LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-218038

An Empirical Study of Block Matching Techniques for the Detection of Moving Objects

N. S. Love, C. Kamath

January 9, 2006

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.

An Empirical Study of Block Matching Techniques for the Detection of Moving Objects

Nicole S. Love and Chandrika Kamath
{love21, kamath2}@llnl.gov
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
7000 East Ave, Livermore, CA 94550

Abstract

The basis of surveillance, event detection, and tracking applications is the detection of moving objects in complex scenes. Complex scenes are difficult to analyze because of camera noise and lighting conditions. Currently, moving objects are detected primarily using background subtraction. We analyze block matching as an alternative for detecting moving objects. Block matching has been extensively utilized in compression algorithms for motion estimation. Besides detection of moving objects, block matching also provides motion vectors (location of motion) which can aide in tracking objects.

Block matching techniques consist of three main components: block determination, search methods, and matching criteria. We compare various options for each of the components with moving object detection as the performance goal. Publicly available sequences of several different traffic and weather conditions are used to evaluate the techniques. A coherence metric and the average magnitude of object motion vector error are used to evaluate block determination approaches and search methods. To compare the matching criteria we use precision-recall curves to evaluate the performance of motion detection.

We present an empirical study of the block matching techniques using these metrics of performance as well as process timing. We found the hierarchical block determination approach has an overall higher coherence of object motion vectors than the simple block determination approach, but with a significant increase in process timing. The average magnitude of object motion vector for the search methods evaluated were comparable, with the cross search method having a better coherence of object motion vectors. Overall the three step search (TSS) detects more moving objects than the cross and 2D-logarithmic search methods. And the mean square difference (MSD) matching criterion has the best precision-recall as well as process timing when using zero motion biasing.

1 Introduction

Detection of moving objects is difficult due to camera noise, lighting conditions, object orientation and size. Detection is primarily done by 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 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 [14, 13] to detect motion. In this report, 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.

2 Block Matching Techniques

Block matching techniques match blocks from the current frame with blocks from a reference frame. The displacement in block location from the current frame to the location in the reference frame is the motion vector. Block matching techniques can be divided into three main components as shown in Figure 1: block determination, search method, and matching criteria.

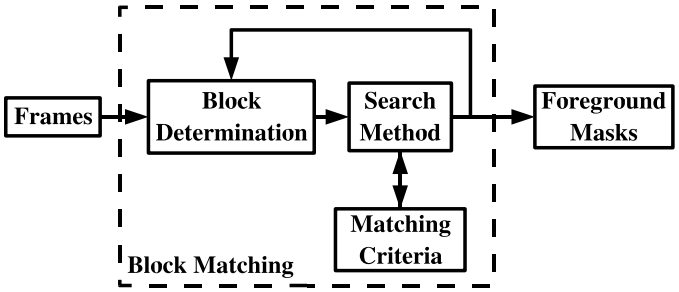


Figure 1: Block Matching Flowchart

The first component, block determination, specifies the position and size of blocks in the current frame, the start location of the search in the reference frame, and the scale of the blocks. We focus on fixed size, disjoint blocks spanning the frame, with initial start location at the corresponding location of the block in the reference frame. In tracking, a predictive method may be used to improve the start location of the search.

The search method is the second component, specifying where to look for candidate blocks in the reference frame. A fully exhaustive search consists of searching every possible candidate block in the reference frame. This search is computationally expensive and other search methods have been proposed to reduce the number of candidate blocks and/or reduce the processing for all candidate blocks. In this report, we concentrate on search methods that reduce the number of candidate blocks.

The third component is the matching criteria. The matching criteria is a similarity metric to determine the best match among the candidate blocks. In faster search methods, the best match so far will also determine the direction of the search (choice of next candidate blocks).

The motion vectors are fed to the block determination to implement multiresolution blocks. A coarse to fine resolution of the blocks is generated. The start location of the search at each resolution is the location of the best match (motion vector) from the previous coarser resolution.

The implementation of block matching using components allows for flexibility; interchanging components produces a large variety of block matching techniques. 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, 21, 16]. 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 implement a variable-size approach.

2.1.1 Simple Block Determination

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.

2.1.2 Hierarchical Block Determination

In this case, a multiresolution approach is used to determine the block location, size, and scale. First, a Gaussian pyramid 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 as shown in Figure 2.

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 [8], 2D-logarithmic search [7], and cross search [5]. Each

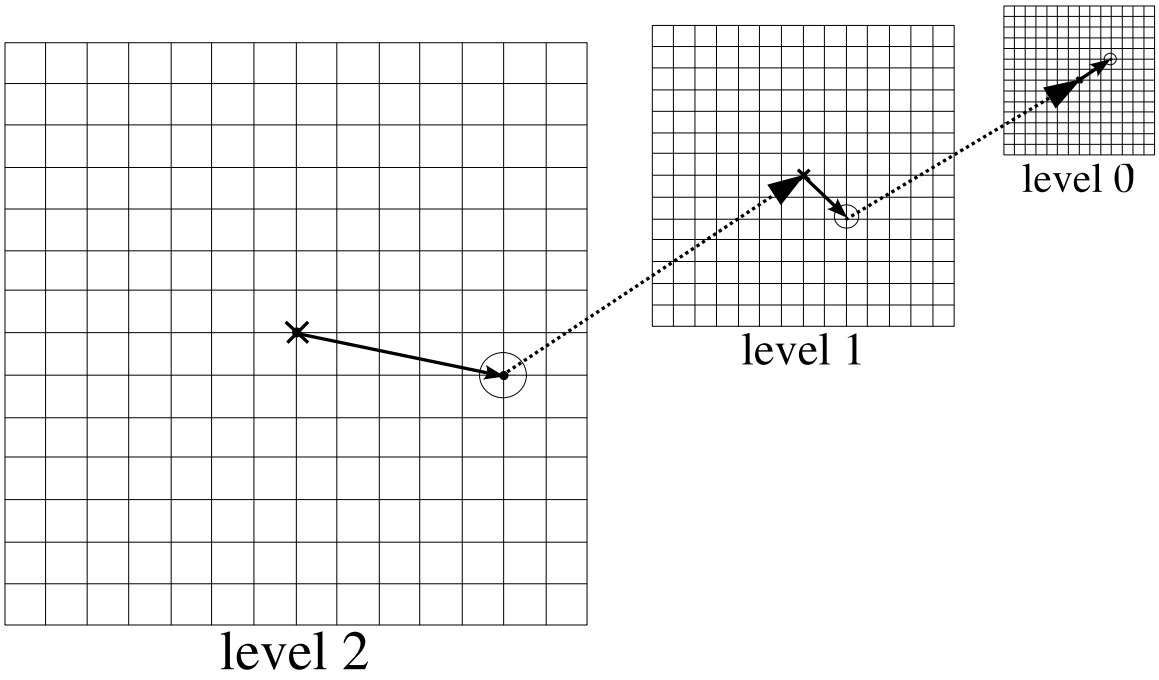


Figure 2: The three blocks are the search areas from three different resolutions. Level 0 is the resolution of the original frame, level 1 and level 2 are successively coarser resolutions. In hierarchical block determination, the $\times$ represents the start location of the search at a particular resolution. The circle is the location of the best match at a resolution. The dashed arrows represent the corresponding location from one resolution to the next. The search begins at level 2, the best match at level 2 becomes the starting location at level 1. This process continues until the best match is found at level 0.

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, 20], the other uses partial sums to speed up processing of a block [1, 3]. These methods concentrate on reducing processing time with a slight reduction in performance and can be incorporated into any search method.

2.2.1 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.

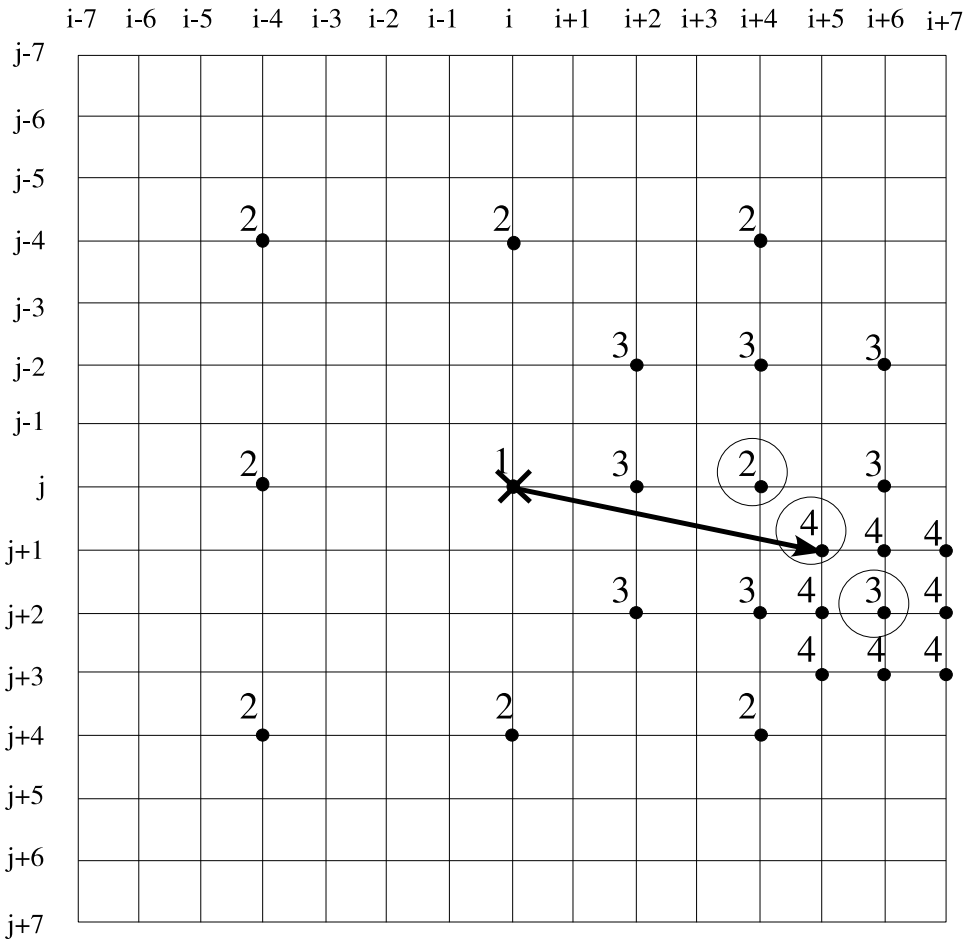


Figure 3: Example of TSS with step size = 4; each number represents the location of candidate blocks at a given iteration. The circles show the location of the best match at each iteration. And the vector is the resulting motion vector. The point at (i, j) indicated by 1 is the location of the block being considered and the location of the first candidate block in the reference frame (zero-motion candidate block). The eight points a step size of four away marked with 2 are the location of candidate blocks for the second iteration. The location of the best match from these candidate blocks is circled, $(i + 4, j)$. The eight neighbors of $(i + 4, j)$ a step size of two away are the next set of candidate blocks indicated by 3, with the location of the best match circled at $(i + 6, j + 2)$. The final iteration has eight neighbors marked with a 4. The resulting best match location is signified with a circle and motion vector from the location of the block being considered to the location of the best match.

2.2.2 Three Step Search - TSS

The three step search 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 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 three step search covers an area of $(2^{(s+1)}-1)^2$ with $8(\log_2 s+1)+1$ candidate blocks. The approach concentrates on directing the search based on the best match from the previous step. With this scheme, however, there is the potential to be trapped in a local minima. An example of the search is given in Figure 3 with circles representing the best match at each iteration. A description of the three step search follows:

Step 1: Set the motion vector of block (i,j) in the current frame, to zero-motion, $\vec{m}(i,j) = (0,0)$ and set the best match value, v_b, to the dissimilarity value of the block (i,j) in the current frame and the block (i,j) in the reference frame, $v_b = M(0,0)$. If v_b is less than the zero-motion bias threshold (defined in Section 2.4), the search stops. Otherwise go to Step 2.

