Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

A Fast Full Search Algorithm for Minimizing the Maximum Difference (MinMax) in Motion Estimation[+]

Kuang-Shyr (Keith) Wu, Jing-Yi Lu, and Ja-Chen Lin*

Department of Computer and Information Science
National Chiao Tung University
Hsinchu, Taiwan 30005
R.O.C.

Abstract

In this paper, a fast full search (FS) algorithm is proposed to accelerate the block matching procedure of motion estimation. Based on the monotonic relation between the Supreme of absolute differences (SAD) obtained for distinct layers of a pyramid structure called Sup-pyramid, the proposed method successfully rejects many impossible candidates considered in the FS. The derivation of the monotonicity relation uses the l_∞ version of Minkowski's Inequality, an inequality which is quite well-known in the field of Math. Simulation results show that the computational complexity is much better than that of the FS. The processing speed of the proposed method is also compared with that of the Three-Step Search (TSS), which is often used for block matching in interframe video coding, although the visual quality performance of TSS is usually poorer than that of the FS.

1. Introduction

The block-matching algorithm is popularly used for the motion estimation of interframe video coding. The ideal approach to block-matching algorithm is the FS, which exhaustively searches for a block whose, say, mean absolute distortion (MAD) to an input block, is minimal among all possible blocks within a search window. To save the hardware requirement of the block-matching algorithm with acceptable video performance and increase the speed of computation in VLSI chip, M. J. Chen *et al.* [1] suggested the use of the matching criterion called Minimized Maximum Difference (MiniMax). Although the FS can find the optimal solution, i.e. the best-matched block mentioned above, the serious computation burden needed will make the hardware implementation difficult[2] if the window size becomes large (in application to HDTV or super high-resolution TV). Many existing methods save computation but degrade the visual quality; examples include: the TSS[3], the orthogonal search[4], or the two algorithms introduced in Reference [5] which use alternating patterns.

In this paper, we present a fast searching algorithm for the block matching. The monotonic relation between the SAD (Supreme of absolute differences) of distinct layers of a pyramid structure called Sup-pyramid will be derived using the l_∞-version of Minkowski's Inequality which is quite well-known in the field of math. With the derived monotonic relation, many impossible candidates can be kicked out without doing time-consuming computation. The proposed method keeps the good visual quality, i.e., the minimized SAD error, of the FS, while the processing speed is greatly improved.

2. Definitions and the proposed algorithm

In this paragraph, we review the definition of MiniMax. For a specified block G of size $2^N \times 2^N$ (the G will be referred to hereafter as the incoming (or input) block) in the current frame t, the corresponding $SAD(\mathrm{x,y})$ is defined as

$$SAD(\mathrm{x,y}) = \max_{1\le i\le 2^N} \max_{1\le j\le 2^N} |f^t(i,j) - f^{t-1}(i+x, j+y)| \quad (1)$$

while the corresponding conventional Mean-Absolute-Distortion $MAD(\mathrm{x,y})$ is defined as

$$MAD(\mathrm{x,y}) = \left(\sum_{i=1}^{2^N} \sum_{j=1}^{2^N} | f^t(i,j) - f^{t-1}(i+x, j+y) | \right) / 2^{2N}.$$

Here, $f^t(i,j)$ represents the gray value at pixel (i,j) of G in frame t, and $f^{t-1}(i+x,j+y)$ denotes the gray value at pixel $(i+x,j+y)$ in frame $t-1$. As for the vector (x,y), which is often called as the "position vector", denotes the relative displacement of a candidate block (in the

[+]This study was supported by the National Science Council, Republic of China, under contract number: NSC85-2213-E009-111.

previous frame *t-1*) to the specified block **G** (in the current frame *t*).

Let $V=(d_x,d_y)$ be the (*x,y*) that solves $\min_{-2^N\le x,y\le 2^N} SAD(x,y)$. In other words, for each displacement (*x,y*) of the block-matching, the maximum of the absolute errors is found by inspecting all pixels of the corresponding block. Then, among all candidate blocks in the previous frame *t-1*, the block whose displacement is (*x,y*) and the corresponding *SAD*(*x,y*) is the smallest will be chosen as the best-matched block.

