

# A description of four motion estimation algorithms

Ramakrishna Kakarala\*

February 11, 2002

## Abstract

This report describes four standard algorithms for motion estimation.

## 1 Introduction

Accurate motion estimation is a key component in high-quality video compression. A recent study[1] of motion estimation algorithms compares 14 different algorithms on the basis of operation complexity (number of additions, subtractions etc) and also quality at a fixed bit rate. Of these, the “three step search” (3SS) and the “projection matching” (PROJ) are shown to compare favorably with others.

In this report, 3SS and PROJ are described, as well as two other well-known motion estimation algorithms, “logarithmic search” (LOG) and “conjugate direction search” (CDS).

## 2 Algorithm description

We assume that the search area is limited to a maximum of  $\pm 8$  pixels in each direction. The aim of motion estimation is to find the best possible motion vector within the set of points  $[-8, 8] \times [-8, 8]$  to match a  $16 \times 16$  macroblock in the current frame to a macroblock in the previous frame. The matching is only performed on luminance values. Let the current macroblock be denoted  $C$ , with upper left corner coordinates  $(x_0, y_0)$ . Let the macroblock referenced in the previous frame be  $P$ , and the motion vector be denoted  $(m_x, m_y)$ . The motion vector should be chosen to minimize the sum of absolute differences

$$\text{SAD} = \sum_{x=0}^{15} \sum_{y=0}^{15} |C[x_0 + x, y_0 + y] - P[x_0 + x + m_x, y_0 + y + m_y]|. \quad (1)$$

The simplest possible algorithm for motion estimation is “full search” (FS), which searches all possible points in  $[-8, 8] \times [-8, 8]$  for the motion vector yielding the minimum SAD. Obviously, no algorithm can improve on FS for accuracy, but significant improvements in speed are possible. Four fast motion estimation algorithms are now described.

---

\*R. Kakarala is with Agilent Technologies, 3500 Deer Creek Rd MS 26U-2, Palo Alto CA 94304, USA. email: kakarala@labs.agilent.com.

## 2.1 Three step search (3SS)

Published 20 years ago[2], the 3SS algorithm is still considered one of the best performing motion estimation algorithms. This method searches in ring-shaped patterns, decreasing the size of the ring at each step. The algorithm is illustrated in Figure 1. In the first step, the set of nine points marked (1) are searched. The spacing between points is 4 pixels in this step. In the next step, the spacing is reduced to 2 pixels and points are again searched around the best of the points searched in the previous step. This is repeated for a third and final step, with the spacing now reduced to 1 pixel.

