A Computational Approach to Semantic Event Detection

Richard Qian Niels Haering[†] Ibrahim Sezan

Sharp Labs of America
Camas, WA 98607
qian,sezan@sharplabs.com

[†]University of Central Florida
School of Computer Science
haering@cs.ucf.edu

Abstract

We propose a three-level video event detection algorithm and apply it to animal hunt detection in wildlife documentaries. The first level extracts texture, color, and motion features, and detects motion blobs. The mid-level employs a neural network to verify whether the motion blobs belong to objects of interest. This level also generates shot summaries in terms of intermediate-level descriptors which combine low-level features from the first level and contain results of mid-level, domain specific inferences made on the basis of shot features. The shot summaries are then used by a domain-specific inference process at the third level to detect the video segments that contain events of interest, e.g., hunts. Event based video indexing, summarization and browsing are among the applications of the proposed approach.

1. Introduction

Existing content-based video indexing and retrieval 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 (a) shot boundary detection and key frame extraction, e.g., [21]; (b) shot clustering, e.g., [19]; (c) table of content creation, e.g., [6]; (d) video summarization, e.g., [14]; and (e) video skimming [17]. 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. The work in the second category tries to classify video sequences into certain categories such as news, sports, action movies, close-ups, crowd, etc. [11, 18]. 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. The work in the third category has been mostly specific to particular domains. For example, methods have been proposed to detect certain events in (a) football games [10]; (b) soccer games [20]; (c) basketball games [16]; (d) baseball games [12]; and (e) sites under surveillance [4]. These methods have the advantage that they are semantically meaningful and significant to users. Unfortunately, many of them 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 query-by-sketch method has also been proposed recently in [1] to detect certain motion events.

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 test the effectiveness of our algorithm, we have applied it to detect animal hunt events in wildlife documentaries. In the next section, we describe the proposed computational framework and its algorithmic components. In Section 3, we present experimental results obtained as we applied the proposed algorithm to detection of animal hunt events in a number of commercially available wildlife video tapes. Implementation details are also furnished in Section 3. Finally in Section 4, we discuss our work and point out future directions.

2. 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 Figure 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

blobs + motion!!

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-speific knowledge.

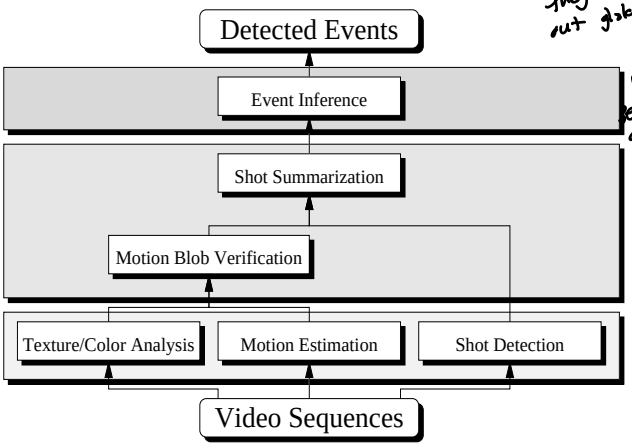


Figure 1. The flowchart of our method.

2.1. Global Motion Estimation and Motion Blob Detection

We assume that the global motion can be estimated with a three parameter system allowing only for zoom, horizontal and vertical translation.

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

A pyramid based correlatioon scheme is employed for recovering the global motion parameters at the four locations shown in Figure 5. Another four estimates are obtained by a local search around the locations predicted by the previous motion estimate.

Patches from uniform areas often result in erroneous displacement estimates. We use the following measure to determine the "amount of texture" and discard patches with insufficient texture.

$$var_x = \sum_{y=0}^{n}(\sum_{x=0}^{m}(p(x,y) - p(.,y))^2 - q_x)^2$$
$$var_y = \sum_{x=0}^{m}(\sum_{y=0}^{n}(p(x,y) - p(x,.))^2 - q_y)^2$$

where p is an $m \times n$ image patch, $p(x,.)$ and $p(.,y)$ are the means of the x^{th} column and y^{th} row of p, and q_x and q_y are the means of $((p(x,y) - p(x,.))^2$ and $(p(x,y) - p(.,y))^2$ for all x and y within p, respectively.