Comparing with the conventional criterion of minimizing MAD, the major benefit of the MiniMax criterion lies in that the MiniMax has less hardware requirements [2]. Therefore, the MiniMax criterion is used from time to time due to the better speed when VLSI chip is used.

In the block-matching algorithms, the current frame *t* is often divided into non-overlapping blocks of size $2^N x 2^N$ (*16x16* in this paper, i.e. *N=4*). The goal of the BMA is to find for each block **G** its best position vector (m,n) within a search window **W** (in this paper the window size is $(2^N+1)x(2^N+1)=33x33$) such that

$$SAD(\mathrm{m,n}) = \min_{-2^N\le x,y\le 2^N} SAD(\mathrm{x,y}). \qquad (2)$$

Hereafter, the position vector which is the best among all *33x33* position vectors {(x,y)} will be called as the "motion vector" (for the block **G**). The basic concept to obtain the motion vector (m,n) quickly is to eliminate from the candidate space **W** those position vectors (x,y) which are impossible to be the best. We first review below a pyramid structure so that those impossible candidates can be eliminated layer by layer.

For every block **B**, the corresponding pyramid of (*N+1*) layers (from layer 0 to layer *N*) is a sequence $\{\mathbf{L}_0, \mathbf{L}_1,\ldots, \mathbf{L}_N\}$, with each layer defined as follows: the layer $\mathbf{L}_N$ of size $2^N x 2^N$ is exactly the original block **B**, the $\mathbf{L}_{N-1}$ of size $2^{N-1}x2^{N-1}$ is a size-reduced layer derived from $\mathbf{L}_N$, the $\mathbf{L}_{N-2}$ of size $2^{N-2}x2^{N-2}$ is a size-reduced layer derived from $\mathbf{L}_{N-1}$,..., and finally, the layer $\mathbf{L}_0$, which consists of a single pixel only, is a size-reduced layer derived from $\mathbf{L}_1$ (see Fig. 1). For each layer $\mathbf{L}_{k-1}$ (where $1\le k\le N$), each pixel value $f_{k-1}(i,j)$ (where $1\le i\le 2^{k-1}$ and $1\le j\le 2^{k-1}$) is obtained by a maximum selection of the *2x2* corresponding pixel values of $\mathbf{L}_k$. Expressed symbolically, the pixel values on layer $\mathbf{L}_{k-1}$ are computed by

