A Semantic Event-Detection Approach and Its Application to Detecting Hunts in Wildlife Video

Niels Haering, Richard J. Qian, and M. Ibrahim Sezan, *Senior Member, IEEE*

***Abstract*—We propose a three-level video-event detection methodology and apply it to animal-hunt detection in wildlife documentaries. The first level extracts color, texture, and motion features, and detects shot boundaries and moving object blobs. The mid-level employs a neural network to determine the object class of the moving object blobs. This level also generates shot descriptors that combine features from the first level and inferences from the mid-level. The shot descriptors are then used by the domain-specific inference process at the third level to detect video segments that match the user-defined event model. The proposed approach has been applied to the detection of hunts in wildlife documentaries. Our method can be applied to different events by adapting the classifier at the intermediate level and by specifying a new event model at the highest level. Event-based video indexing, summarization, and browsing are among the applications of the proposed approach.**

***Index Terms*—Browsing and visualization, content-based indexing and retrieval, video content analysis.**

I. Introduction

THE AMOUNT of video information that can be accessed and consumed from people's living rooms has been ever increasing. This trend may be further accelerated due to the convergence of both technology and functionalities supported by future television receivers and personal computers. To obtain the information that is of interest and to provide better entertainment, tools are needed to help users extract relevant content and to effectively navigate through the large amount of available video information. For ordinary users, such tools may also have to satisfy the following requirements: 1) they should be easy to use in terms of operations and 2) they should be easy to understand and predict in terms of behaviors.

Existing content-based video indexing and retrieval methods do not seem to provide the tools called for in the above applications. Most of those methods may be classified into the following three categories: 1) syntactic structurization of video; 2) video classification; and 3) extraction of semantics. The work in the first category has concentrated on: 1) shot boundary detection and key frame extraction, e.g., [1], [34]; 2) shot clustering, e.g., [32]; 3) table of content creation, e.g., [9]; 4) video summarization, e.g., [22], [35]; and 5) video skimming [28]. These methods are in general computationally simple and their performance is relatively robust. Their results, however, may not necessarily be semantically meaningful or relevant, since they do not attempt to model and estimate the semantic content of the video. For consumer-oriented applications, semantically irrelevant results may distract the user and lead to frustrating search or browsing experience. The work in the second category tries to classify video sequences into categories such as news, sports, action movies, close-ups, crowd, etc. [19], [29]. These methods provide classification results which may facilitate users to browse video sequences at a coarse level. Video-content analysis at a finer level is probably needed to more effectively help users find what they are looking for. In fact, consumers often express their search items in terms of more exact semantic labels, such as keywords describing objects, actions, and events. Work in the third category has been mostly specific to particular domains. For example, methods have been proposed to detect events in: 1) football games [18]; 2) soccer games [33]; 3) basketball games [27]; 4) baseball games [20]; and 5) sites under surveillance [4]. The advantages of these methods include that the detected events are semantically meaningful and usually significant to users. The major disadvantage, however, is that many of these methods are heavily dependent on specific artifacts such as editing patterns in the broadcast programs, which makes them difficult to extend for the detection of other events. A more general method for the detection of events [17] uses "Multijects" that are composed of sequences of low-level features of multiple modalities, such as audio, video, and text.

Query-by-sketch or query-by-example methods have also been proposed recently [7], [36] to detect motion events. The advantage of these methods is that they are domain independent, and therefore may be useful for different applications. For consumer applications, however, sketching needs cumbersome input devices, specifying a query sketch may take undue amounts of time and learning the sketch conventions may discourage users from using such tools.

In this paper, we propose a computational method and several algorithmic components toward an extensible solution to semantic event detection. The automated event-detection algorithm facilitates the detection of semantically significant events in their video content and helps generate semantically meaningful highlights for fast browsing. In contrast to most existing event-detection work, our goal is to develop an extensible computational approach which may be adapted to detect different events in different domains. To achieve this goal, we propose a three-level video event-detection algorithm. The first level extracts color, texture, and motion features, and detects shot boundaries and moving-object blobs. The mid-level employs

Manuscript received December 15, 1998; revised December 28, 1999. This paper was recommended by Associate Editor S.-F. Chang.

N. Haering is with Diamondback Vision, Inc., Reston, VA 20191 USA (e-mail: niels@dbvision.net).

R. J. Qian is with Intel Labs, Hillsboro, OR 97124 USA (e-mail: richard.j.qian@intel.com).

M. I. Sezan is with Sharp Labs of America, Camas, WA 98607 USA (e-mail: sezan@sharplabs.com).

Publisher Item Identifier S 1051-8215(00)07552-2.

a neural network to determine the object class of the moving blobs. This level also generates shot descriptors that combine features from the first level and inferences from the mid-level. The shot descriptors are then used by the domain-specific inference process at the third level to detect video segments that match the user-defined event model. To test the effectiveness of our algorithm, we have applied it to detect animal hunt events in wildlife documentaries. In our implementation, we do not attempt to detect the stalking phase that precedes many hunts; rather we aim to detect the swift or rapid chase of a fleeing or running animal. Since hunts are among the most interesting events in a wildlife program, the detected hunt segments can be composed into a program highlight sequence. The proposed approach can be applied to different domains by adapting the mid- and high-level inference processes while directly utilizing the results from the low-level feature extraction processes [15].

In the following section, we describe the proposed computational method and its algorithmic components. In Section III, we describe implementational details and present experimental results obtained as we have applied the proposed algorithm to the detection of animal hunt events in a number of commercially available wildlife video tapes. Finally, in Section IV, we summarize our work and discuss some future directions.

II. Methodology

The problem of detecting semantic events in video, e.g., hunts in wildlife video, can be solved by a three-level approach, as shown in Fig. 1. At the lowest level, the input video is decomposed into shots, global motion is estimated, and color and texture features are extracted. At this level, motion blobs, i.e., areas due to independent object motion, are also detected after the global motion is compensated.

At the intermediate level, the detected motion blobs are classified as moving-object regions by a neural network. The network uses the color and texture features extracted at the lower level, and performs a crude classification of image regions into sky, grass, tree, rock, animal, etc. This level also generates shot summaries which describe each individual shot in terms of intermediate-level descriptors.

At the highest level the generated shot summaries are analyzed and the presence of the events of interest, e.g., hunts, are detected based on an event-inference model which incorporates domain-specific knowledge.

The feature extraction at the lowest level is entirely domain and event independent. The classifier at the intermediate level is only domain dependent. The event-detection level is event specific (it describes and defines the event of interest). In our hunt-detection example, we limited our domain to wildlife documentaries and the event to animal hunts. Since the submission of this article, we have shown that the same method can be used to detect landing and rocket launch events in videos of different domains [15].

A. Global-Motion Estimation and Motion-Blob Detection

We assume that the motion in many videos can be decomposed into a global (or background) motion component and independent object motion components. We further assume that

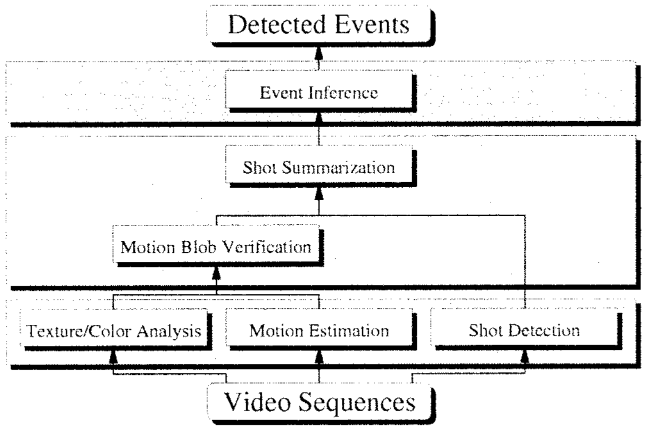


Fig. 1. Flowchart of our method.

the global motion can be modeled by a three-parameter system allowing only for zoom, horizontal, and vertical translation

$$u(x, y) = a_0 + a_2 x$$
$$v(x, y) = a_1 + a_2 y.$$

We correlate patches from consecutive frames to estimate the global-motion parameters. To improve the robustness and reduce the computation of the estimation process, we use a five-level pyramid of reduced resolution representation of each frame. At each level of the pyramid, we consider matches from a 5×5 neighborhood around the location of the patch in the source frame, enabling a maximum matching distance of 62 pixels.