Thus, depending on the "texture" test, we obtain up to eight motion estimates. The motion estimate associated with the best correlation Among the multiple estimates, our algorithm selects the one yielding the best correlation with its source patch.

The estimated global motion paramters 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 using a robust statistical estimation based algorithm [15]. Based on the frame difference result, the algorithm constructs two 1D 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. Figure 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 instantaneous center position and size of a moving object in the video can be estimated based on statistical measurements derived from the two 1D projection histograms. To locate an object in the presence of multiple moving objects, a robust statistical estimation routine has been adopted and described below. Figure 2(b) illustrates this recursive process.

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

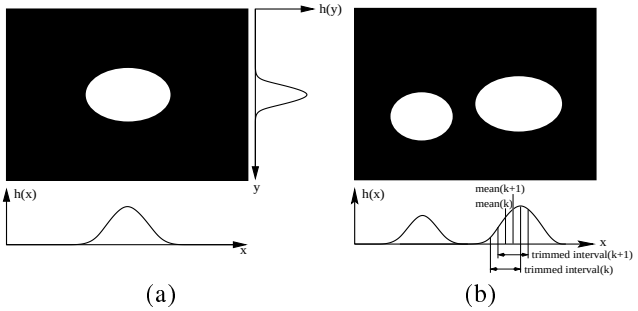
Step 2 Let $\mu_t(0) = \mu$ and $\delta = max(a\ \sigma, b * sampleSpaceWidth)$ where a and b are scaling factors, e.g., $a = 1.0$ and $b = 0.2$.

Step 3 Compute trimmed mean $\mu_t(k+1)$ using 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 Set center-position = the converged mean.

A similar routine has also been adopted for estimating the size of a moving object.



(a) (b)

Figure 2. (a) Two 1D 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.

2.2. Texture and Color Analysis: Low-Level Descriptors

We extract 76 low-level texture and color features: 56 of them are based on the Gray-Level Co-occurrence Matrix, 4 on fractal dimension measures, 12 on Gabor filters, and 4 on color. We found that no single or pair of types of measure (e.g. color and/or Gabor measures) has the power of the combined set of measures [8].

Gabor Filter Measures

Gabor [7] used a combined representation of space and frequency to express signals in terms of "Gabor" functions:

$$F_{\theta,\nu}(\mathbf{x}) = \sum_{i=1}^{n} a_i(\mathbf{x})\, g_i(\theta,\nu) \tag{1}$$

where θ represents the orientation and ν the frequency of the complex Gabor function:

$$g_i(\theta,\nu) = e^{i\nu(xcos(\theta)+ysin(\theta))} e^{-\frac{x^2+y^2}{\sigma^2}} \tag{2}$$

Gabor filter based wavelets have recently been shown [13] to be effective for image retrieval. We obtain 12 features, per pixel, by convolving each frame with Gabor filters tuned to 4 different orientations at 3 different scales.

Graylevel Co-occurrence Matrix Measures

Let $p(i,j,d,\theta) = \frac{P(i,j,d,\theta)}{R(d,\theta)}$, where $P(.)$ is the graylevel co-occurrence matrix of pixels separated by distance d in orientation θ and where $R(.)$ is a normalization constant that causes the entries of $P(.)$ to sum to one.

From previous work [3, 9], we selected the following 14 measures: the angular second moment, difference angular second moment, contrast, inverse difference moment, mean, entropy, difference entropy, difference variance, correlation, shade, and the prominence. As an example of these measures, consider the angular second moment (E), which assigns larger numbers to textures whose co-occurrence matrices are sparse, i.e. regions with few, uniform patches.

$$E(d,\theta) = \sum_{j=1}^{N_g} \sum_{i=1}^{N_g} [p(i,j,d,\theta)]^2 \tag{3}$$

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 $\theta = \{0^o, 45^o, 90^o, \text{and } 135^o\}$ using $d = 1$, yielding a total of 56 measures.