$f_{k-1}(i,j)=max\{f_k(2i-1,2j-1)\ +\ f_k(2i-1,2j)\ +$

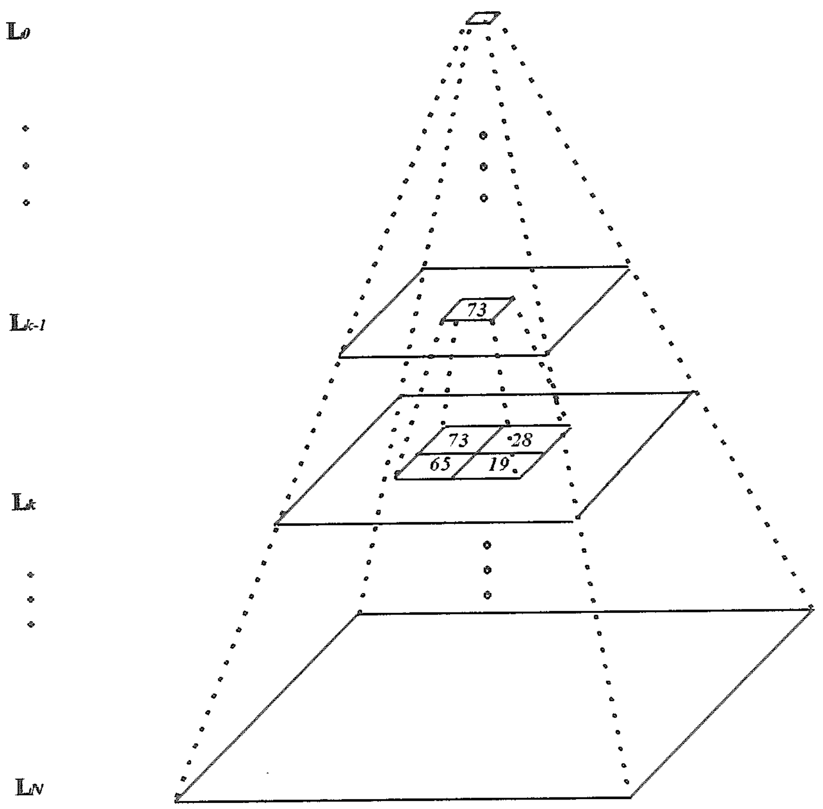


Fig. 1:The structure of the Sup-pyramid. Note that 73=*Max*{73,28,65,19} illustrates the routine way (see Eq. (3)) to construct a layer $\mathbf{L}_{k-1}$ from its previous layer $\mathbf{L}_k$.

$$f_k(2i,2j-1)+f_k(2i,2j)\}$$
for $1\le i,j\le 2^{k-1}$ and $1\le k\le N$. (3')

Since each layer $\mathbb{L}_k$ can be considered as a (low-resolution) image , the $SAD_k(x,y)$ between layer $\mathbb{L}_k$ of an input block $\mathbb{G}$, and layer $\mathbb{L}_k$ of the block corresponding to a motion vector candidate (x,y), can be defined by generalizing (1), namely,

$$SAD_k(x,y) = max\{\ |f_k^t(i,j) - f_k^{[t-1;(x,y)]}(i,j)| : 1\le j\le 2^{k-1},\ 1\le i\le 2^{k-1}\ \} \text{ for } 0\le k\le N. \quad (4)$$

Here, the $f_k^t(i,j)$ represents the (generalized) gray value at the (low-resolution) pixel (i,j) in Layer $\mathbb{L}_k$ constructed from the 2^N-by-2^N block $\mathbb{G}$ in frame t; and $f_k^{[t-1;(x,y)]}(i,j)$ denotes the (generalized) gray value at the (low resolution) pixel (i,j) in Layer $\mathbb{L}_k$ constructed from a 2^N-by-2^N block in frame t-1, with the center of that 2^N-by-2^N block being (x,y) units away from the center of the 2^N-by-2^N block $\mathbb{G}$.

The relation among the SAD of distinct layers is introduced below. For two arbitrarily given sets $\mathbb{P}=\{p_1,p_2,\ldots,p_z\}$ and $\mathbb{Q}=\{q_1,q_2,\ldots,q_z\}$ of non-negative real numbers, where $\mathbb{P}$ and $\mathbb{Q}$ have the same number of elements, the l_∞-version of the very famous Minkowski's Inequality

$$\|\mathbb{P}-\mathbb{Q}\| \ge |\ \|\mathbb{P}\| - \|\mathbb{Q}\|\ |$$

(see Theorem 8.13 of Reference [6] for the definition and proof of the infinitely many versions l_1~l_∞ of the Minkowski's Inequality) implies that

$$\max_{1\le i\le z}|p_i - q_i| = \|\mathbb{P}-\mathbb{Q}\| \ge |\ \|\mathbb{P}\| - \|\mathbb{Q}\|\ | = \left|\max_{1\le i\le z}|p_i| - \max_{1\le i\le z}|q_i|\right| = \left|\max_{1\le i\le z} p_i - \max_{1\le i\le z} q_i\right|$$

