Object and Event Recognition for Aerial Surveillance

Yi Li, Indriyati Atmosukarto, Masaharu Kobashi, Jenny Yuen and Linda G. Shapiro

University of Washington, Department of Computer Science and Engineering, Box 352350, Seattle, WA 98195-2350, U.S.A.

ABSTRACT

Unmanned aerial vehicles with high quality video cameras are able to provide videos from 50,000 feet up that show a surprising amount of detail on the ground. These videos are difficult to analyze, because the airplane moves, the camera zooms in and out and vibrates, and the moving objects of interest can be in the scene, out of the scene, or partly occluded. Recognizing both the moving and static objects is important in order to find events of interest to human analysts. In this paper, we describe our approach to object and event recognition using multiple stages of classification.

Keywords: object recognition, event recognition, machine learning, aerial surveillance

1. INTRODUCTION

Unmanned aerial vehicles (UAVs) are able to provide large amounts of video data over terrain of interest to defense and intelligence agencies. These agencies are looking for significant events that may be of importance for their missions. However, most of this footage will be eventless and therefore of no interest to the analysts responsible for checking it. If a computer system could scan the videos for potential events of interest, it would greatly lessen the work of the analyst, allowing them to focus on the events of possible importance.

Several different processes are needed for the computer analysis of aerial videos. First, the static objects in the video frames must be recognized to determine the context of the events. Static objects might include forests, fields, roads, runways, and buildings, among others. Next the moving objects in the video must be detected, tracked, and identified. Moving objects include vehicles (cars, trucks, tanks, and buses) and people. Given the static objects and moving objects in a set of frames, events are defined by the actions of the moving objects and their interactions with the static objects. For example, two cars might pull off a road and stop together in a field. People might get out of the cars and approach each other for a meeting. A caravan of trucks might travel in one direction on a dirt road for a period of time and then make a U-turn and proceed in the opposite direction. A vehicle might pull up to a building and disappear into an underground garage or tunnel, then reappear some time later. In all of these cases, both the moving objects and the static objects must be recognized and their interactions noted.

We are developing a system for object and event recognition for this purpose. In this paper we describe the structure of our system and give brief overviews of the underlying algorithms.

2. SYSTEM OVERVIEW

In order to recognize events in a video, the moving objects across a sequence of frames and the static objects in each of these frames must be detected and recognized. Then using these moving and static objects as primitives, simple events can be defined in terms of the relationships among the moving objects and between the moving objects and the static ones. Finally, more complex events can be defined as sequences of simple events. For example, simple events such as a vehicle appearing on a road, moving forward for a short distance, and disappearing behind a tree would lead to a complex event as shown in the first row of Figure 1. Three other complex events (a convoy of cars making a U-turn, a car overtaking another car, and a truck making a turn and passing by a line of cars in the opposite direction) are also shown in Figure 1.

Further author information: Send correspondence to Linda Shapiro, E-mail: shapiro@cs.washington.edu, Telephone: 1 206 543 2196; Yi Li is now with Vident Systems, Inc., Sunnyvale, CA, U.S.A.