Fractal Dimension Measures

We use the differential box-counting method outlined in [2] to obtain 4 fractal dimension measures. Among them, three features are calculated based on

- the actual image patch $I(i,j)$,
- the high-graylevel transform of $I(i,j)$,
  $I_h(i,j) = \begin{cases} I(i,j) - L_1 & I(i,j) > L_1 \\ 0 & otherwise \end{cases}$
- the low-graylevel transform of $I(i,j)$,
  $I_l(i,j) = \begin{cases} 255 - L_2 & I(i,j) > 255 - L_2 \\ I(i,j) & otherwise \end{cases}$

where $L_1 = g_{min} + \frac{g_{avg}}{2}$, $L_2 = g_{max} - \frac{g_{avg}}{2}$, and g_{min}, g_{max}, and g_{avg} are the minimum, maximum and average grayvalues 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 graylevel in an image patch centered at position (i,j), let $n_r(i,j) = l - k + 1$, and let $\mathcal{N}_r = \frac{n_r}{N_r}$, then the multi-fractal, D_2, is defined by

$$D_2 = \lim_{r \to 0} \frac{log \sum_{i,j} \mathcal{N}_r^2}{log\, r} \tag{4}$$

We use a range of values for r and linear regression of $\frac{log \sum_{i,j} \mathcal{N}_r^2}{log\, r}$ to estimate of D_2.

Color Features

The color features which are used in our approach are the 3 chromatic color measures r, g, b and the normalized intensity I, i.e., $r = \frac{R}{R+G+B}$, $g = \frac{G}{R+G+B}$, $b = \frac{B}{R+G+B}$, and $I = \frac{R+G+B}{R_{max}+G_{max}+B_{max}}$.

2.3. Region Classification and Motion Blob Verification

We use a multi-layer perceptron neural network to classify image regions based on the, N = 76, extracted texture and color features. Our neural network has a single hidden layer with 20 units and uses the sigmoidal activation function $\Phi(act) = \frac{1}{1+e^{-act}} - 0.5$, where act is the activation of the unit before the activation function is applied. The architecture of the network is shown in Figure 3. The network was trained using the back-propagation training algorithm with a total of M = 15 labels: 9 animal labels (lion, cheetah, leopard, antelope, impala, zebra, gnu, elephant, and an all-other-animal class) and 5 non-animal labels (grass, trees, rocks, sky/clouds, and an all-other-non-animal class) as well as a "don't care" label. After training, we found that the proposed network produced good classification results for the following 5 merged classes: all animals, grass, trees, rocks, and sky/clouds.

The output of the network is then used to verify the motion blob candidates from Section 2.1. In our current implementation, a simple procedure is employed which implements the following test. A region that has high residual

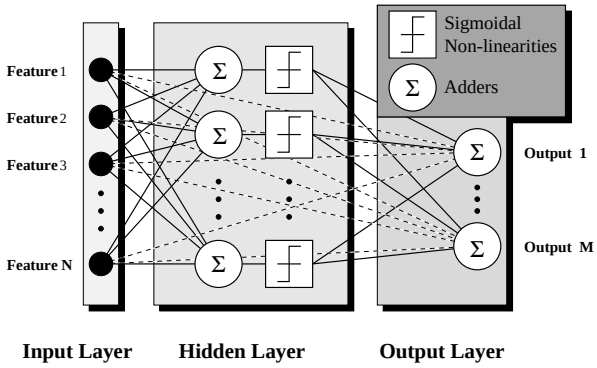


Figure 3. The Neural Network architecture.

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.

2.4. Shot Summarization and Intermediate-Level Descriptors

We use a simple color histogram based technique to decompose video sequences into shots. Additional shot boundaries are also inserted whenever the direction of the global motion changes. Each shot is then summarized in terms of intermediate-level descriptors. To avoid missing important events in extended shots, we also force a shot summary every 200 frames. Intermediate descriptors (1) state the expectations to the low level modules, (2) simplify the algorithm development by making the inference mechanism transparent, and (3) facilitate 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., that indicate the existence of certain objects, (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. [5], that represent motion information about objects and the temporal relations between them.

For the hunt detection application, we currently 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)/(7) the location of the object at the beginning and end of the shot, respectively; (8)/(9) the object size at the beginning and end of the shot; (10) the smoothness of the motion; (11) the precision throughout the shot; and (12) the recall throughout the shot. More precisely, the motion descriptors provide information about the x- and y- translation and zoom components of motion. 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 3.6 for an example and further explanation.