This well-known fact can also be interpreted geometrically as : in the l_∞-norm space, the triangular inequality still holds (The assumption $p_i\ge 0$ and $q_i\ge 0$ for all i makes $\|\mathbb{P}\|_\infty = max|p_i| = max\ p_i$ and $\|\mathbb{Q}\|_\infty = max\ |q_i| = max\ q_i$ when we evaluate the l_∞-norm); that is,

$$\max_{1\le i\le z}|p_i - q_i| = \|\mathbb{P}-\mathbb{Q}\|_\infty \ge \left|\|\mathbb{P}\|_\infty - \|\mathbb{Q}\|_\infty\right| = \left|\max_{1\le i\le z}|p_i| - \max_{1\le i\le z}|q_i|\right| = \left|\max_{1\le i\le z} p_i - \max_{1\le i\le z} q_i\right| \quad (5)$$

Here, we have used the fact that gray values $p_i\ge 0$ and $q_i\ge 0$ for all i and thus $\|\mathbb{P}\|_\infty = \max_{1\le i\le z}|p_i| = \max_{1\le i\le z} p_i$ and $\|\mathbb{Q}\|_\infty = \max_{1\le i\le z}|q_i| = \max_{1\le i\le z} q_i$ hold when evaluating the l_∞-norm.

According to inequality (5), the inequality

$$SAD_k(x,y)\ge SAD_{k-1}(x,y) \text{ for } 1\le k\le N \quad (6)$$

can be proved (the proof is given in the Appendix 1). From the pyramid structure and (6), we can efficiently search for the motion vector, i.e., the best position vector. Without loss of generality, we assume that a part of the $33\times 33=1089$ position vectors in $\mathbb{W}$ have been inspected; the "so far" best position vector is stored in (m,n); and the corresponding distortion SAD(m,n) is stored in SAD_{min}. Then the following corollary is always true:

Corollary: If $SAD_k(x,y) \ge SAD_{min}$, for some $k\in\{0,1,\ldots,N\}$, then
$SAD(x,y) \ge SAD_{min}$,
i.e., in the previous frame t-1, the block obtained by translating the input block (x,y) units will not be better than the block obtained by translating the input block (m,n) units.

This corollary comes directly from (6): because $SAD_N(x,y)$ is in fact $SAD(x,y)$, and $SAD_N(x,y)\ge SAD_{N-1}(x,y) \ge\ldots\ge SAD_0(x,y)$ always holds by (6), we therefore obtain

$$SAD(x,y)=SAD_N(x,y)\ge SAD_k(x,y)\ge SAD_{min}.$$

Hence, (x,y) could not be better than (m,n), and thus, (x,y) should be rejected. Note that the computation of $SAD_k(x,y)$ grows when k increases. More precisely, the computation load of $SAD_k(x,y)$ is about four times heavier than that of $SAD_{k-1}(x,y)$. Therefore, the proposed algorithm uses the "top-down" approach (from $\mathbb{L}_0$ to $\mathbb{L}_N$) to reject impossible candidates (x,y) layer by layer. Some are rejected in $\mathbb{L}_0$, some are rejected in $\mathbb{L}_1$, etc. Of course, the overhead to construct the pyramid (the construction is bottom-up, however) should also be considered. But, we found that, even with this overhead, the total time is still saved somewhat using the above corollary.

For the reader's benefit, we give below the algorithm that utilizes the proposed method to eliminate those "impossible position vectors" for a given $\mathbb{G}$. Note that the motion vector of the corresponding block (i.e., the block having identical location as $\mathbb{G}$) in frame t-1 is used as the initial guess for the motion vector (m,n) of $\mathbb{G}$ of the current frame t.

Algorithm.

Goal: Find the motion vector, i.e., the best position vector within a search window $\mathbb{W}$ in the previous frame t-1 for an incoming block $\mathbb{G}$ in the current frame t.

Input: 1. An incoming block $\mathbb{G}$ of size $2^N x 2^N = 16\times 16$ in frame t.
2. $33\times 33=1089$ candidates, i.e., 1089 position vectors, in $\mathbb{W}$ of frame t-1.