Step 2: The best match is the minimum of v_b (the current best match value) and the dissimilarity values of the eight neighboring blocks a step size, s, away centered at $(i,j)+\vec{m}(i,j)$ (current best match location) in the reference frame. v_b is set to the best match value and $\vec{m}(i,j)$ is set to the corresponding motion vector.

Step 3: If $s > 1$, then halve the step size (set $s = \lceil s/2 \rceil$) and go to Step 2. Otherwise return $\vec{m}(i,j)$.

2.2.3 Cross Search

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 of candidate blocks. The results are even more likely to be a local minima. The steps of the cross search are:

Step 1: Set the motion vector of block (i,j) in the current frame, to zero-motion, $\vec{m}(i,j) = (0,0)$ and set the best match value, v_b, to the dissimilarity value of the block (i,j) in the current frame and the block (i,j) in the reference frame, $v_b = M(0,0)$. If v_b is less than the zero-motion bias threshold (defined in Section 2.4), the search stops. Otherwise go to Step 2.

Step 2: The best match is the minimum of v_b (the current best match value) and the dissimilarity values of the four neighboring blocks in a cross ($\times$) pattern a step size, s, away centered at $(i,j)+\vec{m}(i,j)$ (current best match location) in the reference frame. v_b is set to the best match value and $\vec{m}(i,j)$ is set to the corresponding motion vector.

Step 3: Halve the step size (set $s = \lceil s/2 \rceil$). If $s > 1$, then go to Step 2. Otherwise go to Step 4.

Step 4: If the best match is at the upper right or lower left corner, go to step 5. Otherwise go to Step 6.

Step 5: Find the best match among v_b (the current best match value) and the dissimilarity values of the four neighboring blocks in a plus (+) pattern a step size, s, away centered at $(i,j)+\vec{m}(i,j)$ (current best match location) in the reference frame. v_b is set to the best match value and $\vec{m}(i,j)$ is set to the corresponding motion vector.

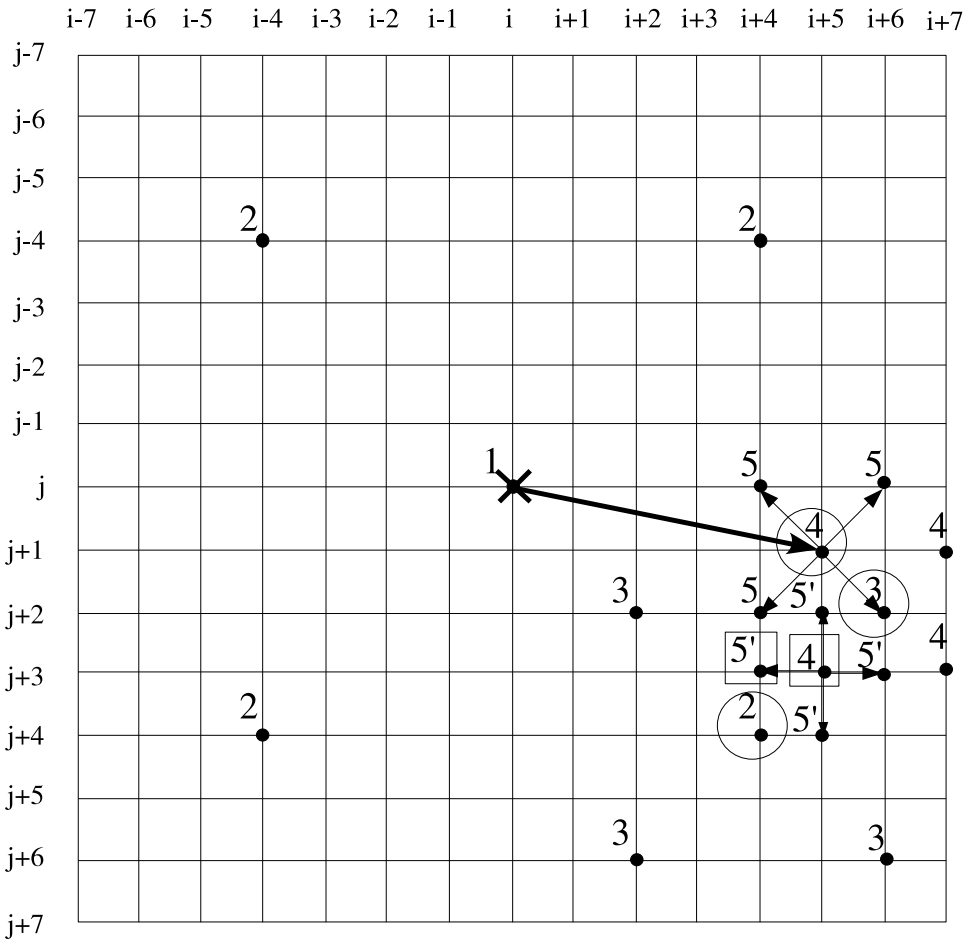


Figure 4: Example of cross search with step size = 4; each number represents the location of candidate blocks at a given iteration. The circles show the location of the best match at each iteration. And the vector is the resulting motion vector. The point at (i, j) indicated by 1 is the location of the block being considered and the location of the first candidate block in the reference frame (zero-motion candidate block). The four points a step size of four away marked with 2 are the location of candidate blocks for the second iteration. The location of the best match from these candidate blocks is circled, $(i + 4, j + 4)$. The four neighbors of $(i + 4, j + 4)$ a step size of two away are the next set of candidate blocks indicated by 3, with the location of the best match circled at $(i + 6, j + 2)$. The 4^{th} iteration has four neighbors marked with a 4. Followed by a final iteration, the resulting best match location is signified with a circle and motion vector from the location of the block being considered to the location of the best match. At the 4^{th} iteration, the squares are an alternative best match leading to the 5' search pattern (+).

Step 6: The best match is the minimum of v_b (the current best match value) and the dissimilarity values of the four neighboring blocks in a cross ($\times$) pattern a step size, s, away centered at $(i,j)+\vec{m}(i,j)$ (current best match location) in the reference frame. v_b is set to the best match value and $\vec{m}(i,j)$ is set to the corresponding motion vector.

The last step in the search is a cross ($\times$) or plus (+) pattern depending on the location of the best match so far. Figure 4 shows an example of the two cases, after the 4^{th} iteration the circle corresponds to a best match which leads to a cross ($\times$) pattern search and the square leads to a plus (+) pattern search.

2.2.4 2D-Logarithmic Search

The 2D-logarithmic search is similar to the cross search, except for the search pattern and when the step size is reduced. The search may be faster than TSS due to the possible reduction of candidate blocks. This search method can become trapped in a local minima as well. The steps for the 2D-logarithmic search follow:

Step 1: Set the motion vector of block (i,j) in the current frame, to zero-motion, $\vec{m}(i,j) = (0,0)$ and set the best match value, v_b, to the dissimilarity value of the block (i,j) in the current frame and the block (i,j) in the reference frame, $v_b = M(0,0)$. If v_b is less than the zero-motion bias threshold (defined in Section 2.4), the search stops. Otherwise go to Step 2.

Step 2: The best match is the minimum of v_b (the current best match value) and the dissimilarity values of the four neighboring blocks in a plus (+) pattern a step size, s, away centered at $(i,j)+\vec{m}(i,j)$ (current best match location) in the reference frame. If the center is not the best match, go to Step 3. Otherwise go to Step 4.

Step 3: v_b is set to the best match value and $\vec{m}(i,j)$ is set to the corresponding motion vector, go to Step 2.

Step 4: If $s > 1$, then halve the step size (set $s = \lceil s/2 \rceil$) go to Step 2. Otherwise go to Step 5.

Step 5: Find the best match of the among v_b (the current best match value) and the dissimilarity values of the eight neighboring blocks centered at $(i,j)+\vec{m}(i,j)$ (current best match location) in the reference frame. v_b is set to the best match value and $\vec{m}(i,j)$ is set to the corresponding motion vector.

Figure 5 shows an example of the 2D-logarithmic search. The step size is only reduced when the center candidate block of the previous iteration is the best match. With the last eight neighboring blocks and the potential for additional iterations, the 2D-logarithmic search is slower than the cross search but covers more area if not restricted. 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.

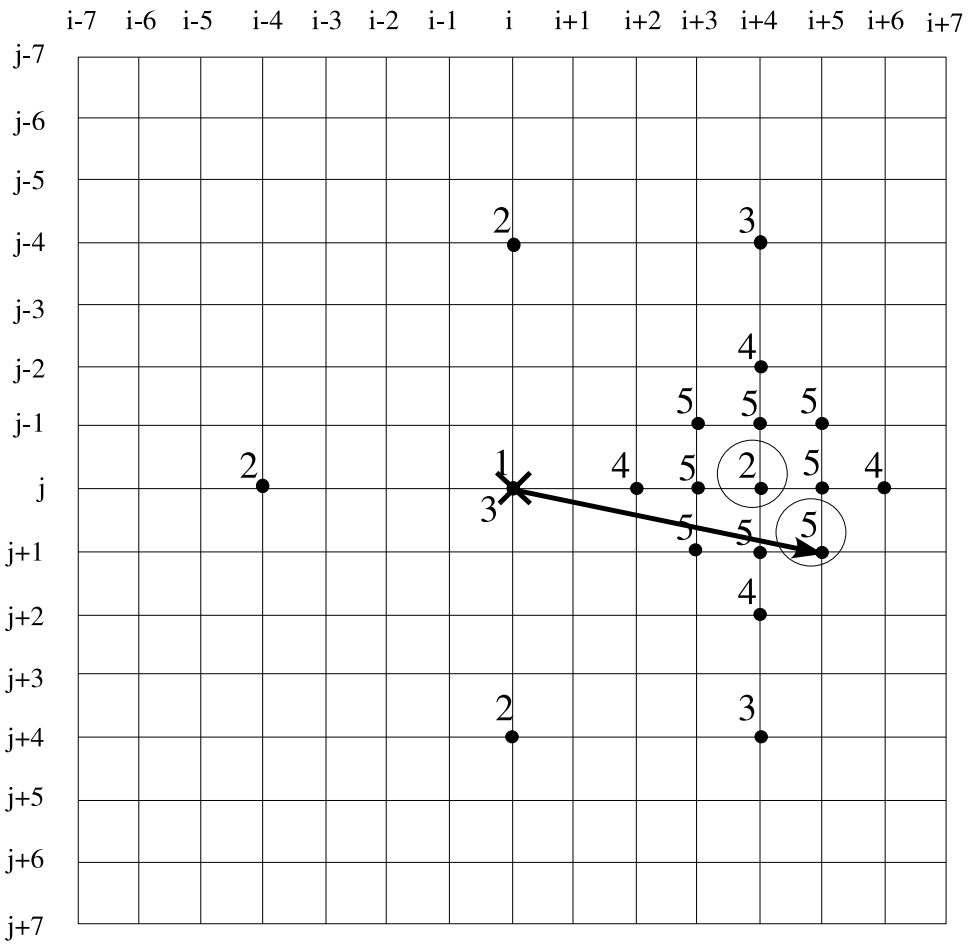


Figure 5: Example of 2D-logarithmic search with step size = 4; each number represents the location of candidate blocks at a given iteration. The circles show the location of the best match at each iteration. And the vector is the resulting motion vector. The point at (i, j) indicated by 1 is the location of the block being considered and the location of the first candidate block in the reference frame (zero-motion candidate block). The four points a step size of four away marked with 2 are the location of candidate blocks for the second iteration. The location of the best match from these candidate blocks is circled, $(i + 4, j)$. Since the location of the best match is not at the center of the current iteration, the step size is not reduced. The four neighbors of $(i + 4, j)$ a step size of four away are the next set of candidate blocks indicated by 3, (there are only three due to the search area limit). The location of the best match has not changed and is now the center of the candidate blocks, therefore the step size is reduced by a factor of 2. The 4^{th} iteration has four neighbors marked with a 4. The location of the best match is again $(i + 4, j)$. The final iteration consists of the eight neighboring points marked with 5. The resulting best match location is signified with a circle and motion vector from the location of the block being considered to the location of the best match.

