Chapter 5

Moving Object Detection and Compression in IR Sequences [⋆]

Namrata Vaswani, Amit K Agrawal, Qinfen Zheng, and Rama Chellappa

Center for Automation Research, University of Maryland, College Park, MD,
{namrata,aagrawal,qinfen,rama}@cfar.umd.edu

Summary. We consider the problem of remote surveillance using infrared (IR) sensors. The aim is to use IR image sequences to detect moving objects (humans or vehicles), and to transmit a few “best-view images” of every new object that is detected. Since the available bandwidth is usually low, if the object chip is big, it needs to be compressed before being transmitted. Due to low computational power of computing devices attached to the sensor, the algorithms should be computationally simple. We present two approaches for object detection — one which specifically solves the more difficult long-range object detection problem, and the other for objects at short range. For objects at short range, we also present techniques for selecting a single best-view object chip and computationally simple techniques for compressing it to very low bit rates due to the channel bandwidth constraint. A fast image chip compression scheme implemented in the wavelet domain by combining a non-iterative zerotree coding method with 2D-DPCM for both low-and high-frequency subbands is presented. Comparisons with some existing schemes are also included. The object detection and compression algorithms have been implemented in C/C++ and their performance has been evaluated using the Hitachi's SH4 platform with software simulation.

5.1 Introduction

Remote monitoring of activities of stationary or moving vehicles and humans is a critical component in surveillance applications. The sensors used in practice are typically of low quality and the available bandwidth for transmission is quite limited. We consider the problem of remotely monitoring a battlefield with IR sensors and present two approaches for object detection. The first approach is useful when the objects are at large distances, are very small, appear

[⋆] Prepared through collaborative participation in the Advanced Sensors Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0008. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation thereon.

to move slowly, and their signatures have low contrast over the background. In such cases, traditional methods based on the analysis of the difference between successive frames and/or image and background or image intensity change will not work. We present a novel algorithm (referred to as the MTI algorithm) based on variance analysis which is useful at long ranges, when the object is small and slowly moving. The algorithm uses temporal variance to detect potential moving spots, spatial variance analysis to suppress false alarms caused by sensor vibration, and object geometrical constraints to filter out false alarms caused by tree branches and sensor noise.

In other scenarios, when the objects are not so far away, the problem of moving object detection is formulated as one of segmenting an image function using a measure of its local singularity as proposed in [1]. When the detected objects are very small (see Figures. 5.1, 5.2), all views are equally good or bad and hence the problem of best-view selection becomes irrelevant. Also, since the object chip is already very small, compression is not necessary and hence any chipped image of the object can be transmitted. Thus, in such cases all the computational power available on the sensor can be used to solve the object detection problem. When the object chips are larger so that over time the object pose changes, we choose one best-view of the object, compress it using computationally simple techniques and then transmit the compressed best view chip. The challenge here is to solve the best view selection problem and to develop techniques which can compress the image to very low bit rates and are yet computationally simple since they have to be implemented in realtime and on hardware with limited computing power. Note that in this work, we do not study techniques for channel coding since we assume a reliable channel is available for transmission.

Performance evaluation of algorithms is critical for determining the memory and computation needs of the algorithm. We present these measures for the algorithms using the Hitachi's SH-4 microprocessor.

The rest of the chapter is organized as follows. Section 5.2 describes the MTI object detection algorithm for long-range, small-size object detection along with results followed by the technique for detecting short-range and larger objects. Section 5.3 describes techniques for best-view selection for large objects. Section 5.4 discusses compression techniques for real-time, low-power, and very low bit rate compression of object chips. Simulations results on SH4–7751 microprocessor for performance characterization are presented in Section 5.5 and conclusions in Section 5.6.

5.2 Object Detection

We present two object detection algorithms, the first one deals with the more difficult problem of long-range object detection when objects are very small and the second one for detecting objects at short range.

5.2.1 Long-Range Object Detection (MTI)

Techniques based on optical flow and image intensity analysis may not work well for long-range object detection. In typical scenarios, a moving object size can be as small as 2×3 pixels and the motion between two adjacent frames can be less than 0.1 pixels. Several challenging issues need to be addressed. The first challenge is to compensate for sensor motion using electronic stabilization methods. Over the years, two dominant approaches have been developed — flow-based and feature based. The feature-based methods extract point, edge or line features and solve for an affine transformation between successive frames, while the flow-based methods compute optical flow and then estimate an affine model. These methods do not perform well in IR images as feature extraction in IR images is not reliable, and flow estimates for IR images suffer from severe bias due to noise. Also, these methods are too complex for real-time computation. Even if stabilization can be solved using one of these approaches, the problem of separating moving objects from the stabilized background is challenging due to low signal-to-clutter ratio in surveillance applications. Other factors that complicate the problem are changes in the background as the sensor is panning, false motion detection due to atmospherics, and other confusing motion (motion of tree leaves, etc).

We assume a stationary sensor. The proposed algorithm uses a variance analysis based approach for moving object detection that can effectively integrate the object motion information over both temporal and spatial domains. Each input frame is first checked for possible errors and then used to update the temporal variance. When the variance for a pixel increases above a given threshold, the pixel is labelled as corresponding to a potential moving object. All candidate-moving pixels are then grouped to form disjoint moving objects. The detected potential moving objects (regions) are verified for their size, shape, and orientation to filter out false detection. After false detection verification, the remaining change regions are reported as moving objects.

The temporal variance at a pixel (i, j) at frame k is computed as

$$\sigma^2_{i,j,k} = S_{i,j,k} - \mu^2_{i,j,k}, \tag{5.1}$$

where

$$S_{i,j,k} = \frac{1}{L} f^2_{i,j,k} + \frac{L-1}{L} S_{i,j,k-1}, \tag{5.2}$$

$$\mu_{i,j,k} = \frac{1}{L} f_{i,j,k} + \frac{L-1}{L} \mu_{i,j,k-1}, \tag{5.3}$$

$$S_{i,j,1} = f^2_{i,j,1}, \tag{5.4}$$

$$\mu_{i,j,1} = f_{i,j,1}. \tag{5.5}$$

Here, $f_{i,j,k}$ is the image value at frame k and location (i, j) and L is the temporal window parameter. A pixel is detected as belonging to potential moving object if the following conditions are satisfied:

$$\sigma^2_{i,j,k} \geq \sigma^2_{i,j,k-1}, \tag{5.6}$$
$$\sigma^2_{i,j,k} \geq T_0(i,j) + T_h, \tag{5.7}$$

or

$$\sigma^2_{i,j,k} \geq \sigma^2_{i,j,k-1}, \tag{5.8}$$
$$\sigma^2_{i,j,k} \geq T_0(i,j) + T_l, \tag{5.9}$$
$$\max_{-1\leq\delta_x\leq 1,-1\leq,\delta_y\leq 1} \sigma^2_{i+\delta_x,j+\delta_y,k} \geq T_0(i,j) + T_h, \tag{5.10}$$

where

$$T_0(i,j) = \max_{-2\leq\delta_x\leq 2,-2\leq\delta_y\leq 2} |f_{i+\delta_x,j+\delta_y,0} - f_{i,j,0}|, \tag{5.11}$$
$$\sigma_t = var_{i,j\in N\times N}(T_0(i,j)), \tag{5.12}$$
$$T_h = \gamma_h \times \sigma_t, \tag{5.13}$$
$$T_l = \gamma_l \times \sigma_t. \tag{5.14}$$