Output: The motion vector (m,n), i.e., the position of the best-matched block.

Initial conditions: 1. Assume that the pyramids for all blocks in frame t-1 have been established. (The overhead shared by each $\mathbb{G}$ is quite small, as is indicated by Appendix 2 and Table 1.)

2. Let the initial guess of the motion vector (m,n) be the motion vector of the corresponding block of **G** in frame *t-1*.

Steps:

```
Step 1: Use Formula (3) to construct the pyramid for G.
Step 2: Evaluate the SAD for the initial guess (m,n) and let $SAD_{\min} = SAD(m,n)$.
Step 3: If all position vectors have been checked, then go to Step 8.
Step 4: Pick up from the search window W a position vector (x,y) which is not processed yet by Step 5.
Step 5: For $k = 0$ to $N$ do
          if ($SAD_k(x,y) \geq SAD_{\min}$ for the current value of $k$) then
              delete (x,y) from W and go to Step 3.
Step 6: Replace the content of $SAD_{\min}$ by $SAD_N(x,y)$ and $(m,n)$ by $(x,y)$, respectively. In symbol, $SAD_{\min} \leftarrow SAD_N(x,y)$, $m \leftarrow x$, and $n \leftarrow y$. (This step updates the "so far" minimal SAD and the "so far" best position vector.)
Step 7: Go to Step 3.
Step 8: Print out the final value of (m,n) because its content now is indeed the motion vector. ❑
```

3. Experimental results

In this section, the FS, TSS, and the proposed method are compared. All three methods estimated motions from the original frames instead of from the reconstructed frames. Table 1 shows the average results of the four test image sequences from the 0th frame to the 30th's.

In Table 1, every entry of the ANSP for our method has a value "6" added to indicate the overhead to construct the pyramids in the previous frame *t-1*, and the reason of using "6" is provided in Appendix 2. As for the way by which the first portion of each entry under the column with the title ANSP is obtained, we collected together all the works used in the experiments to evaluate the MAX_k, $0 \leq k \leq N$, and then expressed the collected works in terms of the work needed to evaluate MAX_N. The collection we just made was used as the first portion of the ANSP.

As for the values of SAD or PSNR, which gauge the visual qualities of the reconstructed images, the proposed method produced the same values as those of the FS. This should be of no surprise because the proposed method is just a kind of fast implementation of the FS. The TSS, however, had worse values in both SAD and PSNR, because TSS is not a kind of FS (the TSS uses a hard rule to by-pass many candidates without even giving these candidates a chance of being checked. Unfortunately, it happens quite often that some of these candidates by-passed by TSS are the ones that yield optimized match).

We had also compared our results with that of another pyramid-based method [7] which used the so-called Mean-pyramid. (The method of [7] was originally developed for Vector Quantization, but we use it here for motion estimation of images.) We found that our SAD is a little better than that of [7], but our MSE (Mean Square Error) is a little worse than that of [7]. This should be of no surprise, however; because the method [7] tried to minimize MSE while our method tried to minimize SAD.

Table 1. Performance comparison among three algorithms when MiniMax was used as the matching criterion.

Images	Algorithms	ANSP	*PSNR*	*SAD*
Suzie	FS	1089	38.385	12.54
	ours	132+6	38.385	12.54
	TSS	33	37.713	14.04
Salesman	FS	1089	36.837	12.97
	ours	46+6	36.837	12.97
	TSS	33	36.596	13.48
Mobile	FS	1089	26.41	63.102
	ours	199+6	26.41	63.102
	TSS	33	23.85	79.573
Football	FS	1089	26.5	50.662
	ours	386+6	26.5	50.662
	TSS	33	24.7	60.48

4. Conclusions

In this paper, a fast searching algorithm for the block matching of motion estimation has been proposed according to the matching criteria called MiniMax. Our MiniMax-based method uses the monotonic relation between the SAD (Supreme Absolute Difference) values obtained for distinct layers of a pyramid structure called Sup-pyramid to reduce the computation load significantly. To prove the monotonic relation, we had used the l_∞ version of the quite well-known Minkowski's Inequality of the normed linear space. Also note that we adopted MiniMax as the matching criterion from the motivation stated in [1] which pointed out that using MiniMax can save the extra hardware requirement and increase the speed of computation in VLSI chip. The experimental results showed that the proposed method really reduced significantly the computations needed by the FS although the optimized quality of FS was completely kept.