At the lowest level of the pyramid, i.e., the full-resolution representation of the frame, the patches used for matching are of size 64×64. Patches from uniform areas often result in erroneous motion estimates. To reduce this effect we discard patches with insufficient "texture." We use a 2-D variance measure to determine the "amount of texture"

$$\text{var}_x = \frac{1}{m}\sum_{x=0}^{m-1}\left(\frac{1}{n}\sum_{y=0}^{n-1}(p(x, y) - p(x, \cdot))^2 - q_x\right)^2$$
$$\text{var}_y = \frac{1}{n}\sum_{y=0}^{n-1}\left(\frac{1}{m}\sum_{x=0}^{m-1}(p(x, y) - p(\cdot, y))^2 - q_y\right)^2$$

where p is an $m \times n$ image patch, $p(x, \cdot)$ and $p(\cdot, y)$ are the means of the xth column and yth row of p, and q_x and q_y are the means of $(1/n)\sum_{y=0}^{n-1}((p(x, y) - p(x, \cdot))^2$ and $(1/m)\sum_{x=0}^{m-1}(p(x, y) - p(\cdot, y))^2$ for all x and y within p, respectively.

We compute motion estimates at each of the four corners of a frame. Since the motion of the tracked objects often does not vary drastically between consecutive frames (i.e., their acceleration is small), we also use the previous best motion estimate to predict the location of the four patches in the next frame. A limited search in a 5×5 neighborhood around the predicted location, improves the motion estimates in many cases. Therefore, we obtain up to eight motion estimates, one pyramid based estimate for each of the four patch locations, and one for each of the four estimates based on a limited search around the predicted match locations. Since some patches may not pass the "texture" test, we may have fewer than eight motion estimates. The highest normalized dot product between a source patch P_1 and matched patch P_2 determines the "correct" global-motion estimate between the current and next frame. The normalized dot product is equal to the cosine of the angle (α) between the two patches (vectors) P_1, and P_2

$$\cos(\alpha)_{P_1,P_2} = \frac{\sum\limits_{i,j} P_1(i,\, j) P_2(i,\, j)}{\sum\limits_{i,j} P_1(i,\, j) \sum\limits_{i,j} P_2(i,\, j)}.$$

The estimated global-motion parameters are used to compensate for the background motion between two consecutive frames. The difference between the current- and the motion-compensated previous frame is then used to detect motion blobs. Areas with low residual differences are assumed to have motion values similar to those of the background and are ignored. The independent motion of foreground objects, on the other hand ,usually causes high residual differences between the current frame and the following motion compensated frame. We use a robust estimation technique developed in [26] to detect motion blobs. Based on the frame difference result, the algorithm constructs two 1-D histograms by projecting the frame difference map along its x and y direction, respectively. The histograms, therefore, represent the spatial distributions of the motion pixels along the corresponding axes. Fig. 2(a) illustrates an ideal frame difference map where there is only one textured elliptical moving object in the input sequence, and the corresponding projection histograms. The center position and size of a moving object in the video can be estimated from statistical measurements of the two 1-D projection histograms. To locate an object in the presence of multiple moving objects, a robust statistical estimation routine has been adopted and described below. Fig. 2(b) illustrates this recursive process.

The center position and size of an object in the image can be estimated based on statistical measurements derived from the two 1-D projection histograms. For example, a simple method estimates the center position and size of a dominant moving object in an input sequence using the sample means and standard deviations of the distributions. More specifically, let $h_x(i),\ i = 0,\, 1,\, \ldots,$ and $h_y(i),\ i = 0,\, 1,\, \ldots$ denote the elements in the projection histograms along the x and y direction, respectively. Then the object center position $(x_c,\, y_c)$ and object width and height $(w,\, h)$ may be estimated as

$$x_c = \frac{\sum\limits_i x_i h_x(i)}{\sum\limits_i h_x(i)}$$

$$y_c = \frac{\sum\limits_i y_i h_y(i)}{\sum\limits_i h_y(i)}$$

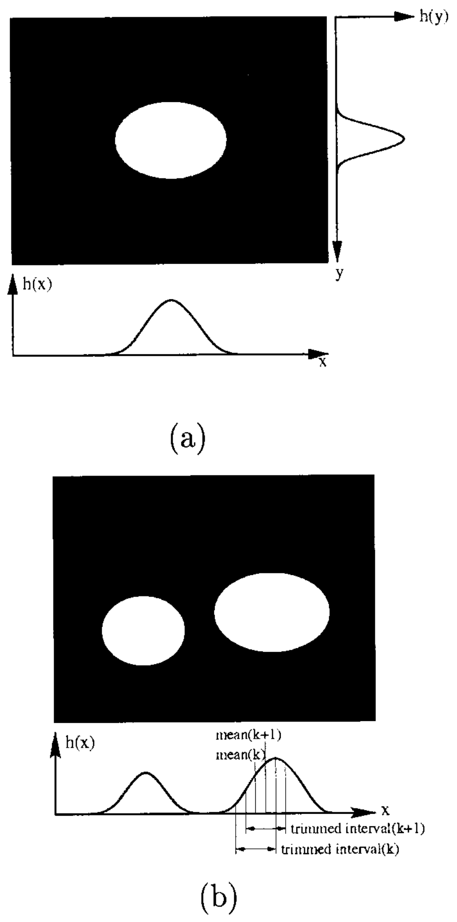


Fig. 2. (a) Two 1-D histograms constructed by projecting the frame difference map along the x and y direction, respectively. (b) Robust mean estimation for locating the center position of a *dominant* moving object.

$$w = \alpha \left[\frac{\sum\limits_i (x_i - \mu_x)^2 h_x(i)}{\sum\limits_i h_x(i)} \right]^{(1/2)}$$

$$h = \beta \left[\frac{\sum\limits_i (y_i - \mu_y)^2 h_y(i)}{\sum\limits_i h_y(i)} \right]^{(1/2)}$$

where α and β are constant scaling factors.

However, the object center position and size derived from the sample means and standard deviations may be biased in the cases where other moving objects appear in the scene. It is, therefore, necessary to develop a more robust procedure to address this problem. We propose the use of robust statistical estimation routines to achieve robust measurements for object center position and size [31]. More specifically, the center position of a dominant moving object in an input sequence is estimated based on the robust (trimmed) means of the two 1D projection histograms in the x and y directions. Fig. 2(b) illustrates the process of the estimation of the motion center.

Step 1) Compute sample mean μ and standard deviation σ based on all the samples of the distribution.

Step 2) Let $\mu_t(0) = \mu$ and $\delta = \max(a\sigma, b * sampleSpaceWidth)$ where a and b are scaling factors, e.g., $a = 1.0$ and $b = 0.2$, and $sampleSpaceSize$ is the width and height of the image for δ_{horiz} and δ_{vert}, respectively.

Step 3) Compute trimmed mean $\mu_t(k+1)$ based on the samples within the interval $[\mu_t(k) - \delta, \mu_t(k) + \delta]$.

Step 4) Repeat Step 3 until $|\mu_t(k+1) - \mu_t(k)| < \epsilon$ where ϵ is the tolerance, e.g., $\epsilon = 1.0$.

Step 5) Let center-position = the converged mean.