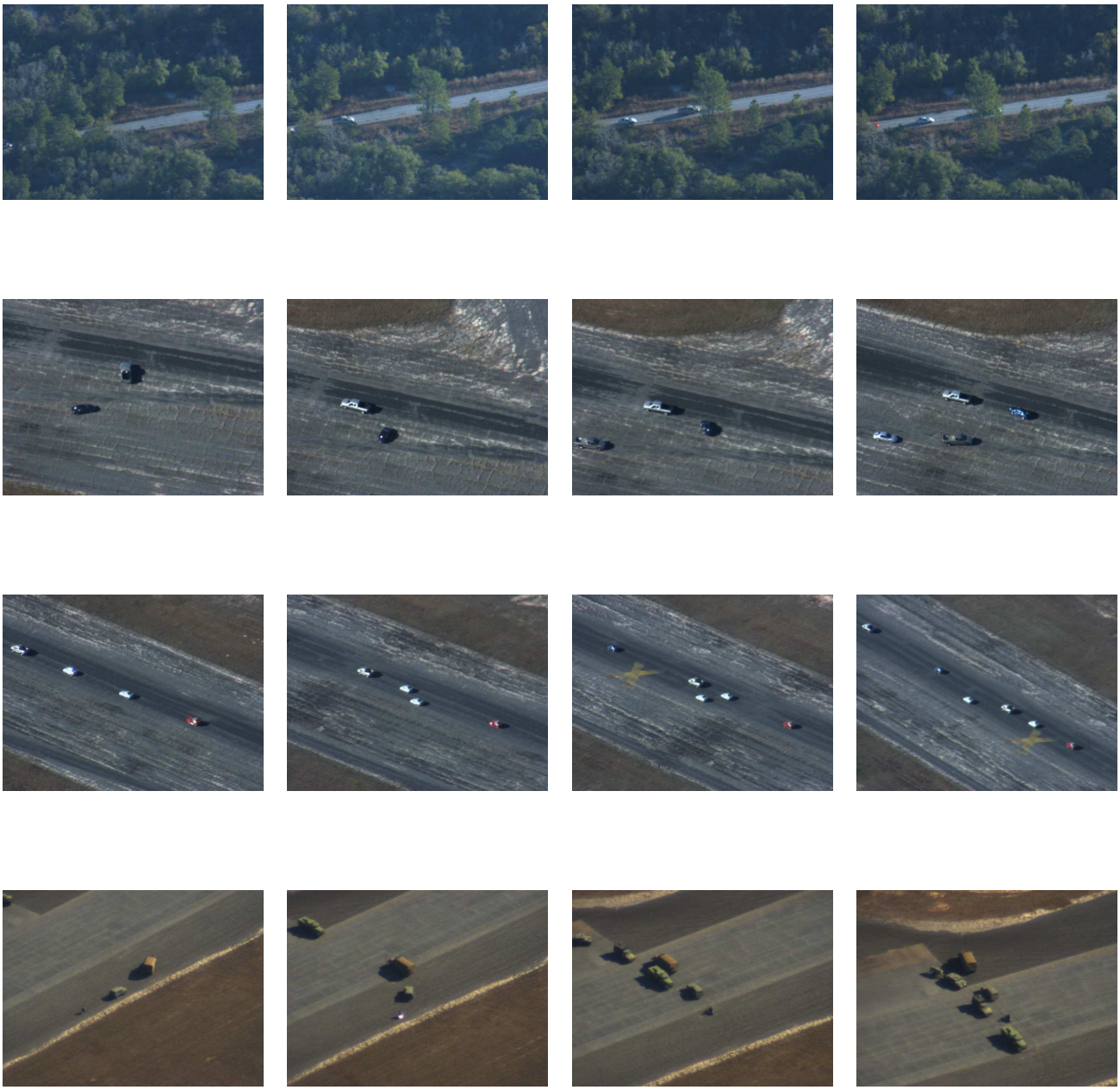


Figure 1. Examples of events. (Row 1) Vehicle disappears behind a tree, (Row 2) Cars making a Uturn, (Row 3) Car overtaking another car, (Row 4) Truck makes a turn and passes by cars moving in the opposite direction.

Figure 2 shows the architecture of our system. The system receives a video sequence as its input. The static feature extraction module will extract region features such as color regions, texture regions and structure regions from the static objects in the video, while the dynamic feature extraction module will extract features from the objects that are moving in the video. The object recognition module will use the features extracted by both the static and dynamic feature extraction modules to label the objects in the frames, while the object tracking module will track the objects that are moving from frame to frame. Relationships between objects over time such as relative position between objects within a frame will be computed by the object relationship extraction module. The results of all three modules: object recognition, object tracking and object relationship extraction will be used by the event recognition module to recognize the events happening in the video and output the results.

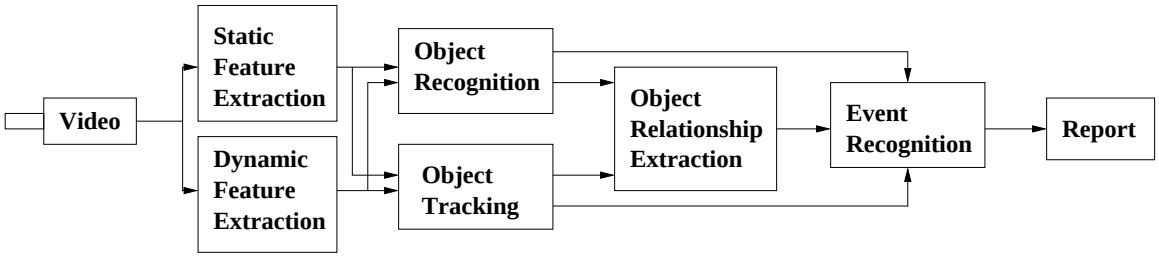


Figure 2. The architecture of our event recognition system.