APPENDIX 1

The statement $SAD_{k+1}(x,y) \geq SAD_k(x,y)$ holds for all $k \in \{0,1,\ldots,N-1\}$.

Proof:

By (3) through (5), we get

$$SAD_{k+1}(x,y) = \max_{1\le i\le 2^{k+1}} \max_{1\le j\le 2^{k+1}} \left| f'_{k+1}(i,j) - f_{k+1}^{[t-1;(x,y)]}(i,j) \right| \qquad \text{(A1)}$$

$$= \max_{1\le i\le 2^k} \max_{1\le j\le 2^k} \left(\max_{0\le a\le 1} \max_{0\le b\le 1} \left| f'_{k+1}(2i-a,2j-b) - f_{k+1}^{[t-1;(x,y)]}(2i-a,2j-b) \right| \right) \qquad \text{(A2)}$$

$$\geq \max_{1\le i\le 2^k} \max_{1\le j\le 2^k} \left| \max_{0\le a\le 1} \max_{0\le b\le 1} f'_{k+1}(i,j) - \max_{0\le a\le 1} \max_{0\le b\le 1} f_{k+1}^{[t-1;(x,y)]}(i,j) \right| \qquad \text{(A3)}$$

$$= \max_{1\le i\le 2^k} \max_{1\le j\le 2^k} \left| f'_k(i,j) - f_k^{[t-1;(x,y)]}(i,j) \right|$$

$$= SAD_k(x,y).$$

The derivation of (A2) from (A1) is just a change of the indices so that the $2^{k+1} \times 2^{k+1}$ pixels in (A1) are decomposed into $2^k \times 2^k$ groups with each group having four adjacent pixels. The derivation of (A3) from (A2) is again by the l_∞-norm version of Minkowski's Inequality (in the 4-dimensional vector space). ❑

APPENDIX 2

For each input block G of size 16×16, the overhead of constructing the needed pyramids in frame *t-1* is no more than six times of the work of evaluating the traditional SAD.

Proof:

In each layer L_{k-1}, $1 \le k \le 4$, each pixel value $f_{k-1}(i,j)$ (where $1 \le i \le 2^{k-1}$ and $1 \le j \le 2^{k-1}$) is obtained by a maximum selection of the *2×2* corresponding pixel values of L_k, and the maximum selection of the *2×2* pixel values needs only three comparisons. Therefore, the total computation to obtain these pixel values for the *W×H* image frame is 3×(total number of pixels). As shown in Fig 2, if we consider all layers, then, due to the overlap of pyramid bases, there are *(W-1)×(H-1)* pixels in layer L_3 for frame *t-1*. Similarly, there are *(W-3) ×(H-3)*, *(W-7)×(H-7)*, *(W-15)×(H-15)* pixels in L_2, L_1, L_0, respectively. The total number of pixels is therefore *(W-1)×(H-1)+(W-3)×(H-3)+(W-7)×(H-7)+(W-15)×(H-15)*, and the total computations required to obtain their pixel values are

$$U = 3[(W-1)\times(H-1)+(W-3)\times(H-3)+(W-7)\times(H-7)+(W-15)\times(H-15)]$$
$$= 12WH-78W-78H+852$$