In addition to the robust estimation of object center position, we propose the following routine for robust estimation of object size. The method first re-projects the frame difference result in a neighborhood of the located center. It then derives the object size based on the robust (trimmed) standard deviation. Given the robust mean μ^* and δ obtained from the above center locating routine, the routine for estimation the size in either x or y direction is as follows.

Step 1) Construct a clipped projection histogram H^{clip} by projecting the color filtering map within the range $[\mu^*_{\text{opp}} - \Delta, \mu^*_{\text{opp}} + \Delta]$ in the opposite direction, where μ^*_{opp} is the robust mean in the opposite direction and Δ determines the number of samples used in the calculation.

Step 2) Based on H^{clip}, compute the trimmed standard deviation δ_t based on the samples within the interval $[\mu^* - \delta, \mu^* + \delta]$.

Step 3) IF $H^{\text{clip}}(\mu^* + d\delta_t) \geq gH^{\text{clip}}(\mu^*)$ OR $H^{\text{clip}}(\mu^* - d\delta_t) \geq gH^{\text{clip}}(\mu^*)$, where e.g., $d = 1.0$ and $g = 0.4$, THEN increase δ_t until the condition is no longer true.

Step 4) Let $size = c\delta_t$ where c is a scaling factor, e.g., $c = 2.0$.

Multiple motion blobs can be located by repeating the above proposed method in an iterative manner. More specifically, the area of the already detected motion blob can be zeroed out in the frame difference map and the above method can be applied to the modified frame difference map to locate the subsequent motion blobs.

B. Texture and Color Analysis: Low-Level Descriptors

To obtain rich, and hence, robust and expressive descriptions of the objects in the video frames, we describe each pixel in terms of 76 color and texture measures: 56 of them are based on the Gray Level Co-occurrence Matrix (GLCM), 4 on fractal dimension estimation methods, 12 on Gabor filters, and 4 on color. The feature space representations of each pixel are classified into the categories sky/clouds, grass, trees, animal, and rock using a back-propagation neural network. The use of these features in conjunction with the back-propagation classifier have previously been shown to enable the detection of deciduous trees in unconstrained images [13]. We decided not to use shape for our description of objects in video frames mostly because the recognition of the following important objects is far beyond the current state-of-the-art in object recognition.

1) Clouds and dust are amorphous objects for which shape models are difficult to construct.
2) Rocks, trees, grass, sky, etc., although not amorphous, can occur in an almost infinite number of different shapes, and furthermore, they rarely appear in isolation, trees grow near other trees, rocks lie with other rocks, etc.
3) Articulated and somewhat deformable objects such as running animals are difficult to describe with shape invariants.
4) In our hunt application, tall grass often hides the legs and lower body of animals, trees occlude themselves, other trees, and animals who seek their cover, etc. But occlusion is also causing severe problems to shape-based object recognition schemes in most other domains.

No single or pair of types of measure (e.g., color and/or Gabor measures) has the power of the combined set of measures [14]. The neural network described in Section II-C is well suited to combine this set of measures and robustly classify image regions into various animal and non-animal classes. Note that we are only computing features from still frames and that motion is included explicitly at a higher level.

1) Gabor Filter Measures: The image (in the spatial domain) is described by its 2-D intensity function. The Fourier Transform of an image represents the same image in terms of the coefficients of sine and cosine basis functions at a range of frequencies and orientations. Similarly, the image can be expressed in terms of coefficients of other basis functions. Gabor [12] used a combined representation of space and frequency to express signals in terms of “Gabor” functions

$$f(x, y) = \sum_i a_i g_i(x, y) \tag{1}$$

where a_i weights the ith complex Gabor basis function

$$g_i(x, y) = e^{j\omega_i(x\cos(\theta_i) + y\sin(\theta_i))} e^{-(\alpha_i^2 x^2 + \beta_i^2 y^2)}. \tag{2}$$

Gabor filters have gained popularity in multi-resolution image analysis [11], [12], despite the fact that they do not form an orthogonal basis set. Gabor filter based wavelets have recently been shown [23] to be fast and useful for the retrieval of image data. We obtain 12 features, per pixel, by convolving each frame with Gabor filters tuned to four different orientations at three different scales.

2) GLCM Measures: Let $p(i, j, d, \theta) = p(i, j, d, \theta) = (P(i, j, d, \theta)/R(d, \theta))$, where $P(\cdot)$ is the GLCM of pixels separated by distance d in orientation θ and where $R(\cdot)$ is a normalization constant that causes the entries of $P(\cdot)$ to sum to one. In texture classification, the following measures have been defined [3], [16]:

The *Angular Second Moment (E)* (also called the Energy) assigns larger numbers to textures whose co-occurrence matrix is sparse

$$E(d, \theta) = \sum_{j=1}^{N_g} \sum_{i=1}^{N_g} [p(i, j, d, \theta)]^2.$$

The *Difference Angular Second Moment (DASM)* assigns larger numbers to textures containing only a few gray-level patches. This and other features use $p_{x-y}(n, d, \theta) = \sum_{j=1}^{N_g} \sum_{i=1 \, |i-j|=n}^{N_g} p(i, j, d, \theta)$

$$\text{DASM}(d, \theta) = \sum_{n=0}^{N_g} p_{x-y}(n, d, \theta)^2.$$

The *Contrast (Con)* is the moment of inertia around the co-occurrence matrix's main diagonal. It is a measure of the spread of the matrix values and indicates whether pixels vary smoothly in their local neighborhood

$$\text{Con}(d, \theta) = \sum_{n=0}^{N_g-1} n^2 \left[\sum_{j=1}^{N_g} \sum_{i=1 \, |i-j|=n}^{N_g} p(i, j, d, \theta) \right].$$

The other GLCM-based measures we use for our texture analysis are the *Inverse Difference Moment, Mean, Entropy, Sum Entropy, Difference Entropy, Difference Variance, Correlation, Shade, and Prominence*. These features are described in [3], [16], [30].

Note that the directionality of a texture can be measured by comparing the values obtained for a number of the above measures as θ is changed. The above measures were computed at $\theta = \{0°, 45°, 90°, \text{and } 135°\}$ using $d = 1$. For further discussion of these GLCM measures (see [3], [16], [30]).

3) Fractal Dimension (FD) Measures: The underlying assumption for the use of the fractal dimension for texture classification and segmentation is that images or parts of images are self-similar at some scale. Various methods that estimate the FD of an image have been suggested.

- Fourier-transform based methods [25];
- box-counting methods [2], [21];
- 2-D generalizations of Mandelbrot's methods [24].

The principle of self-similarity may be stated as: If a bounded set A (object) is composed of N_r non-overlapping copies of a set similar to A, but scaled down by a reduction factor r, then A is self-similar. From this definition, the FD D is given by

$$D = \frac{\log N_r}{\log r}.$$

The FD can be approximated by estimating N_r for various values of r and then determining the slope of the least-squares linear fit of $(\log N_r / \log(1/r))$. The differential box-counting method outlined in Chaudhuri, *et al.* [2] are used to achieve this task.

Three features are calculated based on:

1) the actual image patch $I(i, j)$;
2) the high-gray-level transform of $I(i, j)$

$$I_1(i, j) = \begin{cases} I(i, j) - L_1, & I(i, j) > L_1 \\ 0, & \text{otherwise} \end{cases}$$

3) the low-gray-level transform of $I(i, j)$

$$I_2(i, j) = \begin{cases} 255 - L_2, & I(i, j) > 255 - L_2 \\ I(i, j), & \text{otherwise} \end{cases}$$

where $L_1 = g_{\min} + (g_{avg}/2)$, $L_2 = g_{\max} - (g_{avg}/2)$, and $g_{\min}$, $g_{\max}$, and g_{avg} are the minimum, maximum, and average gray values in the image patch, respectively.

