SANDIA REPORT

SAND2005-0779

Unlimited Release

Printed February 2005

Extracting Meaningful Information from Video Sequences for Intelligent Searches

Trina Russ, Maritza Muguira

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: <http://www.osti.gov/bridge>

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: <http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online>

SAND2005-0779
Unlimited Release
Printed February 2005

Extracting Meaningful Information from Video Sequences for Intelligent Searches

Trina Russ and Maritza Muguira

Security Technology Department
and
Intelligent Systems Controls
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0780

Abstract

Video and image data are knowledge-rich sources of information, but their utility for current and future systems is limited without autonomous methods for understanding and characterizing their content. Semantic-based video understanding may benefit systems dedicated to the detection of insiders, alarm patterns, unauthorized activities in material monitoring applications, etc. A direct benefit of this technology is not only intelligent alarm analysis, but the ability to browse and perform query-based searches for useful and interesting information after video data has been acquired and stored. These searches can provide a tremendous benefit for use in intelligence agency, government, military, and DOE site investigations.

This report provides an initial investigation into the algorithms and methods needed to characterize and understand video content. Such algorithms include background modeling, detecting dynamic image regions, grouping dynamic pixels into coherent objects, and robust tracking strategies. With solid approaches for addressing these problems, analysis can be performed seeking to recognize distinctive objects and their motions leading to semantic-based video searches.

CONTENTS

1	INTRODUCTION	5
2	OBJECT SEGMENTATION	5
2.1	BACKGROUND MODEL ESTIMATION	6
2.2	DETECTING MOVING OBJECTS	7
2.3	FORMING COHERENT OBJECTS	9
2.4	FAST CONNECTED COMPONENT ANALYSIS	11
3	OBJECT FEATURES.....	12
3.1	GLOBAL FEATURES	12
3.2	LOCAL FEATURES	13
4	OBJECT TRACKING	15
4.1	OBJECT BASED TRACKING ALGORITHM	15
4.2	LOCAL FEATURE TRACKING ALGORITHM	15
5	SYSTEM ARCHITECTURE	19
5.1	VIDEO STORAGE.....	19
5.2	DATA ORGANIZATION	20
5.3	SEARCHING BY CONTENT	22
6	CONCLUSIONS AND FUTURE WORK.....	25
7	REFERENCES	26

FIGURES

FIGURE 1.	EXAMPLES OF FOREGROUND PIXELS WITH DIFFERENT BACKGROUND MODELS.	8
FIGURE 2	IMAGES CREATED FOR QUASI-CONNECTED COMPONENTS.	10
FIGURE 3.	QUASI-CONNECTED COMPONENTS EXAMPLE.....	11
FIGURE 4	THE OBJECT BOUNDARY FROM TWO DIFFERENT TIME FRAMES.	12
FIGURE 5.	TWO DIFFERENT VIEWS OF A SPACE TIME OBJECT.	13
FIGURE 6.	HORIZONTAL AND VERTICAL GRADIENT IMAGES.	14
FIGURE 7.	KLT FEATURES IN AN IMAGE.....	14
FIGURE 8.	SEQUENTIAL FRAMES IN DYNAMIC UNCONTROLLED ENVIRONMENT.	17
FIGURE 9.	TRANSLATED SHAPES.....	17
FIGURE 10.	ROTATED SQUARE.....	18
FIGURE 11.	SCALED SQUARE	18
FIGURE 12.	DETECTED CHANGE PIXELS IN SAMPLES OF A 19 FRAME VIDEO SEGMENT. GRAY LEVEL VALUES ONLY ASSOCIATED WITH THE CHANGE PIXELS AND THE REFERENCE FRAMES BEFORE AND AFTER MOTION ARE STORED BY THE ARCHIVING SYSTEM.....	20
FIGURE 13.	AN EXAMPLE VIDEO SEGMENT. THIS FIGURE SHOWS SAMPLE FRAMES FROM A 19 FRAME VIDEO SEGMENT ACQUIRED WITH OUR VIDEO ARCHIVING SYSTEM. THESE IMAGES WERE CREATED FROM THEIR COMPRESSED RUN-LENGTH ENCODED FORMATS.	20
FIGURE 14.	EXAMPLE OF THE DATABASE SCHEMA FOR THE VIDEO INDEXING SEARCH ENGINE.....	21
FIGURE 15.	THE CURRENT USER INTERFACE FOR THE VIDEO INDEXING SEARCH ENGINE.	23
FIGURE 16.	USER INTERFACE VIDEO SEARCH CRITERIA.	24
FIGURE 17.	DESIRED VIDEO SEARCH CAPABILITY.....	25

Extracting Meaningful Information from Video Sequences for Intelligent Searches

1 Introduction

The autonomous organization and characterization of video and imaging data is a problem that exists across a number of domains. This is particularly the case for surveillance applications where information is being stored on a regular basis and the amount of data that is generated can be overwhelming. After surveillance data is observed by security personnel, it is of no use if it is not searchable and retrievable. While numerous efforts have been made to address the browsing and searching of image and video data, these efforts have not considered the necessity of providing a mechanism for searching and retrieving data based on the content of images and video itself.