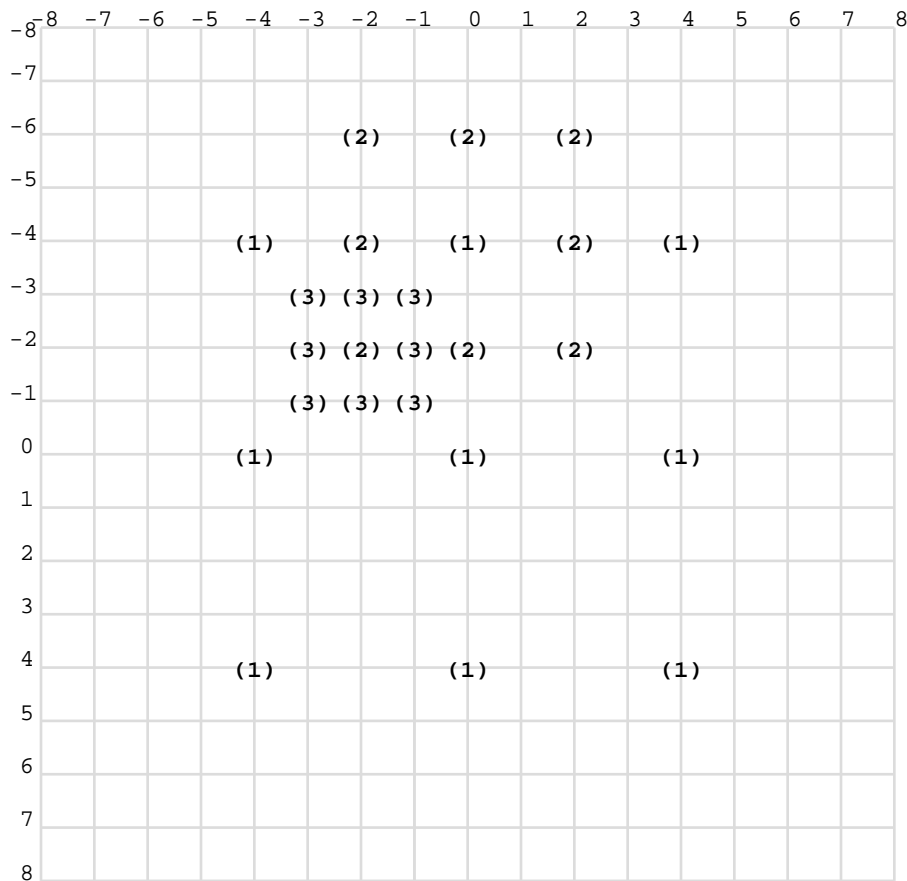


Figure 1: An example of a search pattern for 3SS is shown. The points marked (1), spaced 4 pixels apart, are searched in the first step. The next step searches points marked (2), spaced 2 pixels apart, around the best (lowest SAD) of the points from the first step. The third step searches with spacing 1 pixel around the best of the second step points.

It can be seen that 3SS searches a total of 25 unique points, and therefore requires 25 computations of SAD.

## 2.2 Projection matching (PROJ)

Projection matching is a full-search algorithm that reduces computation by comparing only the horizontal and vertical projections of a macroblock. For the macroblock in the current frame  $C[x, y]$ , the projections are

$$H[y] = \sum_{x=0}^{15} C[x, y], \quad V[x] = \sum_{y=0}^{15} C[x, y].$$

The degree of fit between  $C$  and the displaced macroblock in the previous frame is the sum of absolute differences of projections (PROJSAD). Letting  $H_P$ ,  $V_P$  denote the horizontal, vertical projections of the macroblock from the previous frame (with offset given by the motion vector), the PROJSAD is computed as follows:

$$\text{PROJSAD} = \sum_{y=0}^{15} |H[y] - H_P[y]| + \sum_{x=0}^{15} |V[x] - V_P[x]| \quad (2)$$

The advantage of using projections in full search is that their updates are simpler for small changes in motion vectors: for example, if the motion vector is increased by 1 in the vertical direction, then only one element of  $H_P$  needs to be computed, and the update to  $V_P$  is also simple. The search pattern is illustrated in Figure 2.

## 2.3 Conjugate Direction Search (CDS)

The idea behind CDS [4] is to search in the direction that yielded the best result in the previous step, or switch to the orthogonal direction. The initial step is to search in the star-shaped pattern around the origin indicated by points marked (1) in Figure 3. The search terminates if the center yields the smallest SAD. Otherwise, the next step searches in the direction of the point having the lowest SAD. At each step, the results of the search in the previous two steps are considered. Letting  $\text{SAD}(i)$ ,  $\text{SAD}(i-1)$ ,  $\text{SAD}(i-2)$  denote respectively the results of the current, previous, and next previous steps. If  $\text{SAD}(i)$  is the smallest of the three, the next step proceeds along the direction pointed to by point  $(i)$  from point  $(i-1)$ . If, however,  $\text{SAD}(i-1)$  is the smallest, the next step proceeds along the direction *orthogonal* to the line connecting points  $(i)$  and  $(i-1)$ . The search concludes after a predefined number of steps, or if the center of a star-shaped pattern has the least SAD.

## 2.4 Logarithmic search (LOG)

Logarithmic search [5] is another widely-used algorithm, which has similarities to 3SS. The initial search is conducted on a diamond shaped grid with pixel spacing equal to 4. Each subsequent stage repeats the search around the previous stage's best point, decreasing the grid radius by a factor of 2 if the center point has the best result. An example is shown in Figure 4.