This computation load can be viewed as the search overhead. The total number of non-overlapping blocks G for the current frame is $V=WH/(16)^2$ The computation overhead for each incoming block G of the current frame is

$$\frac{U}{V} = \left(12 - \frac{78}{W} - \frac{78}{H} + \frac{852}{WH}\right) \times (16)^2$$

$$\cong 12(16)^2$$

comparisons, for both *W* and *H* are usually (much) greater than 78. Because each SAD evaluation requires $(2^N)^2 = (16)^2$ subtractions, $(16)^2$ absolute-value-takings, and $(16)^2$-1 comparisons, this means that the average computation overhead for each block G is about 12/3=4 to 12/2=6 SAD evaluations. (Taking absolute value uses less CPU cycle than doing subtraction or comparison. Moreover, the CPU cycle needed to do one comparison is almost identical to that of one subtraction; and hence, one subtraction, one absolute-value-taking and one comparison will approximately equal to the work of doing comparison 2~3 times.) ❑

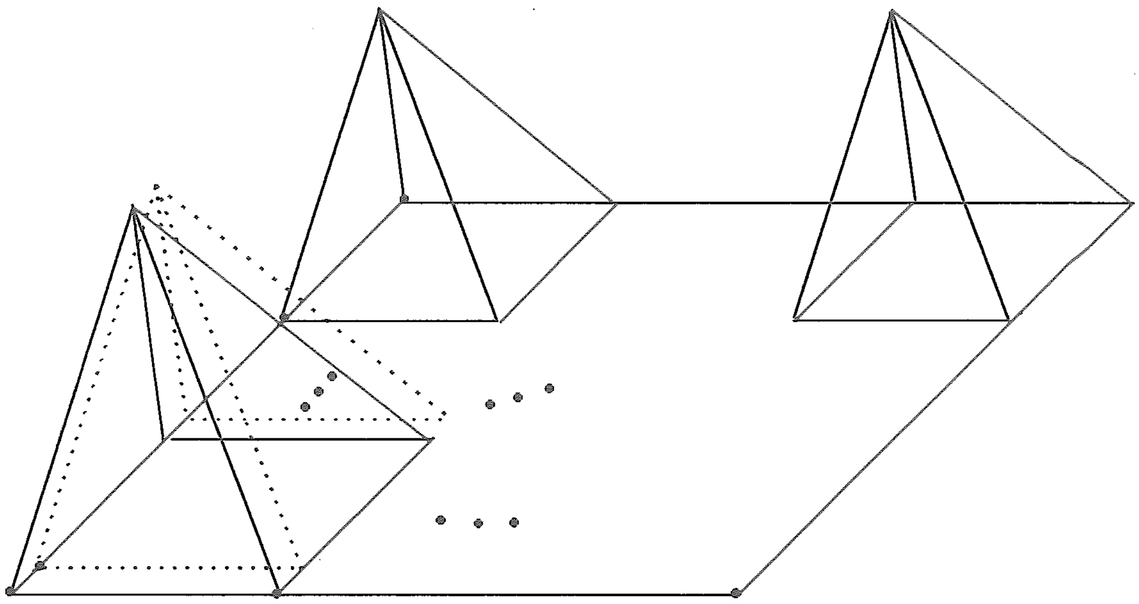
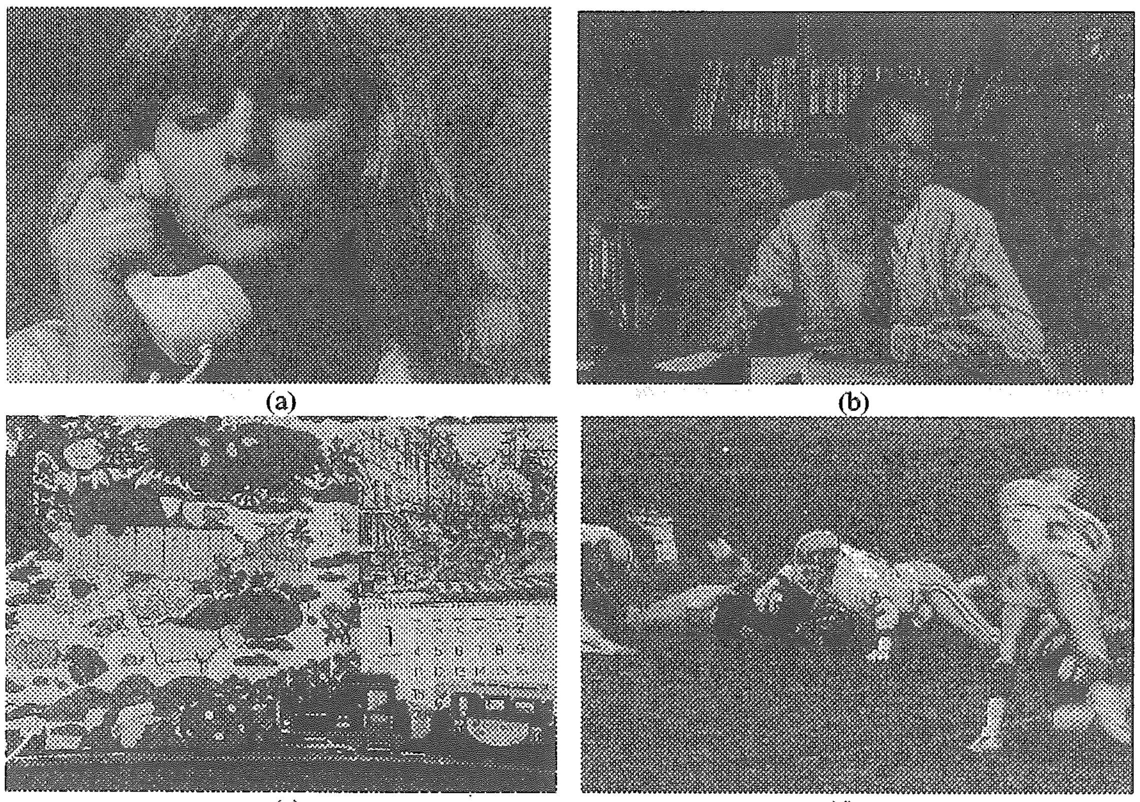