Many products are commercially available for performing content-based video and image indexing by companies like IBM, Virage, Magnifi, DIVAN, LTU, Excalibur, and DataCrystal. Most of these companies examine information at the image level without using high-level object or motion models to extract video content understanding. In addition, the algorithms in these products rely on audio and textual information that may accompany the video to provide content-specific information. Even DataCrystal, who applies advanced signal processing and artificial intelligence technology to model and learn objects on a frame-by-frame basis, never uses the analysis of motion to aid in video content understanding. Instead they rely on text and audio information to resolve difficult scenes. While these approaches may be useful for web-based searches and/or for the broadcast industry, they typically will not function well for surveillance applications where textual and audio information are not readily available.

Other companies like NiceVision and ObjectVideo are making efforts to address the need for intelligent alarm analysis in surveillance applications. However, most of these methods are simple (i.e. determining when certain regions of image are occupied with a dynamic object or determining when motion occurs in a specific image direction). In order to obtain higher-level understanding of video information, higher-level models have to be used to represent its contents from which semantic information can be retrieved. Given that video data encompasses discrete samples of continuous phenomenon, the motion that occurs over the discrete samples should be analyzed to gain understanding about the continuous events.

In this paper the foundational algorithms necessary to perform semantic-based video searches are discussed. In addition, a system architecture is presented to enable searches to be performed using the capability developed under this project. The paper is organized as follows. In Section 2, object segmentation algorithms are discussed using a background modeling based algorithms. Sections 3 and 4 discuss object features and tracking strategies respectively. Section 5 presents the system architecture and finally conclusions are presented in Section 6.

2 Object Segmentation

An important phase of performing video searches is to segment objects of interest that can be later analyzed. Traditional approaches use background subtraction to look for change pixels based on significant differences from a background model. However, for this application it is not sufficient to

just detect pixel level changes, but they must be grouped into coherent objects that can be later analyzed. In order to achieve this successfully good background modeling algorithms are needed in addition to good pixel grouping methods. Accurate background models will enable distinguishing changes from the reference to be identified and good grouping algorithms will enable change regions to be grouped into a description that represents the actual moving object in the scene. These algorithms will be discussed in the following.

2.1 Background Model Estimation

Several approaches have been investigated for background estimation. The most simplistic and first method investigated includes just using a reference frame of the scene before any motion events occur as in (1). While this approach is attractive for its low computational cost, it does not adapt to changes in the background that occur over time due to lighting effects, pixels level noise, and new objects that enter the scene and become part of the background. To better accommodate these types of changes one can slowly update pixel regions that are different from the current background model, but not large enough to be considered a motion region as done in (2) and (3). For instance, only background pixels defined by (7) are updated. In (2) pixel level increments and decrements are provided to model the background and in (3) statistics are used to model a running mean background image based on small changes in the background pixels at time t . One issue with the statistical model is the selection an appropriate α which determines how fast temporally local changes will impact the background model. In many cases an appropriate α will be dependent on the speed of dynamic objects in the scene. Thus, we also investigated a mode-based approach for background modeling which examines the frequency of gray-level occurrence for each pixel to model the background at time t . This approach appears to be more robust to temporally local changes. The storage requirements for this approach are fairly large because it requires the storage of a gray-level histogram for each pixel in the image (i.e. $L \times M \times N$) where an image is of size $M \times N$ and there are L gray-levels of interest. However, the computational cost associated with calculating the most frequently occurring gray level value for each pixel is similar to if not less than other adaptive background model algorithms. The update of the histogram based on image frame I_t is shown in (5). Hence, the mode based background model $B_t^{\text{mode}}(i, j)$ is updated to be $H_t(i, j, I_t(i, j))$ if (6) holds true.

