LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-CONF-221486

An Experimental Comparison of Block Matching Techniques for Detection of Moving Objects

N. S Love, C. Kamath

May 18, 2006

SPIE Applications of Digital Image Processing XXIX
San Diego, CA, United States
August 13, 2006 through August 17, 2006

An experimental comparison of block matching techniques for detection of moving objects

Nicole S. Love and Chandrika Kamath

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
7000 East Ave, Livermore, CA 94550

ABSTRACT

The detection of moving objects in complex scenes is the basis of many applications in surveillance, event detection, and tracking. Complex scenes are difficult to analyze due to camera noise and lighting conditions. Currently, moving objects are detected primarily using background subtraction algorithms, with block matching techniques as an alternative. In this paper, we complement our earlier work on the comparison of background subtraction methods by performing a similar study of block matching techniques. Block matching techniques first divide a frame of a video into blocks and then determine where each block has moved from in the preceding frame. These techniques are composed of three main components: block determination, which specifies the blocks; search methods, which specify where to look for a match; and, the matching criteria, which determine when a good match has been found. In our study, we compare various options for each component using publicly available video sequences of a traffic intersection taken under different traffic and weather conditions. Our results indicate that a simple block determination approach is significantly faster with minimum performance reduction, the three step search method detects more moving objects, and the mean-squared-difference matching criteria provides the best performance overall.

Keywords: block matching, moving object detection

1. INTRODUCTION

Detection of moving objects is difficult due to camera noise, lighting conditions, object orientation and size. Detection is primarily done by first preprocessing the frame to reduce noise and the effect of different lighting conditions, followed by background subtraction. Background subtraction is the process of subtracting a frame that models the background from the current frame. The simplest case of background subtraction is frame differencing, where the background model is the previous frame. After background subtraction, pixels in the resulting frame produce a foreground mask of moving pixels. These pixels are then combined to produce moving objects.

Block matching offers an alternative to background subtraction for object detection. In block matching, blocks in the current frame are matched to blocks in a reference frame (an earlier frame). For each block in the current frame, the reference frame is searched for the best matching block. A matching criteria determines the best match from candidate blocks in the reference frame. If the matched block is not in the same location in the reference frame as in the current frame, the block has moved. A foreground mask of the moving blocks is then generated. Blocks with the same motion can be combined to form moving objects. Block matching adds the additional information of block motion, making block matching attractive for tracking applications. Block matching is extensively used in compression[1] to detect motion. In this paper, we present a detailed evaluation of the various implementations of block matching for use in detecting moving objects.

Block matching techniques can be divided into three components: block determination, searching method, and matching criteria. For each component, a comparison of several options is performed. We begin with Section 2 describing the block matching components studied. Section 3 follows with details on the data sequences evaluated and the results of the metrics used. We end with conclusions drawn from the empirical study in Section 4.

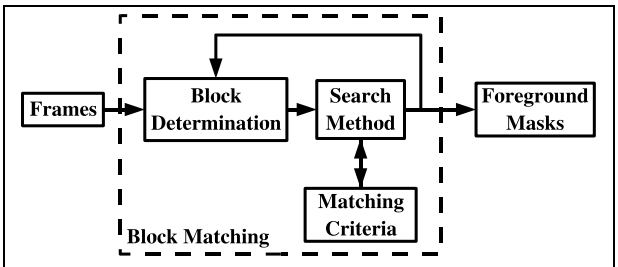

Figure 1. Block Matching Flowchart

2. BLOCK MATCHING TECHNIQUES

Block matching techniques can be divided into three main components as shown in Figure 1: block determination, search method, and matching criteria. The implementation of block matching using components allows for flexibility; interchanging components produces a large variety of block matching techniques and based on the application, components which provide the best results can be chosen with ease.

2.1. Block Determination Approaches

Block matching techniques begin by determining the location, the size, and the scale of blocks, as well as the start location of the search. We examine both a simple and a hierarchical approach. Variable-size approaches are also used in compression algorithms.[2–4] In a variable-size approach the block sizes are not fixed and vary in size based on a homogeneity metric (test for similarity). The variable-size approaches are designed primarily to reduce bit-rate by reducing the number of motion vectors which represent a frame. Since our focus is moving object detection and speed of processing, not compression, we did not consider a variable-size approach.

In the **simple block determination** approach, each block is a portion of the frame at a fixed size. All blocks are disjoint, and there is no change in the scale of a block. A block size which does not encompass an entire object, but allows for several blocks to form an object is chosen. The search is started at the same block location in the reference frame.

In the case of **hierarchical block determination**, a multiresolution approach is used to determine the block location, size, and scale. First, a Gaussian pyramid[5] of the current and the reference frame is constructed. At each level of the pyramid, the frame is divided into disjoint fixed size blocks as in the simple block determination approach. The search begins at the same location of the block in the lowest resolution reference frame; at each subsequent level the search starts at the best match from the previous level.

2.2. Search Method

The search method determines candidate matching blocks in the reference frame. We examine 4 search methods; a window search, three-step search,[6] 2D-logarithmic search,[7] and cross search.[8] Each search method searches the reference frame at a given step size. The step size is the number of pixels from the center candidate block to the other candidate blocks based on the search pattern.

Although we focus our evaluation on search methods which reduce the number of candidate blocks, there are also search methods that reduce the processing within a block. There are two common methods, one uses only a sample of pixels in a block to determine a match,[9,10] the other uses partial sums to speed up processing of a block.[11,12] These methods concentrate on reducing processing time with a slight reduction in accuracy and can be incorporated into any search method.