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, and a block in the reference frame, I_r, shifted by (p, q). These criteria can be characterized by $M(p, q) = \sum_{i=1}^{n} \sum_{j=1} \phi(e)$, where $e = I_c(i, j) - I_r(i + p, j + q)$ and $\phi(e)$ is the criteria function. Figure 6 shows the criteria functions for a given e. We examine four matching criteria which are also known as error or matching functions.

SAD The sum of the absolute values of the differences in the two blocks:

$$M(p, q) = \sum_{i=1}^{n} \sum_{j=1}^{n} |I_c(i, j) - I_r(i + p, j + q)|$$

MAD The mean of the absolute values of the differences in the two blocks:

$$M(p, q) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} |I_c(i, j) - I_r(i + p, j + q)|$$

MSD The mean of the square of the differences in the two blocks:

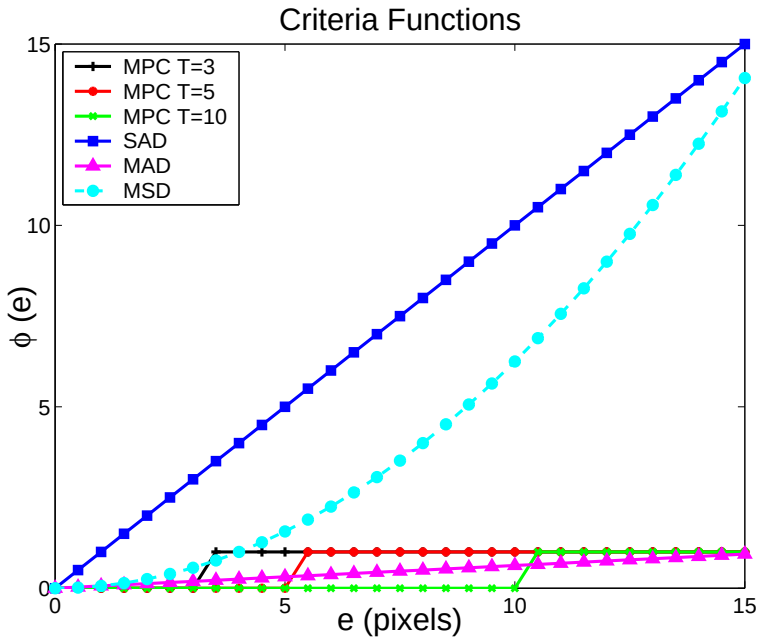
$$M(p, q) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (I_c(i, j) - I_r(i + p, j + q))^2$$

MPC The sum of the non-matching pixels in the two blocks, a match is determined by the absolute value of the difference being less than a threshold, t_{MPC}.

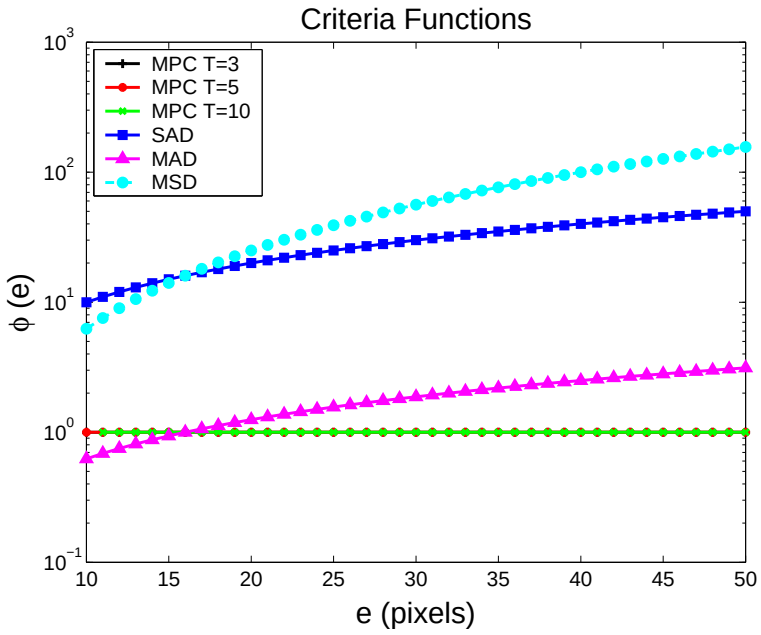
$$M(p, q) = \sum_{i=1}^{n} \sum_{j=1}^{n} D(I_c(i, j), I_r(i + p, j + q))$$

$$D(a, b) = \begin{array}{ll} 0 & \text{if } |a - b| \leq t_{MPC} \\ 1 & \text{otherwise} \end{array}$$

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 a block with one or more large differences. MPC on the other hand equally weights any difference above a threshold.

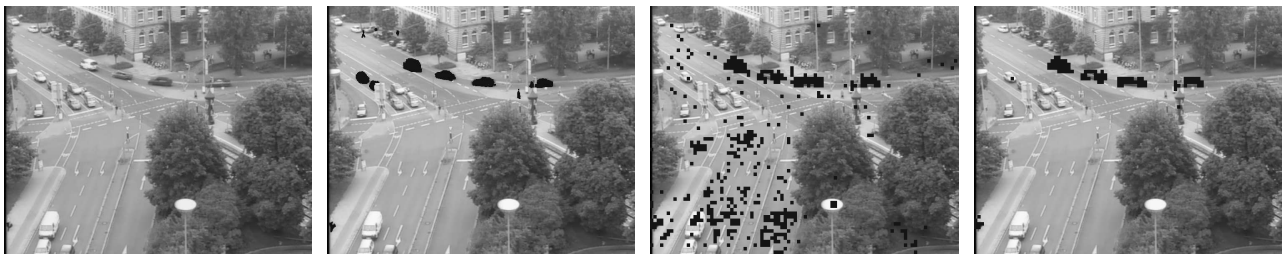


(a) Error range (0, 15)



(b) Error range (10, 50)

Figure 6: The matching criteria as functions of the difference of pixels, $M(p,q) = \sum_{i=1}^{n}\sum_{j=1}\phi(e)$. The plots are the various criteria functions $\phi(e)$ for an individual error, e. Plot (a) is for an error ranging from 0 to 15 pixels and plot (b) is for an error ranging from 10 to 50 pixels. *Note that the y-axis in plot(b) is in the log scale.*



(a) Original (b) Ground truth frame (c) No biasing (d) Zero-motion biasing

Figure 7: Example of reduced noise and false motion using zero-motion biasing. The original frame is shown in (a). The manually highlighted ground truth of the original frame is in (b). The frame in (c) contains highlighted motion blocks from block matching using the simple block determination approach with 8×8 block size, the window search method with a step size of one and 15×15 window size, and SAD matching criteria. The frame in (d) is the same block matching technique with zero-motion biasing, the zero-motion bias threshold is 250.0.

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 7, zero-motion biasing reduces false motion; it also reduces the processing time by eliminating searches.

The zero-motion bias threshold is dependent upon the matching criteria, overall motion in the frame and block size. From Figure 6, a given threshold permits different amounts of error per matching criteria. Camera motion requires a higher zero-motion bias threshold to remove blocks with background motion. Matching criteria which sum differences between the blocks are directly related to the size of the block. In the case of mean-based matching criteria (e.g. MAD and MSD), a large block size will be sensitive to the overall motion of the frame and may set the motion to zero for a block with the desired motion. In contrast, sum-based matching criteria like SAD for a given block size are more sensitive to large differences of a few pixels. 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.

3.1 Data

The traffic sequences tested are publicly available from a website maintained by KOGS/IAKS Universitaet Karlsruhe.[1] In Figure 8, the first column is sample frames from the four 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 block matching results to evaluate the different techniques. We used both ground truth masks and object motion vectors as ground truth measurements for our experiments.

Masks 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. Samples of frames with "blocky" versions of the ground truth masks are shown in Figure 8.

Object Motion Vectors 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.

3.2 Results

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

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

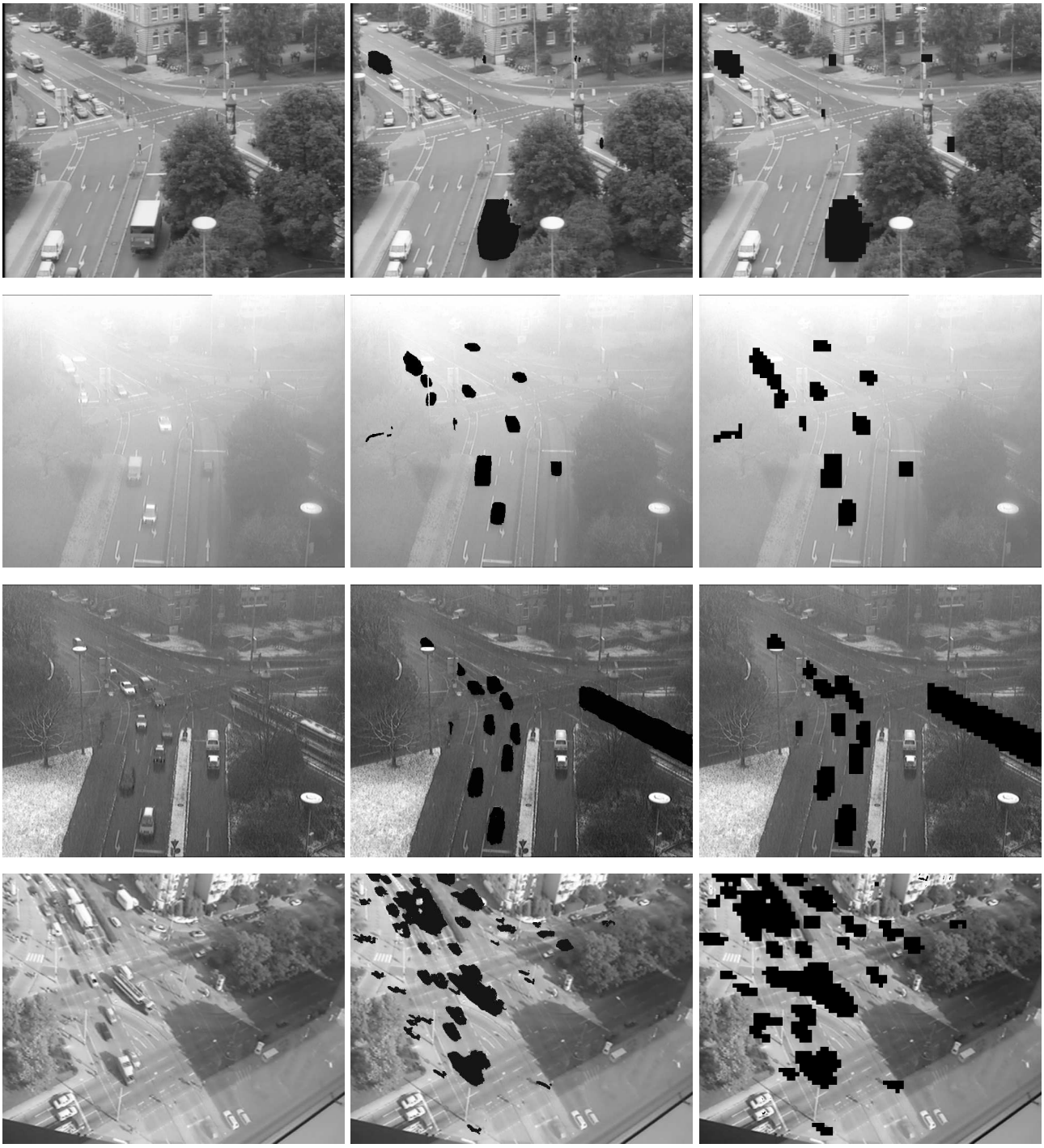


Figure 8: Sample frames (first column), manually highlighted ground truth frames (second column), and "blocky" (8×8 blocks) ground truth frames (third column) from each sequence. Each row is a sequence: row 1: *Bright*, row 2: *Fog*, row 3: *Snow*, and row 4: *Busy*.