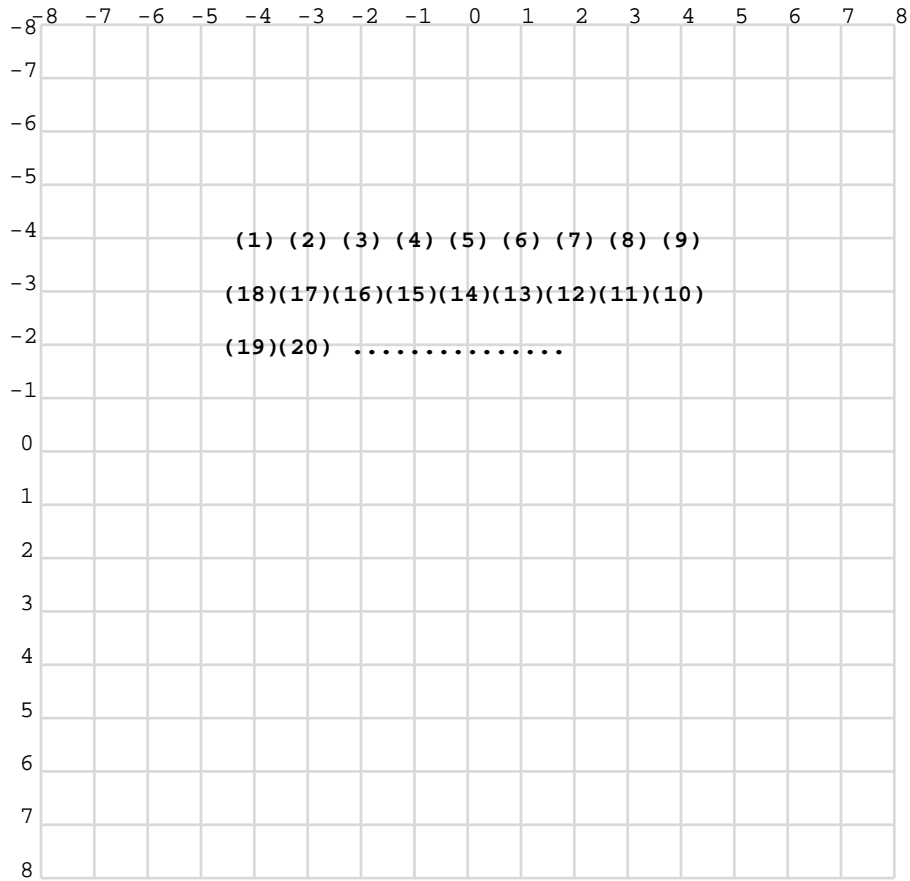


Figure 2: An example of a search pattern for PROJ is shown. Note that this is basically full search, with the order of search being arranged to maximize reuse of previously-computed projection data. For simplicity the pattern starts at  $(-4,4)$ , instead of  $(-8,8)$ .