- In the **window search**, a window in the reference frame is searched at a given step size. The size of the window is dependent upon the motion of the objects in the frame. The window size is chosen to be slightly larger than the maximum possible motion of the objects. This reduces the search area based on the application. A step size equal to one pixel is a full search of the window, finding the optimal motion vectors. A larger step size increases the speed by reducing the number of candidate blocks, but increases motion vector error.

- The **three-step search (TSS)** begins with eight candidate neighbor blocks a given step size away. The search is a recursive process with each iteration centered at the best match from the previous iteration and the step size, s, is halved, until a step size of one is reached. The three-step search reduces the number of candidate blocks and covers a large area, making it a fast search technique. The approach concentrates on directing the search based on the best match from the previous step.

- The **cross search** is similar to TSS, except the candidate blocks are limited to four neighbors (cross pattern, $\times$) in each iteration rather than eight as in the case of TSS. The search is faster than TSS due to the further reduction in candidate blocks. The last step in the search is a cross ($\times$) or plus ($+$) pattern depending on the location of the best match so far.

- The **2D-logarithmic search** has four neighbors (plus pattern, $+$) in each iteration. The step size is only reduced when the center candidate block of the previous iteration is the best match. The last step in the search is an eight neighbor pattern.

  With the last eight neighboring blocks and the potential for additional iterations (step size not reduced at each iteration), the 2D-logarithmic search is slower than the cross search but covers more area. In order to limit the search area the 2D-logarithmic search can be restricted to a search window reducing the number of possible candidate blocks.

The three-step, 2D-logarithmic, and the cross search are designed for speed to reduce the number of candidate blocks. However, unlike the window search, there is a potential to be trapped in a local minima in these searches. The window search with a step size of one is a full search trying all possible candidate blocks.

2.3. Matching Criterion

Given an $n \times n$ block, a matching criteria, $M(p,q)$, measures the dissimilarity of a block in the current frame, I_c at (i,j), and a block in the reference frame, I_r, shifted by (p,q). These criteria can be characterized by $M(p,q) = \sum_i^{n+i-1} \sum_j^{n+j-1} \phi(e)$, where $\phi(e)$ is the criteria function and $e = I_c(i,j) - I_r(i+p, j+q)$. We examine four matching criteria (also known as error or matching functions): the sum of the absolute values of the differences **(SAD)**, the mean of the absolute values of the differences **(MAD)**, the mean of the square of the differences **(MSD)**, and the sum of the non-matching pixels in the two blocks **(MPC)**, where a match is determined by the absolute value of the difference being less than a threshold, t_{MPC}.

SAD and MAD only differ by a constant in the case of fixed size blocks and can be used interchangeably in our comparison. Practically, SAD is faster due to the removal of the divide operation. While MAD incorporates large differences, MSD penalizes more for large differences. MPC on the other hand equally weights any difference above a threshold.

2.4. Zero-Motion Bias

To reduce the effect of noise, we incorporate zero-motion biasing into all block matching techniques. All searches begin with a check for zero-motion, that is, the current block is compared with the block at the same location in the reference frame. If zero-motion is a "good" match (within a threshold), the search is terminated, resulting in a motion vector with no motion, $\vec{m}(i,j) = (0,0)$. Otherwise one of the search methods is used. As shown in Figure 2, zero-motion biasing reduces false motion; it also reduces the processing time by eliminating searches. In our work, the choice of zero-motion bias threshold is determined empirically based on the application.

3. EXPERIMENTS

Our focus is on the comparison of block matching techniques for moving object detection. The experiments consist of four image sequences of traffic. We examine the detection of moving objects and processing speeds for a variety of block matching components.

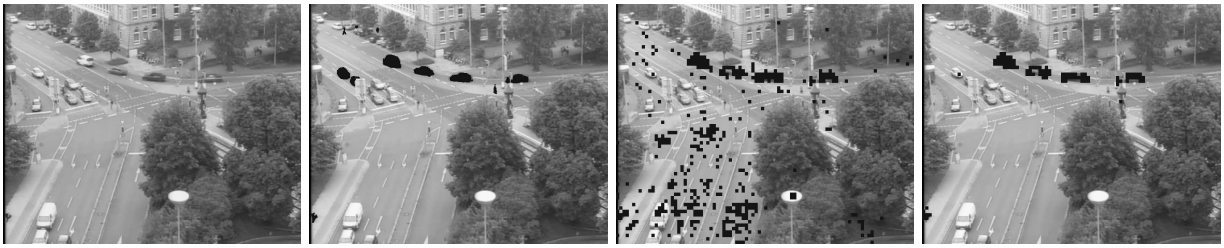

Figure 2. Example of reduced noise and false motion using zero-motion biasing. From left to right the frames are, original (left); manually highlighted ground truth (center left); highlighted motion blocks from block matching (center right); and the same block matching technique with zero-motion biasing (right). The simple block determination approach is applied with 8×8 block size, the window search method with a step size of one and 15×15 window size, and SAD matching criteria, with the zero-motion bias threshold of 250.0.

3.1. Data

The traffic sequences tested are publicly available from a website maintained by KOGS/IAKS Universitaet Karlsruhe.* Figure 3 shows a sample frame from each of the traffic sequences used. The sequences are of varying traffic and weather conditions:

- *Bright* (1500 frames): Bright daylight with "stop-and-go" traffic
- *Fog* (300 frames): Heavy fog with traffic patterns similar to the *Bright* sequence.
- *Snow* (300 frames): Snow with low to moderate traffic.
- *Busy* (300 frames): Busy intersection with vehicles and pedestrians and a large shadow from a building.

Bright, Fog, and *Snow* are sequences of the same intersection. The *Fog* and *Snow* sequences are converted from color to luminance only.