3.2.1 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,

$$P = \frac{\text{\# of moving pixels correctly detected}}{\text{\# of moving pixels detected}}$$

and the recall, R, is a measure of moving objects detected,

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

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 9 shows the results for each sequence. As the zero-motion bias threshold increases (right-to-left in Figure 9), the noise is reduced (increasing precision) until eventually (at the knee of the curve), moving object pixels are removed (decreasing recall). MSD gives the best overall results. 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 shown in Figure 10. The average precision-recall curves are generated by averaging the results of all sequence for a given matching criterion and threshold. 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 $t_{MPC} = 10$.

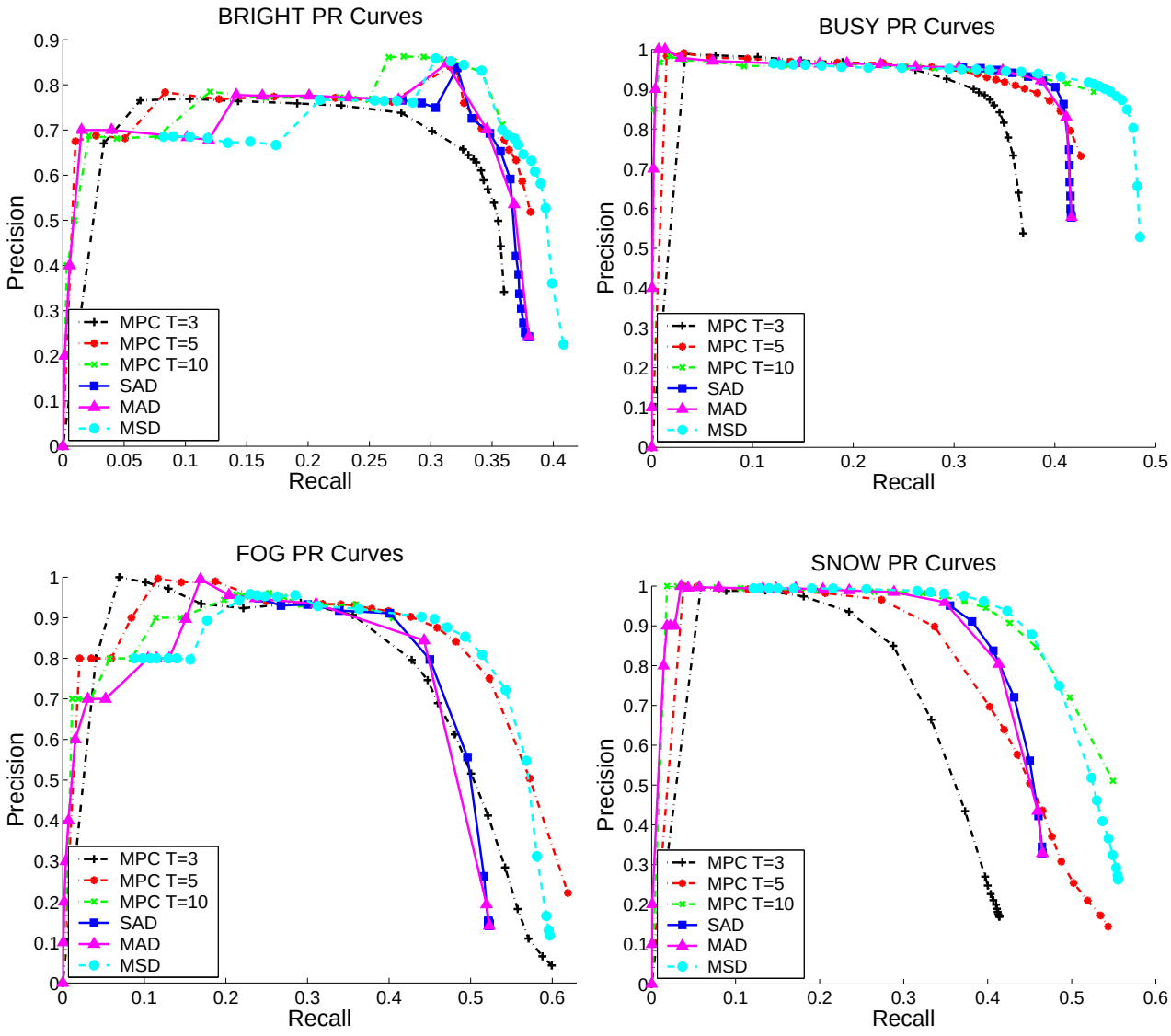


Figure 9: Precision-recall curves of matching criteria with varying zero-motion bias threshold for the four video sequences.

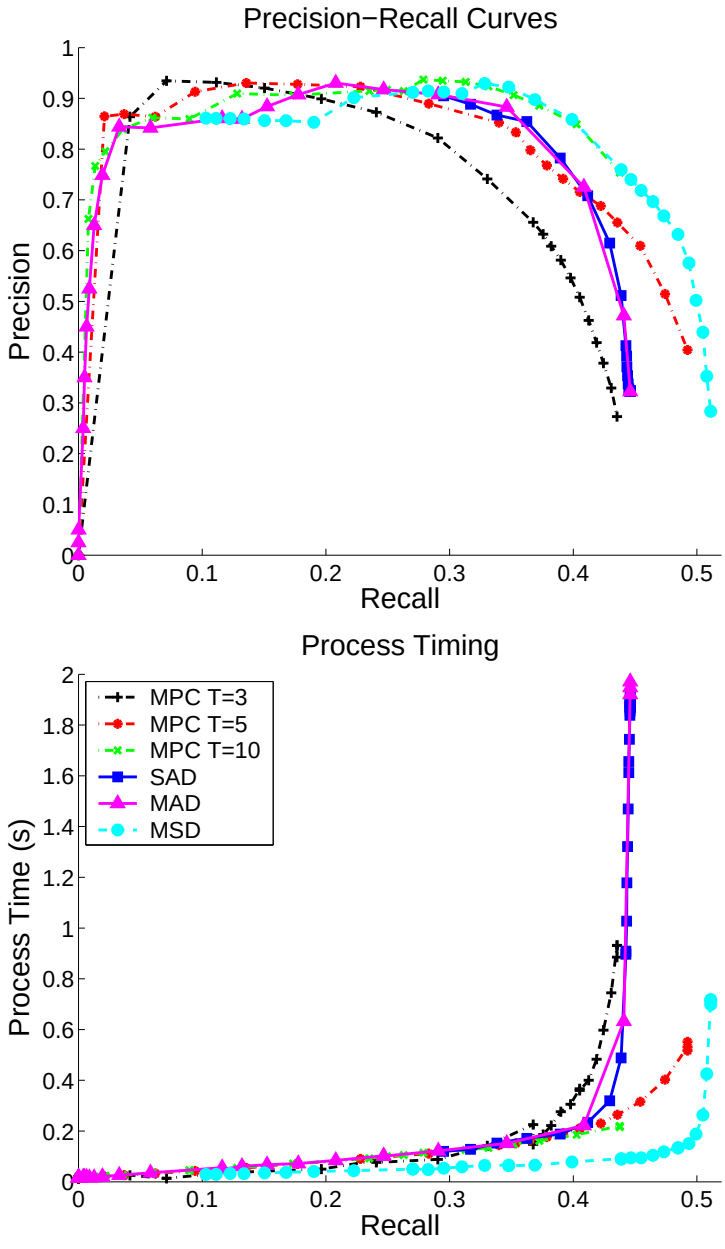


Figure 10: The top plot is the average precision-recall curves of the four sequences for matching criteria with varying zero-motion bias threshold. The bottom plot is the corresponding average process timing over the frames and sequences for a given zero-motion bias threshold.

3.2.2 Search Method

Evaluation Metrics 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. As previously stated, this is obtained by first determining an object's motion vector. Each connected component (object) in the ground truth frame is manually moved to find the best match in the subsequent frame; this motion is the object motion vector. Every pixel within the object is given the object motion vector and the error is the absolute difference between the object's motion vector and the motion vector of a pixel from block matching. We have manually determined a motion vector for the objects in one frame per sequence.

Comparison 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 seen in Figures 11, 13, 17, 19. Each error ellipse is formed from the covariance matrix of an object [12]. A tight ellipse indicates that the motion vectors within an object are coherent.

Although Figures 11, 13, 17, 19 gives insight into the coherence of an object, there are co-occurring ellipses, hiding the overall performance of a search method. To address this, Figures 12, 14, 18, 20 show the percentage of objects detected with error ellipses that fit in a circle of radius ρ (an object motion vector error with standard deviation $\leq \rho$). The higher the percentage, the better the method, with 100% indicating all objects detected.

Coherence Metric

Bright: In Figure 11, the cross method has the tighter error ellipses, signifying more coherence of the object motion vectors than the other search methods. But from Figure 12, we see the cross search has fewer objects detected overall and fewer objects with standard deviations within 3 pixels than the TSS and the window search.

Fog: In Figure 13, 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 14, the 2D-logarithmic search has a higher percentage of objects detected within a given error range. Figures 15 and 16 are results from a frame in the *Fog* sequence. We see in the 2D-logarithmic search a bias in the vertical direction. A comparison of the vehicle at the bottom also shows the coherence of the 2D-logarithmic search compared to the other search methods.

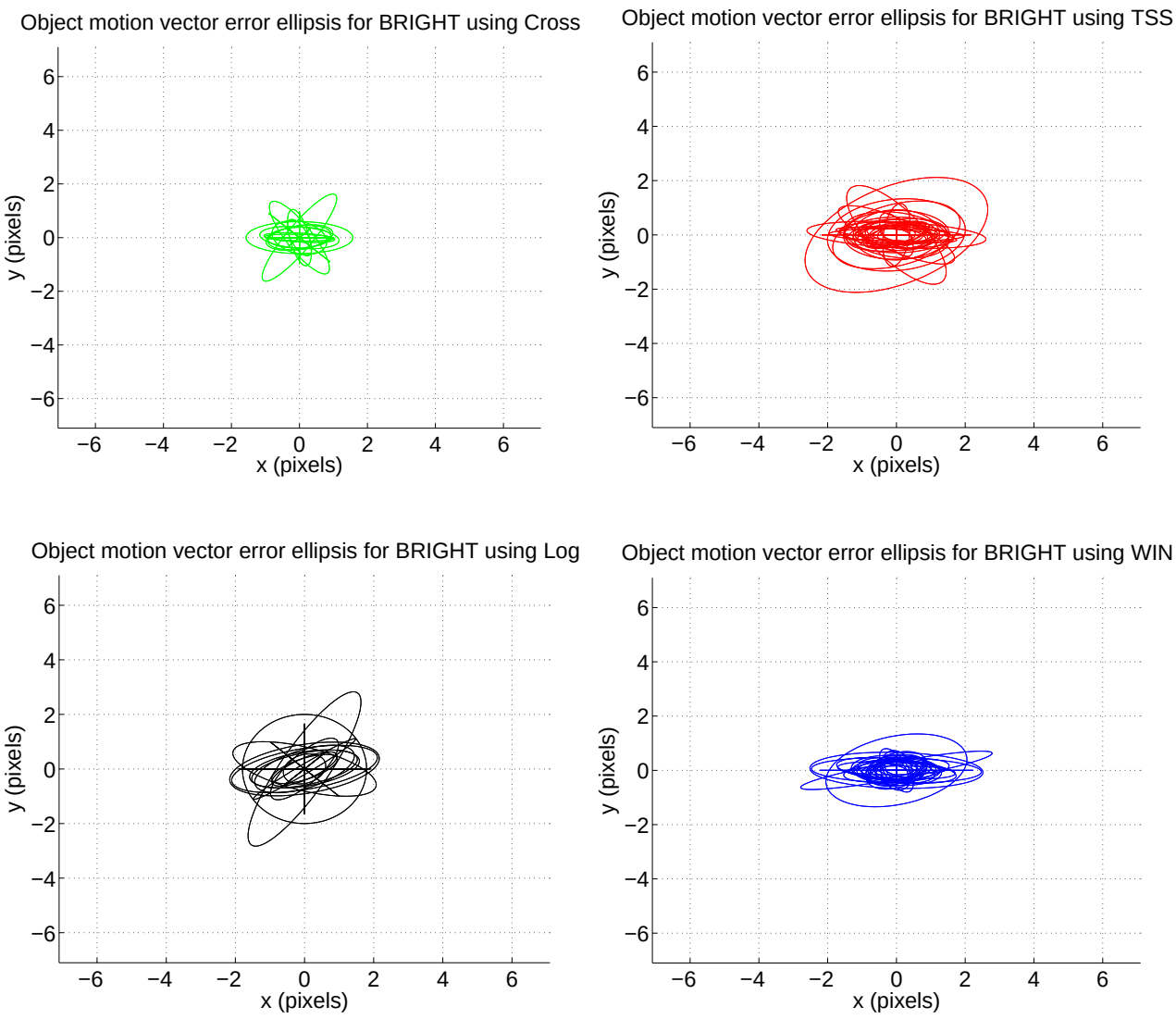


Figure 11: The plots show error ellipses of motion vectors within an object for the *Brignt* sequence using the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The cross, TSS, and 2D-logarithmic searches have a step size of 4 and the window search has a step size of 1.

