reduction in prediction errors for a given fixed number of computations. This requirement is based on the following background.

In general, when motion vector detection is implemented in devices that operate in real time, the desired motion vector needs to be found within some constant time, regardless of whether the method is implemented at a hardware or software level. In other words, the desired motion vector needs to be found within a upper limit on the number of computations. This upper limit varies depending on the method used for motion vector search and the required power consumption of the target device. Therefore, methods that help reduce the prediction error as much as possible within an arbitrary upper limit on the number of computations are in demand.

If we examine the above-mentioned classes in terms of this demand, the first class uses a constant number of computations and do not normally anticipate change in the number of computations (i.e., the process being cutoff midway). Similarly, in the second class, the number of computations required generally varies depending on the image being processed, and these methods are not intended to execute the process within a fixed number of computations.

For this reason, the authors have proposed a new algorithm (hereinafter ALM) that includes a mechanism to avoid local minima in a motion vector search method that uses the steepest descent method. ALM is able to avoid local solutions without an additional procedure for setting the initial search point in the vicinity of the optimal solution. Furthermore, within an arbitrary upper limit on the number of computations, ALM is able to perform motion vector detection, and is able to obtain prediction results close to FS if the number of computations is increased.

Because the search path in ALM forms a tree structure, it can be implemented using a number of different search orders. In this paper, two different implementations of ALM are demonstrated using two major search methods for tree structures, depth first search and breadth first search. Furthermore, it is shown that by avoiding local solutions, both of these implementations of ALM are able to obtain smaller prediction errors compared to conventional motion search methods using the steepest descent method, and are able to perform motion vector search within an arbitrary upper limit on the number of computations. The effects that differences in the search order have on the effectiveness of avoiding local solutions are also presented.

2. ALGORITHM

2.1 Motion Detection Algorithm Using the Steepest Descent Method

The steepest descent method [7] is a method for searching for the minimum value of a function by iteratively performing the process of "setting the next search point to the point formed by advancing by some distance in the direction where the gradient of the function decreases the most from the current search point."

If we let $e(u, v)$ (hereinafter matching error) be the sum of the absolute errors between a template block in a template image (the search source) and a reference block with a motion vector (u, v) in a reference image (the search target), the motion vector can be found using the steepest descent method as follows.

Step 1 Let the current motion vector $(u, v) \leftarrow (0, 0)$ and find $e(u, v)$.

Step 2 Find the unknown $e(u+s, v+t)$, $-1 \le s \le 1, -1 \le t \le 1$ for the current motion vector (u, v).

Step 3 Let (x, y) be the value of (s, t) that gives the minimum value of $e(u+s, v+t)$ when $-1 \le s \le 1, -1 \le t \le 1$ and $(s, t) \ne (0, 0)$. If $e(u, v) \ge e(u+x, v+y)$, let the current motion vector $(u, v) \leftarrow (u + x, v + y)$ and return to Step 2. Otherwise, let the motion vector giving the minimum matching error $(\hat{u}, \hat{v}) \leftarrow (u, v)$ and end the search.

2.2 Motion Detection Algorithm for Avoiding Local Solutions Using the Steepest Descent Method

2.2.1 Overview

The steepest descent method is a method that searches the "valley floor" by advancing along the direction with the largest decrease in gradient, and is known to be generally susceptible to falling into local solutions. The proposed method therefore begins searching the "valley floor" in multiple directions where the gradient is downwards. Once the "valley floor" has been reached, the search then continues along multiple "valleys", searching for even lower "valley floors".

- The search is performed from the search starting point in D directions where the reduction in gradient is large, i.e., the gradient is small. For these D directions, the gradient is allowed to increase as long as it is within the smaller D directions.

- The search for minimal points in D directions is performed by proceeding along the direction with the minimum value of gradient, the same as the method of steepest descent. If the direction of advancement has a minimum value of gradient from among all the directions, a direction where the gradient increases is also allowable. However, during the process of searching for the minimum point in the corresponding direction, increases in the gradient shall be permitted up to C times, and if a new minimum point that updates the minimum value is not found by then, search in that direction is terminated.