3.1.1. Ground truth

Ground truth measurements are accurate results used for comparison with the results from the block matching algorithms to evaluate system performance of these algorithms. We used both ground truth masks and object motion vectors as ground truth measurements for our experiments.

The ground truth masks are manually highlighted moving pixels (including shadows). The frames used to generate the ground truth masks are selected from the latter part of each sequence at regular intervals. Ten ground truth masks from each test sequence are created. Connected components in the ground truth masks are considered an object. There are a total of 81 objects in the ground truth masks of the *Bright* sequence, 73 objects in the *Fog* sequence, 83 objects in the *Snow* sequence, and 251 objects in the *Busy* sequence.

In order to accurately compare the block matching techniques, we use a "blocky" version of the ground truth. The frame is divided into the block sizes tested; if a block contains highlighted pixels from the ground truth mask the entire block is highlighted. A sample frame with the "blocky" version of the ground truth masks is shown in Figure 3.

Object motion vectors are obtained by manually finding the best match for an object (connected components). This process is extremely time consuming and therefore is only done for one frame per sequence. Object motion vectors are found from the first ground truth mask in each sequence. There are a total of 13 objects in the first ground truth mask of the *Bright* sequence, 11 objects in the *Fog* sequence, 11 objects in the *Snow* sequence, and 40 objects in the *Busy* sequence.

*All sequences are copyrighted by H.-H. Nagel of KOGS/IAKS Universitaet Karlsruhe. `http://i21www.ira.uka.de/image_sequences`

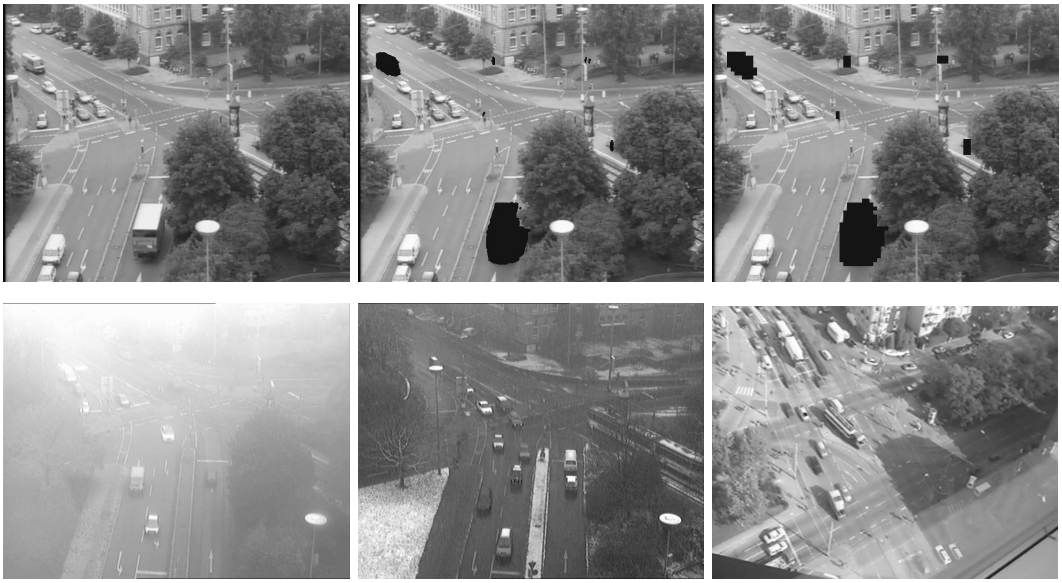

Figure 3. The first row contains frames from the *Bright* sequence: sample frame (left), manually highlighted ground truth frame (center), and the corresponding "blocky" (8×8 blocks) ground truth frame (right). The second row has a sample frame from the other sequences: *Fog* (left), *Snow* (center), and *Busy* (right).

3.2. Results

We begin with a comparison of the matching criteria, followed by the search methods, and, finally, the block determination approaches.

3.2.1. Evaluation of matching criteria

In order to compare the matching criteria, matching blocks were determined using the simple block determination approach with 8×8 blocks, and a window search with step size of 1 and a 15×15 window. This is the equivalent of a full search with zero-motion biasing. As a measure of performance, precision-recall curves were generated for each of the matching criteria by varying the zero-motion bias threshold. As MPC also has a threshold to determine pixel matches, we examine three values of the MPC threshold: 3, 5, and 10.

The precision, P, is a measure of the accuracy of the detection and the recall, R, is a measure of moving objects detected.

$$P = \frac{\text{\# of moving pixels correctly detected}}{\text{\# of moving pixels detected}} \quad \text{and} \quad R = \frac{\text{\# of moving pixels correctly detected}}{\text{\# of moving pixels in the ground truth}}.$$

Ideally, a precision and recall of one is the goal. Most often, an increase in recall leads to a decrease in precision, as more moving pixels are being detected along with a greater number of non-moving object pixels.

Figure 4 shows the results for each sequence. As the zero-motion bias threshold increases (right-to-left in Figure 4), the noise is reduced increasing precision until eventually (at the knee of the curve), moving object pixels are removed and the recall decreases. MSD gives the best overall results and MPC with $t_{MPC} = 10$ closely follows for each sequence. Recall is reduced by about 10% in the sequences when using SAD (or MAD) instead of MSD.

The average precision-recall curves over all four sequences with the corresponding average process timing is also shown in Figure 4. The average precision-recall curves are generated by averaging the results of all sequence for a given matching criterion and threshold. We observe that the MSD and MPC ($t_{MPC} = 10$) matching criteria have comparable results. The other matching criteria have a visible reduction in precision and recall at the knee of the curves. The corresponding process timing shows that MSD is faster than MPC with $t_{MPC} = 10$.