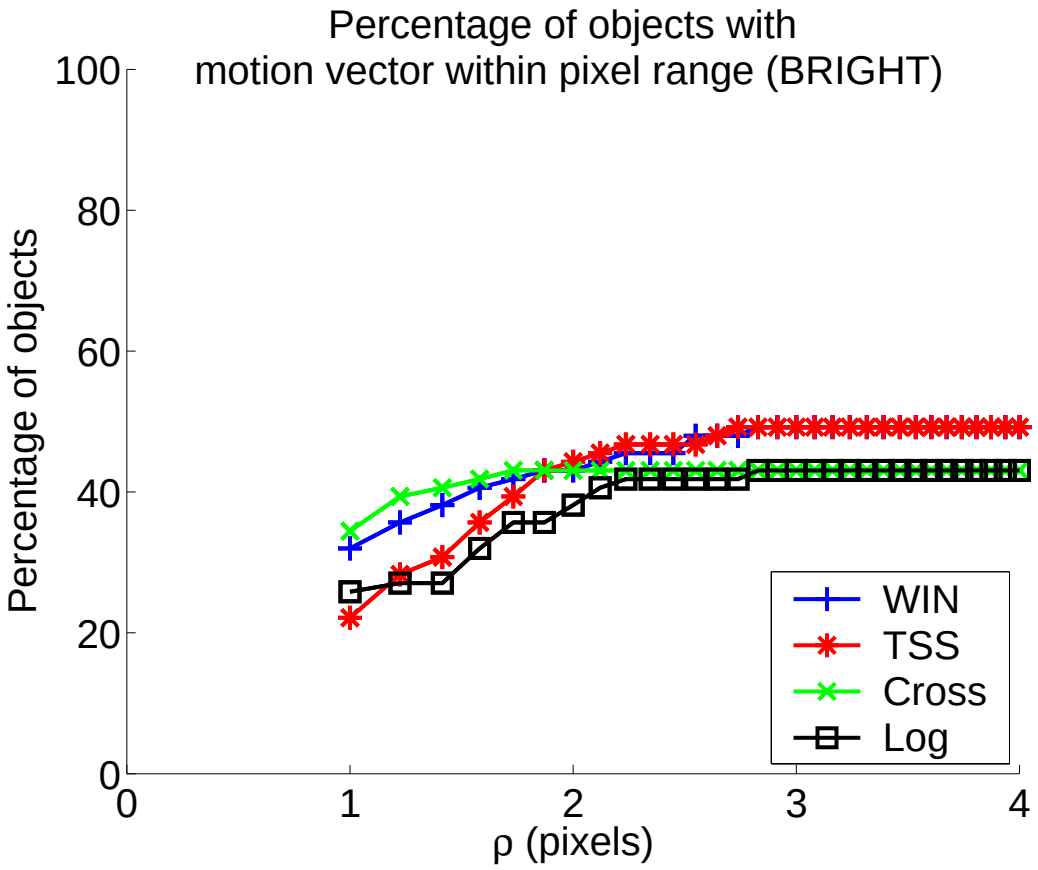


Figure 12: *Bright* sequence: percentage of objects that fit in an error circle of radius ρ for each search method using the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The cross, TSS, and 2D-logarithmic searches have a step size of 4 and the window search has a step size of 1.

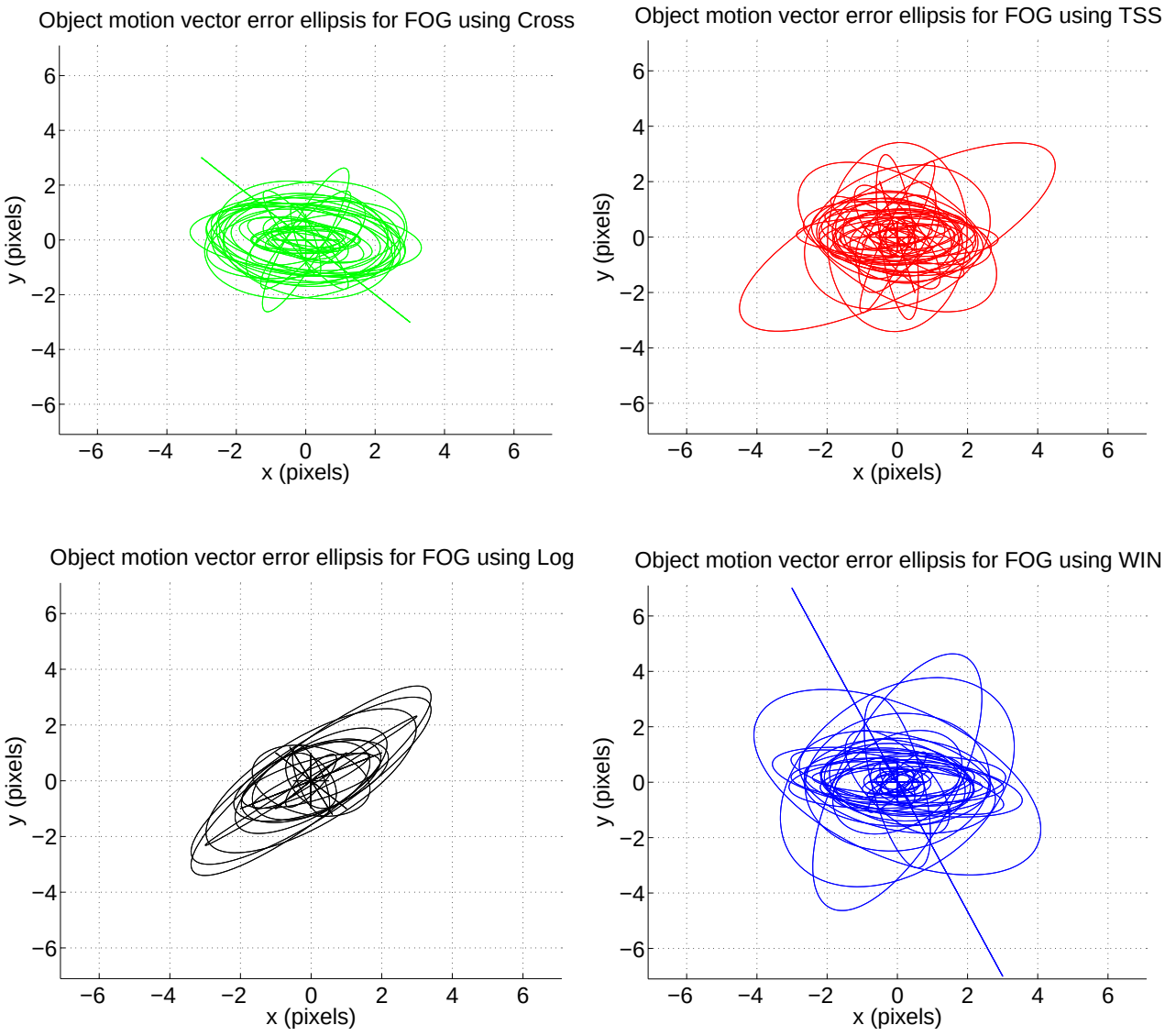


Figure 13: The plots show error ellipses of motion vectors within an object for the *Fog* sequence using the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The cross, TSS, and 2D-logarithmic searches have a step size of 4 and the window search has a step size of 1.

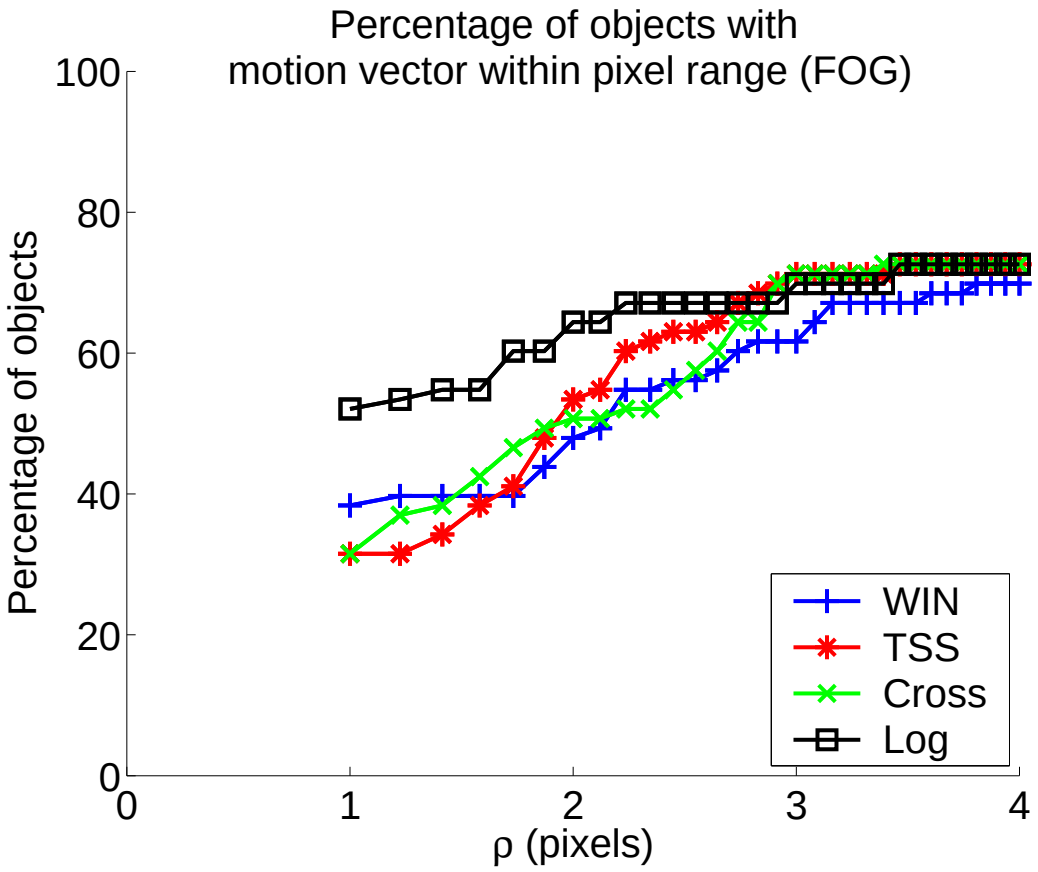


Figure 14: *Fog* sequence: percentage of objects that fit in an error circle of radius ρ for each search method using the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The cross, TSS, and 2D-logarithmic searches have a step size of 4 and the window search has a step size of 1.