$$B_t^{\text{static}} = I_0 \quad (1)$$

$$B_t = B_{t-1} + W_t \quad (2)$$

$$B_t^{\text{RunAvg}} = (1 - \alpha)B_{t-1}^{\text{RunAvg}} + \alpha I_t \quad (3)$$

$$B_t^{\text{mode}}(i, j) = \max_{l \in \{0, 255\}} H_t(i, j, l) \quad (4)$$

$$H_t(i, j, I_t(i, j)) = H_{t-1}(i, j, I_{t-1}(i, j)) + 1 \quad (5)$$

$$H_t(i, j, I_t(i, j)) > H_t(i, j, B_{t-1}^{\text{mode}}(i, j)) \quad (6)$$

2.2 Detecting Moving Objects

In order to limit the sensitivity of the background model to dynamic objects in the scene, the model is only updated if a pixel (i, j) is classified as a background pixel. Thus, foreground pixels, those pixels identified as deviating significantly from the current background model will not change the background model. Foreground pixels are detected using background subtraction as shown in (7) and (8). The threshold value can be calculated for each pixel or a global threshold can be used such that $TH_{t-1}(i, j) = T$.

$$D_t(i, j) = |B_{t-1}(i, j) - I_t(i, j)| \quad (7)$$

$$\begin{aligned} D_t(i, j) > TH_{t-1}(i, j) & \quad \text{then } (i, j) \in \text{Foreground} \\ \text{otherwise} & \quad (i, j) \in \text{Background} \end{aligned} \quad (8)$$

In the case of (3) a running standard deviation σ_{t-1} can be computed as in (9) and provide information for performing pixel level thresholds. This advances one beyond the global threshold approach so that characteristics of lighting and pixel noise that change across an image can be better modeled. This approach models fitting one Gaussian distribution over the histogram. In this case, the pixel level threshold can be chosen as (10) where k is a constant factor. However, this approach does not handle multi-modal backgrounds; a mixture of Gaussian model would be required to deal with this case.

$$\sigma_t^2 = (1 - \alpha)\sigma_{t-1}^2 + \alpha(B_{t-1}^{RunAvg} - I_{t-1})^2 \quad (9)$$

$$TH_{t-1}(i, j) = k\sigma_{t-1}(i, j) \quad (10)$$

For the purpose of our investigation, we decided to use the mode based background model with a global threshold. We found that this model gave us the most consistent and robust performance for the types of sequences examined. An example of the pixel change regions detected with different background models are shown in Figure 1. For the global threshold pixel change approaches shown in Figures 1a, 1b, 1c, and 1e a threshold of 30 gray-level values is used. The pixel level threshold result shown in Figure 1d uses a k of 5. Thus, a pixel is a foreground pixel if its deviation from the background model is greater than five times the running standard deviation. In addition, for the results in Figures 1c and 1d α is set to .1. Comparing Figures 1c and 1d, one can see that the running Gaussian detects more of the object than the running average; however, it is still sensitive to the high pixel changes around the lights seen at the top of the image. Increasing k will probably remove these detections, but one will also detect less of the object of interest. The mode based approach of Figure 1e does not appear to detect more of the object of interest because parts of the object are so close to the current background that they are not detected. However, it is important to note that this approach has less spurious objects than all of the other approaches presented.

a) Static Background Model

b) Weighted Update Model

c) Running Average

d) Running Gaussian

e) Mode Background Model

f) Current Image

Figure 1. Examples of Foreground Pixels with Different Background Models.

2.3 Forming Coherent Objects

In order to form objects from which descriptions can be obtained, pixel grouping must be performed on the foreground pixels detected in the previous section. While numerous criteria can be used for grouping pixels the most straightforward is the connectivity of pixels in the scene. This is achieved by computing connected components [9]. While this is a good first step, it is often not sufficient if object segmentations that closely resemble the actual object are desired and if it is also desirable to filter out random motions due to blowing leaves, clouds, etc. which often occur in outdoor scenes. The problem with the basic connected component analysis approach is that there will often be gaps in the object description, not because part of the object is occluded, but because part of the object closely resembles the background model and is thus, not detected as a foreground pixel.

Researchers often augment their connected component algorithms using morphological processing. However, in [1] an alternative approach called quasi-connected components (QCC) was used and has shown promising results. This idea was explored in our work and has proven to be quite successful in obtaining more coherent object descriptions. QCC uses the concept of hysteresis thresholding and image downsampling to produce a low resolution image from which connected components can be obtained. Hysteresis thresholding has been very successful in extracting edges from a variety of gradient operators [9]. Hysteresis thresholding typically identifies strong gradients by detecting values with magnitudes greater than a strong threshold T_h and then identifies intermediate values that are greater than a low or minimum threshold value T_l , but less than T_h . The idea is to start at the high threshold pixels and grow the edge to neighboring high or low threshold values. When finished if low threshold values are not connected to higher ones then they are not considered gradient points of interest (i.e. edges).