3. STATIC OBJECT CLASS RECOGNITION

3.1. Abstract region features

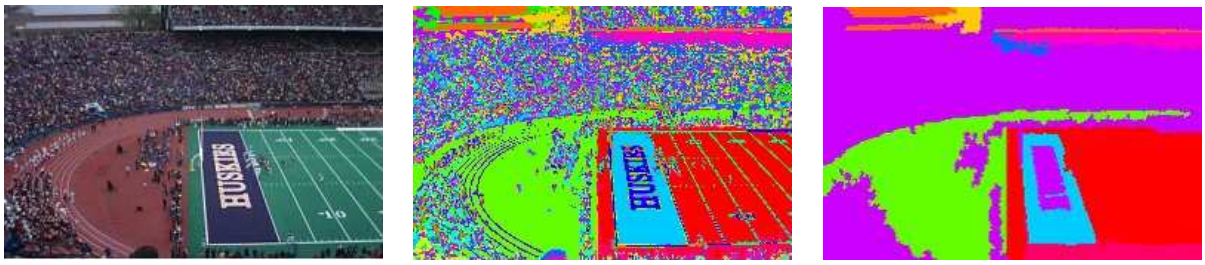
Our methodology for object recognition has three main parts:

1. Select a set of features that have multiple attributes for recognition and design a unified representation for them.
2. Develop methods for encoding complex features into feature vectors that can be used by general-purpose classifiers.
3. Design a learning procedure for automating the development of classifiers for new objects.

The unified representation we have designed is called the *abstract region* representation. The idea is that all features will be regions, each with its own set of attributes, but with a common representation. The regions we are using in our work are color regions, texture regions and structure regions defined as follows.

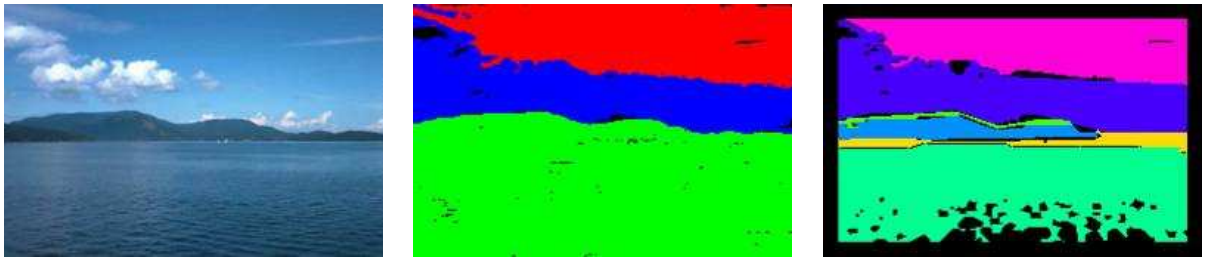
Color regions are produced by a two-step procedure. The first step is color clustering using a variant of the K-means algorithm on the original color images represented in the CIELab color space.[1] The second step is an iterative merging procedure that merges multiple tiny regions into larger ones. Figure 3 illustrates this process on a football image in which the K-means algorithm produced hundreds of tiny regions for the multi-colored crowd, and the merging process merged them into a single region. Our texture regions come from a color-guided texture segmentation process. Color segmentation is first performed using the K-means algorithm. Next, pairs of regions are merged if after a dilation they overlap by more than 50%. Each of the merged region is segmented using the same clustering algorithm on the Gabor texture coefficients. Figure 4 illustrates the texture segmentation process.

The features we use for recognizing man-made structures are called *structure features* and are obtained using the concept of a *consistent line cluster*.[2] These features are obtained as follows:



Original Color Merged

Figure 3. Illustration of merging of tiny color regions.



Original image Color segmentation Texture segmentation

Figure 4. The texture segmentation is color-guided: it is performed on regions of the initial color segmentation.

1. Apply the Canny edge detector[3] and ORT line detector[4] to extract line segments from the image.
2. For each line segment, compute its orientation and its color pairs (pairs of colors for which the first is on one side and the second on the other side of the line segment).
3. Cluster the line segments according to their color pairs, to obtain a set of color-consistent line clusters.
4. Within the color-consistent clusters, cluster the line segments according to their orientations to obtain a set of color-consistent orientation-consistent line clusters.
5. Within the orientation-consistent clusters, cluster the line segments according to their positions in the image to obtain a final set of consistent line clusters.