Figure 15: An example frame of detected motion vector blocks from the *Fog* sequence using the cross search (top) and TSS search (bottom) methods, the simple block determination approach with 8×8 blocks, the MSD matching criterion, and step size of 4.

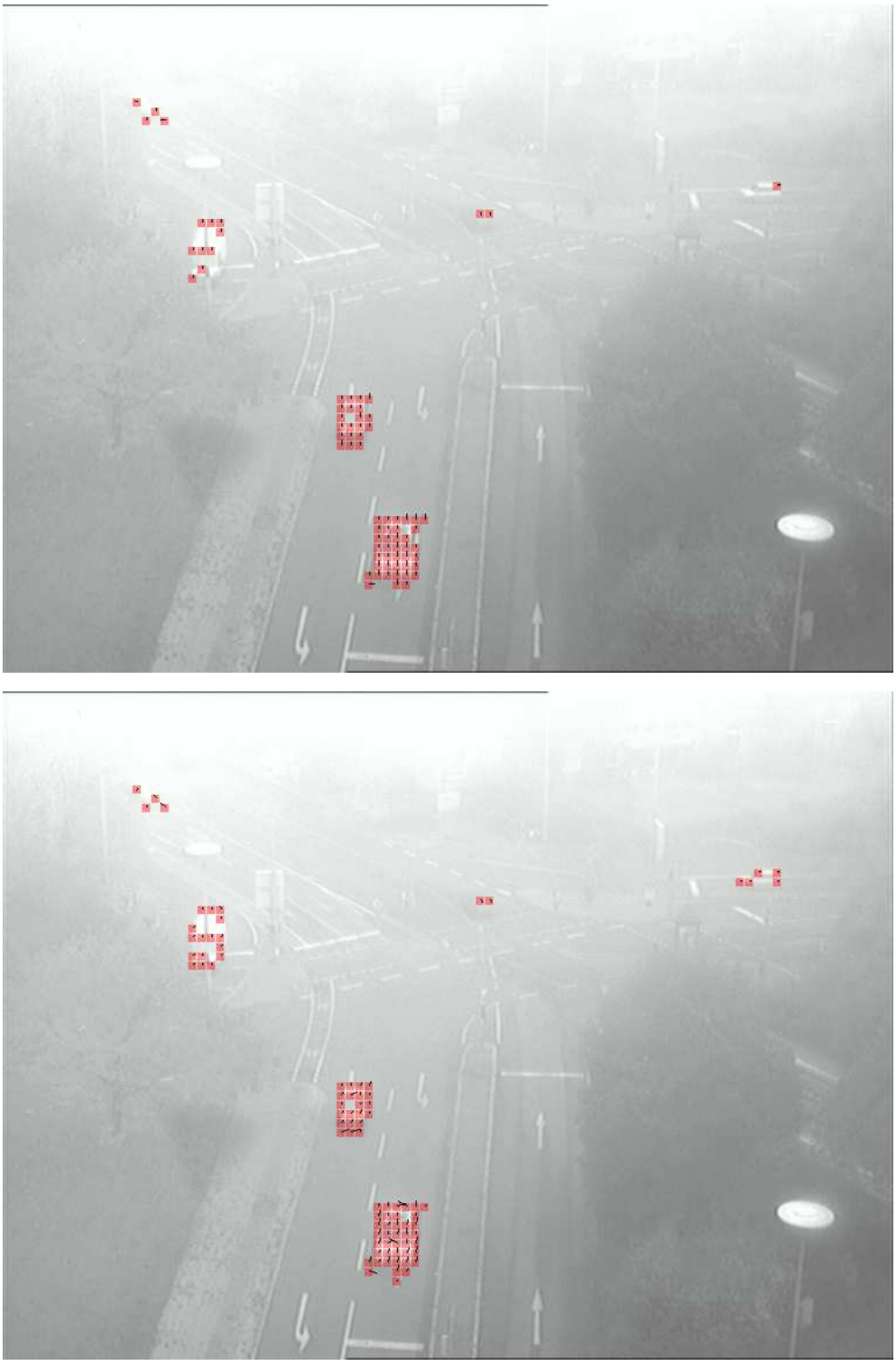


Figure 16: An example frame of detected motion vector blocks from the *Fog* sequence using the 2D-logarithmic search (top) and window search (bottom) methods, the simple block determination approach with 8×8 blocks, the MSD matching criterion. The 2D-logarithmic search method with step size of 4 and the window search method with step size of 1.

Snow: In Figure 17, there is no significant difference between the search methods. Figure 18 confirms this observation.

Busy: In Figure 19, the cross method has the tightest error ellipsis, but the window search seems to have a denser core. In Figure 20, the window search and TSS have no difference, but there is a clear distinction from the 2D-logarithmic search and the cross search.

Magnitude Metric 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 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.

$\lvert\hat{\vec{E}}\rvert$	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.

3.2.3 Block Determination

We compare two block determination approaches: a simple approach and a hierarchical approach. These block determination approaches are evaluated using the coherence metric and average magnitude of object motion vector error. Figure 21 shows the hierarchical approach clearly has tighter error ellipsis in the *Fog* and *Snow* sequences. In Figure 22, we see that the hierarchical approach has a larger or equal percentage of objects overall pixel ranges. Figures 23-24 show an example frame from the *Fog* and

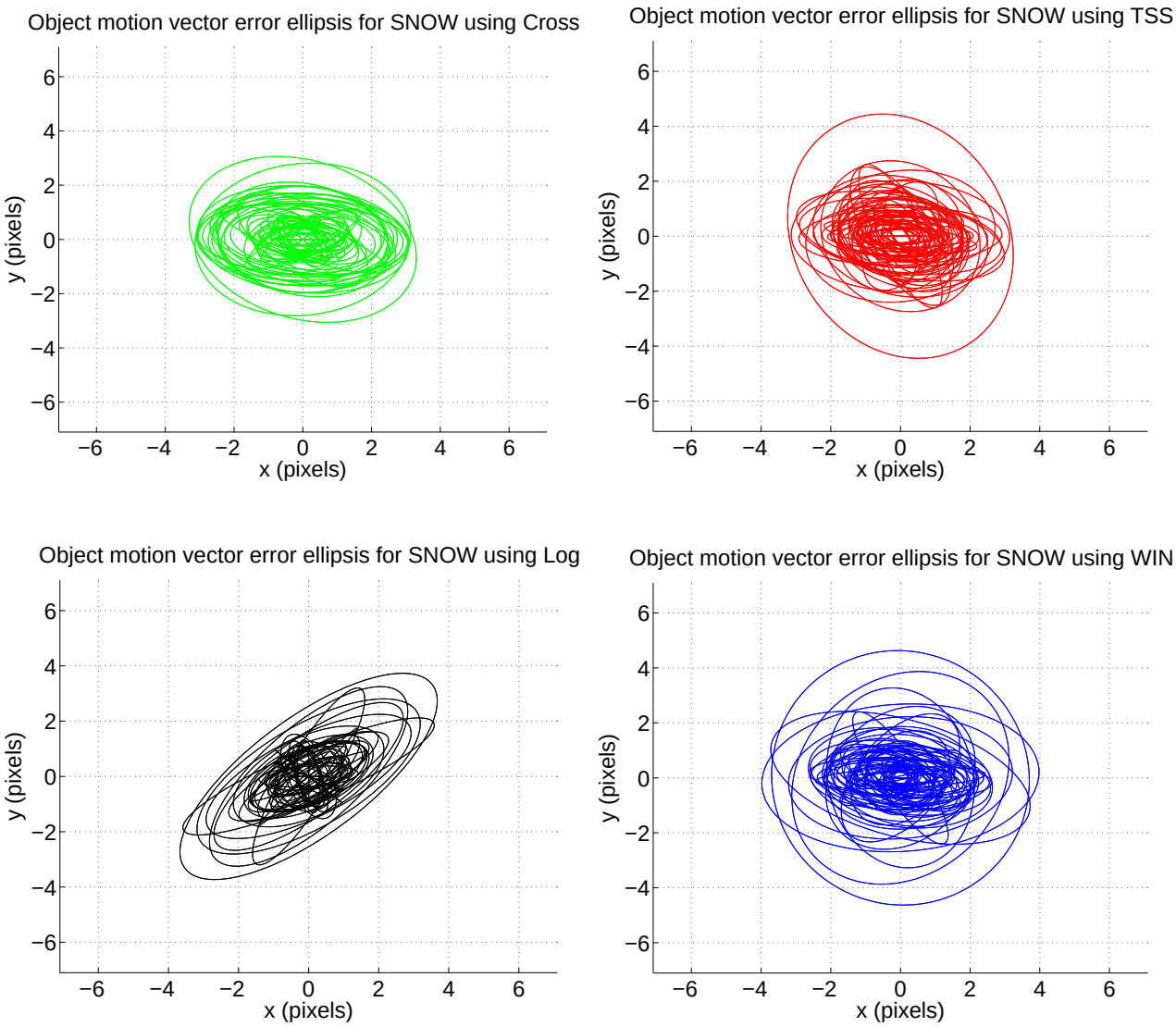


Figure 17: The plots show error ellipses of motion vectors within an object for the *Snow* sequence using the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The cross, TSS, and 2D-logarithmic searches have a step size of 4 and the window search has a step size of 1.

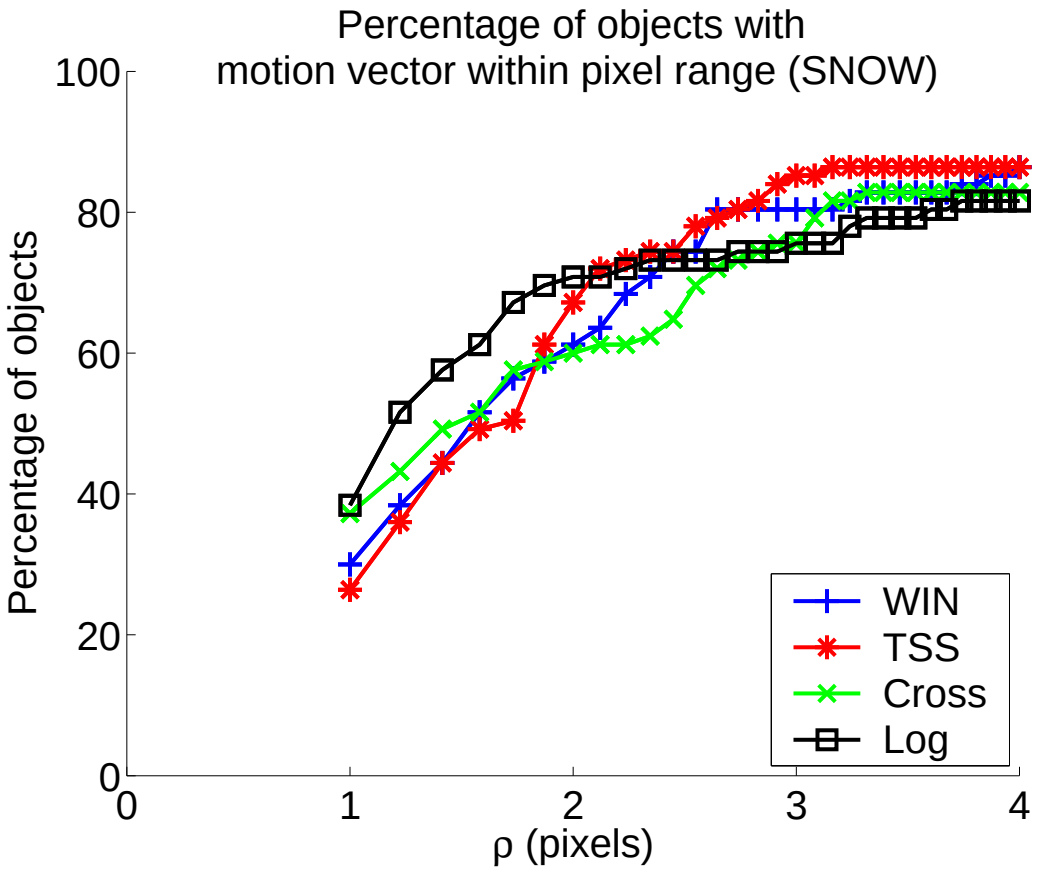


Figure 18: *Snow* sequence: percentage of objects that fit in an error circle of radius ρ for each search method using the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The cross, TSS, and 2D-logarithmic searches have a step size of 4 and the window search has a step size of 1.