The quasi-connected algorithm gathers information about the number of change pixels above the high threshold in an image block of the difference image defined in (7) and stores it as an image value in a lower resolution image on which connected component analysis is performed. An example of this is shown in Figure 2, where Figure 2a represents the high/low threshold image where H and L denote the high and low threshold pixels found in the difference image respectively. In this example, the parent image represents the downsampling of the original image by a factor of 2 in both the horizontal and vertical directions. The numbers shown in the image of Figure 2b represents the number of high threshold pixels detected in each 2x2 image block of Figure 2a and likewise for Figure 2c except it represents the number of low threshold pixels in each image block. Connected component analysis is performed on a parent image computed by (11), given that Figure 2b represents P_H and Figure 2c represents P_L .

$$P = P_H \cup P_L \quad (11)$$

In the final analysis connected components of interest in the low resolution image can be maintained by analyzing both the size of the each connected component and the number of high threshold pixels contained within each component. These parameters provide two threshold values that can control the kind of connected components maintained. Note, since the parent image stores the number of high and low threshold pixels in the default resolution difference image, actual component size in the default resolution can be computed from the parent image. While the method is sensitive to the threshold values selected, it does provide a more intelligent method for obtaining coherent object descriptions than morphological processing. This is because morphological operators dilate and erode binary images based on local neighborhood properties, not object properties. While

these approaches are beneficial, they cannot adequately deal with the large gaps that exist in objects that have parts that closely resemble the current background without potentially distorting the object shape. A representation of each object in the default resolution can be obtained by upsampling the lower resolution connected component image. While this does results in the loss of some detail around the border of the object it will be minor as long as the reduction of the default resolution image is reasonable with respect to the size of image objects under investigation. In addition, approaches exist for better preserving object shape when generating the low resolution parent image and can be explored in future work.

(a) Low/High Threshold Image (b) High Threshold Parent Image (c) Low Threshold Parent Image

Figure 2. Images Created for Quasi-Connected Components.

Figure 3 demonstrates the steps needed to create quasi-connected components. Figure 3a is the difference image generated by (7). High/low image thresholding is applied to the difference image to produces the high/low image shown in Figure 3b. The high threshold values are shown in white and the low threshold values are shown in gray. In the high/low image, 4x4 image blocks are processed to produce the parent image. The binary parent image is shown in Figure 3c where white indicates that values were mapped to the specified location. Figure 3d shows an image of the connected components of interested after applying thresholding based on object size and high pixel threshold strength. The final result is shown in Figure 3e which is the up sampled version of the low resolution connected component. For comparison with Figure 3e, the current image is shown in Figure 3f. While we observe that some detail in the boundary of the object has been lost, we also observe that most of the object of interest is identified. The ability to detect and remove shadows will improve the quality of the extracted representation.

a) Difference Image

b) High Low Threshold Image

c) Parent Image High Low Threshold Evidence

d) Parent Image Connected Components

e) High Resolution Connected Components
determined from Parent Image.

f) Current Image

Figure 3. Quasi-connected Components Example.

2.4 Fast Connected Component Analysis

In video analysis applications, speed of processing is very important to achieve real-time or close to real-time processing. Connected component analysis is known to be an $O(K^3)$ algorithm where K is the number of labels created during the algorithm (many of which are equivalent), thus, it is an

appropriate place for optimization. Some of the cost associated with connected component analysis has been reduced by choosing to operate on a lower resolution image as is done in QCC, however, additional benefit can be observed by optimizing this code. For this project, we chose to investigate and implement the Fast Connected Component algorithm implemented in [2]. The algorithm partitions each binary image into $N \times N$ sub-images and performs equivalence resolution (determining which labels are equivalent) on each sub-image while keeping track of global equivalences with a list of pointers to equivalence lists. Considerable improvement in speed can only be realized if N is significant. The results in [2] demonstrated that N should be at least 10.

3 Object Features

In order to define credible criteria for object searches and object correspondences needed for tracking, information has to be obtained about the objects of interest determined by the segmentation algorithm. Such feature information can be classified as global or local features and will be imperative to producing semantic-based search capability. The global features used for this work are simplistic, but are useful for providing meaningful criteria for searches and recognition. Such features include: object size, object center of mass, object boundary descriptions, moments, etc. In contrast, local features are obtained from local neighborhood regions of an object and are defined as features of interest if they adhere to some defined criteria. In this work, the local features are used to provide additional track points for corresponding objects. Later these object trajectories can be analyzed to determine meaningful motions.

3.1 Global Features

Many of the global features are used to define search criteria for objects. However, some of them can be very useful for aiding in the correspondence of objects over time or tracking. Simple features may include object size and location in an image. However, attributes of an object boundary may also be useful. An example of how an object boundary of a human evolves over 90 frames is shown in Figure 5 from two different views. This figure illustrates the gradual evolution of object shape over time as well as the global motion of the object. The 3D representation allows us to observe changes in object shape, object size, and object position over the temporal sequence. In order to better understand the sequence, the first and last boundary images of Figure 5 are shown in Figure 4.