In our experiments, we set $\gamma_h = 3.4$, $\gamma_l = 0.8 * \gamma_h$, and $L = 16$.

5.2.2 MTI Algorithm Results

Figure 5.1 shows the object detection result for frame 200 and 300 on an IR sequence (IRseq10) for the case of people walking at a long distance. Even though the width of the objects (people) is 2 to 3 pixels, the algorithm correctly detects and tracks them. In this sequence, the sensor was stationary and its direction was fixed. Figure 5.2 shows the detection of a moving vehicle for frames 160 and 250. In this case the sensor pans and stops and is quickly able to find the object again. Hence, the algorithm is capable of recovering after sensor panning motion.

As a comparison with background subtraction techniques, Figure 5.3 shows the result using background subtraction based on nonparametric background modelling [2] on IRseq10. We can see that there are a lot of false detections whose sizes are the same or even greater than the object size. So, the objects cannot be detected reliably. Also background subtraction techniques need some number of frames for background modelling before they can start the detection process. Figure 5.4 shows the ROC curve for the MTI algorithm for different IR sequences for γ_h varying from 3.0 to 5.5. The detection percentage is calculated as the ratio of object detections in all frames and the total objects actually present in all frames. The false alarms percentage is calculated as the ratio of the false alarms in all frames and the total objects in all frames. Since we do not have ground truth, it was generated manually as follows. For all the sequences considered, the objects are far away and are moving approximately parallel to the image plane with uniform speed. The objects locations were hand-picked every thirty frames and are interpolated

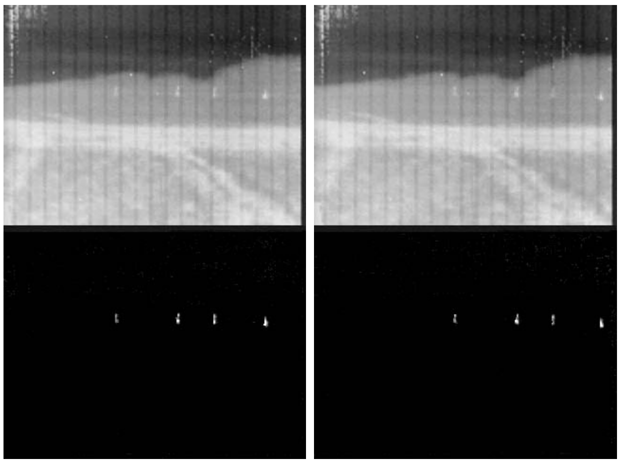


Figure 5.1. Object detection: Frames 200 and 300 of IRseq10 along with detected objects.

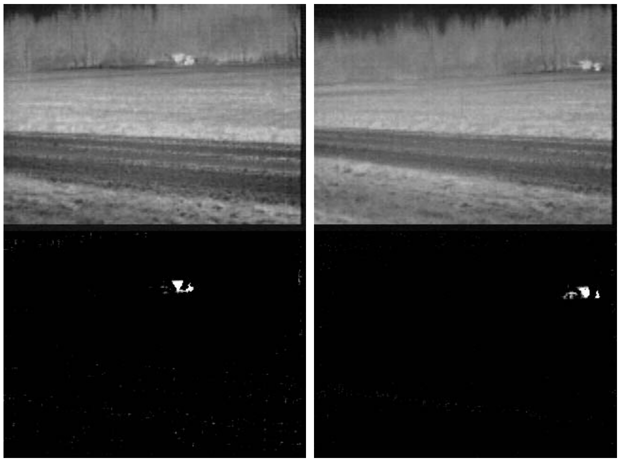


Figure 5.2. Frames 160 and 250 of a moving vehicle IR sequence along with detected objects (sensor recovering after panning motion).

in between assuming constant object velocity. Table 5.1 shows the effect of parameter L on detection rates and false alarms for different sequences.

The MTI algorithm has been implemented on a Dual Pentium 550-MHz personal computer. It can perform moving object detection for 974×436 image sequences at a sustained rate of 12 frames per second (including reading data from the hard drive and sending image to a video display). The algorithm has been tested on fourteen $974 \times 436 \times 3600$ sequences with objects at various distances, moving speeds, moving directions, and motion patterns, and with different numbers of moving objects. From the ROC curves (Figure 5.4) and

Figure 5.3. Detection result using background subtraction on IRseq10.

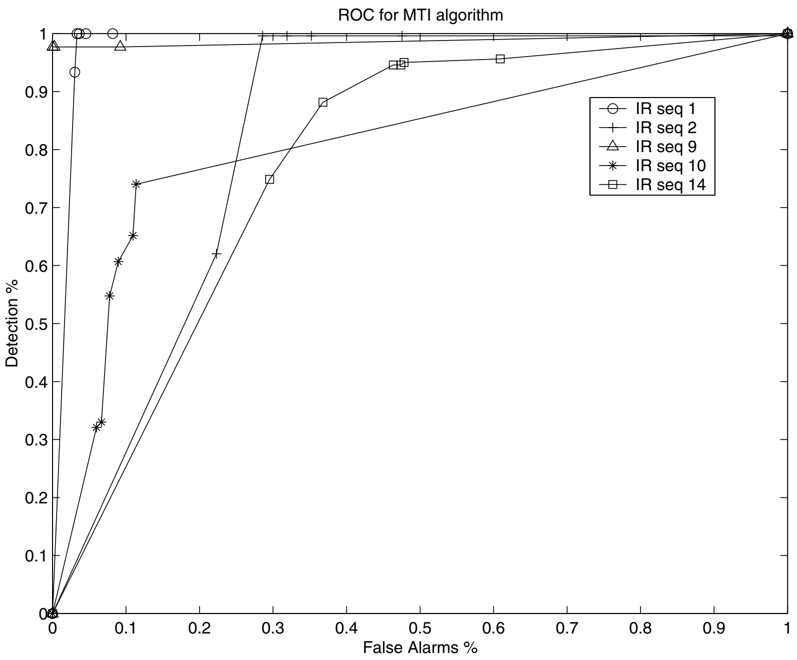


Figure 5.4. ROC curve for the MTI algorithm for different IR sequences.