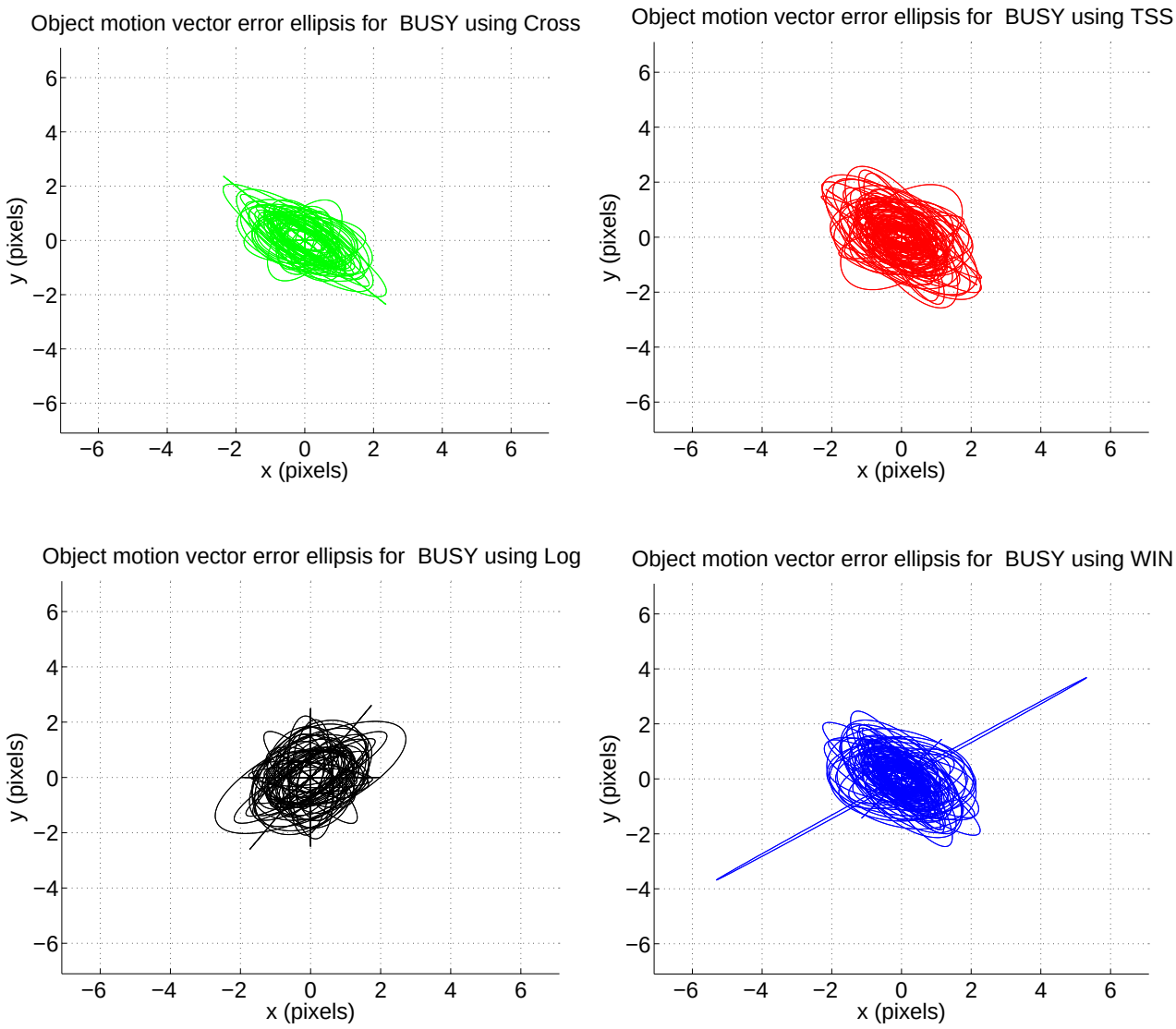


Figure 19: The plots show error ellipses of motion vectors within an object for the *Busy* sequence using the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The cross, TSS, and 2D-logarithmic searches have a step size of 4 and the window search has a step size of 1.

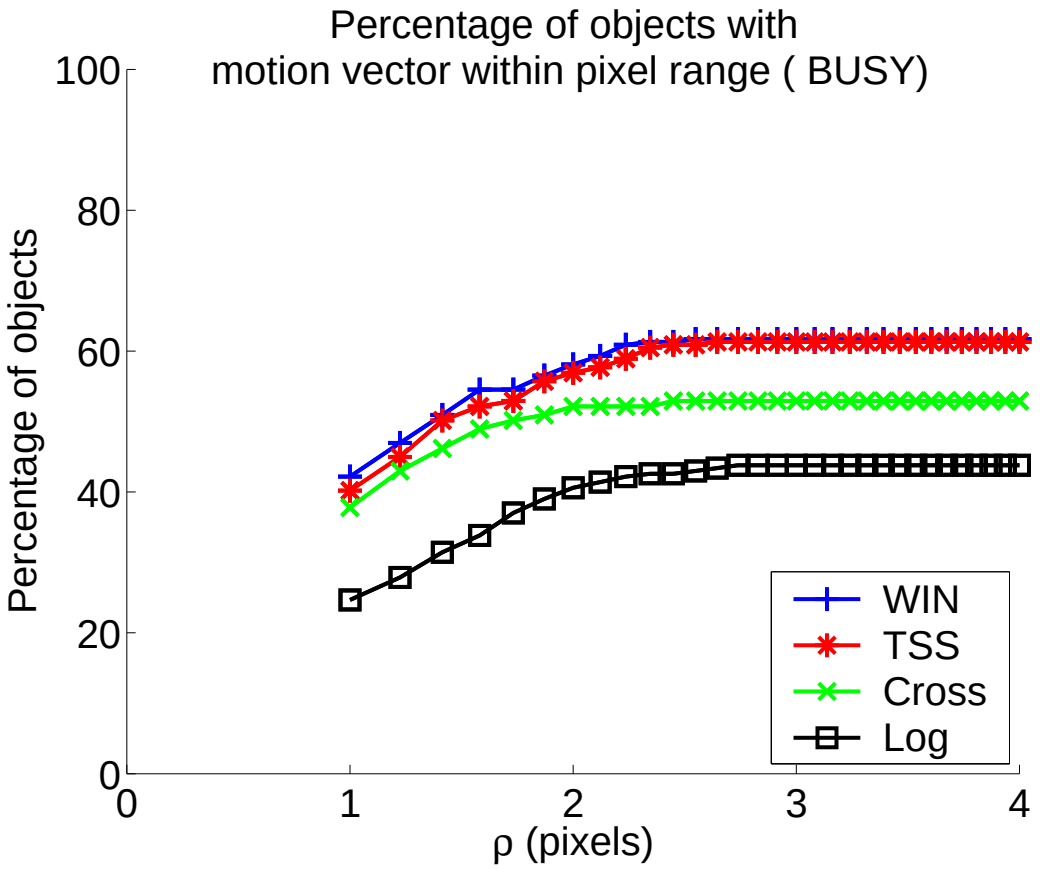


Figure 20: *Busy* sequence: percentage of objects that fit in an error circle of radius ρ for each search method using the simple block determination approach with 8×8 blocks, and the MSD matching criterion. The cross, TSS, and 2D-logarithmic searches have a step size of 4 and the window search has a step size of 1.

Snow sequences using the two block determination methods. The hierarchical approach is seen as more coherent than the simple approach.

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 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.

$\|\hat{\vec{E}}\|$	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.

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 the simple block determination approach though not as coherent as the hierarchical approach in the *Fog* 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.

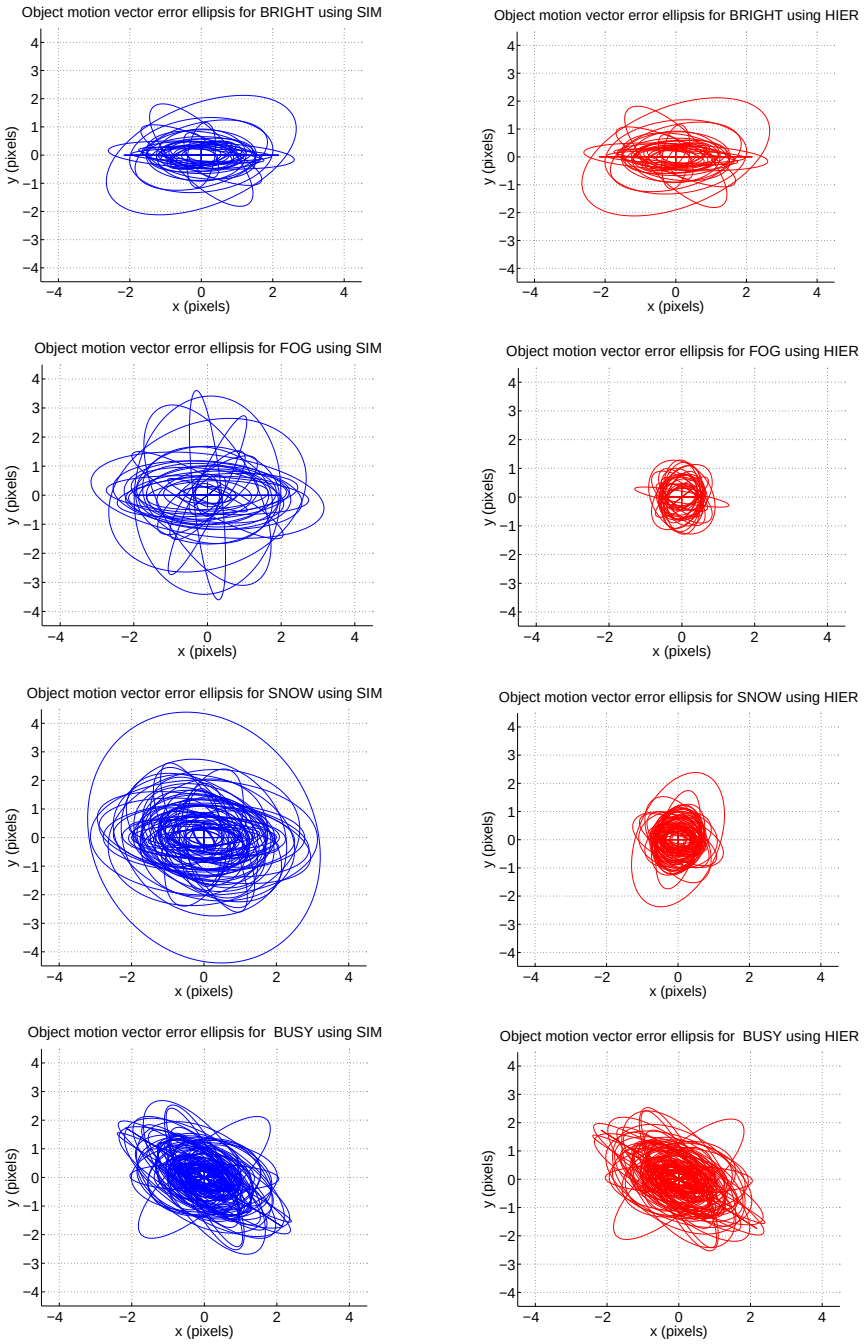


Figure 21: Each row is from a different sequence, row 1: *Bright*, row 2: *Fog*, row 3: *Snow*, and row 4: *Busy*. The plots show error ellipses of motion vectors within an object, column 1 uses the simple block determination approach and column 2 uses the hierarchical block determination approach, all plots use 8×8 blocks, the TSS search method, the MSD matching criterion, and step size of 4.