Figure 4. The Object Boundary from Two Different Time Frames.

Figure 5. Two Different Views of a Space Time Object.

3.2 Local Features

Tomasi and Kanade reported good tracking capabilities [4] using the Kanade-Lucas-Tomasi (KLT) tracking method proposed in [5]. The KLT feature selection requires the computation of the

gradient images g_x and g_y from image I where x denotes the horizontal direction and y the vertical direction. From these images we can calculate a local covariance matrix

$$C = \begin{bmatrix} g_x^2 & g_{xy} \\ g_{xy} & g_y^2 \end{bmatrix}. \quad (12)$$

A good feature candidate is identified if one of the eigenvalues, λ_1 and λ_2 , of the local covariance matrix exceeds a predefined threshold.

$$\lambda : \min(\lambda_1, \lambda_2) > \lambda, \quad (13)$$

Figure 6 shows the horizontal and vertical gradients that have been scaled to have values between 0 and 255 (note, gradients can have positive and negative values). Figure 5b displays in white the selected KLT features from the segmented object in Figure 5a. Here we can see than many of the features are clustered around the object boundary where strong gradients exist.

(a) Horizontal Gradients

(b) Vertical Gradients

Figure 6. Horizontal and Vertical Gradient Images.

(a) Intensity image of a segmented object

(b) Detected KLT features. Ten percent of the features with the highest strength were selected.

Figure 7. KLT Features in an Image.

Another type of feature that has received a lot of attention in the literature is the affine invariant feature. As it is named, this feature type is known for its invariance to affine transformations. Mikolajczyk and Schmid demonstrate an iterative method to estimate an affine invariant neighborhood in [6]. They use an affine-adapted version of the Harris detector

$$\mu(x, \sigma_I, \sigma_D) = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} L_x^2(x, \sigma_D) & L_x L_y(x, \sigma_D) \\ L_x L_y(x, \sigma_D) & L_y^2(x, \sigma_D) \end{bmatrix} \quad (14)$$

where σ_I is the integration scale, σ_D is the derivation scale, g is the Gaussian, and L is the Gaussian smoothed image. In general, affine transformations do not change the scale equally in both directions. Thus, a two dimensional, symmetrical Gaussian will not properly scale image regions. Mikolajczyk and Schmid recommend using an elliptical window to convert to an affine scale-space which may be generated by convolution with the non-uniform Gaussian kernel:

$$g(\Sigma) = \frac{1}{2\pi\sqrt{\det \Sigma}} \exp \frac{-x^T \Sigma^{-1} x}{2}. \quad (15)$$

This affine invariant feature will be examined in Section 4.2.

4 Object Tracking

An important capability for intelligent video searches is the tracking of dynamic objects. Two types of object motion are of interest: global object motion and local object motion. Global object motion will characterize any translation and potentially rotation of the object of interest. Local object motion is often referred to as articulated object motion for non-rigid objects and will describe how different rigid object parts move with respect to each other. Local object motions will provide additional detail needed to understand and characterize object motions. In order to define these types of motions features need to be autonomously extracted from an image (as discussed in Section 3), described in a unique fashion and matched in subsequent images in order to track their motion. In this section we will discuss an object based tracking algorithm and a local feature tracking algorithm.

4.1 Object Based Tracking Algorithm

The purpose of the blob based tracking strategy is to create a temporal object consisting of collection of objects at sequential time frames. This results in a collection of pixels that have been grouped in space and time. Initially this is performed using object size and location to determine corresponding objects. The algorithm is initialized with the very first time frame from which a temporal object is created for each object in the image frame (note each temporal object has length 1). For each subsequent frame each object is visited and the list of temporal objects is searched for the most similar object in terms of location in the image and size of the object. For simple scenes containing a few objects that are well separated this approach is reasonable. For more complicated scenes additional criteria should be used such as boundary shape features, global shape features, or even local features.