2.5. Event Inference

Hunt events are detected by an event inference module that 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 has 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 which are followed by subsequent shots with slower or no animal motion. In other words, the event inference module looks for a prescribed number of shots in which (a) there is at least one animal of interest; (b) the animal is moving in a consistently fast manner for an extended period; and (c) the animal stops or slows down drastically after the fast motion. Figure 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 any event 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 3.7, we provide an example and further explanation for using this inference model for hunt detection.

3. 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.

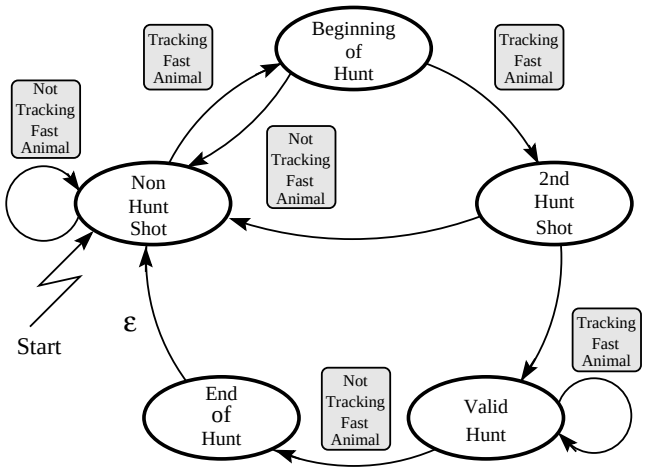


Figure 4. The state diagram of our hunt detection method. Initially the control is in the Non-Hunt state. When a fast moving animal is detected the control moves to the Beginning of Hunt state. When three consecutive shots were found to track fast moving animals then the Valid Hunt flag is set. The first shot thereafter, that does not track a fast moving animal, takes the control to the End of Hunt state, before returning to the Non-Hunt state.

3.1. 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 per second and the digitized frame resolution is 360 x 243 pixels. A total of 10 minutes of footage $\triangleq$ 18000 frames $\triangleq$ 100 shots have been processed so far.

3.2. Global Motion Estimation

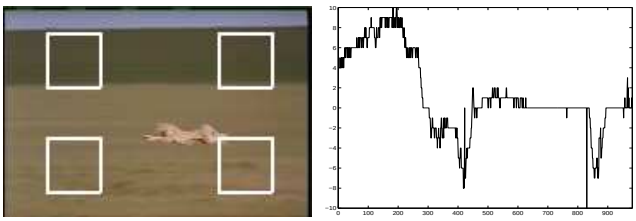
Figure 5(a) shows the size (64×64) and locations of the four regions at which the global motion is estimated. Figure 5(b) shows the motion estimates during a typical hunt. Section 2.1 describes our method for estimating the global motion.

3.3. Motion Blob Detection

Figure 6 shows the motion blob detection results. Reliable estimation and compensation of global motion simplifies the motion blob detection task. When its accuracy is poor, the performance relies on the robustness of the motion blob detection algorithm described in Section 2.1.

3.4. Texture and Color Features

Figure 7 shows the feature space representation of the first frame in Figure 6. The features shown in order are the results of the 56 Gray-Level Co-occurrence Matrix based measures, the 4 Fractal Dimension based measures, the 4 color based measures, and the 12 Gabor filter bank measures.



(a) (b)

Figure 5. (a) The locations used to estimate the global motion, and (b) the horizontal motion estimates during a hunt.

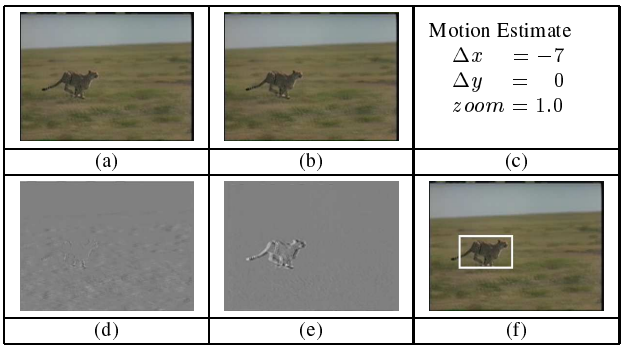


Figure 6. Two consecutive frames from a hunt (a) and (b), the difference image (c), the estimated motion between the two frames (d), the motion compensated difference image (e), and the box around the area of largest residual error in the motion compensated difference image (f).