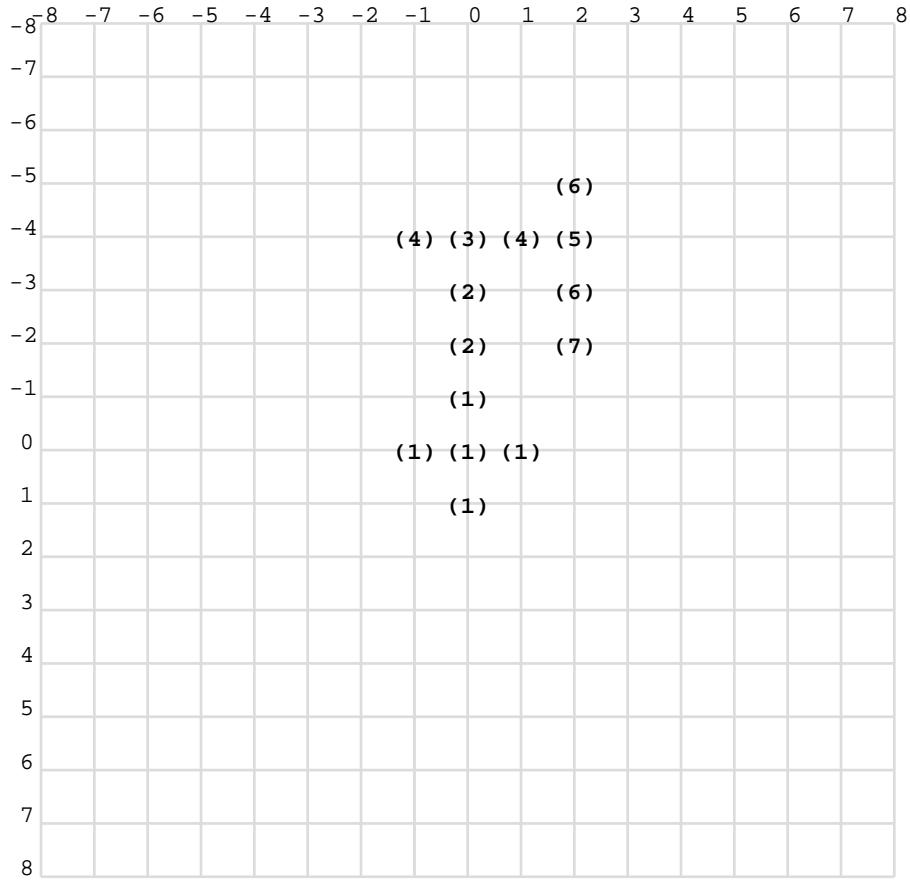


Figure 3: An example of a search pattern for CDS is shown. The points marked (1), spaced 1 pixel apart, are searched in the first step. The next step, (2), searches in the direction of the best point from the first step. At each step, the direction of the best point from the previous step is used, or a switch is made to the orthogonal direction. Note that when the orthogonal direction is chosen, two points are searched to determine the sign of next displacement step.