Fig. 2 :Some of the $(M-15)\times(H-15)$ pyramids in the previous frame $t-1$. Assume that the image size is $M\times H$, and the block size (the size of the pyramid base) is 16×16.



(a) (b) (c) (d)

Fig. 3. The four test image sequences. (a) Suzie, (b) Salesman, (c) Mobile, and (d) Football.

References

[1] M.J. Chen, L.G. Chen, T.D. Chiueh and Y.P. Lee, "A new block-matching criterion for motion estimation and its implementation," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 5, No. 3, pp. 231-236, June 1995.

[2] L.W. Lee, J.F. Wang, J.Y. Lee and J.D. Shie, "Dynamic Search-Window Adjustment and Interlaced Search for Block-Matching Algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 3, No. 1, pp. 85-87, Feb. 1993.

[3] T. Koga, K. Iinuma, A. Hirano, Y. Iijima and T. Ishiguro, "Motion Compensated Interframe Coding for Video Conference," *Proc. Nat. Telecommun. Conf.*, pp. G5.3.1-G5.3.5, Nov/Dec 1981.

[4] A. Puri , H. M. Hang and D. L. Schilling, "An Efficient Block-matching Algorithm for Motion Compensated Coding," *Proc. IEEE ICASSP* pp.25.4.1-25.4.4. 1987.

[5] B. Liu and A. Zaccarin, "New Fast Algorithms for the Estimation of Block Motion Vectors," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 3, No. 2, pp. 148-157, Apr. 1993.

[6] R. L. Wheeden and A. Zygmund, "Measure and Integral: An Introduction to Real Analysis," p. 131, Marcel Dekker, Inc., New York, 1977.

[7] C. H. Lee and L H. Chen, "A fast search algorithm for vector quantization using means pyramids of codewords," *IEEE Trans.* on *Commun.*, Vol. 437 No. 2/3/4, pp. 1697-1702, Feb/Mar/Apr 1995.