4.2 Local Feature Tracking Algorithm

Local feature tracking requires each feature to be independently matched from the current image I_t to the subsequent image I_{t+1} using a local $w_a \times w_y$ image patch centered around feature point (u, v) . The matching is accomplished by minimizing the error

$$\min_{d_x, d_y \in D} \varepsilon(u, v, d_x, d_y) \quad (16)$$

$$\varepsilon(u, v, d_x, d_y) = \sum_{x=u-w_x}^{u+w_x} \sum_{y=v-w_y}^{v+w_y} \left(I_t(x, y) - I_{t+1}(x + d_x, y + d_y) \right)^2 \quad (17)$$

where d_x and d_y are the displacements for the horizontal and vertical directions respectively, and D defines the search region. Furthermore, the features may be searched in coarser resolution images by smoothing and decimating the images to form a scale space pyramid level P defined as

$$\begin{aligned} I^P(x, y) = & \frac{1}{4} I^{P-1}(2x, 2y) + \\ & \frac{1}{8} [I^{P-1}(2x-1, 2y) + I^{P-1}(2x+1, 2y) + I^{P-1}(2x, 2y-1) + I^{P-1}(2x, 2y+1)] + \\ & \frac{1}{16} [I^{P-1}(2x-1, 2y-1) + I^{P-1}(2x+1, 2y+1) + I^{P-1}(2x-1, 2y+1) + I^{P-1}(2x+1, 2y-1)] \end{aligned} \quad (18)$$

For our application, we focused on dynamic objects and thus selectively chose features only from identified changed regions as described in Section 2. Reducing the areas from which to extract and search for features greatly improves the throughput of the video stream. Tracking would be more robust if features could be correlated with corresponded objects identified from frame to frame. Unfortunately, we were unable to complete this capability prior to the completion of the project.

Features of interest were selected as a small percentage of the number of pixels in each identified object. The search for corresponding features in subsequent frames was confined to a window D as described in (16). Large search windows allow the accommodation of greater motion, but smaller windows typically provide greater localization accuracy. The matching of feature points based upon a few pixels (i.e. a small local window $w_x \times w_y$) is not an optimal solution especially in uncontrolled dynamic environments. For instance, since humans have articulated members, the overall shape and relative positioning of the test subject can change considerably. Furthermore, the object rotation causes exposure to new surfaces as well as occlusion of previous ones. Consider the changes found from frame to frame in Figure 8. The subject's face, albeit at a different scale, is similarly posed in both frames. If we select the features from a small area, we risk ambiguity. However, the analysis of larger regions may include more points from articulated objects with non-rigid surfaces whose relative positioning will not hold.

We tested tracking with both KLT and affine invariant feature points through 81 sequential frames using a 5×5 feature support window. Although some of the tracking points bounced around within the 9×9 displacement window, the tracking maintained the feature on the object or near the perimeter through four frames. In fact, features were commonly found along the perimeter of the moving objects. Perimeter points do not make good features in rotating or articulated objects since they change drastically. We tried feature selection and tracking in the different pyramid levels described in (18) but found the best performance in the highest resolution level. Feature selection in coarse resolution levels identify perimeters points which are not good features, however, the more detailed features identified in higher resolutions of the image pyramid are lost as the images are

decimated. Overall, it was difficult to assess the exact cause of lost feature points with such drastic changes in scale, translation, and rotation between frames.

Figure 8. Sequential Frames in Dynamic Uncontrolled Environment.

In order to quantify the performance of the point feature tracking algorithm, we devised a controlled experiment to test a square and circle through a series of translated, rotated, and scaled images. Figures 5, 6, and 7 show sequential image frames as they undergo the respective test transformations. Again, we found the features bounced around during tracking, however, we did not consider them lost if they were maintained on the target or close to it.

Figure 9. Translated Shapes

Figure 10. Rotated Square

Figure 11. Scaled Square

Table 1 summarizes the finding of these experiments. It is crucial to note that these tolerances provide an upper bound of maximum performance for these objects. A more cluttered environment will only serve to degrade the performance as the points are more apt to be falsely matched with background points or other objects. Also, note that the maximum tolerances exceed the displacement window size. This is possible because we allowed the points to move along an edge without considering it lost. This sort of low correlation matching may suffice with a well characterized background and limited foreground objects and motion. The maximum tolerances for these samples should not be confused to indicate the maximum tolerances for any shape in general. Because of the “loose” matching, the shape of the object as related to the direction of motion will directly effect this value.

Table 1. Transformation Tolerances					
Transformation	Maximum Tolerance (for these samples)	Feature Selector	Displacement Window (pixels)	Number of Features	Features Lost
Translation	37 pixels	KLT	20	4	3
Translation	42 pixels	KLT	40	4	2

Translation	42 pixels	Affine Invariant	20	6	4
Translation	42 pixels	Affine Invariant	40	6	2
Rotation	20 degrees	KLT	20	12	2
Rotation	30 degrees	KLT	40	17	6
Scale	41 pixels	KLT	20	10	2
Scale	41 pixels	KLT	40	10	0

5 System Architecture

In order to demonstrate system capability and to address issues pertinent to video searches a preliminary video indexing system was developed. The implemented video indexing system was partitioned into three major units: the video archiving unit, the video processing unit, and the video query unit. The video archiving unit enables the compact efficient acquisition of large amounts of video information over long periods of time. The video processing unit examines the stored image data for the video archive to create object clusters from which object attributes can be extracted and on which tracking algorithms can be performed. The important information extracted from this stage is then written to a database file that the video query unit can examine for interesting video segments based on user input. The current organization was motivated by the need to efficiently store video to address the large storage requirements without significantly compromising video quality. In addition, the ability to separate the video processing/analysis from the video streaming was desired. This enables a flexible organization in the event that some of the processing capability is computationally expensive and cannot be computed in a real-time environment.