- Once a minimum point is found that updates the minimum value, the search for further minimum points is performed from that point in D directions where the gradient is low, the same as for the search starting point.

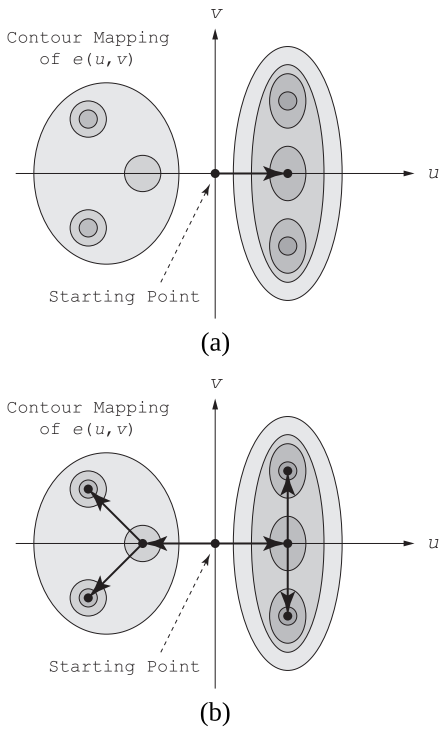
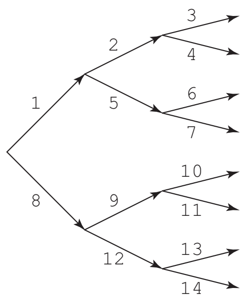


Fig. 1: Outlines of search paths. (a) SDM. (b) ALM.

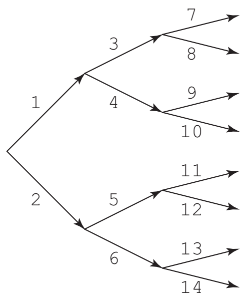
Figure 1 shows outlines of the search paths in the method shown in 2.1 (hereinafter SDM) and in ALM. The diagram uses contour lines to display an outline of $e(u, v)$, and the arrows in the diagram show the search path. As shown in the diagram, the search path of ALM is a tree structure. ALM can therefore be implemented using different search orders by using depth first search or breadth first search, etc., as shown in 2.2.3 and 2.2.2. Figure 2 shows an overview of the search order in depth first search and breadth first search.

2.2.2 Algorithm Using Breadth First Search

ALM using breadth first search (hereinafter ALMB) is composed of the three procedures : top_B, $target\_B(u, v)$, and $search\_next\_target(u, v, s, t)$ as shown below.



(a)



(b)

Fig. 2: Overview of search order. (a) Depth first search. (b) Breadth first search.

top_B

Step 1 Let the current motion vector $(u,v) \leftarrow (0,0)$, find $e(u,v)$, and let the flag that represents belonging to a motion vector search path $p(u,v) \leftarrow \mathrm{TRUE}$. Let the motion vector that gives the minimum matching error $(\hat{u},\hat{v}) \leftarrow (u,v)$. Add (u,v) to the queue for holding motion vectors waiting to be searched (hereinafter search queue).

Step 2 Take the first motion vector (k,l) from the search queue, and perform $target\_B(k,l)$.

Step 3 If the search queue is empty, the search ends. Otherwise, return to Step 2.

$\boldsymbol{target\_B(u,v)}$

Step 1 Find the unknown $e(u+s,v+t)$ for the current motion vector (u,v) where $-1 \le s \le 1, -1 \le t \le 1$.

Step 2 Let (x_i,y_i) be the value of (s,t) that gives the i'th smallest value of $e(u+s,v+t)$ where $-1 \le s \le 1, -1 \le t \le 1, (s,t) \ne (0,0)$ and $p(u+s,v+t) = \mathrm{FALSE}$. Let the current search direction (x_i,y_i) be $i \leftarrow 1$.