Table 5.1. Effect of parameter L on detections and false alarms for various IR sequences.

Sequence	L	Detections(%)	False Alarms(%)
IRSeq1	8	98.33	13.8
IRSeq1	16	99.83	28.3
IRSeq9	8	97.70	2.24
IRSeq9	16	97.70	0.05
IRSeq2	8	99.80	32.09
IRSeq2	16	99.60	57.33

Table 5.1, we see that the MTI algorithm gives good results (high detection rate and very low false alarm rate) on almost all of the test sequences with no running parameters to be adjusted.

5.2.3 Short-Range Object Detection

The problem of object detection is formulated as one of segmenting an image function using a measure of its local singularity as proposed in [1]. The method combines the problem of stabilization, object detection, and tracking into a single process when interframe motions are restricted to lateral translations or tilts and scale changes, and has the advantage of exploiting these sensor motion constraints for performing simultaneous activity detection and stabilization. The algorithm makes use of the Holder exponent of a hybrid capacity (derivative of Gaussian along the X- and Y- axis). Using this measure, Lipschitz signatures which reflect the singularity of the image function along each spatial axis are defined. The Lipschitz signatures are used for detection and tracking of objects. The proposed measure is obtained by applying the operators $G_{x,\sigma}$ and $G_{y,\sigma}$ to the images which are the derivatives of the Gaussian applied along the *x- and* y- axes, respectively. The Lipschitz signatures can then be defined as the projection of the these measure along the *x- and* y- axes. The main assumption of the algorithm is that the "active regions" of the image exhibit some higher level of singularity in the Lipschitz signatures. In other words, the singularities can be detected and tracked over time. The algorithm is robust to image scale variations and can handle multiple moving objects. It also involves projection along spatial axis and hence can be done in real time. Spatio-temporal information on the objects in the scene can also be inferred.

5.3 Best View Selection

As mentioned earlier, the best-view selection is not required when the detected object chip is small (few pixels), but becomes important in case the object chip is large and the object pose changes from time to time.

Three different techniques were tried for best-view selection. We work on the assumption that the side view is the "best view" since it has most of the identifiable features (See Figure 5.5). Eigenspace classification has successfully been used for pose detection [3] and face recognition applications [4, 5]. In the first approach, we formulate the best-view selection problem as a pose matching problem in eigenspace. Another approach to best-view selection would be to wait while the object size keeps increasing (it is approaching the sensor) and transmit the largest-sized image or transmit the last frame before the object takes a turn. This can done by estimating the focus of expansion (FOE) which can be used to calculate the velocity direction. Since both techniques mentioned above are computationally intensive, they are not suitable for real-time hardware implementation. The eigenspace technique also suffers from the drawback that it is not generalizable, i.e., to use it for a different type of vehicle would require a new training phase. It works well only when similar objects are available in its training database. Hence, the third approach, a size-based detection method, was finally implemented. In what follows, we describe these techniques in detail.

5.3.1 Eigenspace Classification

This technique is useful for best-view selection when there are multiple sensors capturing an object from different orientations and a database of multiple views of the object is available. In our experiments we have used multiple views of tanks taken from different directions. A view closest to the side view is classified as the best view.

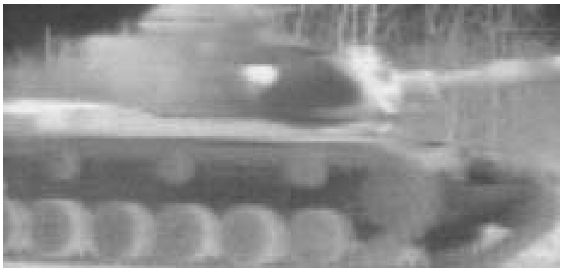


Figure 5.5. Tank2 (side view).

Algorithm

Construct an eigenspace using images of tanks in various orientations. Place the camera at 45-degree spacings around the tanks to obtain eight possible views of each tank. Obtain the mean image of each of the eight orientations and save its coordinates in eigenspace. For a query image, classify it in eigenspace and calculate the distance from each of the orientations using a distance metric.

Since the required data for doing this was not available, we constructed an eigen-space using all the tank images available and obtained class means corresponding to front, side and back views. On classification of a query image, it got mapped to the correct class most of the times.

Eigenspace Construction

Instead of directly obtaining the eigenvectors of an $N^2 \times N^2$ covariance matrix, the first M (M is the number of training vectors) eigenvectors can be obtained by first calculating the eigenvectors of an $M \times M$ covariance matrix of the transpose of the image data and then obtaining the image eigenvectors by taking linear combinations of the training images weighted by the eigenvector component as described in [4, 5, 6].

Scale and Intensity Invariance

Since eigenspace classification is sensitive to scale variations, all images were scaled down to a fixed size before classifying and since the image chip contained the tank only, scaling down the image to a fixed size implied obtaining the tank image of a fixed size. Both for eigenspace construction and classification, the images were normalized by their total energy.

Distance Metrics

The distance metric can either be the simple Euclidean distance (ED) or the distance along each component normalized by the corresponding eigenvalue (ND). The latter gives better results since it gives more weight to those directions where the noise variance is lower.

A better solution is to obtain variance along each component in each class and calculate the distance from a particular class using those eigenvectors which have low variance in that class but overall high variance in eigenspace. For scaling the distance from class k, the variance in class k is used rather than the global eigenvalue for scaling. In this way, the intraclass variance can be suppressed while the interclass variance can be emphasized. We call this measure the class normalized distance (CND). All the three distance metrics for some sample images are tabulated in Table 5.2. It can be seen from the results that the CND is the best metric for classification for the reason stated above.