Figure 5 illustrates the abstract regions for several representative images. The first image is of a large campus building at the University of Washington. Regions such as the sky, the concrete, and the large brick section of the building show up as large homogeneous regions in both color segmentation and texture segmentation. The windowed part of the building breaks up into many regions for both the color and the texture segmentations, but it becomes a single region in the structure image. The structure-finder also captures a small amount of structure at the left side of the image. The second image of a park is segmented into several large regions in both color and texture. The green trees merge into the green grass on the right side in the color image, but the texture image separates them. No structure was found. In the last image of a sailboat, both the color and texture segmentations provide some useful regions that will help to identify the sky, water, trees and sailboat. The sailboat is captured in the structure region. It is clear that no one feature type alone is sufficient to identify the objects.

In our framework for object and concept class recognition, each image is represented by sets of abstract regions and each set is related to a particular feature type. To learn the properties of a specific object, we must know which abstract regions correspond to it. Once we have the abstract regions from an object, we extract the common characteristics of those regions as the model of that object. Then given a new region, we can compare it to the object models in our database to decide to which it belongs. We designed the algorithm to learn

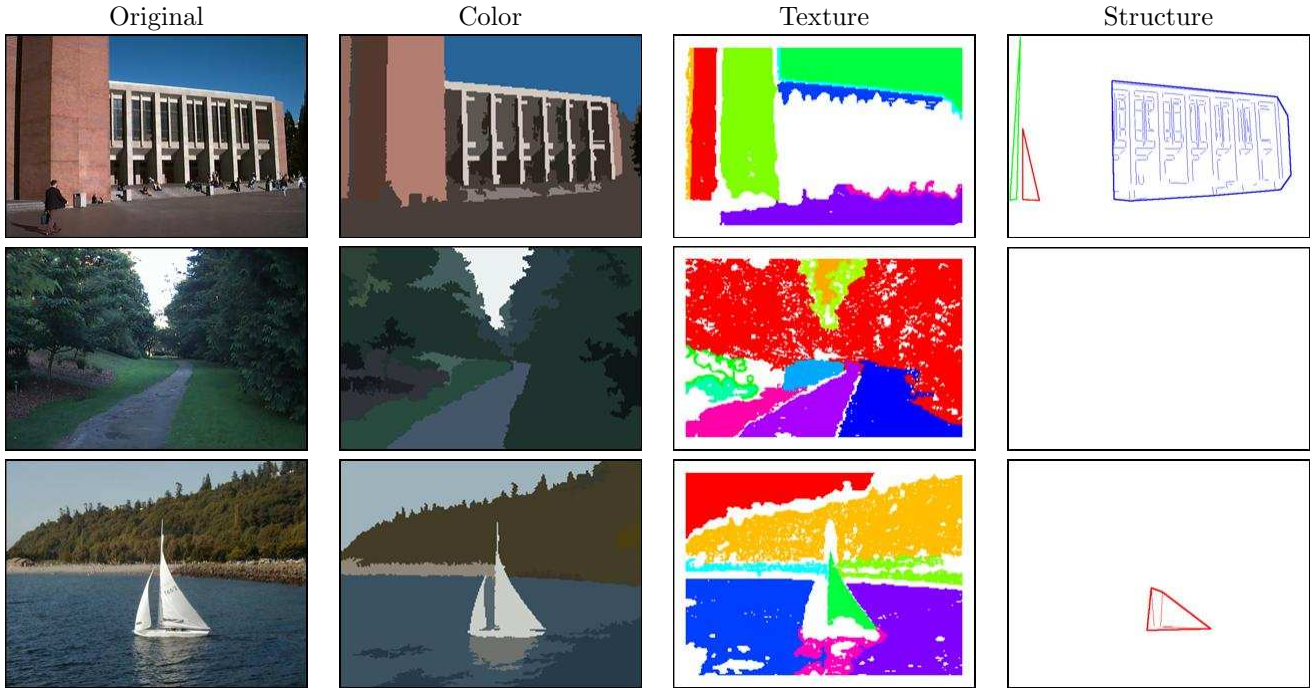


Figure 5. The abstract regions constructed from a set of representative images using color clustering, color-guided texture clustering and consistent-line segment clustering.

correspondences between regions and objects in the training images to require only the list of objects in each training image. With such a solution, not only is the burden of constructing the training data largely relieved, the principle of keeping the system open to new image features is upheld.