Step 3 If $i > D$, end. Otherwise, search for the vector that gives the next minimum point by making use of $search\_next\_target(u,v,x_i,y_i)$, and if the motion vector that gives the next minimum point $(\bar{u},\bar{v})$ exists, add that vector to the search queue.

Step 4 Let $i \leftarrow i+1$, and return to Step 3.

$\boldsymbol{search\_next\_target(u,v,x,y)}$

Step 1 Let the number of increases in the gradient $cnt \leftarrow 0$, and the minimum value updated flag $renew \leftarrow \mathrm{FALSE}$.

Step 2

Case 1 If $e(u,v) \ge e(u+x,v+y)$, let the current motion vector $(u,v) \leftarrow (u+x,v+y)$ and $p(u,v) \leftarrow \mathrm{TRUE}$. If $e(u,v)$ updates the minimum value, let $renew \leftarrow \mathrm{TRUE}$, let the motion vector that gives the next minimum value $(\bar{u},\bar{v}) \leftarrow (u,v)$, and let the vector that gives the minimum matching error $(\hat{u},\hat{v}) \leftarrow (u,v)$.

Case 2 If $e(u,v) < e(u+x,v+y)$ and $renew = \mathrm{TRUE}$, the motion vector that gives the next minimum value $(\bar{u},\bar{v})$ exists, and the procedure ends.

Case 3 If $e(u,v) < e(u+x,v+y)$ and $renew = \mathrm{FALSE}$ and $cnt = C$, the motion vector that gives the next minimum value $(\bar{u},\bar{v})$ does not exist, and the procedure ends.

Case 4 Otherwise, let $cnt \leftarrow cnt+1$, let the current motion vector $(u,v) \leftarrow (u+x,v+y)$, and let $p(u,v) \leftarrow \mathrm{TRUE}$.

Step 3 Find the unknown $e(u+s,v+t)$ where $-1 \le s \le 1, -1 \le t \le 1$ for the current motion vector (u,v).

Step 4 Let the current direction (x,y) be the value of (s,t) that gives the minimum value of $e(u+s,v+t)$ for (u,v) where $-1 \le s \le 1$, $-1 \le t \le 1$, $(s,t) \ne (0,0)$, and $p(u+s,v+t) = \mathrm{FALSE}$, and return to Step 2.

2.2.3 Algorithm Using Depth First Search

ALM using depth first search (hereinafter ALMD) can be implemented by using a stack instead of the queue that is used to store motion vectors waiting to be searched in the ALMB procedure shown in 2.2.2.

Alternatively, ALMD can also be implemented by a system of using recursive function calls, as shown below.

ALM where the depth first search is implemented by means of recursive calls consists of the three procedures, top_D, $target\_D(u,v)$, and $search\_next\_target(u,v,s,t)$, as shown below. The $search\_next\_target(u,v,s,t)$ procedure is the same as in 2.2.2.

top_D

Step 1 Let the current motion vector $(u,v) \leftarrow (0,0)$, find $e(u,v)$, and let the flag that indicates the motion vector belong to the search path $p(u,v) \leftarrow \mathrm{TRUE}$. Let the motion vector that gives the minimal matching error $(\hat{u},\hat{v}) \leftarrow (u,v)$.

Step 2 Perform $target\_D(u,v)$.

target_D(u, v)

```
Step 1  Find the unknown $e(u + s, v + t)$ for the current
        motion vector $(u, v)$ where $-1 \leq s \leq 1, -1 \leq t \leq 1$.
Step 2  Let $(x_i, y_i)$ be the value of $(s, t)$ that gives the $i$'th
        smallest value of $e(u + s, v + t)$ where $-1 \leq s \leq 1$,
        $-1 \leq t \leq 1$, $(s, t) \neq (0, 0)$, and $p(u + s, v + t) =$
        FALSE. Let the current search direction $(x_i, y_i)$ be
        $i \leftarrow 1$.
Step 3  If $i > D$, end. Otherwise, search for the vector
        that gives the next minimum point by making use of
        $search\_next\_target(u, v, x_i, y_i)$, and if the motion
        vector that gives the next minimum point $(\bar{u}, \bar{v})$ exists,
        perform $target\_D(\bar{u}, \bar{v})$.
Step 4  Let $i \leftarrow i + 1$, and return to Step 3.
```