Table 5.2. Eigenspace Classification Results

Class	Class Normalized Distance(CND)	Euclidean Distance(ED)	Normalized Distance(ND)
tank2	4	10	24
tank6	15	20	35
tank9	15	19	28
btank12	30	44	56
sftank5	17	21	31

5.3.2 Focus of Expansion Estimation

The focus of expansion is the point in the image sequence of a moving body from which all the motion vectors appear to diverge. The FOE may not always lie inside the image boundary. Mathematically, the FOE(X_f,Y_f) is defined as

$$X_f = \frac{T_x}{T_z}, \tag{5.15}$$

$$Y_f = \frac{T_y}{T_z}, \tag{5.16}$$

where T_x,T_y, T_z are the translational motion vectors. Therefore, assuming the ground is the X-Z plane, the direction of motion (velocity angle) of the tank is given by

$$\theta = \tan^{-1}\frac{V_x}{V_z} = \tan^{-1}\frac{T_x}{T_z} = \tan^{-1} X_f. \tag{5.17}$$

A modification of the partial search technique for FOE estimation developed by Srinivasan in [7] was used. The equations to be solved are

$$u(x,y) = -(x - x_f)h(x,y) + xy\omega_x - (1 + x^2)\omega_y + y\omega_z, \tag{5.18}$$

$$v(x,y) = -(y - y_f)h(x,y) + (1 + y^2)\omega_x - xy\omega_y - x\omega_x, \tag{5.19}$$

where $u(x,y)$ and $v(x,y)$ are the optical flow estimates at (x,y), x_f, y_f are the x and y coordinates of the FOE in the image(pixel) coordinates, $h(x,y)$ is the inverse of the depth at point (x,y) and ω_x, ω_y, ω_z are the x-, y-, and z- direction rotations. The FOE estimation algorithm requires calculation of the optical flow which is done using the overlapped basis functions technique developed by Srinivasan and Chellappa [8]. The FOE in the world coordinates is given by

$$X_f = \frac{x_f + x_{\text{offset}} - (N-1)/2}{f} \tag{5.20}$$

which is used for direction of motion calculation in (5.17).

In the modified partial search FOE estimation technique, we use only the optical flow estimates of the part of the image containing the moving object since the flow estimates of the background are unreliable. Since the size of the image for which optical flow is available is smaller, the FOE is also searched over a smaller region. This speeds up the FOE calculation. Also, the FOE estimate of the previous frame can be used to select an initial offset for the FOE in the current frame to speed up the FOE calculation over successive frames. The direction of motion is estimated at regular intervals and we keep waiting if the tank is approaching towards the camera. The frame at which the tank takes a turn away from the camera or the tank's size increases above a certain threshold could be chosen as the "best view" in this approach.

5.3.3 Size-Based Best-View Selection

Since the application requires best view selection and compression to be done in realtime on hardware with low computing power, we need very simple techniques for best-view selection. Thus, the final algorithm that was implemented simply waits till the size of the image chips exceeds a predefined threshold. If the size starts decreasing, it simply chooses the maximum size frame in the last 90 frames and sends it. Actually a single frame buffer is used to store the maximum sized chip in the last few frames, and as soon as the size starts decreasing or increases beyond the threshold the stored frame is compressed and transmitted. The algorithm rejects chips which are very close to the image boundary (the tank may not be complete). Spurious frames (produced as a result of wrong object detection) are also rejected based on thresholding the height-to-width ratio.

5.3.4 Best-View Selection Results

In this section, we present the results of best-view selection using the three approaches discussed above.

Eigenspace Classification

An eigenspace of front, side, and back views of various tanks is constructed and the class means for each class are precalculated. In Table 5.2, results for distances from the perfect side view ("tank2") class are shown.

Figure 5.6 shows the mean image of the tank2 class. The distance of a query tank2 image (side view, see Figure 5.5) is a minimum. Distance of tank6 (side-back view) shown in Figure 5.8 is higher than tank2 but lower than tank12 (back view) shown in Fig. 5.7. Hence tank2 is the "best-view" in this case. As can be seen from Table 5.2, the CND (class normalized distance) has the maximum variation (4 for perfect side view and 30 for back view), and thus it is the best metric for classification among the metrics used.

The eigenspace classification is not a very good method for best-view selection because it is sensitive to the lighting conditions, the type of IR sensor

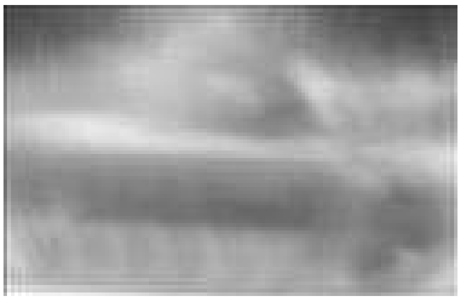


Figure 5.6. Mean image of the itank2 class.

Figure 5.7. Query image 3: tank12 (back view).

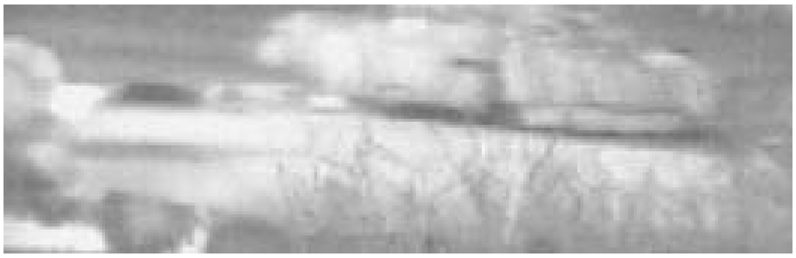


Figure 5.8. Query image 2: tank6 (back-side view).

used, and to the scale of the image. If the actual sensor is different from the sensor used in the database, classification could fail. Moreover a very large database of tank images in various poses is required for a robust eigenspace construction which may not be feasible.

Focus of Expansion Estimation

The FOE estimation algorithm was run on IR video sequences of the tanks. Since most of the tanks are moving almost horizontally in front of the camera in the test sequences, the FOE values are very large (tending to infinity). The FOE in pixel coordinates for a sample sequence is shown in Table 5.3.

Table 5.3. FOE estimates using the optical flow shown in Figure 5.10.

Frame No.	FOE_x	FOE_y
121	−244	145
123	−369	152
125	−201	140
127	−253	136
129	−553	110
131	−583	110

5.9.

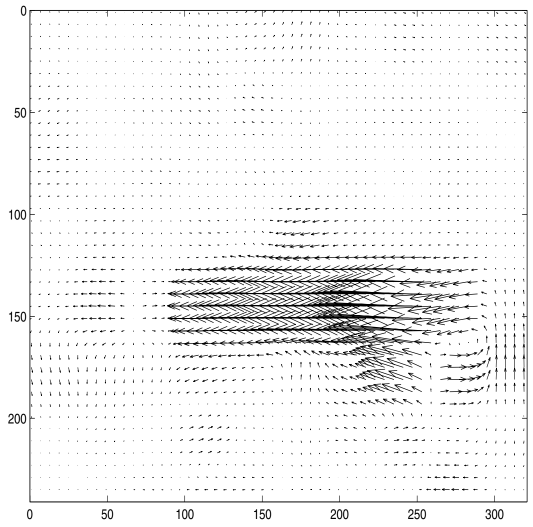


Figure 5.9. Optical flow estimate of a typical frame (125) for FOE Estimation.