The fourth feature is based on multi-fractals which are used for self-similar distributions exhibiting non-isotropic and inhomogeneous scaling properties. Let k and l be the minimum and maximum gray level in an image patch centered at position (i, j), let $n_r(i, j) = l - k + 1$, and let $\mathcal{N}_r = (n_r / N_r)$, then the multi-fractal, D_2 is defined by

$$D_2 = \lim_{r \to 0} \frac{\log \sum_{i,j} \mathcal{N}_r^2}{\log r}.$$

A number of different values for r are used and the linear regression of $(\log \sum_{i,j} \mathcal{N}_r^2 / \log r)$ yields an estimate of D_2.

4) The Color Features: The final set of features are the three normalized color measures r, g, b and the intensity I

$$r = \frac{R}{R+G+B}$$
$$g = \frac{G}{R+G+B}$$
$$b = \frac{B}{R+G+B}$$
$$I = \frac{R+G+B}{R_{\max} + G_{\max} + B_{\max}}.$$

We generally observed that although our feature set is theoretically redundant it is beneficial for the classifier to use all the measures rather than a carefully selected subset.

C. Region Classification and Motion-Blob Verification

We use a back-propagation neural network to arbitrate between the different features describing the image. Our back-propagation neural network [10] has a single hidden layer with 20 hidden units and uses the sigmoidal activation function $\Phi(act) = (1/1 + e^{-act}) - 0.5$, where act is the activation of the unit before the activation function is applied. A single hidden layer in a back-propagation neural network has been shown to be sufficient to uniformly approximate any function (mapping) to arbitrary precision [5]. Although this existential proof does not state that the best network for some task has a single hidden layer, we found one hidden layer adequate. The architecture of the network is shown in Fig. 3. The back-propagation algorithm propagates the (input) function values layer by layer, left to right (input to output), and back-propagates the errors layer by layer, right to left (output to input). As the errors are propagated back to the input units, part of each unit's error is being corrected.

A number of factors prevent zero error results. A few of these complicating factors are that *often there is no correct classification*. For instance, should bushes be labeled as tree or non-tree areas? What if a bush is actually a small tree? In general, it is difficult to label class border pixels correctly, and misclassifications need not all be equally important. Misclassifying a distant herd of animals as trees or rocks is not as severe a mistake as, for example, classifying a nearby lion as sky.

We trained the network using a total of $M = 15$ labels. Nine animal labels (lion, cheetah, leopard, antelope, impala, zebra,

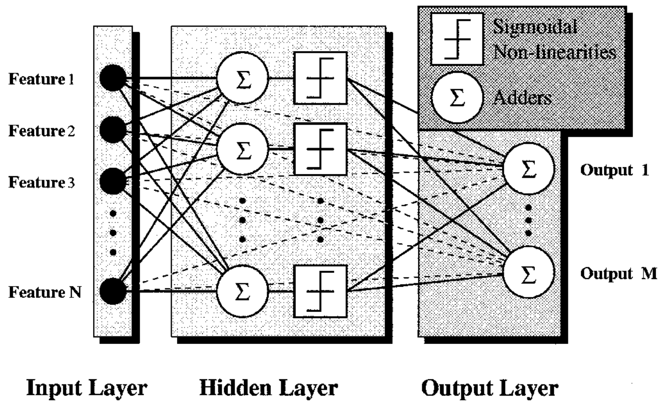


Fig. 3. Neural network architecture.

gnu, elephant, and an all-other-animal class) and five non-animal labels (rock, sky/clouds, grass, trees, and an all-other-non-animal class) as well as a "don't care" label.

After training, we found that the proposed network performed well at classifying five merged classes: all animals, grass, trees, rocks, and sky/clouds. However, it is difficult for the network to classify the animals in our wildlife documentary videos, namely, lions, cheetahs, leopards, antelopes/impalas, gnus, hyenas, and even zebras, rhinos and elephants each into different groups. This is probably due to the fact that many of these animals differ mostly in their shape and size, which we do not model. Hence, while the network was still trained on the different animal labels, we artificially grouped those labels into a single "animal" label when using the network for animal region verification. We also found that the network did not perform well at solving the opposite problem of classifying, grass, trees, rocks, and sky together as a single "non-animal" group. The differences between the appearance of instances of these groups are severe. Asking the network to assign one label to them and a different label to animals proves to be more difficult than the classification into the individual non-animal groups.

The output of the network is then used to verify the motion-blob candidates from Section II-A. In our current implementation, a simple procedure is employed which implements the following test. A region that has high residual motion after motion compensation and that contains a significant amount of animal labels, as detected by the neural network, is considered as a possible moving animal region.

D. Shot Summarization and Intermediate-Level Descriptors

We use a simple color histogram based technique to decompose video sequences into shots. Since some shots last for 50 frames or less and others last for 1000 s of frames we also *force* a shot summary every 200 frames to impose a degree of regularity onto the shot summaries and to avoid missing important events in extended shots. A third kind of shot boundary is inserted whenever the direction of the global-motion changes. Shot boundaries of this last kind ensure that the motion within shots is homogeneous. Each shot is then summarized in terms of intermediate-level descriptors. The purpose of generating intermediate-level shot summaries is two-fold. First, the shot summaries provide a way to encapsulate the low-level feature and motion analysis details so that the high-level event inference module may be developed independent of those details, rendering it robust against implementational changes. Second, the shot summaries abstract the low-level analysis results so that they can be read and interpreted more easily by humans. This simplifies the algorithm development process and may also facilitate video indexing, retrieval, and browsing in video database applications.

In general, the intermediate-level descriptors may consist of: 1) *object*, descriptors, e.g., "animal," "tree," "sky/cloud," "grass," "rock," etc. indicate the existence of certain objects in the video frames; 2) *spatial* descriptors, e.g. "inside," "next to," "on top of," etc., that represent the location and size of objects and the spatial relations between them; and 3) *temporal* descriptors, e.g. "beginning of," "while," "after," etc. [6], [8], that represent motion information about objects and the temporal relations between them.

For the hunt-detection application, we employ a particular set of intermediate-level descriptors which describe: 1) whether the shot summary is due to a forced or detected shot boundary; 2) the frame number of the beginning of the shot; 3) the frame number of the end of the shot; 4) the global motion; 5) the object motion; 6) the initial object location; 7) the final object location; 8) the initial object size; 9) the final object size; 10) the smoothness of the motion; 11) the precision throughout shot; and 12) the recall throughout shot. More precisely, the motion descriptors provide information about the x- and y-translation and zoom components of motion. The location and size descriptors indicate the location and size of the detected dominant motion blob at the beginning and the end of the shot. The precision is the average ratio of the number of animal labels within the detected dominant motion blob versus the size of the blob, while the recall is an average of the ratio of the animal labels within the detected dominant motion blob versus the number of animal labels in the entire frame. In addition, we also employ descriptors indicating: 13) that tracking is engaged; 14) that object motion is fast; 15) that an animal is present; 16) the beginning of a hunt; 17) number of consecutive hunt shot candidates found; 16) the end of a hunt; and 19) whether a valid hunt is found. See Section III-F for an example and further explanation.

E. Event Inference

The event-inference module determines whether segments of video contain events of interest. If a contiguous sequence of shot summaries matches the event model, then the presence of that event is asserted. We decided to design the event inference module manually for two reasons: 1) the design of many events is straightforward given the intermediate representation of the depicted objects and their motions and 2) a rule-based event model allows a high level of transparency.