3. COMPUTATIONAL EXPERIMENTS

We next show using computational experiments that by avoiding local solutions, the two types of ALM are able to produce smaller prediction errors compared to SDM as described in 2.1 and are able to detect the motion vector under an arbitrary upper limit on the number of computations. Furthermore, the effect that differences in the search order have on the effectiveness of avoiding local solutions is shown.

3.1 Experimental conditions

In the experiments, the template block size was taken to be 16×16 pixels, the reference image of the search target was taken to be the image of the previous frame, and the search range was taken to be $\pm$ 15 in both the horizontal and vertical directions. Frame numbers 150 to 599 (excluding the title frame) of the four types of SIF size evaluation sequences shown in Fig. 3 [8] were used.

In the experiments, the same evaluation indicators as used in [4] and [5] were used. That is, mean square error (MSE) was used as an indicator for evaluating prediction performance. Furthermore, $\mathrm{CPX} = 100 \cdot (\bar{N}/\bar{M})$ was used as an indicator for evaluating the number of computations where $\bar{N}$ is the average number of reference blocks searched per template block using this method, and $\bar{M}$ is the average number of reference blocks searched per template block in FS.

3.2 Experimental Results

3.2.1 Effectiveness of Avoiding Local Solutions

Table 1 shows the values of MSE and CPX as the number of search directions D and number of gradient increase permitted C is varied. The case of $D = 1$ and $C = 0$ is equivalent to SDM.

As shown in the table, in all the evaluation sequences, the value of MSE approached the value of MSE from FS monotonically as the values of D and C were increased, and it can be judged that both ALMD and ALMB were effective in avoiding local solutions. Furthermore, although CPX increases as D and C increase, even at $D = 4$ and $C = 7$, the value of CPX is less than 15%, and it is clear that the number of computations is small compared to FS. Furthermore, although there was a trend for the value of MSE to be slightly smaller and CPX to be slightly larger for ALMD compared to ALMB, this difference is not considered to be significant.

3.2.2 Motion detection performance under an upper limit on the number of computations

Next, an upper limit on the number of computations per template block MAXCPX was applied and the variations in prediction performance were found as MAXCPX was changed. MAXCPX refers to $\mathrm{MAXCPX} = 100 \cdot (N/M)$ where $M = 31 \times 31$ is the number of reference blocks within the search domain of the template block and N is the number of reference blocks that are permitted to be searched per template block.

Figure 5 shows the value of MSE for ALMD, ALMB, and FS as the upper limit on the number of computations MAXCPX was varied. In ALMD and ALMB, the number of search directions D was taken to be 4 and the number of gradients increases permitted C is taken to be 4. In FS, the search was cutoff midway according to MAXCPX by performing searches in the spiral shaped order shown in Fig. 4.

As shown in Fig. 5, the value of MSE decreased monotonically in all the methods as MAXCPX was increased, and tended to converge on the value of MSE for FS when no upper limit is applied on the number of computations. In addition, there was a tendency for the value of MSE to be smaller for ALMB compared to ALMD when MAXCPX was approximately 5 to 10%.

4. CONCLUSION

In this paper, two types of motion vector detection algorithms using the steepest descent method that are effective in avoiding local solutions were demonstrated using depth first search and breadth first search. Furthermore, computer experiments demonstrated that these two types of algorithms were able to obtain prediction errors smaller than conventional motion detection methods using the steepest descent method, and were able to detect motion vectors under an arbitrary limit on the number of computations. Furthermore, these two algorithms did not exhibit large differences in the effectiveness of avoiding local solutions. Issues for the future are progressing with comparisons with other methods and improving prediction performance under arbitrary limits on the number of computations.