3.2. EM-Variant Approach to Object Classification

Our object recognition methodology uses whole images of abstract regions, rather than single regions for classification. A key part of our approach is that we do not need to know where in each image the objects lie. We only utilize the fact that objects exist in an image, not where they are located. We have designed an EM-like procedure that learns multivariate Gaussian models for object classes based on the attributes of abstract regions from multiple segmentations of color photographic images.[5]

The objective of this algorithm is to produce a probability distribution for each of the object classes being learned. It uses the label information from training images to supervise EM-like iterations. In the initialization phase of the EM-variant approach, each object is modeled as a Gaussian component, and the weight of each component is set to the frequency of the corresponding object class in the training set. Each object model is initialized using the feature vectors of all the regions in all the training images that contain the particular object, even though there may be regions in those images that do not contribute to that object. From these initial estimates, which are full of errors, the procedure iteratively re-estimates the parameters to be learned. The iteration procedure is also supervised by the label information, so that a feature vector only contributes to those Gaussian components representing objects present in its training image. The resultant components represent the learned object classes and one background class that accumulates the information from feature vectors of other objects or noise. With the Gaussian components, the probability that an object class appears in a test image can be computed. The EM-variant algorithm is trained on a set of training images, each of which is labeled by the set of objects it contains. For each test image, it computes the probability of each of the object classes appearing in that image. Figure 6 shows some sample classifications using abstract regions with both color and texture properties.

Figure 6. Classification results for grass and tree using the EM-variant approach with regions having both color and texture attributes.

3.3. Generative/Discriminative Approach to Object Classification

Our two-phase generative/discriminative learning approach addresses three goals[6]:

1. We want to handle object classes with more variance in appearance.
2. We want to be able to handle multiple features in a completely general way.
3. We wish to investigate the use of a discriminative classifier to add more power.

Phase 1, the generative phase, is a clustering step that can be implemented with the classical EM algorithm (unsupervised) or the EM variant (partially supervised). The clusters are represented by a multivariate Gaussian mixture model and each Gaussian component represents a cluster of feature vectors that are likely to be found in the images containing a particular object class. Phase 1 also includes an aggregation step that has the effect of normalizing the description length of images that can have an arbitrary number of regions. The aggregation step produces a fixed-length feature vector for each training image whose elements represent that image's contribution to each Gaussian component of each feature type.

Phase 2, the discriminative phase, is a classification step that uses the feature vectors of Phase 1 to train a classifier to determine the probability that a given image contains a particular object class. It also generalizes to any number of different feature types in a seamless manner, making it both simple and powerful. We currently use neural net classifiers (multi-layer perceptions) in Phase 2.

The two-phase generative/discriminative approach has several advantages. It is able to combine any number of different feature types without any modeling assumptions. Regions from different segmentations do not have to align or to correspond in any way. Segmentations that produce a sparse set of features, such as our structure features, can be handled in exactly the same manner as those whose feature cover the entire image. This method can learn object classes whose members have several different appearances, such as trees or grass. It can also learn high-level concepts or complex objects composed of several simpler objects, such as football stadium, which has green turf, a structural pattern of white lines, and red track around it. Figure 7 shows sample classifications using color, texture, and structure features in the generative/discriminative approach.

forest(94.37), house(64.09), car(46.5), dirt road(23.44), paved road(4.77) tree(2.29), airplane(1.47), runway(0.03), field(0.02), people(0)

runway(99.98), field(98.66), car(96.24), people(10.04), airplane(2.74), paved road(2.39), forest(0.82), house(0.48), dirt road(0.41), tree(0)

runway(100), car(99.23), field(98.07), dirt road(92.1), house(85.24), tree(19.43), paved road(5.77), airplane(3.56), forest(2.85), people(0.07)

runway(99.98), car(99.84) field(99.27), paved road(18.28), people(13.13), tree(8.71), airplane(7.94), forest(1.67), house(0.14), dirt road (0.08)

car(94.3), dirt road(91.7), field(16.17), tree(14.23), paved road(5.34), airplane(5.17), people(3.91), forest(0.53), house(0.47), runway (0.41)

car(97.92), forest(94.2), paved road(85), dirt road (72.94), tree(68.84), airplane(39.13), house (33.17), people (12.97), field(2.38), runway (0.04)

Figure 7. Sample classification using the generative/discriminative approach. The boldface labels are true labels (groundtruth) of the images.