Hunt events are detected by an event inference module which utilizes domain-specific knowledge and operates at the shot level based on the generated shot summaries. From observation and experimentation with a number of wildlife documentaries, a set of rules have been deduced for detecting hunts. The rules reflect the fact that a hunt usually consists of a number of shots exhibiting smooth but fast animal motion, followed by subsequent shots with slower or no animal motion. In other

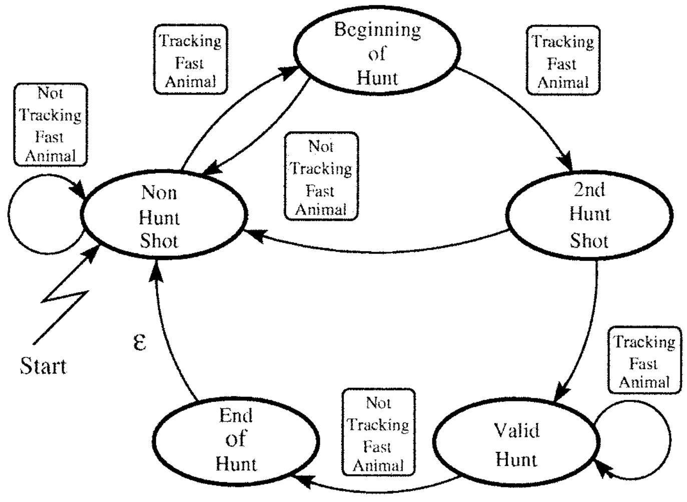


Fig. 4. State diagram of our hunt-detection method. Initially, the control is in the Non-Hunt state on the left. When a fast moving animal is detected, the control moves to the beginning of hunt state at the top of the diagram. When three consecutive shots are found to track fast-moving animals, then the Valid Hunt flag is set. The first shot afterwards that does not track a fast moving animal takes the control to the End of Hunt state, before again returning to the Non-Hunt state.

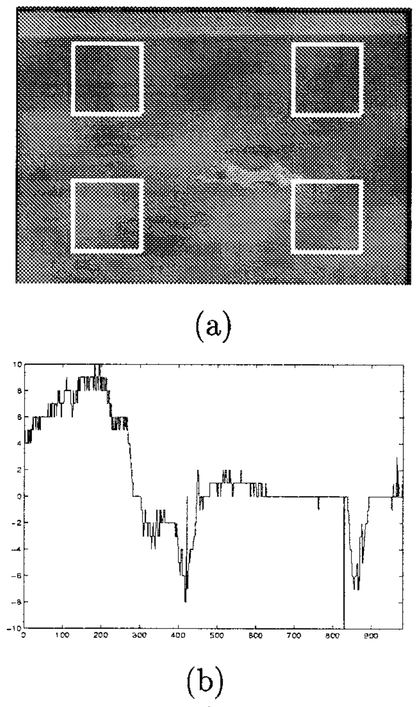


Fig. 5. (a) Locations used to estimate the global motion. (b) Motion estimates during a hunt.

words, the event-inference module looks for a prescribed number of shots in which: 1) there is at least one animal of interest; 2) the animal is moving in a consistently fast manner for an extended period; and 3) the animal stops or slows down drastically after the fast motion. Fig. 4 shows and describes a state diagram of our hunt-detection inference model.

Automatic detection of the properties and sequences of actions in the state digram is non-trivial and the low-level feature and motion analysis described earlier in this paper are necessary to realize the inference. Since many events can be defined by the occurrence of objects involved and the specification of their spatio-temporal relationship, the proposed mechanism, of combining low-level visual analysis and high-level domain-specific rules, may be applicable to detect other events in different domains. In Section III-G, we provide an example and further explanation for using this inference model for hunt detection.

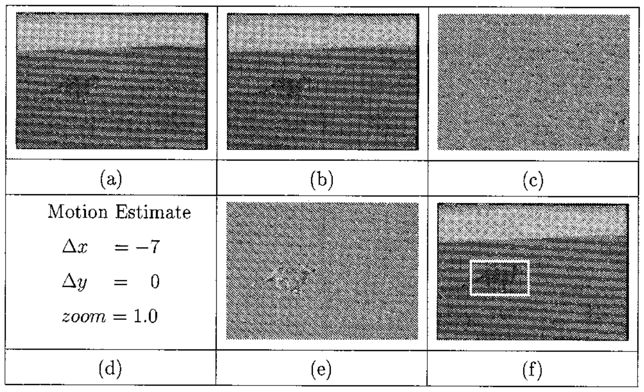


Fig. 6. (a) and (b): Two consecutive frames from a hunt. (c) Difference image. (d) Estimated motion between the two frames. (e) Motion-compensated difference image (e), and (f) the box around the area of largest residual error in the motion-compensated difference image.

III. Experimental Results

The proposed algorithm has been implemented and tested on wildlife video footage from a number of commercially available VHS tapes from different content providers. In the following sections, we show example results of the global-motion estimation, motion-blob detection, extracted texture and color features, region classification, and shot summarization. Then we present the final hunt event-detection results.

A. Test Data

About 45 minutes of actual wildlife video footage have been digitized and stored as test data for our hunt-detection experiments. The frame rate of the video is 30 frames/s and the digitized frame resolution is 360×243 pixels. A total of 10 min of footage $\triangleq 18\,000$ frames $\triangleq 100$ shots have been processed so far.

B. Global-Motion Estimation

Fig. 5(a) shows the size (64×64) and locations of the four regions at which the global motion is estimated. For each pair of frames, motion estimates are computed using a five-level pyramid scheme at the shown patch locations. In addition, the previous motion estimate is taken as the current motion estimate and a tight local search around the four *predicted* patch locations yields another four patch matches. The best match of any of these eight patch comparisons becomes the motion estimate for the current frame pair. Fig. 5(b) shows the motion estimates during a hunt.

C. Motion-Blob Detection

Fig. 6 shows an example of the motion-blob-detection results. It is apparent that reliable estimation and compensation

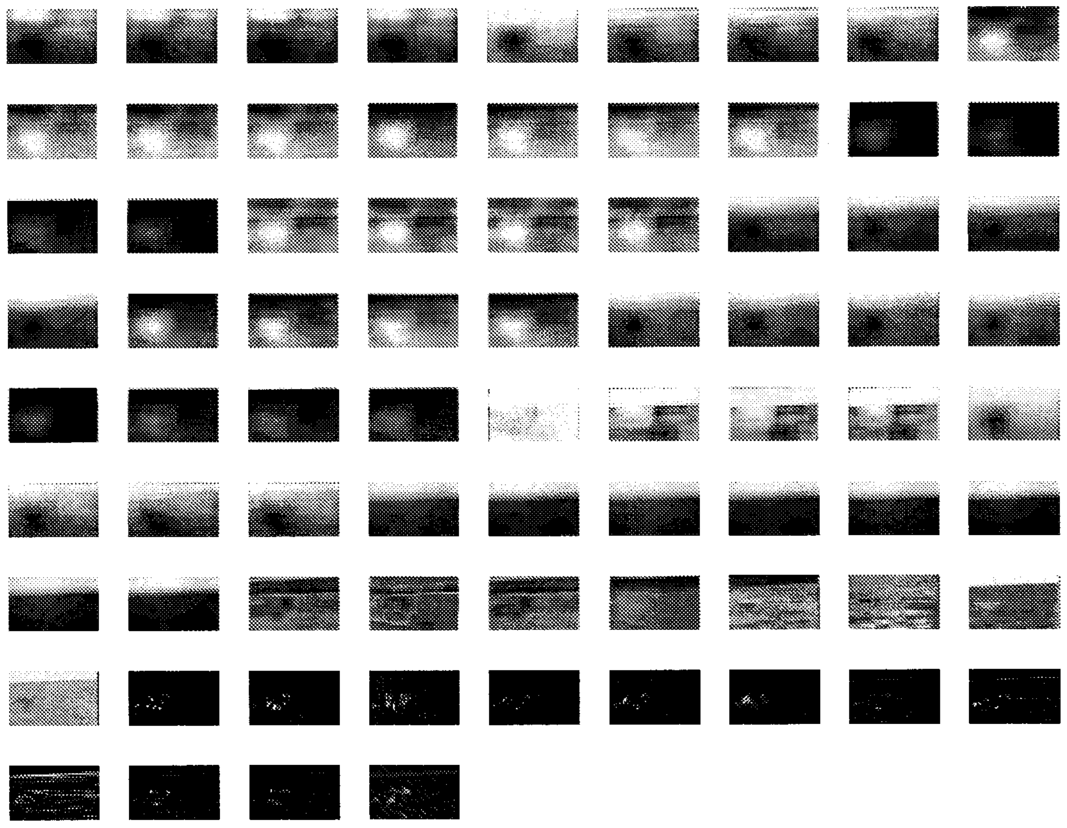


Fig. 7. Feature-space representation of the first frame in Fig. 6.

of global motion makes the task of motion-blob detection relatively easier. When the accuracy of the global-motion estimation results are poor, the performance of the motion-blob detection relies largely on the robustness of the motion-blob detection and tracking algorithm described in Section II-A.