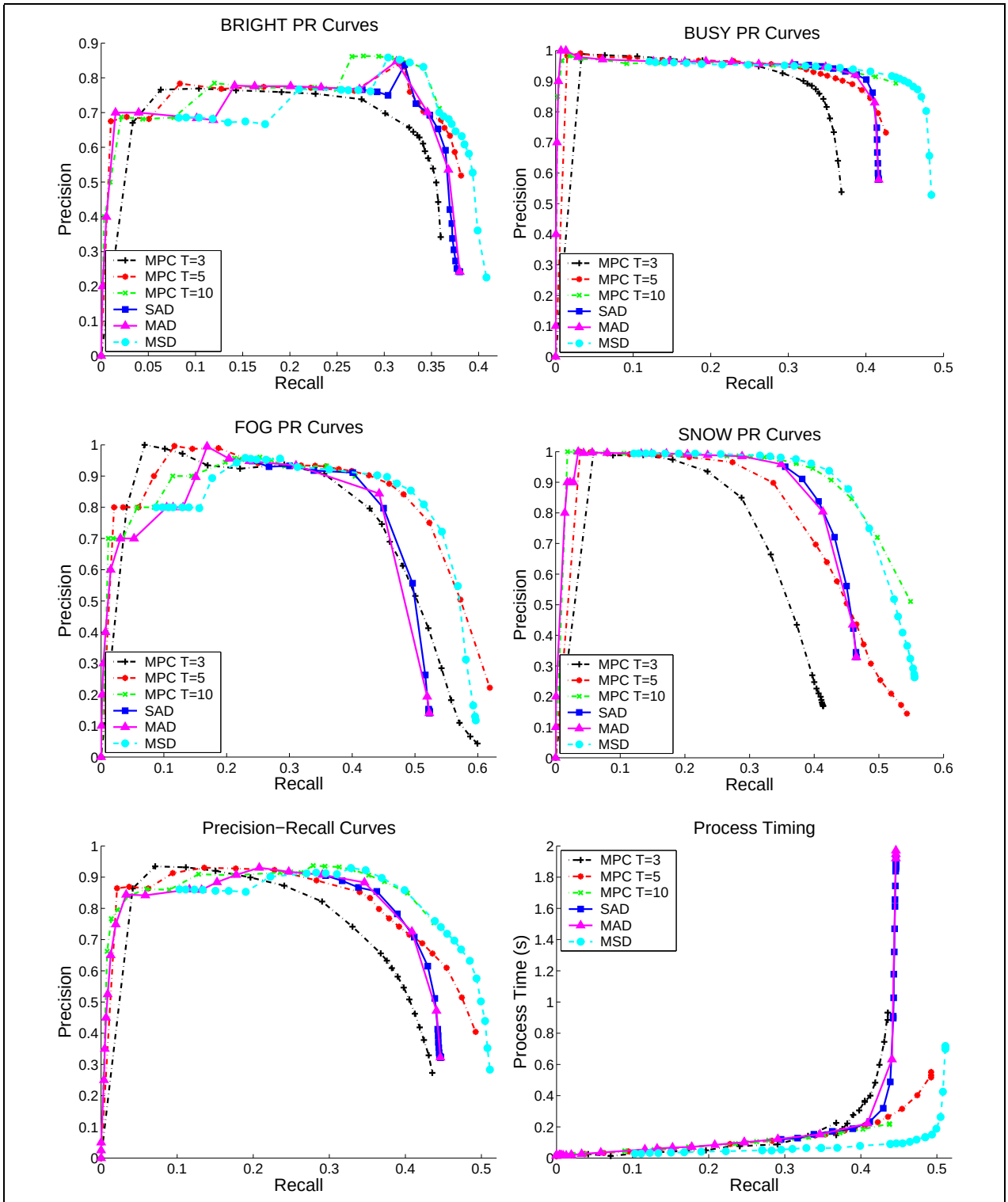

Figure 4. The first two rows are precision-recall curves of matching criteria with varying zero-motion bias threshold for the four video sequences. The third row has on the left, the average precision-recall curves of the four sequences for matching criteria with varying zero-motion bias threshold, and on the right is the corresponding average process timing over the frames and sequences for a given zero-motion bias threshold.

3.2.2. Evaluation of search methods

To compare search methods and block determination approaches, we use two metrics. These two metrics require the identification of an object, which is defined as a connected component in the ground truth frames. The first metric is a coherence metric, which measures the deviation of block motion within an object. The second metric is the average magnitude of the object motion vector error.

The search methods are compared using the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The TSS, 2D-logarithmic, and cross search methods begin with a step size of 4 and the window search has a step size of 1. The sequences have a maximum observed motion vector of 5.7 pixels. The motion vector coherence within an object is illustrated using error ellipses formed from the covariance matrix of an object.[13] A tight ellipse indicates that the motion vectors within an object are coherent. Although the error ellipses give insight into the coherence of an object, they may be co-occurring ellipses, and a simple visual inspection may not accurately indicate the overall performance of a search method. To address this, the percentage of objects detected with error ellipses that fit in a circle of radius ρ (an object motion vector error with standard deviation $\leq \rho$) are also examined. The higher the percentage, the better the method, with 100% indicating all objects detected.

We make the following observations for the performance measured using the coherence metric.