3.5. Region Classification

A neural network is trained on a number of training frames from wildlife video. The network is then used to classify unseen wildlife video. Rows 1, 3, and 5 of Figure 8 show a number of frames from hunts together with their classification results (rows 2, 4, and 6).

3.6. 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.

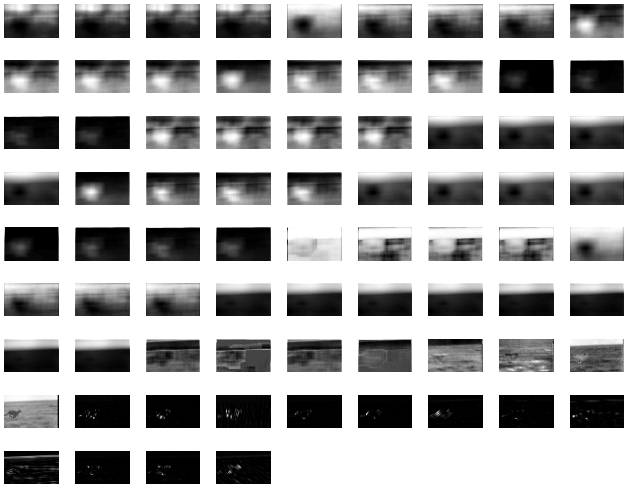


Figure 7. The feature space representation of the first frame in Figure 6.

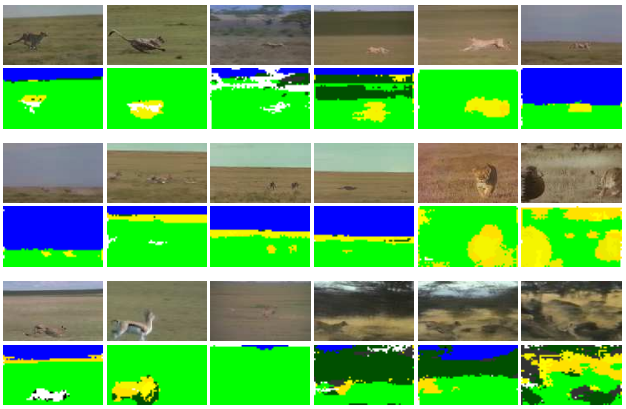


Figure 8. Color and texture based segmentation results.

When the 50 frame global motion averages before and after the current frame have significant *magnitudes* and differ in *sign*, we insert an artificial shot boundary. In the second stage each shot is summarized as 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 of inferences based on those statistics.

The above example summarizes the first candidate hunt shot following a `forced` shot boundary. The descriptors in the above summary were described in Section 2.4. Note that the smoothness measure described here provides a *quantitative* measure of the smoothness of the estimated motion. The detection of a reversal of the global motion direction described earlier, on the other hand, serves as a *qualitative* measure. The `Tracking` predicate is set to 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 set to true if the translational components of the estimated global motion are sufficiently large in magnitude, and the `Animal` predicate is set to true if the precision is sufficiently large. The remaining predicates are determined and used by the inference module as described below.

3.7. 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 3.6. It employs four predicates, `Beginning of hunt`, `Number of hunt shot candidates`, `End of hunt`, and `Valid hunt`. If the intermediate descriptors `Tracking`, `Fast` and `Animal` are true for the first time, `Beginning of hunt` is set to 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 3, `Valid hunt` is set to 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 7 hunt events in the 10 minutes (18000 frames) of wildlife video footage which we have processed. Table 1 shows the actual versus the detected frame numbers of the 7 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.

Table 1. A comparison of the 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%

4. Discussion

The proposed algorithm detects semantic events by detecting certain spatio-temporal phenomena which are physically associated with the events in the real world. More precisely, the physical phenomenon which we attempt to capture in the hunt detection application 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 postproduction editing patterns or other artifacts. The drawback of detecting specific editing patterns or other artifacts is the fact 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, at best, 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.