5. REFERENCES

[1] ISO/IEC 14496-2, “Information technology - Coding of audio-visual objects - Part2 : Visual,” International Standard, 1999.

Table 1: Dependence of MSE and CPX on D and C.

(a) Woman with bird cage. (FS : MSE 59.5, CPX 100.0%)

			C															
			0		1		2		3		4		5		6		7	
			MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX
D	1	ALMD	74.1	1.5%	71.5	1.9%	70.2	2.4%	69.4	2.8%	68.8	3.2%	68.2	3.6%	67.7	4.0%	67.3	4.4%
		ALMB	74.1	1.5%	71.5	1.9%	70.2	2.4%	69.4	2.8%	68.8	3.2%	68.2	3.6%	67.7	4.0%	67.3	4.4%
	2	ALMD	71.8	1.7%	67.8	2.6%	66.0	3.5%	64.8	4.4%	63.9	5.4%	63.3	6.3%	63.0	7.1%	62.7	8.0%
		ALMB	71.8	1.7%	67.6	2.7%	65.9	3.6%	64.8	4.5%	64.1	5.4%	63.6	6.3%	63.2	7.1%	63.0	7.8%
	3	ALMD	70.5	1.9%	65.5	3.3%	63.8	4.4%	62.6	5.8%	61.8	7.1%	61.4	8.3%	61.2	9.4%	61.0	10.4%
		ALMB	70.5	1.9%	65.5	3.4%	64.1	4.5%	63.2	5.7%	62.6	6.9%	62.3	7.9%	62.0	8.9%	61.8	9.8%
	4	ALMD	69.5	2.1%	63.9	3.9%	62.3	5.2%	61.3	7.0%	60.7	8.5%	60.5	9.9%	60.3	11.2%	60.2	12.4%
		ALMB	69.5	2.1%	64.0	4.0%	62.8	5.2%	62.1	6.6%	61.6	7.9%	61.2	9.1%	61.0	10.3%	60.8	11.3%

(b) Whale show. (FS : MSE 356.5, CPX 100.0%)

			C															
			0		1		2		3		4		5		6		7	
			MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX
D	1	ALMD	400.2	1.5%	391.3	1.9%	385.9	2.3%	383.1	2.7%	380.9	3.1%	379.0	3.5%	377.3	4.0%	376.1	4.4%
		ALMB	400.2	1.5%	391.3	1.9%	385.9	2.3%	383.1	2.7%	380.9	3.1%	379.0	3.5%	377.3	4.0%	376.1	4.4%
	2	ALMD	389.8	1.7%	374.6	2.6%	369.0	3.5%	366.0	4.4%	364.2	5.4%	363.0	6.3%	362.2	7.3%	361.6	8.1%
		ALMB	389.8	1.7%	373.7	2.7%	369.1	3.6%	366.5	4.6%	364.8	5.4%	363.7	6.3%	363.0	7.2%	362.4	8.0%
	3	ALMD	383.9	1.9%	368.5	3.3%	364.4	4.5%	362.4	5.9%	361.1	7.3%	360.3	8.6%	359.9	9.8%	359.5	10.9%
		ALMB	383.9	1.9%	368.1	3.5%	364.7	4.7%	363.0	5.9%	362.1	7.1%	361.3	8.3%	360.8	9.4%	360.5	10.4%
	4	ALMD	381.0	2.0%	365.4	4.0%	362.3	5.4%	360.8	7.3%	360.0	8.9%	359.5	10.5%	359.1	11.9%	358.8	13.2%
		ALMB	381.0	2.0%	365.3	4.1%	362.8	5.5%	361.5	7.0%	360.7	8.4%	360.3	9.8%	359.8	11.0%	359.5	12.1%

(c) Soccer action. (FS : MSE 95.6, CPX 100.0%)