3.4. Localization

Since the probabilities computed by the system are based on abstract regions, we can analyze the learning procedure to determine which regions are most important in the decision-making process. The localization procedure assumes that those regions that contribute most to the decision of whether an object has a high probability of being in an image are most likely to contain that object. The procedure uses the output of the multi-layer perceptron of Phase 2 to see the contribution of the regions. The input to the perceptron is the concatenation of the feature vectors for all feature types. The contribution of a particular region is calculated by looking at the difference in the likelihood if that particular region is included in the feature vector input to the perceptron and if it is not included. The difference between the outputs of the multi-layer perceptron with the two different inputs represents the contribution of the particular region to the perceptron output through a particular component type. For the implementation, the algorithm actually works at the pixel level. The algorithm calculates the contribution of a pixel to the multi-layer perceptron output through all the components of all the feature types. At the end, each pixel is finally labeled with the object of highest likelihood. Figure 8 shows some sample results from the localization phase. In the result images, the intensities represent the probability of occurrence of the indicated object; the lighter the intensity, the more probable is the object.

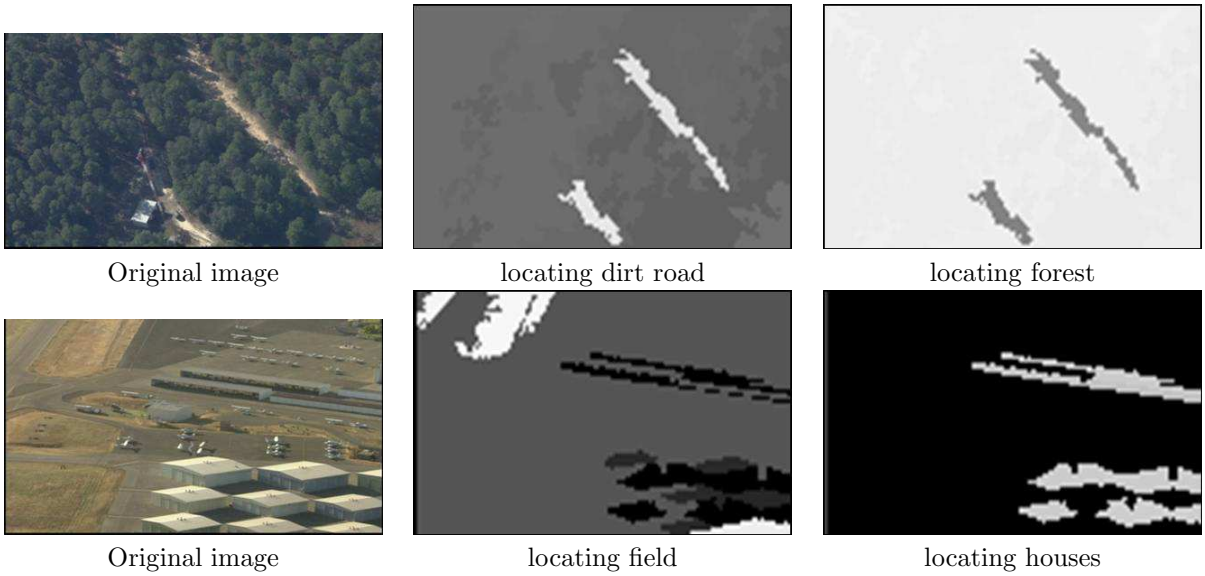


Figure 8. Sample results from the localization phase.

4. TRACKING MOVING OBJECTS IN AERIAL VIDEOS

Videos created by UAVs tend to have challenging conditions for video analysis systems. The problem with these videos is that they are generally not well focused and have substantial noise. In addition both external and internal camera parameters are constantly changing due to the UAV's ego-motion and the frequent and abrupt changes in pan, tilt, rotation, and zoom of the camera triggered by the remote camera operator. Because of these difficult conditions and the frequent illumination changes, which are inevitable in real outdoor videos, conventional optical flow detectors, which are designed with the assumption of constant illumination and high signal-to-noise ratio, do not work well.