An immediate focus of future work is to develop a full 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. We also plan to adopt machine learning techniques into the current event inference algorithm.

References

[1] S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong, "A Fully Automated Content Based Video Search Engine Supporting Spatio-Temporal Queries," *IEEE Trans. CSVT*, vol. 8, no. 5, pp. 602-615, 1998.

[2] B.B. Chaudhuri, N. Sarkar, and P. Kundu, "Improved Fractal Geometry Based Texture Segmentation Technique," *IEE Proceedings*, part E, vol. 140, pp. 233-241, 1993.

[3] R.W. Conners and C.A. Harlow, "A Theoretical Comparison of Texture Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 2, no 3, pp. 2 04-222, 1980.

[4] J. D. Courtney, "Automatic Video Indexing via Object Motion Analysis," *Pattern Recognition*, vol. 30, no. 4, pp. 607-626, 1997.

[5] A. Del Bimbo, E. Vicario, and D. Zingoni, "A Spatial Logic for Symbolic Description of Image Contents," *J. Visual Languages and Computing*, vol. 5, pp. 267-286, 1994.

[6] P. England, R.B. Allen, M. Sullivan, and A. Heybey, "I/Browse: The Bellcore Video Library Toolkit," *SPIE Proc. Storage and Retrieval for Image and Video Databases*, pp. 254-264, 1996.

[7] D. Gabor, "Theory of communication," *J. IEE*, vol. 93, pp. 429-457, 1946.

[8] N. Haering and N. da Vitoria Lobo, "Features and Classification Methods to Locate Deciduous Trees in Images," to appear in *J. CVIU*, 1999.

[9] R.M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," *IEEE Trans. SMC*, vol. 3, no. 6, pp. 610-621, 1973.

[10] S. S. Intille, "Tracking Using a Local Closed-World Assumption: Tracking in the Football Domain," *Master Thesis, M.I.T. Media Lab*, 1994.

[11] G. Iyengar and A. Lippman, "Models for Automatic Classification of Video Sequences", *SPIE Proc. Storage and Retrieval for Image and Video Databases*, pp. 216-227, 1997.

[12] T. Kawashima, K. Tateyama, T. Iijima, and Y. Aoki, "Indexing of Baseball Telecast for Content-based Video Retrieval," *Proc. ICIP*, pp. 871-875, 1998.

[13] B.S. Manjunath and W.Y.Ma, "Texture Features for Browsing and Retrieval of Image Data," *IEEE Trans. PAMI*, vol. 18, no. 8, pp. 837-859, 1996.

[14] R. L. Lagendijk, A. Hanjalic, M. Ceccarelli, M. Soletic, and E. Persoon, "Visual Search in a SMASH System", *Proc. ICIP*, pp. 671-674, 1997.

[15] R. J. Qian, M. I. Sezan and K. E. Matthews, "A Robust Real-Time Face Tracking Algorithm", *Proc. International Conference on Image Processing*, pp. 131-135, 1998.

[16] D. Saur, Y.-P. Tan, S.R. Kularni, and P.J. Ramadge, "Automated Analysis and Annotation of Basketball Video," *SPIE Proc. Storage and Retrieval for Image and Video Databases*, pp. 176-187, 1997.

[17] M. Smith and T. Kanade, "Video Skimming for Quick Browsing Based on Audio and Image Characterization," *CMU CS Tech. Rep. CMU CS-95-186*, 1995.

[18] N. Vasconcelos and A. Lippman, "A Bayesian Framework for Semantic Content Characterization," *Proc. CVPR*, pp. 566-571, 1998.

[19] M. Yeung, and B.-L. Yeo, "Video Visualization for Compact Presentation and Fast Browsing of Pictorial Content," *IEEE Trans. CSVT*, vol. 7, no. 5, pp. 771-785, 1996.

[20] D. Yow, B.L.Yeo, M. Yeung, and G. Liu, "Analysis and Presentation of Soccer Highlights from Digital Video," *Proc. ACCV*, 1995.

[21] H. J. Zhang, S. W. Smoliar, and J. H. Wu, "Content-Based Video Browsing Tools," *SPIE Proc. Storage and Retrieval for Image and Video Databases*, pp. 389-398, 1995.