			C															
			0		1		2		3		4		5		6		7	
			MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX
D	1	ALMD	176.3	1.5%	160.3	1.9%	152.0	2.3%	146.7	2.8%	143.1	3.2%	140.3	3.6%	138.1	4.1%	136.2	4.5%
		ALMB	176.3	1.5%	160.3	1.9%	152.0	2.3%	146.7	2.8%	143.1	3.2%	140.3	3.6%	138.1	4.1%	136.2	4.5%
	2	ALMD	162.5	1.6%	134.8	2.7%	122.6	3.7%	114.7	4.7%	110.9	5.7%	108.0	6.8%	106.5	7.8%	105.4	8.8%
		ALMB	162.5	1.6%	133.3	2.8%	122.0	3.8%	115.6	4.8%	111.6	5.8%	109.6	6.8%	108.7	7.8%	107.6	8.7%
	3	ALMD	154.7	1.7%	124.6	3.5%	113.2	4.9%	106.6	6.4%	103.3	7.9%	101.5	9.3%	100.4	10.7%	100.0	11.9%
		ALMB	154.7	1.7%	123.3	3.7%	113.6	5.1%	108.4	6.5%	105.7	7.9%	104.4	9.2%	103.5	10.5%	102.7	11.7%
	4	ALMD	150.2	1.8%	117.0	4.2%	107.4	6.0%	101.8	7.9%	99.6	9.7%	98.7	11.4%	98.1	13.0%	97.7	14.4%
		ALMB	150.2	1.8%	116.5	4.4%	108.5	6.1%	104.2	7.8%	102.0	9.4%	101.1	11.0%	100.2	12.4%	99.7	13.7%

(d) Marching in. (FS : MSE 195.1, CPX 100.0%)

			C															
			0		1		2		3		4		5		6		7	
			MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX	MSE	CPX
D	1	ALMD	196.2	1.0%	195.8	1.4%	195.6	1.9%	195.5	2.4%	195.5	2.8%	195.5	3.3%	195.4	3.8%	195.4	4.3%
		ALMB	196.2	1.0%	195.8	1.4%	195.6	1.9%	195.5	2.4%	195.5	2.8%	195.5	3.3%	195.4	3.8%	195.4	4.3%
	2	ALMD	196.2	1.0%	195.5	1.8%	195.3	2.7%	195.2	3.6%	195.2	4.6%	195.2	5.5%	195.2	6.4%	195.2	7.3%
		ALMB	196.2	1.0%	195.5	1.9%	195.3	2.7%	195.2	3.6%	195.2	4.6%	195.2	5.5%	195.2	6.4%	195.2	7.3%
	3	ALMD	196.2	1.0%	195.4	2.3%	195.3	3.3%	195.2	4.6%	195.2	6.0%	195.2	7.3%	195.2	8.5%	195.2	9.7%
		ALMB	196.2	1.0%	195.4	2.3%	195.2	3.3%	195.2	4.7%	195.2	6.0%	195.2	7.3%	195.2	8.5%	195.2	9.7%
	4	ALMD	196.2	1.0%	195.4	2.7%	195.2	3.9%	195.2	5.6%	195.2	7.3%	195.2	8.9%	195.2	10.3%	195.1	11.7%
		ALMB	196.2	1.0%	195.4	2.8%	195.2	4.0%	195.2	5.6%	195.2	7.2%	195.2	8.8%	195.2	10.2%	195.1	11.6%

[2] ITU-T Recommendation H.263, “Video coding for low bit rate communication,” 1998.

[3] T. Koga, K. Iinuma, A. Hirano, Y. Iijima and T. Ishiguro, “Motion compensated interframe coding for video conferencing,” Proc. National Telecommunication Conf., G.5.3, 1981.

[4] L. K. Liu and E. Feig, “A Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding,” IEEE Trans. Circuits Syst. Video Technol., vol. 6, no. 4, pp. 419–422, August 1996.