Our object tracking system is based on the mechanism of multi-scale local feature correspondence. We convolve each candidate circular region centered at a pixel with a Gaussian filter at four different scales of radius from 2 to 6 pixels. For each scale, directional information is collected by computing piecewise RGB differences along each of the eight equally spaced radials coming from the center. The feature correspondence is decided based on a similarity function, which takes the multi-scale and directional information as arguments. The function is designed to be robust against minor illumination changes by using both the absolute RGB values and the normalized RGB values relative to the average RGB in the adjacent region. Points with low difference features are ignored, since they are likely to be in a homogeneous region.

Initially the points for checking correspondence are set to the corner points of each 8 x 8 grid square. The frame distance between two frames of video to be compared for finding correspondences of feature points is initially set to 3. In the case of detecting too little motion (i.e. too little overall displacement between corresponding features) for more than a certain number of consecutive frame pairs, the frame distance automatically increases gradually up to 10. This is necessary, since too little displacements tend to generate spurious directional vectors due to the granularity of digitization.

In the next stage detection of moving objects is done by comparing the displacement vectors for each feature points. If the ego-motion of the camera and UAV are translation only, the detection of the moving objects can be done simply by finding irregular flows. However, in the real situation where pan, tilt, rotation, and zooming can take place, we need to take special steps to distinguish moving objects from stationary ones. Our system does this by taking advantage of the fact that in any affine or perspective transformation, the amount of displacement is similar between stationary objects of close vicinity. The system checks each small region for an abnormal amount of displacement, which indicates a moving object.

Once a moving object region is located, its local properties (color, variance of color, dominant gradients, size, location) are recorded for tracking. Once a target for tracking is set, the candidate points for checking feature correspondence are increased in and around the region of the target. Tracking is performed using the recorded properties of the target region as well as the displacement vectors between feature points in and around the target region.

Overall, the multi-scale feature of our design makes it robust against noise and frequent changes in the camera parameters. The use of both relative and absolute colors makes it robust against minor illumination changes.

5. EVENT RECOGNITION

In order to recognize an event in a video, one first has to define what that event is. Consider first the case of a car passing another car on a road. This is a relatively easy concept for us as humans to recognize. However, to teach a computer to recognize the event, it needs to be broken down into the relationships between the objects at various points in time. The Video Event Representation Language (VERL) is a formal technique for defining events.[7] The objective of VERL is to define an event in terms of the objects involved, and their relationships with each other over time. A VERL definition for the passing event above might look as follows:

```
PROCESS (passing-same-direction (vehicle x, vehicle y),
         SEQUENCE(
              behind(vehicle x, vehicle y),
              behind(vehicle y, vehicle x)))
```

This is a relative simplistic definition. First, it ignores that the cars are on a road, and can therefore be used for two vehicles crossing a desert. It also labels the objects involved as vehicles, and thus would apply to trucks as well as cars. Therefore, this definition might have wider applicability to general situations than a more narrow definition restricted to cars on a road. The definition also makes no distinction between whether both cars are moving or one is standing still. Depending on the recognition objective, more or less constraints can be added to the definition. The definition indicates what parameters need to be measured from the video. First it is necessary to recognize the vehicles (cars). Next, the spatial relationship between the cars needs to be recognized (behind). The relationship 'behind' implies knowledge of the orientation of the vehicles, and is a primitive element that needs to be measured directly from the video sequence itself. In many UAV video sequences, it is not possible to determine the front from the back of a car due to the size of the vehicle in the image or blur or other factors. Another measurement of the spatial relationship is to use the vehicles' direction of motion as a measure of orientation. This assumes that the vehicle is nominally moving in the forward direction (high probability of being correct). Defining even simple events can be a difficult task depending on the specific set of constraints that are of interest for a particular event to be recognized.

Higher-level events can be defined in terms of lower-level events. Consider the case of a car crossing an intersection. This is only slightly more complicated than the passing event, but illustrates the use of simple events to construct more complex events.

```
PROCESS(cross-intersection(vehicle x, intersection y),
      AND(
          E1:SEQUENCE(
                 E2:enter(x, intersection y),
                 E3:exit(x, intersection y))
          EQUIV(
                 velocity(x, direction, at-time(E2)),
                 velocity(x, direction+/- tol, at-time(E3)))
          less-than(duration(E1, threshold)))
```

This event makes use of the sub-events 'enter' and 'exit'. These events are in turn defined in terms of the change in relationship between the vehicle and the intersection. In the sub-event 'enter', the vehicle is first

'outside' of the intersection, and then 'inside' the intersection. More complex events can be defined using simple events, which leads to a hierarchical structure for event recognition. The event recognition scheme is based on two components, primitive relationship recognition followed by event recognition. The primitive recognition component recognizes spatial and temporal relationships between objects. Examples of the primitive relationships are: 'moving', 'ahead', 'behind', 'left', 'right', 'inside-of', etc. These relationship primitives are determined directly from the video data and the parameters of the recognized objects. A very important component of the relationship recognition is a measure of the likelihood of the relationship, which is also output with the relationship description. The output of the relationship recognition component is an XML description of the relationship and the objects that are involved in the relationship. This is then used as input by the event recognition module.