- *Bright:* In Figure 6, the cross method has tighter error ellipses, signifying greater coherence of the object motion vectors than the other search methods. But from Figure 7, we see the cross search has fewer objects detected overall than the TSS and the window search.
- *Fog:* In Figure 8, the cross method appears to have the tighter error ellipses, and the 2D-logarithmic search has elongated ellipses showing the tendency of the error in a particular direction. From Figure 9, the 2D-logarithmic search has a higher percentage of objects detected within a given error range. Figure 5 shows results from a region in a frame in the *Fog* sequence. We see in the 2D-logarithmic search a bias in the vertical direction.

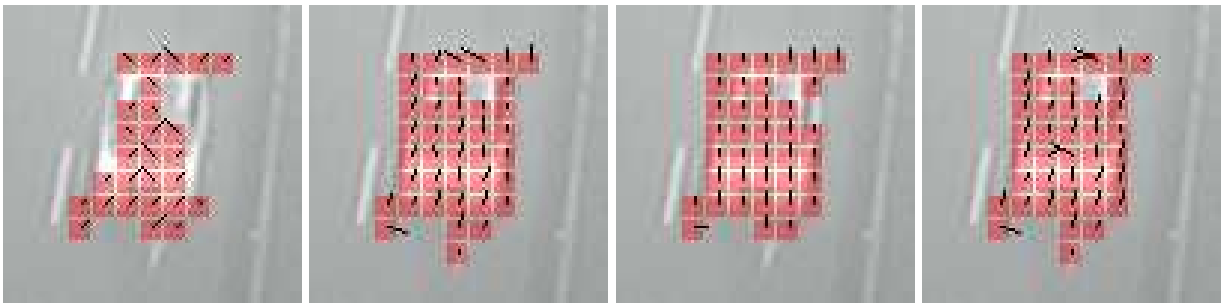

Figure 5. An example frame illustrating the motion vector blocks from the *Fog* sequence using the **cross search** (far left), **TSS search** (center left) methods, **2D-logarithmic search** (center right), and **window search** (far right) methods, the simple block determination approach with 8×8 blocks, the MSD matching criterion. The search methods have step size of 4, except the **window search** method which has step size of 1.

- *Snow:* In Figure 10, there is no significant difference between the search methods. Figure 11 confirms this observation.
- *Busy:* In Figure 12, the cross method has the tightest error ellipses, but the window search seems to have a denser core. In Figure 13, the window search and TSS have no significant difference, but there is a clear distinction from the 2D-logarithmic search and the cross search, which detect fewer objects.

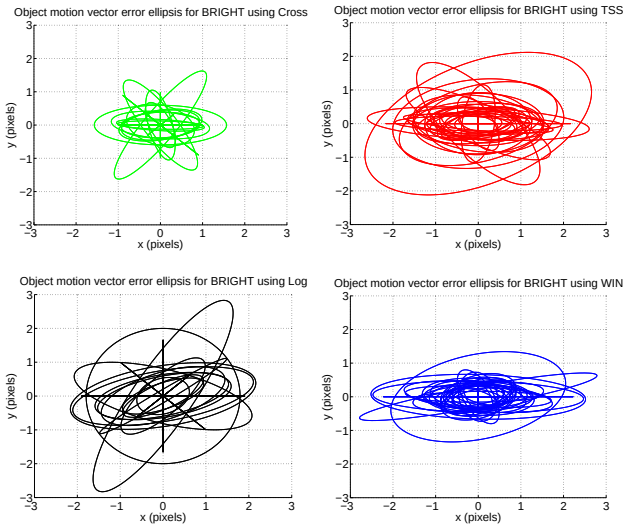

Figure 6. The plots show error ellipses of motion vectors within an object for the *Bright* sequence.

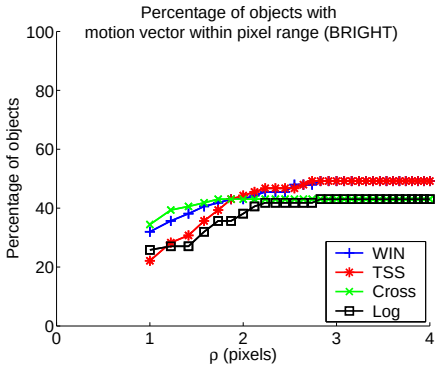

Figure 7. *Bright* sequence: percentage of objects that fit in an error circle of radius ρ for each search method.

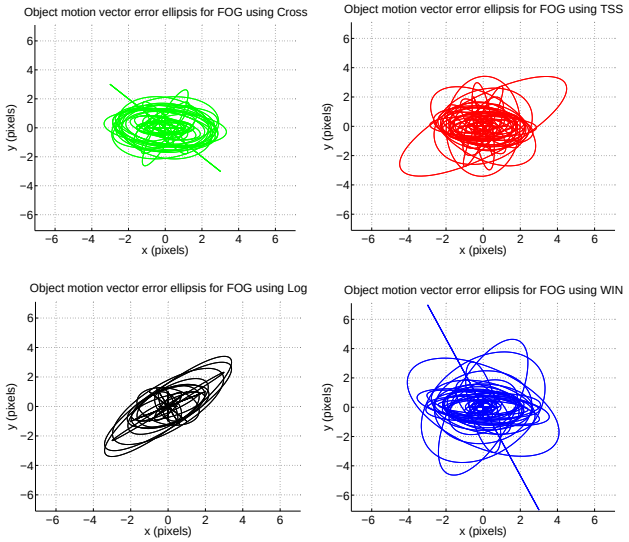

Figure 8. The plots show error ellipses of motion vectors within an object for the *Fog* sequence.