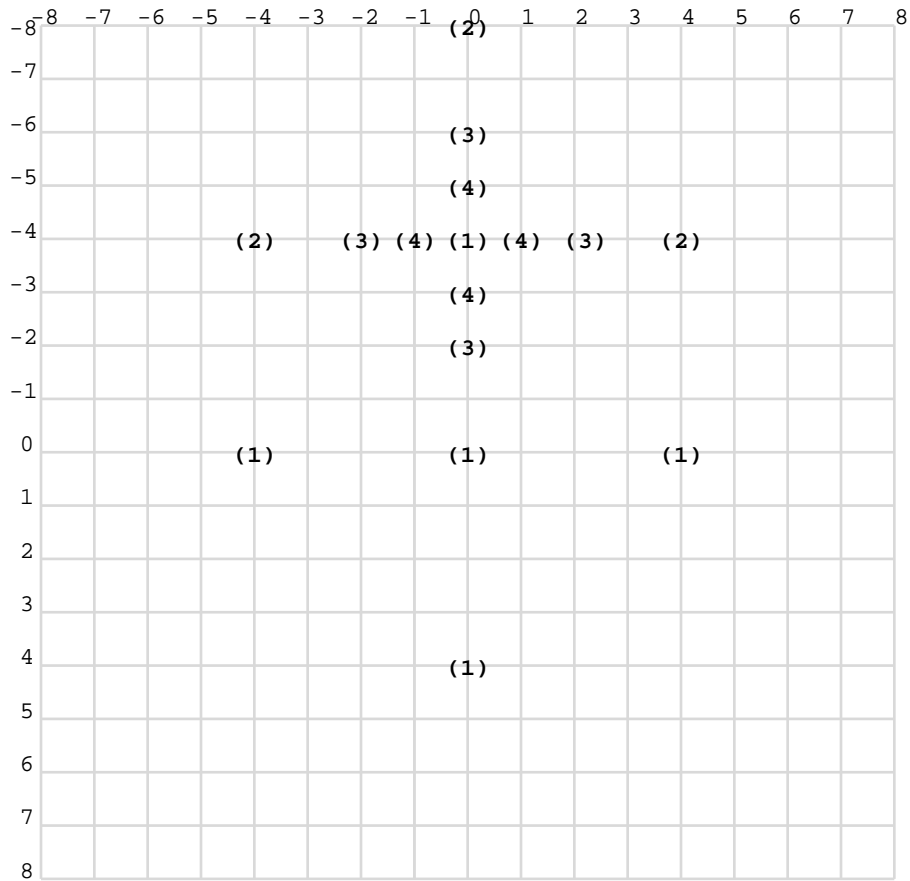


Figure 4: An example of a search pattern for LOG is shown. The points marked (1), spaced 4 pixels apart, are searched in the first step. Each subsequent stage repeats a diamond-shaped search around the previous stage's best point, decreasing the search radius if the best point is in the center.

### 3 Subpixel motion estimation

The algorithms previously described carry out pixel-level motion estimation. Subpixel motion estimation provides significant improvement in compression. Subpixel estimation requires that the  $16 \times 16$  macroblock from the previous image have interpolated values, since its pixels lie in between existing pixel values. For half-pixel interpolation, the MPEG-4 Visual Standard requires that the decoder use the following formulae. Consider the four integer-coordinate pixels  $A$ ,  $B$ ,  $C$ ,  $D$ , and half-pixel coordinate pixels  $a$ ,  $b$ , and  $c$ , shown in the array below.

$$\begin{array}{ccc} A & a & B \\ b & c & \\ C & & D \end{array}$$

With “/” denoting integer division (discard fractional portion), the formulae are:

$$\begin{aligned} a &= (A + B + 1)/2, \\ b &= (A + C + 1)/2, \\ c &= (A + B + C + D + 2)/4. \end{aligned}$$

Motion vectors are coded in MPEG-4 to half-pixel resolution.

### 4 Motion compensation

Once the motion vectors have been computed, the next step is determine the residual after subtracting the best matching macroblock from the previous frame with the macroblock in the current frame. This is the process of motion compensation. The residual is what is coded, along with the motion vector. Motion compensation must be carried out for both luminance and chrominance values. Since MPEG-4 uses a  $4 : 2 : 0$  sampling structure, with one  $8 \times 8$   $Cb$  and  $Cr$  block for each  $16 \times 16$  luminance block. For a chrominance block, motion vectors are divided by 2, and their fractional parts are rounded up to the nearest half-pixel value.

MPEG-4 allows a motion vector to point outside of the frame boundary. This capability is called *unrestricted motion compensation* (UMC). In that case, the value of a pixel outside the boundary is chosen to be equal to the value of the nearest pixel inside the boundary. Suppose that the frame boundaries are  $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ . If  $P[x, y]$  denotes the reference image used for motion compensation, the value used for any motion vector  $(m_x, m_y)$  is

$$P[\text{clip}(x + m_x, x_{\min}, x_{\max}), \text{clip}(y + m_y, y_{\min}, y_{\max})]$$

The “clip” function is defined as follows:

$$\text{clip}(x, x_{\min}, x_{\max}) = \begin{cases} x & x_{\min} < x < x_{\max} \\ x_{\min} & x \leq x_{\min} \\ x_{\max} & x \geq x_{\max} \end{cases}$$

### 5 Conclusions

Four well-known motion estimation algorithms are described.

## References

- [1] P. Kuhn, *Algorithms, complexity analysis and VLSI architectures for MPEG-4 motion estimation*, Boston: Kluwer Academic Publishers, 1999.
- [2] T. Koga, I. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," *Proceedings of the IEEE 1981 National Telecommunications Conference*, Part G, pp 5.3.1-G.5.3.5.
- [3] S. B. Pan, S. S. Chae, R.-H. Park, "VLSI architectures for block matching algorithms using systolic arrays," *IEEE Transactions on circuits and systems for video technology*, Vol. 6, No. 1, Feb. 1996, pp 67-73.
- [4] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Transactions on communications*, Vol. COM-33, No. 8, August 1985, pp 888-896.
- [5] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions on communications*, Vol. COM-29, Dec. 1981, pp 1709-1808.