[5] O. T.-C. Chen, “Motion Estimation Using a One-Dimensional Gradient Descent Search,” IEEE Trans. Circuits Syst. Video Technol., vol. 10, no. 4, pp. 608–616, June 2000.

[6] K. Imamura, Y. Nakanishi and H. Hashimoto, “Improvement of Local Minima Problem and Its Effectiveness for Motion Vector Detection Using Steepest Descent Method,” IPSJ Journal, vol. 45, No. 11, pp. 2528–2531, November 2004.

[7] T. Nagao, Optimization Algorithms, Shokodo, Tokyo, 2000.

[8] The Institute of Image Information and Television Engineers, Standard Test Sequences (SIF), NHK Engineering Services, Inc., Tokyo, 2003.

(a)

(b)

(c)

(d)

Fig. 3: Evaluation sequences. (a) Woman with bird cage. (b) Whale show. (c) Soccer action. (d) Marching in.

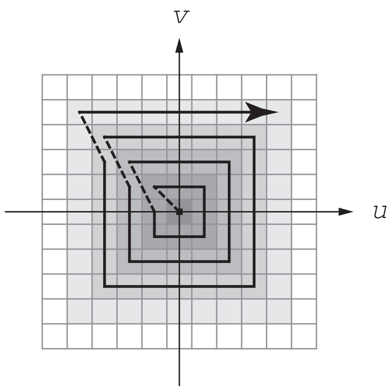
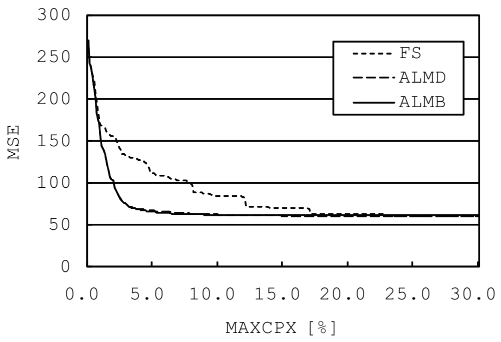
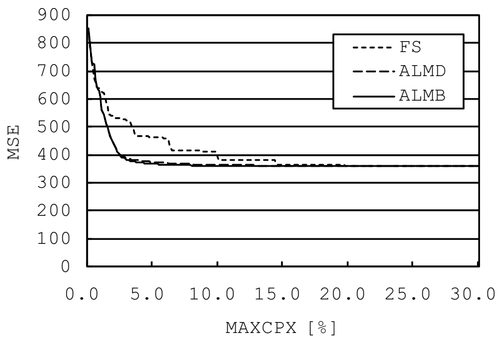


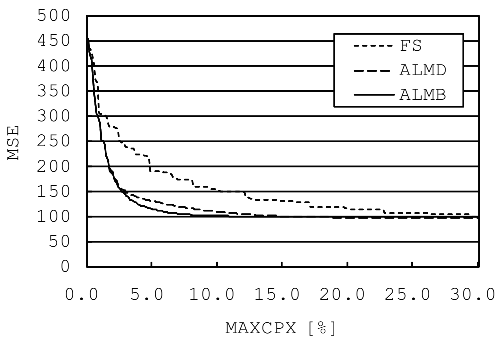
Fig. 4: Search order in FS.



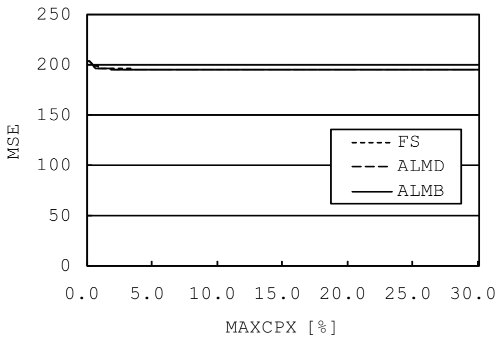
(a) Woman with bird cage.



(b) Whale show.



(c) Soccer action.



(d) Marching in.

Fig. 5: Dependence of MSE on MAXCPX.