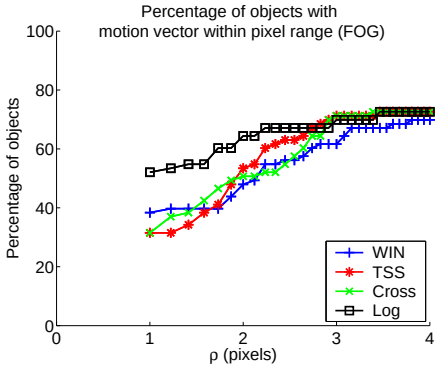

Figure 9. *Fog* sequence: percentage of objects that fit in an error circle of radius ρ for each search method.

The plots are generated from the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The searches have a step size of 4, except the window search which has a step size of 1.

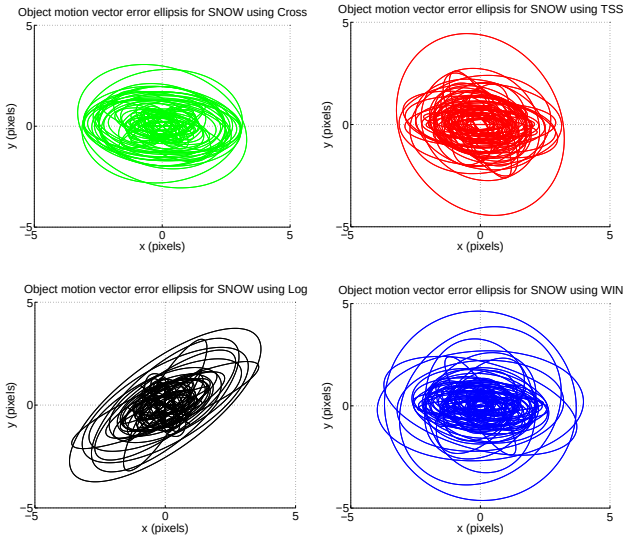

Figure 10. The plots show error ellipses of motion vectors within an object for the *Snow* sequence.

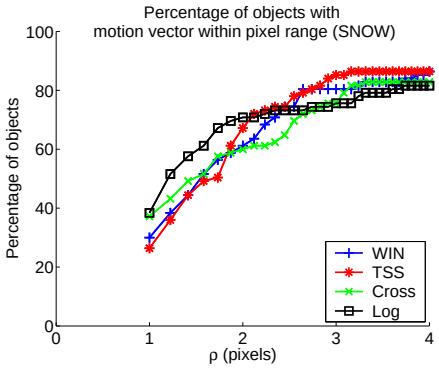

Figure 11. *Snow* sequence: percentage of objects that fit in an error circle of radius ρ for each search method.

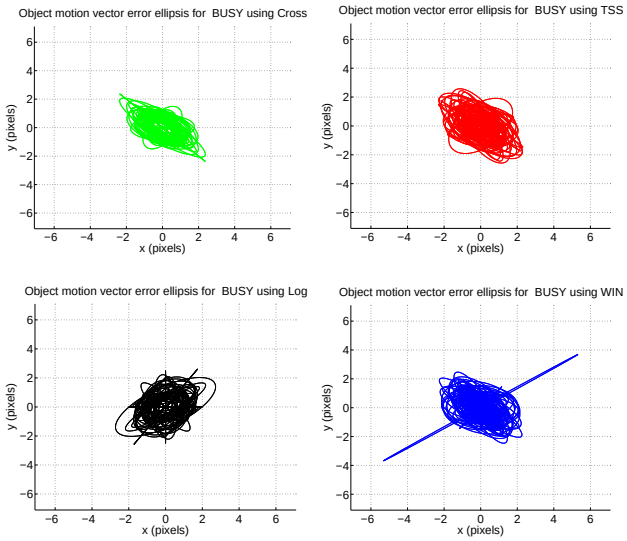

Figure 12. The plots show error ellipses of motion vectors within an object for the *Busy* sequence.

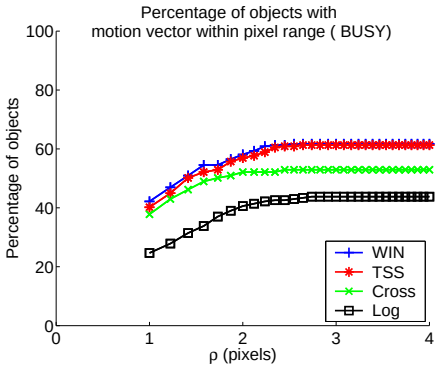

Figure 13. *Busy* sequence: percentage of objects that fit in an error circle of radius ρ for each search method.

The plots are generated from the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The searches have a step size of 4, except the window search which has a step size of 1.

$\|\hat{\vec{E}}\|$	Cross	TSS	Log	WIN
Bright	2.6	2.5	2.3	2.5
Fog	2.1	2.1	1.9	2.2
Snow	1.8	1.7	1.9	1.8
Busy	1.5	1.6	1.7	1.7

Table 1. Average magnitude of object motion vector error using the simple approach with 8×8 blocks, the MSD matching criterion. Cross, TSS, and Log search methods have a step size of 4 and WIN has a step size of 1.

Timing (ms)	Cross	TSS	Log	WIN
Bright	28	19	24	31
Fog	24	27	22	30
Snow	27	30	27	43
Busy	25	25	25	78

Table 2. Process timing of the simple block determination approach with 8×8 blocks, the MSD matching criterion. Cross, TSS, and Log search methods have a step size of 4 and WIN has a step size of 1.