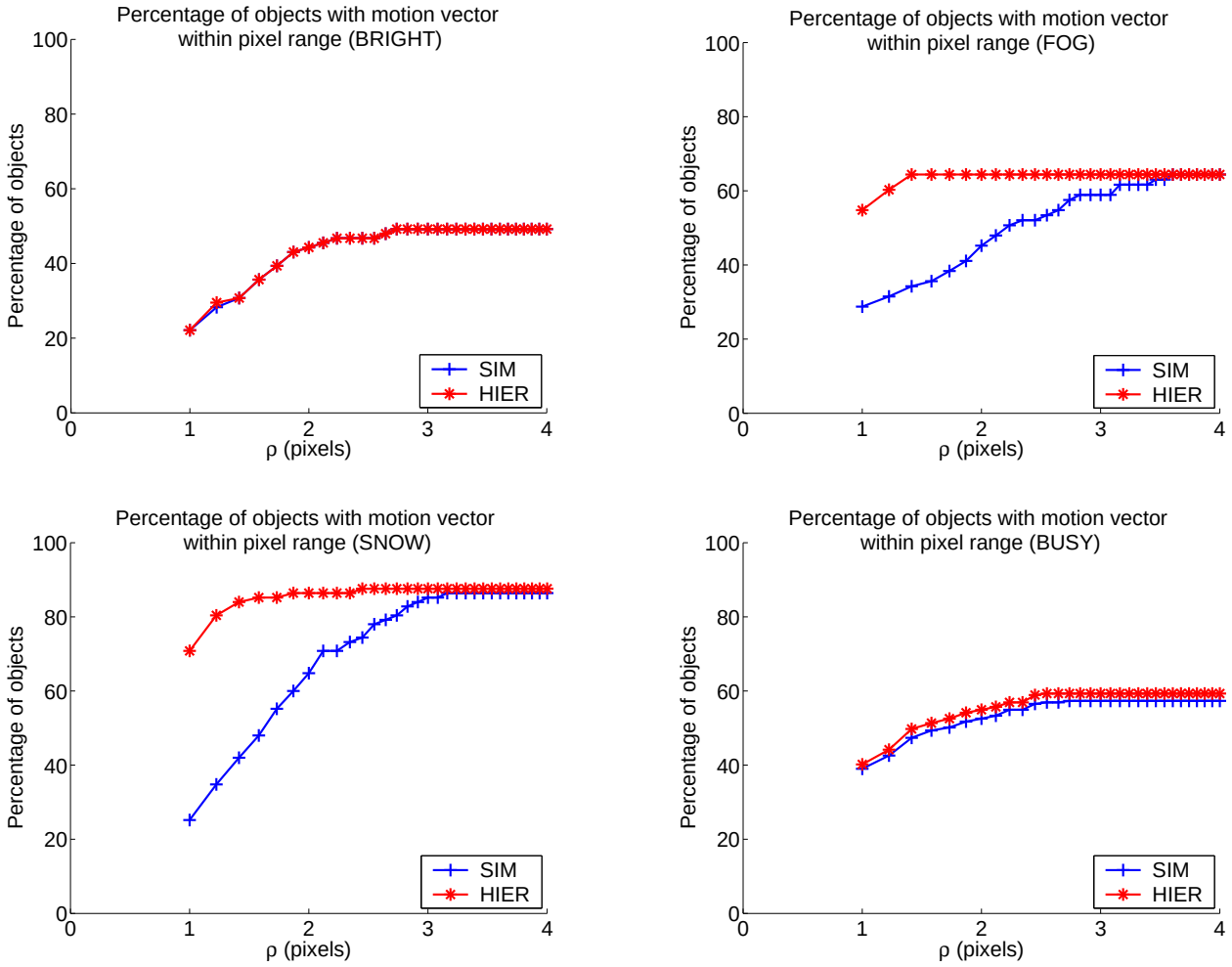


Figure 22: Percentage of objects that fit in an error circle of radius ρ for simple and hierarchical block determination approaches with 8×8 blocks, the TSS search method, the MSD matching criterion, and step size of 4.

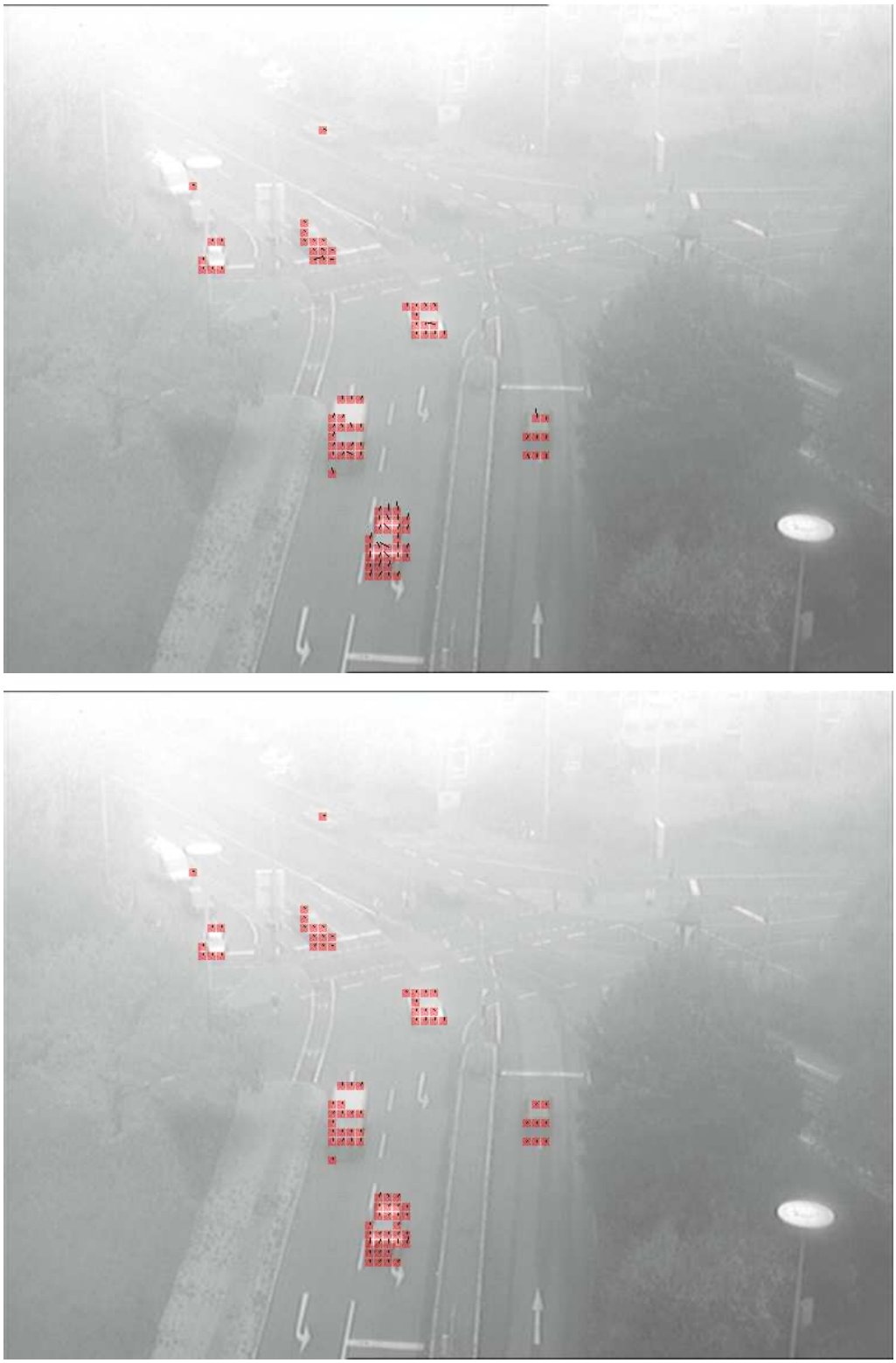


Figure 23: An example frame of detected motion vector blocks from the *Fog* sequence using the simple block determination (top) and hierarchical block determination (bottom) approaches with the TSS search method, 8×8 blocks, the MSD matching criterion, and step size of 4.

Figure 24: An example frame of detected motion vector blocks from the *Snow* sequence using the simple block determination (top) and hierarchical block determination (bottom) approaches with the TSS search method, 8×8 blocks, the MSD matching criterion, and step size of 4.

UCRL-TR-218038: 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] BEI, C.-D., AND GRAY, R. An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Transactions on Communications 33* (October 1985).

[2] CHAN, M. H., YU, Y. B., AND CONSTANTINIDES, A. G. Variable size block matching motion compensation with applications to video coding. In *IEE* (August 1990), vol. 137.

[3] CHEUNG, C.-K., AND PO, L.-M. Normalized partial distortion search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology 10*, 3 (April 2000).

[4] CHEUNG, S. S.-C., AND KAMATH, C. Robust background subtraction with foreground validation for urban traffic video. *Eurasip Journal on Applied Signal Processing 14* (2005), 2330–2340.

[5] GHANBARI, M. The cross-search algorithm for motion estimation. *IEEE Transactions on Communication 38*, 7 (1990), 950–953.

[6] GYAOUROVA, A., KAMATH, C., AND CHEUNG, S.-C. Block matching for object tracking. Tech. Rep. UCRL-TR-200271, Lawrence Livermore National Laboratory, 2003.

[7] JAIN, J., AND JAIN, A. Displacement measurement and its application in interframe image coding. *IEEE Transactions on Communication COMM-29*, 12 (1981), 1799–1808.

[8] KOGA, T., IINUMA, K., HIRANO, A., IIJIMA, Y., AND ISHIGURO, T. Motion compensated interframe coding for video conferencing. In *National Telecommunications Conference* (1981), pp. G5.3.1–5.3.5.

[9] LIU, B., AND ZACCARIN, A. New fast algorithms for the estimation of block motion vectors. *IEEE Transactions on Circuits and Systems for Video Technology 3*, 2 (April 1993).

[10] LIU, L.-K., AND FEIG, E. A block-based gradient descent search algorithm for block motion estimation in video coding. *IEEE Transactions on Circuits and Systems for Video Technology 6*, 4 (August 1996), 419–422.

[11] MANNING, C. Block matching algorithms for motion compensated video compression. Master's thesis, National University of Ireland, 1996.

[12] MESTER, R., AND HOTTER. Robust displacement vector estimation including a statistical error analysis. In *5th International Conference on Image Processing and Its Applications* (July 1995), pp. 168–172.

[13] MIZUKI, M. M. Edge based video image compression for low bit rate applications. Master's thesis, Massachusetts Institute of Technology, September 1996.

[14] MUSMANN, H. G., PIRSCH, P., AND GRALLERT, H.-J. Advances in picture coding. In *Proceedings of IEEE* (1985), vol. 73, pp. 523–548.

[15] PICCARDI, M. Background subtraction techniques: a review. *IEEE International Systems Man and Cybernetics 4* (2004), 3099–3104.

[16] RHEE, I., MARTIN, G. R., MUTHUKRISHNAN, S., AND PACKWOOD, R. A. Quadtree-structured variable-size block-matching motion estimation with minimal error. *IEEE Transactions on Circuits and Systems for Video Technology 10*, 1 (February 2000).

[17] SYMES, P. D. *Video Compression Demystified*. McGraw Hill, 2000.

[18] TEKALP, A. M. *Digital Video Processing*. Prentice-Hall, Upper Saddle River, NJ, 1995.

[19] WANG, Y., OSTERMANN, J., AND ZHANG, Y.-Q. *Video Processing and Communications*. Prentice-Hall, 2001.

[20] WANG, Y., WANG, Y., AND KURODA, H. A globally adaptive pixel-decimation algorithm for block-motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology 10*, 6 (September 2000).

[21] ZHANG, K., BOBER, M., AND KITTLER, J. Variable block size video coding with motion prediction and motion segmentation. In *SPIE* (1995), vol. 2419, pp. 62–70.