D. Feature-Space Representation of the Video Frames

Fig. 7 shows the feature-space representation of the first frame in Fig. 6. The features shown in order are the results of the 56 GLCM-based measures, the four fractal-dimension-based measures, the four color-based measures, and the 12 Gabor filter-bank measures.

E. Region Classification

Global-motion estimates, such as the ones in Fig. 5, are used to detect moving objects, as shown in Fig. 6. The locations of these moving object blobs are then verified using a neural network classifier that combines color and texture information. The classifier is trained on a number of training frames from wildlife video. Rows 1, 3, and 5 of Fig. 8 show frames from hunts together with their classification results (rows 2, 4, and 6).

F. Shot Summarization

The intermediate level process consists of two stages. In the first stage, the global-motion estimates are analyzed and directional changes are detected in the x and y directions. When the *signs* of the 50-frame global-motion averages before and after the current frame differ and their *magnitudes* are greater than 1 pixel per frame, we insert an artificial shot boundary. In the second stage, each shot is then summarized as in the example shown below:

```
--------General Information--------
Forced/real shot summary :        0
First frame of shot :             64
Last frame of shot :              263
Global motion estimate (x, y) :   (−4.48, 0.01)
Within frame animal
  motion estimate (x, y) :        (−0.17, 0.23)
Initial position (x, y) :         (175, 157)
Final position (x, y) :           (147, 176)
Initial size (w, h) :             (92, 67)
Final size (w, h) :               (100, 67)
Motion smoothness throughout
  shot (x, y) :                   (0.83, 0.75)
Precision throughout shot :       (0.84)
Recall throughout shot :          (0.16)

--------Hunt Information--------
Tracking :                          1
Fast :                              1
Animal :                            1
Beginning of hunt :                 1
Number of hunt shot candidates :    1
End of hunt :                       0
Valid hunt :                        0
```

The summary consists of two parts, the first part, under `General Information` shows general statistics extracted for this shot, while the second, under `Hunt Information` consists

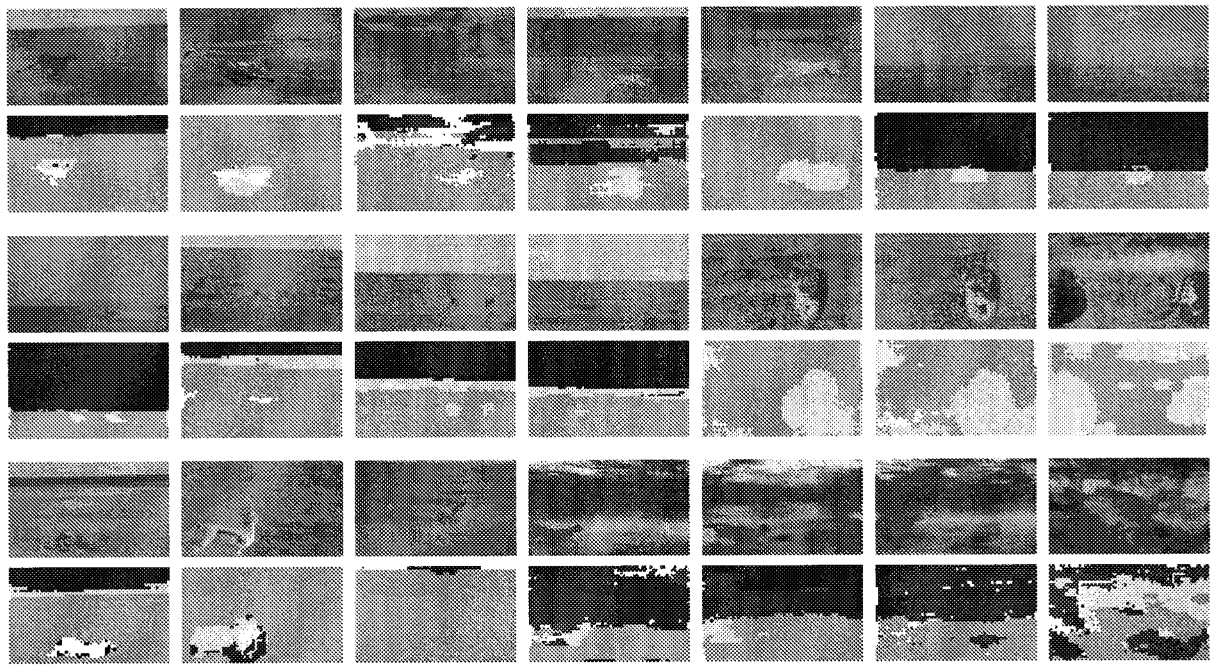


Fig. 8. Color- and texture-based segmentation results.

of inferences based on those statistics for the hunt-detection application.

The first row of the General Information part of the summary shows whether the shot boundary corresponding to this shot summary was real, i.e., whether it was detected by the shot boundary detector, or if it was forced because the maximum number of frames per shot was reached or the global motion has changed. The next two rows show the first and last frame numbers of this shot. The following measurements are shot statistics, i.e., the average global motion over the entire shot on row four, and the average object motion within the shot on row five. The next four rows measure the initial position and size, as well as the final position and size of the detected dominant motion blob. The third-to-last row shows the smoothness of global motion where values near 1 indicate smooth motion and values near zero indicate unstable motion estimation. The following (3) shows how the smoothness measure is computed:

$$S = \frac{1}{N} \sum_{i=1}^{N} v_i \tag{3}$$

where N is the number of frames in the shot and v_i is defined as follows:

$$v_i = \begin{cases} 1, & q_i * r_i \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

where q_i and r_i are the minimum and maximum values of the horizontal components of the global-motion estimates for the five most recent frames, including the ith frame and the four previous frames in the shot. The smoothness measure is large when consecutive motion estimates have the same sign. Likewise, the smoothness measure is small when the motion estimates of consecutive frames frequently differs in sign. The smoothness measure, therefore, provides a *quantitative* measure of the smoothness of the estimated motion.

The detection of a reversal of the global-motion direction, described above, is based on a long term average of the motion estimates around the current frame, indicates a *qualitative* change in the global motion. Finally the last two rows show the average precision and recall for the entire shot. As defined in Section II-D, the precision is the average ratio of the number of animal labels within the detected dominant motion blob versus the size of the blob, while the recall is an average of the ratio of the animal labels within the detected dominant motion blob versus the number of animal labels in the entire frame.

The hunt information part of the shot summary shows a number of predicates that were inferred from the statistics in part one. The shot summary shown above summarizes the first hunt shot following a `forced` shot boundary. The system is indicating that it is `Tracking` a `Fast` moving `Animal` and hence, that this could be the `Beginning of a hunt`. The `Tracking` predicate is true when the motion-smoothness measure is greater than a prescribed value and the motion-blob-detection algorithm detects a dominant motion blob. The `Fast` predicate is true if the translational components of the estimated global motion are sufficiently large in magnitude, and the `Animal` predicate is true if the precision, i.e., the number of animal labels within the tracked region, is sufficiently large. (The recall measure has not been used in our current implementation.) The remaining predicates are determined and used by the inference module as described below.