The event recognition module uses Bayesian networks to account for uncertainty in the recognition of objects and relationships, and also the uncertainty in the recognition of lower level sub-events. A Bayesian network is defined for every event that is to be recognized based on the VERL definition of that event. Inputs to the Bayesian network are the relationships between the objects, the parameters of the objects, e.g. direction, and the likelihoods associated with the objects and the relationships. For higher-level event networks, the inputs also include the state of lower-level events. In most cases the desired output has two states; either the event occurred or it did not with some likelihood. In some cases, the event recognition output can have more than just the two states. As a result the output accommodates multi-state outputs, and the associated probabilities for each output state. Recognizing that the output state of an event can be used as the input to a higher-level event, the input to the Bayesian network is also capable of dealing with multi-state inputs. This also applies to the inputs from the relationship recognition component.

One of our goals has been to reduce the amount of processing at each level of the architecture. Relatively large amounts of processing are required for each frame to recognize the objects and their parameters. The relationship processing deals only with the objects that are recognized, but still has to be done for each frame. However, the event recognition only needs to be done when there is a change in the relationships. The state of each relationship is stored and an event is processed only when the relationship changes.

Sample video data can be used to train the network and develop the conditional probabilities when there are sufficient amounts of video data representing the events of interest. However, it is the unusual events that are often of most interest, and by definition, there will probably not be much video data representing unusual events. Bayesian networks are also sometimes called belief networks. This represents another approach to defining the networks and the conditional probabilities. An analyst can develop the networks based on the event definition and assign the conditional probabilities based on his experience and beliefs about the system. Therefore large amounts of video data are not necessary to build an initial capability.

Both the relationship processing and the event processing have been tested with some synthetic video data. As the object recognition capability is not yet complete, synthetic data was created to represent the video scenarios. The synthetic data is an XML list of the objects and parameters that are expected to be output by the object recognition and tracking processing modules for the particular scenarios. This list of the objects was input to the relationship processing module, which then appended the relationships that were recognized for each frame of the synthetic video. This output from the relationship processing module was then used as an input to the event processing module.

6. CONCLUSIONS

This paper presents an approach to object and event recognition using multiple stages of classification for aerial surveillance videos. The system uses a uniform feature representation called abstract regions. We described two different object recognition approach: EM-Variant and Generative/Discriminative approach. The EM-Variant approach learns the multivariant Gaussian models for the object classes based on the abstract regions. The Generative/Discriminative approach is divided into two phases: the generative phase, which normalizes the description length of images, and the discriminative phase, which learns what images, as represented by the fixed-length description, contain the target object. The location of the target objects in the images is then obtained by the localization stage. The location of the objects and the relationship between objects in each frame provide information for event recognition in the full video.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under grant IIS0097329 and by ARDA through a subcontract from the Boeing company. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the sponsors.

REFERENCES

1. Y. Gong, "Advancing content-based image retrieval by exploiting image color and region features," *Multimedia System* **7**(6), pp. 449–457, 1999.
2. Y. Li and L. G. Shapiro, "Consistent line clusters for building recognition in CBIR.," in *Proceedings of the International Conference on Pattern Recognition*, pp. 952–956, 2002.
3. J. Canny, "A computational approach to edge detection," *IEEE Transaction Pattern Analysis Machine Intelligence* **8**(6), pp. 679–698, 1986.
4. A. Etamadi, "Robust segmentation of edge data," in *IEEE Image Processing Conference*, 1992.
5. Y. Li and L. G. Shapiro, "Object class recognition using images of abstract regions.," in *Proceedings of the International Conference on Pattern Recognition*, pp. 40–43, 2004.
6. Y. Li, *Object and Concept Recognition for Content-Based Image Retrieval*. PhD thesis, Department of Computer Science and Engineering, University of Washington, 2005.
7. R. Nevatia, J. Hobbs, and R. Bolles, "An ontology for video event representation," in *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, **7**, 2004.