The optical flow estimate for the full frame 125 is shown in Figure The region of significant motion that is segmented out and used for FOE estimation finally is shown in Figure 5.10. The FOE estimation technique is not very suitable for our application because both optical flow calculation and FOE estimation are computationally intensive. Also for the IR images, the optical flow estimates are not very accurate and as a result the FOE estimates are also not accurate enough.

Size-Based Best-View Selection

Figure 5.11 shows a best view selected by this approach.

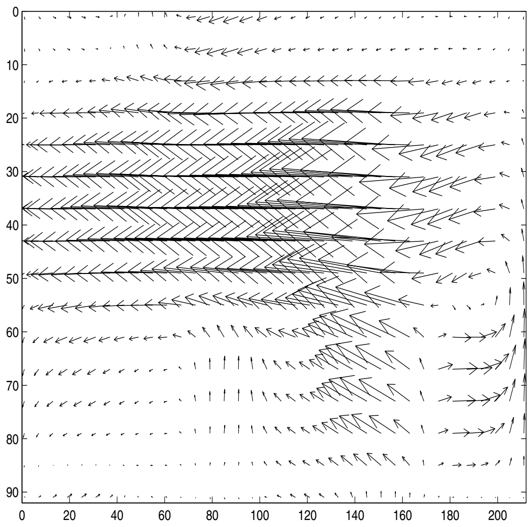


Figure 5.10. Segmented optical flow estimate for frame 125 which is finally used in FOE estimation (magnified view).

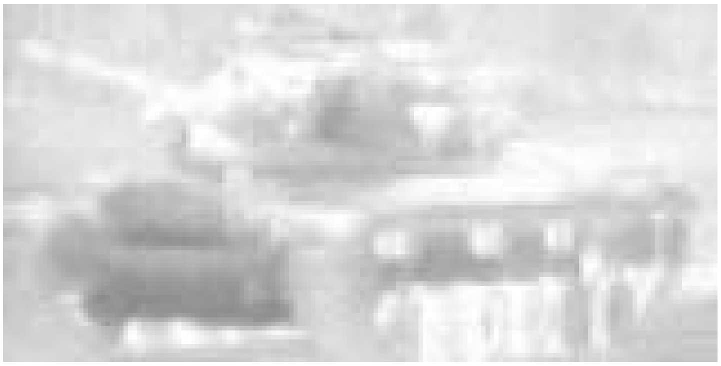


Figure 5.11. Size-based best-view selection.

5.4 Compression

For long-range objects, since the detected image chip is already very small, compression is not really necessary. Also since the object chip contrast can be very low compared to the background, it is preferable to just transmit the binary image of the object (instead of transmitting a grayscale one) which itself provides an 8 : 1 compression ratio. If more computational power is available, the binary image could be compressed by run-length coding else just the raw bits can be transmitted.

For objects at short range (which are large), the image chip chosen by the "best-view selection" algorithm has to be compressed before transmission.

Since the available channel bandwidth is low, we have tried to develop compression schemes which can provide very high compression ratios while at the same time maintaining a reasonable image quality. Since the algorithm is to be implemented in real-time using limited computing hardware, the computational complexity should be low. We have developed realtime algorithms for image compression in the wavelet domain. We first provide a background of existing image compression schemes and discuss the theoretical background for wavelet transforms and the compression techniques. Following that, we discuss the compression schemes implemented and compare against other techniques. We compare our scheme, combined zerotree and DPCM coding, against three existing schemes all of which have low computational complexity, viz. scalar quantization (SQ), zerotree coding, and DPCM coding. The more efficient compression schemes like JPEG and LZW are not compared here because they have a much higher computational cost associated with their implementation. We then provide a performance analysis of the coding schemes. Finally, experimental results are provided followed by computational cost analysis. We use PSNR (peak signal-to-noise ratio) which is a standard metric for image compression schemes to compare decompressed image quality.

5.4.1 Previous Work

A wavelet zerotree coding scheme for compression is presented in [9], but since it uses vector quantization (VQ), it cannot be used for our application due to high computational cost. Reference [10] presents an embedded predictive wavelet image coder. But it uses arithmetic coding which is not suitable for implementation on a embedded processor such as Hitachi's SH4. So we have developed algorithms using the Haar wavelet transform followed by noniterative zerotree coding [9] and 2D-DPCM for all subbands. The computational complexity of these algorithms is only marginally higher than simple scalar quantization (SQ) of the entire image.

5.4.2 Wavelet Transform Properties

The wavelet transform is an atomic decomposition that represents a signal in terms of shifted and dilated versions of a prototype bandpass wavelet function and shifted versions of a low-pass scaling function. In discrete time, the bandpass wavelet function is a high-pass filter at different scales and the scaling function is a low-pass filter. The image is low-pass and high-pass filtered first along rows and then along columns to generate LL, LH, HL, and HH images each of which is subsampled by two. This process is repeated on the subsampled LL image. The wavelet transform has the following properties which make it suitable for compression.

- *Multiresolution:* The image is decomposed into wavelets at N scales, and only the top few coarsest scales need to be transmitted to obtain a reasonable image quality. Depending on the available channel bandwidth, more finer scale coefficients can be transmitted to improve the reconstructed image quality.

- *Entropy Reduction:* The wavelet transform of a real image generates a large number of small coefficients (which can be set to zero) and a small number of large coefficients which can be encoded. This property is based on the fact that a real world image will not have information in all frequencies at all points in space. At most points except edges, the higher-frequency information is almost zero.
- *Clustering and Persistence:* The wavelet transform attempts to decorrelate the image, but the decorrelation is not complete (since the filters are constant, not data dependent). There is a residual dependency between adjacent coefficients at the same scale (clustering) and between coefficients in adjacent scales but the same spatial location (persistence). Our coding schemes attempt to remove these correlations in the image.