G. Event-Inference and Final-Detection Results

The event-inference module infers the occurrence of a hunt based on the intermediate descriptors as described in Section III-F. It employs four predicates, `Beginning of hunt`, `Number of hunt shot candidates`, `End of hunt`, and `Valid hunt`. If the intermediate descriptors `Tracking`, `Fast`

TABLE I
COMPARISON OF ACTUAL AND DETECTED HUNTS IN TERMS OF THE FIRST AND LAST HUNT FRAME, AND THE ASSOCIATED PRECISION AND RECALL

Sequence Name	Actual Hunt Frames	Detected Hunt Frames	Precision	Recall
hunt1	305 - 1375	305 - 1375	100 %	100 %
hunt2	2472 - 2696	2472 - 2695	100 %	99.6%
hunt3	3178 - 3893	3178 - 3856	100 %	94.8%
hunt4	6363 - 7106	6363 - 7082	100 %	96.8%
hunt5	9694 - 10303	9694 - 10302	100 %	99.8%
hunt6	12763 - 14178	12463 - 13389	67.7%	44.2%
hunt7	16581 - 17293	16816 - 17298	99.0%	67.0%
Average			95.3%	86.0%

and `Animal` are all true for a given shot, `Beginning of hunt` is set to be true.

The value of `Number of hunt shot candidates` is incremented for every consecutive shot during which the three descriptors remain true. When the `Number of hunt shot candidates` is equal to or greater than three, `Valid hunt` is set to be true. Finally, the inference module sets `End of hunt` to be true if one of the intermediate descriptors `Tracking`, `Fast` and `Animal` becomes false, which implies either the animal is no longer visible or trackable, or the global motion is slow enough, indicating a sudden stop after fast chasing.

In our results, hunt events are specified in terms of their starting and ending frame numbers. There are seven hunt events in the 10 min (18 000 frames) of wildlife video footage which we have processed, Table I shows the actual and the detected frames of the seven hunts. The table also shows the retrieval performance of our method in terms of the two commonly used evaluation criteria: 1) precision and 2) recall. Our method detected the first five hunt events very accurately. The frame numbers of the detected and actual hunt frames match so closely because they coincide with shot boundaries which both humans as well as our method take as the boundaries of events. Hunt 6 was detected rather poorly because: 1) at the beginning of the hunt the well camouflaged animals chasing each other in tall grass were not detected and 2) at the end of the hunt, both animals disappear behind a hill. The camera keeps panning and the two eventually re-emerge on the other side of the hill before the predator catches the prey. Since both animals are occluded for a prolonged period of time, the event-inference module resets itself, signaling a premature end of this hunt. For Hunt 7, the recall measure indicates that our method missed quite a few frames at the beginning of that hunt. The human observers who we had asked to determined the "actual" beginning and end of the hunt included part of the stalking phase into the hunt. Indeed, it is difficult to draw a clear line between the stalking phase and the hunt phase of that hunt. It is likely that the detection of stalking animals requires a detailed animal gesture analysis which goes well beyond the scope of our coarse motion and object analysis.

IV. SUMMARY AND DISCUSSION

In this paper, we have presented a new computational method and a number of enabling algorithmic components for automatic event detection in video and applied it to detect hunts in wildlife documentaries. Our experimental results have verified the effectiveness of the proposed algorithm. The developed method decomposes the task of extracting semantic events into three stages where visual information is analyzed and abstracted. The first stage extracts low-level features and is entirely domain independent. The second stage analyzes the extracted low-level features and generates intermediate-level descriptors, some of which may be domain specific. In this stage, shots are summarized in terms of both domain-independent and domain-specific descriptors. To generate the shot summaries, regions of interest are detected, verified, and tracked. The third and final stage is domain specific. Rules are deduced from specific domains and an inference model is built based on the established rules. In other words, each lower stage encapsulates low-level visual processing from the higher stages. Therefore, the processes in the higher stages can be stable and relatively independent of any potential detail changes in the lower level modules. In order to detect different events: 1) the object classifier may need to be adjusted in the second stage of our method and 2) a new set of rules describing and defining the event are needed in the third stage. The proposed algorithm also provides several reusable algorithmic components. In fact, the extracted low-level texture and color features are entirely domain independent since many objects have texture and color signatures. The neural network used for image region classification can be easily re-configured or extended to handle other types of objects [13]. The robust statistical-estimation-based object-tracking method has already been used in different applications and its robustness and simplicity are verified in experiments repeatedly [26].

We would like to point out that the proposed algorithm detects hunt events by detecting spatial-temporal phenomena which are physically associated with a hunt event in nature. More precisely, the physical phenomenon which we attempt to capture is the combination of the presence of animals in space and their movement patterns in time. This is in contrast to many existing event-detection methods which detect events by detecting artificial post-production editing patterns or other artifacts. The drawbacks of detecting specific editing patterns or other artifacts are that those patterns are often content provider dependent and it is difficult, if not impossible, to modify the detection methods and apply them to the detection of other events. It is also important to point out that our algorithm solves a practical problem and the solution is needed in the real world. In the wildlife video tapes which we obtained, the speech from the audio track and the text from the close-caption are loosely correlated with the visual footage. It is therefore unlikely that the hunt segments may be accurately located by analyzing the audio track and close-caption. In other words, given the existing wildlife tapes, a visual-information-based detection algorithm is needed to locate the hunt segments otherwise manual annotation is required. We believe the limitation to a specific domain, such as wildlife documentaries, does not limit our approach significantly, since such high-level information is readily available from the content provider.

The use of audio information represents one important difference to related work [17] that proposes a two-level method using "Multijects" to combine low-level feature information directly. Two other differences are: 1) the simplicity of the visual features they use to represent video frames and 2) their use of adaptive components (Hidden Markov Models) to learn the entire event from examples. At present, the authors only use color histograms and color histogram differences of entire frames to represent the video content. In contrast, our approach captures information on what is moving, where and how based on a richer analysis using color, texture, and motion. Although adaptive components are desirable for a general event-detection scheme, they tend to reduce the transparency of the event inference process. Seeing that many events are easily described in terms of intermediate object and motion descriptors, we decided to describe and design the event inference processes manually.

An immediate focus of future work is to develop a richer set of intermediate-level descriptors for generating shot summaries. The purpose of developing the descriptors is to provide a wider coverage over different domains and events so that fewer domain-specific descriptors need to be added in new applications. Other future work is to improve the procedure which detects and tracks regions of interest. It would also be interesting to investigate the usefulness of learning techniques for the event inference engine. One goal might be the automatic tuning of the performance of the event inference module.

Finally, we would like to point out that since the submission of this paper, we have successfully applied the proposed method to two other events, namely landings and rocket launches in unconstrained videos [15]. As described in this article, the only changes necessary to handle these new events were the classifier and the event inference module. The absence of shape-based object information in our method allows us to detect landing events independent of the exact identity of the landing object (aircraft, bird, space shuttle, etc.) or the exact type of rocket or launch pad. It is not surprising that approximate object-motion information can aid object recognition and the interpretation of events in which these objects are involved.

References

[1] F. Arman, R. Depommier, A. Hsu, and M.-Y. Chiu, "Content-based browsing of video sequences," in *Proc. ACM Multimedia*, 1994, pp. 97–103.

[2] B. B. Chaudhuri, N. Sarkar, and P. Kundu, "Improved fractal geometry based texture segmentation technique," *Proc. Inst. Elect. Eng.—E*, vol. 140, pp. 233–241, 1993.