The average magnitude of the object motion vector error, $|\hat{\vec{E}}|$, is used to evaluate the search methods and block determination approaches. Table 1 shows $|\hat{\vec{E}}|$ in pixels for each search method using the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The cross, TSS, and 2D-logarithmic searches have a step size of 4 and the window search has a step size of 1. The average magnitude of the object motion vector error is within 0.3 pixels of each other for all sequences, showing no significant difference in performance. The process timing of the search methods seen in Table 2 are also comparable for the three fast searches with the window search having a significantly higher process timing as expected.

3.2.3. Evaluation of block determination approaches

We compare two block determination approaches: a simple approach and a hierarchical approach. These approaches are evaluated using the coherence metric and average magnitude of object motion vector error. Figure 14 shows that the hierarchical approach clearly has tighter error ellipses in the *Fog* and *Snow* sequences. In Figure 15, we see that the hierarchical approach has an equal or larger percentage of objects over all pixel ranges. Figures 16-17 show an example frame from the *Fog* and *Snow* sequences using the two block determination methods. The hierarchical approach is seen as more coherent than the simple approach.

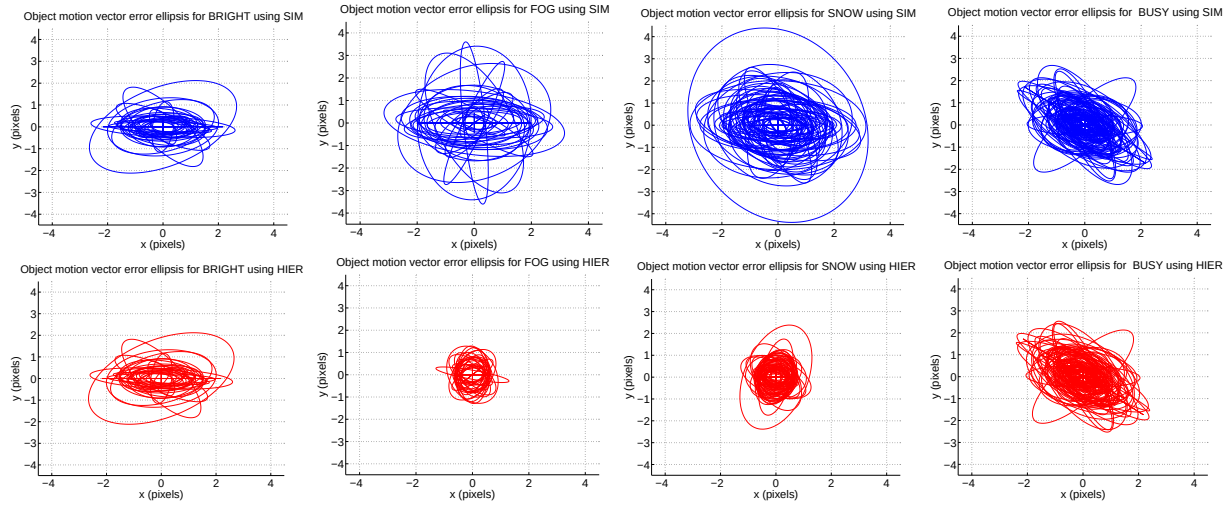

Figure 14. Each column is from a different sequence, column 1: *Bright*, column 2: *Fog*, column 3: *Snow*, and column 4: *Busy*. The plots show error ellipses of motion vectors within an object, row 1 uses the simple block determination approach and row 2 uses the hierarchical block determination approach, all plots use 8×8 blocks, the TSS search method, the MSD matching criterion, and step size of 4.

The hierarchical approach has a slightly lower average magnitude of object motion vector error than the simple block determination approach as seen in Table 3, but the processing time of the hierarchical approach

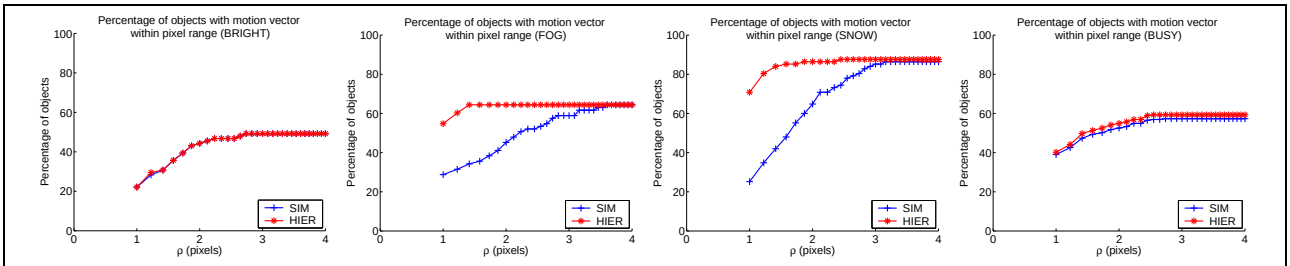

Figure 15. Percentage of objects that fit in an error circle of radius ρ for simple and hierarchical block determination approaches with 8×8 blocks, the TSS search method, the MSD matching criterion, and step size of 4.

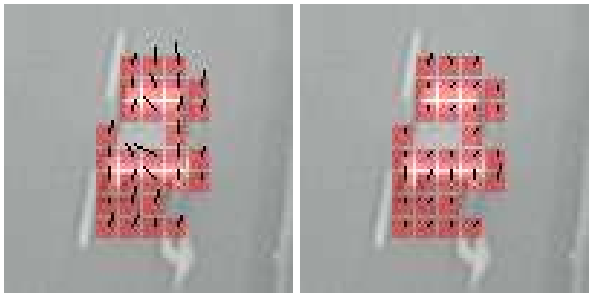

Figure 16. An example region of detected motion vector blocks from the *Fog* sequence using the **simple block determination** (left) and **hierarchical block determination** (right) approaches with the TSS search method, 8×8 blocks, the MSD matching criterion, and step size of 4.