5.4.3 A-DPCM for Scaling Coefficient(LL) Encoding

The scaling coefficients (LL subband) contain the maximum information and thus more bits are allocated for its encoding. But it is also the most highly correlated subband and this fact can be exploited to maximize compression. An adaptive-DPCM scheme is used for encoding the LL subband. The current pixel is predicted based on a linear combination of three causal nearest neighbors. The predicted value of the pixel, $\hat{X}$ is obtained as

$$\hat{X} = l(\bar{Q}) = \bar{w}.\bar{Q} = \sum w_k Q_k. \tag{5.21}$$

The predictor coefficients $\bar{w}$ are calculated to minimize the mean squared prediction error as

$$\bar{w} = E(\bar{Q}\bar{Q}^T)^{-1}E(X.\bar{Q}). \tag{5.22}$$

where X is the pixel to be predicted, Q_i are the quantities based on which the pixel would be predicted (in this case the nearest neighbors), and $\bar{w}$ are the predictor coefficients. Instead of quantizing the pixel value, the error between the actual and the predicted value $(X - \hat{X})$ is quantized, which requires fewer bits since the error would be much smaller than the original pixel value if the prediction is good. Calculation of LMSE predictor coefficients can be done offline on a set of similar images.

5.4.4 Zerotree Coding

In multiresolution wavelet decomposition, each coefficient X_i, except those in the LL subband and the three highest subbands, is exactly related to 2×2 coefficients of the immediately higher subband. These four *children* coefficients correspond to the same orientation and spatial location as the parent coefficient X_i. Each of the four children coefficients is in turn related to 2×2

coefficients in the next higher subband, and so on. These coefficients are collectively called the descendants of the parent X_i. All coefficients with magnitude less than threshold T are called *insignificant coefficients* and their collection is known as a zerotree. In order to obtain a real-time implementation [9], the search for insignificant coefficients is started in the lowest-frequency subbands except baseband and continued in higher-frequency subbands. When a coefficient is decided as insignificant and set to zero, all its descendants are also set to zero. Thus, one needs to transmit only the escape code for the zerotree root vector besides encoding the nonzero coefficients. The zerotree root positions at each scale can be encoded efficiently using the Run-length Coding(RLC). The non-zero coefficients can be scalar quantized and transmitted. This type of simple threshold based zerotree coding, RLC and SQ are computationally simple algorithms for hardware implementation.

However, it is possible that there are significant descendants even though their parent is insignificant. These mispredictions are inevitable in a non-iterative search method, but the conditional probability of this happening is very small, as discussed in [9]. The misprediction error can be reduced by predicting the value of a "zero" coefficient based on its nearest nonzero neighbors (causal and noncausal) while decoding.

5.4.5 DPCM on Wavelet Coefficients

The zerotree coding exploits the persistence property of wavelet coefficients. But there is also a residual correlation in the high-frequency subbands, especially the LH and the HL bands with horizontal and vertical neighbor, respectively. Hence applying an A-DPCM scheme like that discussed for the LL subband can give additional compression. Also while obtaining the prediction value for the current pixel we can exploit both clustering and persistence properties, i.e., obtain a prediction for the current pixel based on its vertical (for HL) or horizontal neighbor(for LH) and its parent coefficient. Again as in the case of the LL subband, the predictor coefficients can be calculated offline for a sequence of similar images using (5.22). In this case the predictors are the parent coefficient at the same spatial location and the horizontal or vertical neighbor. This scheme is motivated by a similar scheme discussed in [10] for visual images.

5.4.6 Compression Schemes

Four different schemes for encoding the wavelet coefficients were compared. In all cases the LL subband was encoded using the A-DPCM scheme discussed in Section 5.4.3.

Scalar Quantization

This scheme involves scalar quantization (SQ) of the wavelet coefficients and DPCM encoding of the LL coefficients. Variable bits are allocated to the subbands based on their variances as discussed in [9].

Zerotree Coding

Zerotree coding is applied as discussed in Section 5.4.4. This not only gives a significantly reduced bits per pixel (BPP) value than the SQ (as expected), but also gives reduced MSE value compared to SQ. The reason is that the quantization error is higher than the thresholding error for high-frequency subbands which are coarsely quantized.

2D Predictive DPCM on Wavelet Subbands

Only DPCM coding is applied as discussed in Section 5.4.5 with no zerotree coding. The performance of this scheme is bad because the “noisy data” close to zero, cannot be predicted correctly and hence the prediction errors obtained are sometimes larger than the original pixel value. Hence the MSE is significantly higher.

Combined Zerotree and DPCM Coding

We propose to combine zerotree coding and the DPCM encoding (ZT/DPCM) of wavelet coefficients to achieve maximal compression. First a simple zerotree coding is applied to the subbands. This is followed by DPCM coding of the “nonzeroed” coefficients. The value of a “zeroed” neighbor is predicted as follows. If we predict $C_{x,y}$ based on $C_{x-1,y}$ which is “zeroed” and the zeroing threshold is T, we estimate $C_{x-1,y}$ as follows

$$S = C_{x-2,y} + C_{x-1,y-1},$$

$$\hat{C}_{x-1,y} = \begin{cases} 0 & \text{if } S = 0, \\ -T & \text{if } S < 0, \\ +T & \text{if } S > 0. \end{cases}$$

This is based on the assumption that since the next coefficient is nonzero, the previous one would be close to the threshold. DPCM combined with zerotree coding works much better because the noisy coefficients have been set to “zero” and we do not try to predict their value. The prediction model is applicable only to those subbands for which enough (> 2) bits have been allocated and the prediction error energy obtained while calculating the predictor coefficients is less than 25% of the subband energy. For other subbands, SQ is used.

5.4.7 Performance Analysis

The aim of any compression scheme is to minimize the mean squared error (maximize the PSNR) and the entropy per pixel (entropy rate, ER). In SQ, each pixel is coded independently and the correlation in the image is not exploited. So the entropy rate is higher. Entropy rate will be minimized if each pixel is coded based on all past pixels on which it depends, i.e. (for a 1D signal)

$$h(X_n) > h(X_n|X_{n-1}) > h(X_n|X_{n-1}, ..., 1). \tag{5.23}$$

If we assume a one-step Markov model,

$$h(X_n|X_{n-1} \cdots 1) = h(X_n|X_{n-1}) \tag{5.24}$$

For 2D data (assuming a Markov random field model), this translates to $X_{n,n}$ depending only on $X_{n-1,n}$ and $X_{n,n-1}$.

The quantization MSE will be minimized for a given bit rate if the mean square value of the quantity to be quantized is minimum. Hence instead of quantizing $X_{n,n}$, in 2D predictive DPCM, we predict a value $(\hat{X_{n,n}})$ based on past values and quantize the difference $(X_{n,n} - \hat{X_{n,n}})$. $\hat{X_{n,n}}$ is calculated as discussed in (5.21) to minimize $E[X_{n,n} - \hat{X_{n,n}}]^2$ and hence the quantization MSE over all linear estimators. Also for a given quantization step size (fixed MSE), reduced data variance means reduced entropy.