[3] R. W. Conners and C. A. Harlow, "A theoretical comparison of texture algorithms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 2, pp. 204–222, PAMI-1980.

[4] J. D. Courtney, "Automatic video indexing via object motion analysis," *Proc. Pattern Recognit.*, vol. 30, no. 4, pp. 607–626, 1997.

[5] G. Cybenko, "Approximation by superposition of sigmoidal function," *Mathemat. Contr., Signals, Syst.*, pp. 303–314, 1989.

[6] A. Del Bimbo, E. Vicario, and D. Zingoni, "A spatial logic for symbolic description of image contents," *Vis. Lang. Comput.*, vol. 5, pp. 267–286, 1994.

[7] Y. Deng and B. S. Manjunath, "Content-base search of video using color, texture, and motion," *Proc. IEEE Int. Conf. Image Processing*, vol. 2, pp. 534–537, 1998.

[8] N. Dimitrova and F. Golshani, "Motion recovery for video content classification," *ACM Trans. Inform. Syst.*, vol. 14, no. 4, pp. 408–439, 1995.

[9] P. England, R. B. Allen, M. Sullivan, and A. Heybey, "I/browse: The bellcore video library toolkit," in *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1996, pp. 254–264.

[10] S. Fahlman, "Faster-learning variations on back-propagation: An empirical study," in *Proc. Connectionist Models Summer School*, 1988.

[11] I. Fogel and D. Sagi, "Gabor filters as texture discriminator," *Biol. Cybern.*, vol. 61, pp. 103–113, 1989.

[12] D. Gabor, "Theory of communication," *J. Inst. Elect. Eng.*, vol. 93, pp. 429–457, 1946.

[13] N. Haering, Z. Myles, and N. da Vitoria Lobo, "Locating deciduous trees," in *Proc. IEEE Workshop Content-Based Access of Image and Video Libraries*, 1997, pp. 18–25.

[14] N. Haering and N. da Vitoria Lobo, "Features and classification methods to locate deciduous trees in images," Comput. Vis. Image Understanding, 1999, to be published.

[15] N. Haering, "A Framework for the Design of Event Detectors," Ph.D. dissertation, Univ. of Central Florida, Orlando, FL, 1999.

[16] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 610–621, 1973.

[17] M. R. Naphade, T. Kristjansson, and T. S. Huang, "Probabilistic multimedia objects (MULTIJECTS): A novel approach to video indexing and retrieval in multimedia systems," *Proc. IEEE Int. Conf. Image Processing*, vol. 3, pp. 536–540, 1998.

[18] S. S. Intille, "Tracking using a local closed-world assumption: Tracking in the football domain," Master's thesis, M.I.T. Media Lab, Cambridge, MA, 1994.

[19] G. Iyengar and A. Lippman, "Models for automatic classification of video sequences," in *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1997, pp. 216–227.

[20] T. Kawashima, K. Tateyama, T. Iijima, and Y. Aoki, "Indexing of baseball telecast for content-based video retrieval," in *Proc. IEEE Int. Conf. Image Processing*, 1998, pp. 871–875.

[21] J. M. Keller and S. Chen, "Texture description and segmentation through fractal geometry," *Comput. Vis., Graph., Image Processing*, vol. 45, pp. 150–166, 1989.

[22] R. L. Lagendijk, A. Hanjalic, M. Ceccarelli, M. Soletic, and E. Persoon, "Visual search in a SMASH system," in *Proc. IEEE Int. Conf. Image Processing*, 1997, pp. 671–674.

[23] B. S. Manjunath and W. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 837–859, Aug. 1996.

[24] S. Peleg, J. Naor, R. Hartley, and D. Avnir, "Multiple resolution texture analysis and classification," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 518–523, 1984.

[25] A. P. Pentland, "Fractal-based description of natural scenes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 661–674, 1984.

[26] R. J. Qian, M. I. Sezan, and K. E. Matthews, "A robust real-time face tracking algorithm," in *Proc. IEEE Int. Conf. Image Processing*, 1998, pp. 131–135.

[27] D. Saur, Y.-P. Tan, S. R. Kularni, and P. J. Ramadge, "Automated analysis and annotation of basketball video," in *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1997, pp. 176–187.

[28] M. Smith and T. Kanade, "Video skimming for quick browsing based on audio and image characterization," Computer Science Department, Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU CS-95-186, 1995.

[29] N. Vasconcelos and A. Lippman, "A Bayesian framework for semantic content characterization," in *Proc. IEEE Computer Vision and Pattern Recognition*, 1998, pp. 566–571.

[30] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparative study of texture measures for terrain classification," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-6, pp. 269–285, 1976.

[31] R. R. Wilcox, *Introduction to Robust Estimation and Hypothesis Testing*, ser. statistical modeling and decision science. New York: Academic, 1997.

[32] M. Yeung and B.-L. Yeo, "Video visualization for compact presentation and fast browsing of poictorial content," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 771–785, 1996.

[33] D. Yow, B. L. Yeo, M. Yeung, and G. Liu, "Analysis and presentation of soccer highlights from digital video," in *Proc. Asian Conf. Computer Vision*, 1995.

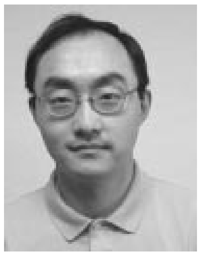
[34] H. J. Zhang, S. W. Smoliar, and J. H. Wu, "Content-based video browsing tools," in *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1995, pp. 389–398.

[35] H. J. Zhang, J. Y. A. Wang, and Y. Altunbasak, “Content-based video retrieval and compression: A unified solution,” *Proc. IEEE Int. Conf. Image Processing*, vol. 1, pp. 13–16, 1997.

[36] D. Zhong and S.-F. Chang, “Spatio-temporal video search using the object based video representation,” *Proc. IEEE Int. Conf. Image Processing*, vol. 1, pp. 21–24, 1998.

Niels Haering received the joint honors B.S. degree in computer science and artificial intelligence from Edinburgh University, Edinburgh, Scotland, in 1993, and the Ph.D. degree in computer science from the University of Central Florida, Orlando, FL in 1999.

He is currently a Senior Software Engineer at Diamondback Vision, Inc., Reston, VA, where he is working on the company’s MPEG-4 video codec and on a DARPA-funded STTR on video surveillance and monitoring.



Richard Qian received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1986, and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign in 1992 and 1996, respectively.

He is currently a Researcher in the Video and Audio Technologies Lab, Intel Labs, Hillsboro, OR. From 1996 to 1999, he was a Researcher at Sharp Labs, Camas, WA. His research interests include video and image analysis, multimedia indexing and retrieval, and infotainment.

M. Ibrahim Sezan (S’80–M’82–SM’91) received the B.S. degrees in electrical engineering and mathematics from Bogazici University, Istanbul, Turkey, in 1980. He received the M.S. degree in physics from Stevens Institute of Technology, Hoboken, NJ, and the Ph.D. degree in electrical computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1982 and 1984, respectively.

He is currently a Senior Manager at the Digital Video Department, Sharp Laboratories of the America, Camas, WA, where he is heading a group focusing on algorithm and system development for video-resolution enhancement, visual-quality optimization, and smart appliances for audiovisual information access, management, and consumption. From 1984 to 1996, he was with Eastman Kodak Company, Rochester, NY, where he founded and headed the Video and Motion Technology Area in the Imaging Research and Advanced Development Laboratories during 1992–1996. He has contributed to a number of books on image recovery, image restoration, medical imaging, and video compression, and has edited *Selected Papers in Digital Image Restoration* (Bellingham, WA: SPIE, 1992) and co-edited *Motion Analysis and Image Sequence Processing* (Norwell, MA: Kluwer, 1993).