5.1 Video Storage

One of the main issues in video archiving is the storage space required for logging video information. A single, moderately sized black and white digital image of 640x480 pixels requires just over 300 kilobytes of digital storage space (uncompressed). At 30 frames per second, the amount of memory required for just one second of video is approximately 10 megabytes. One minute of video requires approximately 600 megabytes. One hour requires nearly 36 gigabytes and one day requires nearly 1000 gigabytes. Storing video images at one frame per second (as is done in a time-lapse video tape recording) reduces storage requirements to approximately 30 gigabytes per day. Using the best available compression technology to further compress data reduces storage requirements by another factor of 10 to approximately 3 gigabytes per day. This still is an exceedingly large requirement to prove cost effective using currently available digital-storage media.

Because of the large amount of data typically acquired in surveillance applications it was necessary to acquire pertinent video information in a compressed format that would enable the acquisition of data acquired over a long period of time without requiring a significant amount of storage capacity. For most surveillance systems, cameras are deployed at particular locations for the purpose of

monitoring activity within a defined region. The adopted approach for acquiring sample video archives followed this philosophy.

An intelligent digital-video archiving system was developed that uses image processing to identify only “interesting” portions of a video sequence for storage. An example of “interesting” content could include frames in which “purposeful motion” or changes have been detected. The motion of humans, animals, and vehicles generally fits this category. Static imagery is mostly non-interesting as is motion or changes due to environmental factors. We have leveraged the capabilities of current video motion detection systems developed at Sandia to provide the necessary capabilities. We also developed computationally simple video compression technology specifically suited to video archiving.

A video archive is defined by a set of video segments that are composed of a set of image frames over which significant motion is detected. Motion and change regions were identified using the methods discussed in Section 2 and the result is shown in Figure 12. For each video segment, we store the frame before motion, the frame after motion, a “stats” record and a run-length encoded (RLE) image sequence. Run-length encoding is a computationally simple technique we have employed to compress and store the locations of change pixels and their corresponding gray-levels. Using this technique, images with motion, as portrayed in Figures 1 and 2 below, are typically compressed to about 5 Kilobytes per image. The stats record contains additional low-level information such as image type (reference image before motion sequence, motion image, reference image after motion sequence), time of image capture, and coarse locations of detected activity within motion images. From the RLE images and the image frame before motion a close approximation to the original video segment can be created as shown in Figure 13.

Figure 12. Detected Change Pixels in Samples of a 19 Frame Video Segment. Gray level values only associated with the change pixels and the reference frames before and after motion are stored by the archiving system.

Figure 13. An Example Video Segment. This figure shows sample frames from a 19 frame video segment acquired with our video archiving system. These images were created from their compressed run-length encoded formats.

5.2 Data Organization

Video archive data was decomposed into several hierarchical levels in which each lower level is a member (or child) of the higher level. This facilitates the acquisition and storage (to a database (DB)) of information pertinent to that level and enables users to perform high-level (video archive) or low-level (video object) searches. These levels in order from highest to lowest are as follows: video archive, video segment, image frame, and image object. In addition, a temporal object structure was developed that embodies important attributes associated with how an object evolves over time.

Each of the hierarchical levels is represented by a Microsoft Access database table containing members with searchable criteria. An example of the simple video indexing data schema is shown in Figure 14. This figure clearly displays the relationships between the data tables. A video archive is composed of a collection of video segments. A video segment is composed of a collection of image frames. An image frame consists of a collection of objects. In addition, a dynamic object or temporal object consists of a collection of objects related in time. A brief discussion of the attributes stored in each table is presented below.

Figure 14. Example of the Database schema for the Video Indexing Search Engine.

- **Video Archive Descriptions** – Attributes obtained and stored to a DB table are as follows: number of video segments.
- **Video Segment Descriptions** - Attributes obtained and stored to a DB table are as follows: time, date, frame length, number of objects (NOO) at segment start, and NOO at segment end.
- **Image Frame Descriptions** - Attributes obtained and stored to a DB table are as follows: time, date, number of change pixels, and NOO in frame.
- **Object Descriptions** – Attributes obtained and stored to the DB table include: size, location, date, time, and central moment features (up to the second order).