In zerotree coding, the PSNR is higher than SQ because the zeroing error is lower than the quantization error for high-frequency subbands which are coarsely quantized. Zeroing also reduces entropy since the number of symbols to be compressed is reduced. The 2D MRF model with second-order dependencies (correlations) fits well for the LL subband, but does not fit well for the wavelet subbands and the prediction fails completely for very small values (only noise). This is the reason why DPCM on wavelet subbands gives the worst PSNR values. Combined zerotree and DPCM (ZT/DPCM) gives best results in terms of PSNR and entropy rate. The noisy coefficients are zeroed and hence not predicted and thus the quantization error remains low. Because of LMSE prediction, the entropy is minimum and zerotree coding further reduces the entropy rate by reducing the number of symbols to be coded.

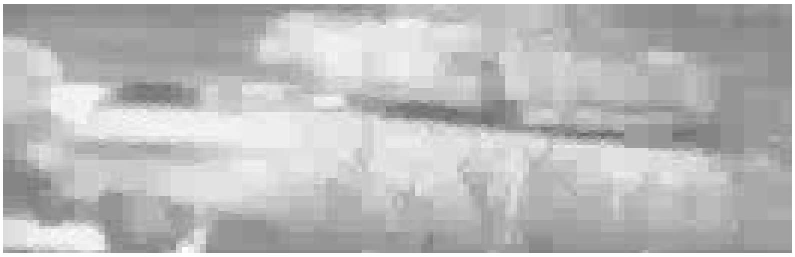
5.4.8 Image Compression Results

Various types of low-pass and high-pass filters satisfying the prefect reconstruction property can be used. In our implementation, the Haar transform is used because of its simplicity and ease of hardware implementation. Using a longer length filter will not be useful because the low-pass filter will tend to average over a very large area and thus lose the localization property of wavelet transforms. The Haar wavelet is built using a two-tap low-pass filter $[1, 1]$ and a two-tap high-pass filter $[1, -1]$.

Figure 5.5 shows the original tank2 image and Figure 5.12 shows the compressed tank2 images using combined zerotree/DPCM coding and zerotree coding.Table 5.4 shows the compression results for two sample IR images and the Lena image.

Table 5.4. The bpp, PSNR[$10 \log_{10} 255^2/MSE$] and entropy for three sample images using zerotree (ZT), zerotree and DPCM (ZT/DPCM), scalar quantization (SQ) and only DPCM coding schemes.

Image	Coder	Total BPP	PSNR	Entropy (Non-zero)	RLC BPP
tank2	ZT/DPCM	0.5628	31.73	0.0920	0.2757
	ZT	0.5628	31.61	0.2112	0.2757
tank12	ZT/DPCM	0.5232	31.75	0.0880	0.2649
	ZT	0.5232	31.65	0.2045	0.2649
lena	ZT/DPCM	0.5066	29.40	0.0851	0.2286
	ZT	0.5066	29.30	0.1542	0.2286
	SQ	0.7947	13.07	0.3156	
	DPCM	0.7947	25.34	0.1508	



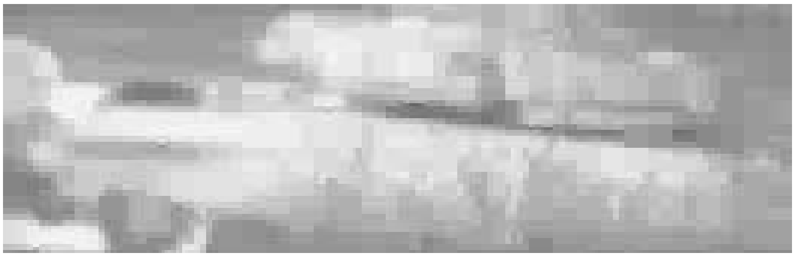


Figure 5.12. Compressed tank2 by (a) combined zerotree and DPCM coding (b) zerotree coding.

The results have been obtained by allocating a total of 0.5 BPP to various subbands proportional to the logarithm of their variances. Since for this low value of BPP, the lowermost subbands get negative bits allocated to them (which are set to zero), the actual BPP obtained is higher than 0.5.

In Table 5.4 we have compared the total BPP, PSNR, BPP, for RLC coding and entropy rate for three different images (two from the IR sequence and the Lena image). The entropy rate is the minimum bits/pixel that can be theoretically achieved for the image. Due to hardware constraints we have not implemented any form of entropy coding (Arithmetic/Huffman).

As can be seen from the values of RLC BPP, almost half the bits are used up in encoding the zerotree information. More efficient binary encoding schemes can be employed to reduce this value and this could considerably improve the BPP. Also, in most cases the combined zerotree and DPCM scheme gives the best results both in terms of PSNR values and entropy. In some cases like the Lena image, the PSNR is higher for simple zerotree coding, but the entropy of the combined scheme is less. Ideally, one would assume that applying a DPCM encoding would cause a significant reduction in PSNR. This is definitely true for the LL subband, but the reduction for higher-frequency subbands is not so much because of lesser correlation. Another reason is the uncertainty in predicting the value of a pixel based on a neighboring "zeroed" pixel. We are experimenting with better methods to improve the prediction model for combined zerotree and DPCM encoding.

The BPP without zerotree coding is consistently higher for all the images and hence the zerotree coding is advantageous even though half the BPP is used up in RLC coding of the zerotree. Also surprisingly, the PSNR for SQ is lower than for zerotree coding even when the BPP is higher. The reason for this is that the quantization error in SQ is higher than the zeroing error for the high-frequency subbands which are coarsely quantized. From Table 5.4, we observe that DPCM encoding without zeroing is the worst scheme. This is because a lot of the coefficients below the zeroing threshold in the high-frequency subband are actually "noise." Thus, in DPCM we are trying to predict the value of these "noise" pixels or use them to predict other pixels and hence the predictions are very bad, thus leading to a higher PSNR. Hence the DPCM model fails in the absence of zerotree coding, while it provides a reasonably good model for the image when combined with zerotree coding.