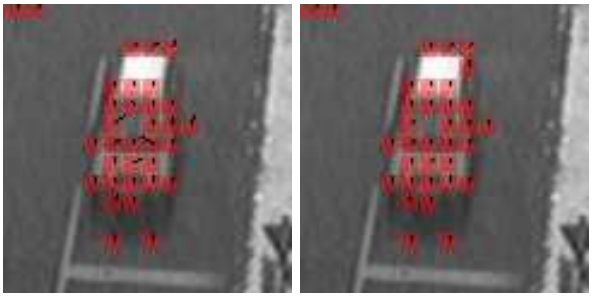

Figure 17. An example region of detected motion vector blocks from the *Snow* sequence using the **simple block determination** (left) and **hierarchical block determination** (right) approaches with the TSS search method, 8×8 blocks, the MSD matching criterion, and step size of 4.

is significantly higher seen in Table 4. From this study it is clear the improvement of coherent object motion vectors in the hierarchical approach is at the high cost of process timing.

4. CONCLUSIONS

We have presented an empirical study of a variety of options for block matching techniques with a focus on moving object detection. In videos illustrating several different traffic and weather conditions, the MSD matching criteria outperforms the other matching criteria for both moving object detection (precision-recall) and process speed using zero-motion biasing. For search methods, the methods are comparable (in coherence and average magnitude of object motion vector error metrics) with the 2D-logarithmic search performing the best in the *Fog* sequence and TSS has the best performance in the *Busy* sequence. Overall TSS detects more moving objects. Also, the simple block determination approach, though not as coherent as the hierarchical approach in the *Fog*

$\lvert\hat{\vec{E}}\rvert$	SIM	HIER
Bright	2.5	2.5
Fog	2.1	1.5
Snow	1.8	1.6
Busy	1.6	1.6

Table 3. Average magnitude of object motion vector error using the simple block determination and hierarchical approach with 8×8 blocks, the TSS search method, the MSD matching criterion, and step size of 4.

Timing	SIM	HIER
Bright	24ms	766ms
Fog	22ms	1.142s
Snow	8ms	1.138s
Busy	21ms	625ms

Table 4. Process timing of block matching techniques using 8×8 blocks, the TSS search method, the MSD matching criterion, and step size of 4.

and *Snow* sequences, has comparable results in the average magnitude of object motion vector error and 1-2 orders of magnitude improvement in processing speed.

ACKNOWLEDGMENTS

We would like to thank Erick Cantú-Paz for help in the initial software development of the block matching components and Sen-ching S. Cheung for creating the ground truth frames.

UCRL-CONF-221486: This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

REFERENCES

1. H. G. Musmann, P. Pirsch, and H.-J. Grallert, "Advances in picture coding," in *Proceedings of IEEE*, **73**, pp. 523–548, 1985.
2. M. H. Chan, Y. B. Yu, and A. G. Constantinides, "Variable size block matching motion compensation with applications to video coding," in *IEE*, **137**, August 1990.
3. K. Zhang, M. Bober, and J. Kittler, "Variable block size video coding with motion prediction and motion segmentation," in *SPIE*, **2419**, pp. 62–70, 1995.
4. I. Rhee, G. R. Martin, S. Muthukrishnan, and R. A. Packwood, "Quadtree-structured variable-size block-matching motion estimation with minimal error," *IEEE Transactions on Circuits and Systems for Video Technology* **10**, February 2000.
5. P. J. Burt and E. H. Adelson, "Laplacian pyramid as a compact image code," *IEEE Transactions on Communications* **31**, pp. 532–540, April 1983.
6. T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *National Telecommunications Conference*, pp. G5.3.1–5.3.5, 1981.
7. J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions on Communication* **COMM-29**(12), pp. 1799–1808, 1981.
8. M. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Transactions on Communication* **38**(7), pp. 950–953, 1990.
9. B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Transactions on Circuits and Systems for Video Technology* **3**, April 1993.
10. Y. Wang, Y. Wang, and H. Kuroda, "A globally adaptive pixel-decimation algorithm for block-motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology* **10**, September 2000.
11. C.-D. Bei and R. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," *IEEE Transactions on Communications* **33**, October 1985.
12. C.-K. Cheung and L.-M. Po, "Normalized partial distortion search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology* **10**, April 2000.
13. R. Mester and Hotter, "Robust displacement vector estimation including a statistical error analysis," in *5th International Conference on Image Processing and Its Applications*, pp. 168–172, July 1995.
14. M. Piccardi, "Background subtraction techniques: a review," *IEEE International Systems Man and Cybernetics* **4**, pp. 3099–3104, 2004.
15. S. S.-C. Cheung and C. Kamath, "Robust background subtraction with foreground validation for urban traffic video," *Eurasip Journal on Applied Signal Processing* **14**, pp. 2330–2340, 2005.
16. A. Gyaourova, C. Kamath, and S.-C. Cheung, "Block matching for object tracking," Tech. Rep. UCRL-TR-200271, Lawrence Livermore National Laboratory, 2003.
17. L.-K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Transactions on Circuits and Systems for Video Technology* **6**, pp. 419–422, August 1996.
18. P. D. Symes, *Video Compression Demystified*, McGraw Hill, 2000.
19. A. M. Tekalp, *Digital Video Processing*, Prentice-Hall, Upper Saddle River, NJ, 1995.
20. Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*, Prentice-Hall, 2001.
21. C. Manning, "Block matching algorithms for motion compensated video compression," Master's thesis, National University of Ireland, 1996.