- **Dynamic or Temporal Object Descriptions:** Attributes that can be stored in the database include the objects it consists of, start and end frames, and start and end times. Additional attributes that can be added include speed of motion, direction of motion, and other criteria derived from space-time trajectories.

5.3 Searching by Content

Initial GUI and database search engines have been implemented to support the searching of video archives for content-specific information. This system essentially provides a GUI in which a user can input limited search criteria and subsequently the query system searches the DB tables created in the video processing unit and returns a list of the relevant video segments. For the returned video segments the user can both view video segment properties and the associated image frames.

An example of the current user interface is shown in Figure 15. It operates by enabling either a video archive to be loaded for processing or a database to be loaded for searching. Under “Query Search Parameters” criteria are defined for searching as shown in Figure 16 for video segments, frames, and/or objects. When the “Search Database” button is pushed, the database is searched and video segments meeting the defined criteria are listed in the table at the bottom of the GUI under “Query Results”. Once the results are loaded, one can view all attributes of the video segment including frames and objects. A double-click on a video segment row will bring up the video segment image for browsing.

Figure 15. The Current User Interface for the Video Indexing Search Engine.

(a) Video Segment Search Criteria

(b) Video Frame Search Criteria

(c) Image Object Search Criteria

Figure 16. User Interface Video Search Criteria.

Throughout this project we have been formulating the type of searches that would be advantageous to a user. In fact, the information that we have obtained from the object clusters support these ideas. A few of the searches that can be performed based on information obtained thus far are listed below.

1. Search for objects within a certain size interval
2. Search for image frames that contain motion in particular image regions.
3. Search for image frames based on the number of change pixels observed.
4. Search for image frames that contain multiple moving objects. A number of objects can be specified.
5. Search for image frames obtained during a certain time interval.
6. Look for video segments where a change in background was detected. For example, items that were part of the background disappeared during the segment or new items became a part of the background during the segment.

In addition to the singular criteria listed above, searches can be performed using multiple attributes. For instance, we can look for image frames containing objects of a certain size occupying a well-defined image region. Also once we improve our tracking methods we can begin to perform searches based on objects moving in a particular direction and/or with a certain velocity. The vision of the system is to enable of host of searches at different levels as shown in Figure 17. We are currently about half way through this vision; additional algorithm development is needed to complete it.

Figure 17. Desired Video Search Capability

6 Conclusions and Future Work

An initial system has been implemented to perform video searches based on content. Additional work is needed to achieve the desired goal of semantic based searches but this work has laid a solid foundation to pursue this goal in the future in terms of implementation of foundational algorithms and system architecture development. In order to achieve semantic-based searches the challenges are many ranging from accurate background modeling to developing novel techniques for grouping pixels to determining robust criteria for corresponding objects and matching local object features. In addition, sophisticated algorithms may be required to analyze this information and recognize objects, their motions, and how objects interact with each other. The spatiotemporal boundary shown in Figure 3 provides a visual summary of what occurs over a temporal sequence and a similar representation should be exploited to address issues related to spatiotemporal data analysis and intelligent searches.

7 References

- [1] T. E. Boulton, R. J. Michaels, X. Gao, and M. Eckmann, “Into the woods: visual surveillance of non-cooperative and camouflaged targets in complex outdoor settings”, *Proceedings of the IEEE*, vol. 89, no. 10, Oct. 2001, pp. 1382-13402.
- [2] J. Park, C. G. Looney, and H. Chen, “Fast connected component labeling algorithm using a divide and conquer technique”, *International Conference on Computers and their Applications*, March 2000, pp. 373-376.
- [3] J. Shi and C. Tomasi, “Good features to track”, *IEEE Conference on Computer Vision and Pattern Recognition*, June 1994, pp. 593-600.
- [4] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision”, *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981.
- [5] C. Tomasi and T. Kanade, “Detection and tracking of point features”, *Carnegie Mellon University Technical Report CMU-CS-91-132*, April 1991.
- [6] K. Mikolajczyk and C. Schmid, “An affine invariant interest point detector”,
- [7] D. Lowe, “Distinctive image features from scale-invariant features”, *International Journal on Computer Vision*, vol. 60(2), pp. 91-110, 2004.
- [8] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2003.
- [9] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, McGraw Hill, Bost, MA, 1995.

DISTRIBUTION:

1	MS0780	Stephen Ortiz, 04128
3	MS0780	Trina Russ, 04128
1	MS1003	Maritza Muguira, 15234
1	MS9018	Central Technical files, 8945-1
2	MS0899	Technical Library, 9616
1	MS0188	D. Chavez, LDRD Office, 1030