5.4.9 Computational Complexity: Hardware Issues

The entire coding scheme is computationally very simple. The Haar wavelet transform involves a single addition operation per pixel. Zerotree coding requires one comparison to a threshold (per pixel) and a multiplication by two (a shift operation) to calculate the descendant position. The DPCM operation involves three real multiplications and two additions to calculate the predicted vale and one subtraction to obtain the error. Run-length coding is again a counting operation requiring one addition per pixel. Thus the additional cost over scalar quantizing the entire image is (which is the minimum one has to do to compress an image) is three multiplications and a few additions per pixel. For an $N^2 * N^2$ image with a three-level wavelet decomposition, the additional cost for our scheme is given by

$$AC_{\text{Haar}} = (N^2 + N^2/4 + N^2/16)C_A, \tag{5.25}$$

$$AC_{\text{Zeroing}} = (N^2 + N^2/4 + N^2/16)(C_C + C_S), \tag{5.26}$$

$$AC_{RLC} = N^2 C_A, \tag{5.27}$$

$$AC_{DPCM} = N^2(3C_M + 3C_A), \tag{5.28}$$

where AC is additional cost. C_A is the cost for one addition, C_M is the cost for one multiplication, C_C is the cost for one comparison, and C_S is cost for one shift (multiply by two) operation . Since comparison and shift are single operations, $C_C = C_S = 1$. Hence, total additional cost is

$$AC = (N^2 + N^2/4 + N^2/16)(C_A + 2) + N^2(3C_M + 4C_A). \tag{5.29}$$

5.5 Performance Evaluation of Algorithms

We have developed a C/C++ implementation of the object detection and compression/decompression algorithms. The C/C++ implementation of image compression part of the system currently uses zerotree coding with SQ and RLC. Performance evaluation of the C/C++ code was done using Hitachi's SH4–7751 microprocessor. SH4–7751 is a high-performance superscalar RISC microprocessor designed for embedded applications. Some of the features of SH4 include 167-mHZ clock frequency, upto 360 MIPS capability, and on-chip cache for instruction and data. More information on SH4 can be found at http://semiconductors.hitachi.com. We evaluated the code performance using the Hitachi Embedded Workshop (HEW) which is an SH4 simulator provided by Hitachi. The results were obtained using the profile utility of HEW. Note that these results are obtained by directly cross-compiling the C code using Hitachi's cross-compiler. Hence, certain features of SH4 architecture (such as floating point unit) which can enhance real-time performance are not used. In practice, the performance can be improved by designing the assembly language code for computationally expensive procedures so as to take advantage of such features of the object microprocessor. The performance results of the algorithms in terms of code size, run-time memory requirement and instruction cycles on SH4 are as follows.

5.5.1 MTI Algorithm

The code size required for the MTI algorithm was 9.8 KB. For an input frame size of 120×160 maximum run-time memory required was 1.086 MB. Cycles required per frame were equal to 8.5 M which corresponds to 19.54 frames per second (30 fps frame rate).

5.5.2 Image Chipping Algorithm

Motion Detection and Best-View Selection

For motion detection using the technique described in Section 5.2.3 and best-view selection, the code size required was 16 KB. The run-time memory required for frame size of 240×320 was 2.4 MB and the processing frame rate (for 120×160 frame size) was 4–5 frames per second.

Compression

The compression part of algorithm required a code size of 30.5 KB. For a typical image chip of size 60×120, run-time memory required was 0.75 MB and cycles on SH4 needed to compress the images was 31.5 M which corresponds to run-time of 0.188 sec. The run-time is calculated as cycles/clock frequency.

Decompression

Decompression required a code size of 21.5 KB and for a typical image chip of size 60×120, run-time memory required was 0.75 MB. Note that this memory requirement will increase with chip size. Table 5.5 gives the instruction cycles required for different PSNR values of reconstructed sample tank chip.

Table 5.5. Cycles and run-time required for decompression of a sample target chip of size 60×120 for different PSNR.

PSNR(db)	Cycles(million)	Runtime on SH4(sec)
30.21	20.73	0.120
34.36	21.00	0.125

5.6 Conclusions

A novel variance-analysis-based algorithm has been presented for moving object detection. The algorithm is especially suitable for detection of long-range, small, slow moving objects. An initial test on several IR sequences has revealed high detection rates with low false alarms for vehicles at distances of several kilometers. The algorithm is robust and requires no tuning of parameters by the operator. For objects at short distances, technique based on detecting and tracking image singularities have been discussed. Also for objects at short range, methods for best-view selection and compression were presented. Three different approaches to the best-view selection problem were compared. The approach based on classification in eigenspace is suitable when multiple views of the same object are available but has limitations when significant scale and illumination variations are present. The FOE estimation approach would be a useful method for direction of motion calculation but is computationally expensive. The size-based technique is the fastest and gives reasonable results and is used in our implementation. A new scheme combining noniterative zerotree coding with 2D DPCM for LL and for the high-frequency subbands was presented. This method gives better results than simple scalar quantization and simple zerotree coding both in terms of BPP and PSNR at a marginally increased computational cost. The algorithms were implemented in C and their performance results on SH4 processor were presented.

The image compression results can be further improved by using some form of entropy coding (since the entropy rate of our scheme is significantly lower) and by replacing the run length coding method with more efficient binary coding techniques. Also the zeroing thresholds can be calculated for the required PSNR values.

References

[1] Shekarforoush, H., Chellappa, R.: A multi-fractal formalism for stabilization, object detection and tracking in FLIR sequences. In: International Conference on Image Processing. Volume 3. (2000) 78–81

[2] Elgammal, A., Harwood, D., Davis, L.: Nonparametric background model for background subtraction. In: Proc. 6th European Conf. Computer Vision. Volume 2. (2000) 751–767

[3] Murase, S., Nayar, S.: Visual learning and recognition of 3-d objects from appearance. International J. Computer Vision **14** (1995) 5–24

[4] Turk, M., Pentland, A.: Eigenfaces for recognition. J. Cognitive Neuroscience **3** (1991) 71–86

[5] Turk, M., Pentland, A.: Face recognition using eigenfaces. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition. (1991) 586–591

[6] Sirovich, L., Kirby, M.: Low dimensional procedures for the characterization of human faces. J. Optical Society of America **4** (1987) 519–524

[7] Srinivasan, S.: Extracting structure from optical flow using fast error search technique. International J. Computer Vision **37** (2000) 203–230

[8] Srinivasan, S., Chellappa, R.: Noise-resilient optical flow estimation using overlapped basis functions. J. Optical Society of America **16** (1999) 493–509

[9] Paek, S., Kim, L.: A real-time wavelet vector quantization algorithm and its vlsi architecture. IEEE Transactions on CSVT **10** (2000) 475–489

[10] Buccigrossi, R., Simoncelli, E.: Image compression via joint statistical characterization in the wavelet domain. IEEE Transactions on Image Processing **8** (1999) 1688–1700