A Trajectory Based Video Indexing System For Street Surveillance

Emile Sahouria and Avideh Zakhor

March 4, 1999

Abstract

We have developed a system to analyze and index surveillance videos of a street scene based on the motions of objects in the scene, such as cars and people. A segmentation and tracking system extracts trajectories from compressed video, which are represented in a multiresolution manner and stored in a database. Once the motion segmentation has been accomplished, the three dimensional volume of data representing the object moving through time can be used to robustly estimate other features of objects, including size and color. Hand-drawn queries can be submitted to the system for imprecise searches. The system was tested on 20 minutes of real video footage with numerous moving objects; the success rates for tracking and recall in most cases exceeded 70 percent. Larger, more rigidly moving objects such as cars were more reliably dealt with than smaller objects such as people.

1 Introduction

A valuable tool in the management of visual records is the ability to automatically "describe" and index the content of video sequences in a meaningful manner. Such a facility would allow recovery of desired video segments or objects from a very large database of image sequences. The efficient use of stock film archives and identification of specific activities in surveillance videos are as potential applications.

While vast energies have been expended in the development of video databases, relatively little of it has focused on using motion to describe object activity. Further, of the systems that use motion, nearly all focus on segmentation and tracking technologies, rather than on indexing schemes. Exceptional examples include the works of Dmitrova and Golshani[4] and Ioka and Kurokawa [6], who used macroblock tracing and clustering to derive trajectories, and some form of dynamic programming to determine similarity between trajectories. Ardizzone [2] used frame-wide optical flow to specify more global motion characteristics. Several systems have also used more statistical descriptions of motion [10] [7]. More recently, commercial systems have been developed to track and catalog objects in video [1].

In this paper we describe a motion based video indexing system for street surveillance. A segmentation and tracking program analyzes compressed video of a scene and extracts the trajectories of moving objects including cars, people and bicycles, represented as two dimensional curves parameterized by time. The coarse-scale components of these trajectories are stored as keys in an index. A user who wishes to find an object moving in a particular way draws a trajectory, which is then matched against those in the index. The use of the coarse scale representation, the resulting ability to index trajectories, and the novelty of the street surveillance as a whole, are the key elements missing from most systems developed so far. These are the primary contributions of this work.

The current system will be presented in the next section. The discussion revolves around the central issues of feature extraction and feature representation for searching. The testing strategy is then described in section 3, and the efficacy of the various system parts evaluated in section 4 on a database of several hundred objects.

2 The System

The apparatus for extracting trajectories and indexing the video on the basis of these trajectories is shown schematically in Figure 1. It comprises components to achieve trajectory extraction, index building, and easy user interaction. Motion vectors from MPEG-1 compressed video form the sole input to the system. In the trajectory extraction and key generation stages, the trajectories of objects in a fixed scene are extracted and represented by their wavelet transforms. The term "representation" will henceforth mean all or part of the set of wavelet coefficients, and any other data useful in characterizing the trajectory. The database comprises an index structure as well as the store of compressed video. The index consists of pairs of representations and pointers to the video clips that they represent; the representation and pointer are collectively called a "key" in the sequel. The search function compares the representation generated by the user query to those in the index.

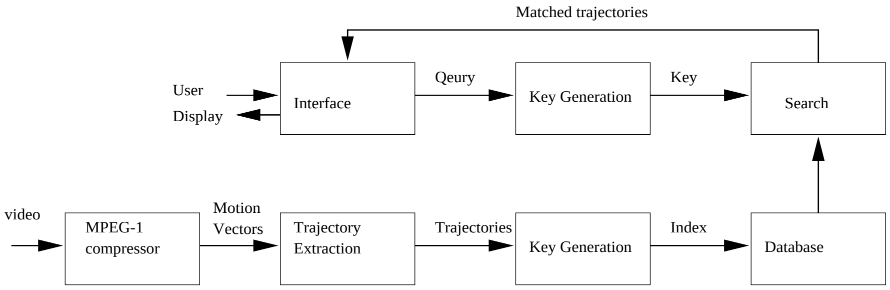

Figure 1: *The database system, showing the interaction of the three components.*

2.1 Trajectory Extraction

Object trajectories, approximating the movement of the center of mass of objects, are produced by the trajectory extraction. Specifically, a "trajectory" is a set of tuples $\{(x_k, y_k) : k = 1 \ldots N\}$, where (x_k, y_k) is the center of mass of the object in the kth frame of a length-N trajectory. This function is achieved by segmenting motion vector fields corresponding to pairs of frames, and then determining the correspondences of objects between frames via tracking. A similar scheme was recently used in [3]; the two schemes differ mainly in the assumptions made of object motion, and in the heuristics used to improve correspondences.

The motion segmentation algorithm is reminiscent of the morphological motion tracking scheme proposed by Gu and Kunt[5], albeit in simpler form. Meng and Chang[9]used a similar system as well. Median filtering is first performed to remove suspected noise from the vector field. Although multidimensional morphological filtering could be used for the two dimensional vector field, one-dimensional filtering is used instead, based on the observation that the magnitude of the MPEG vectors, rather than the angle, is more susceptible to impulsive noise. In other words, since a one-dimensional function of the vector is sufficient to characterize noise, filtering is done only on this function (the magnitude) of the motion vectors. A morphological opening, again based on the magnitude signal, follows. The opening removes small features and smooths the vector field, making it more amenable to clustering. By using only the magnitudes, one avoids separating objects that undergo non-translational motion into pieces. This requires the assumption that the motion of all blocks comprising an object will have about the same magnitude, if not very close angles. Finally, I-type blocks, which have no motion information, are considered still. A sufficiently small number occur in surveillance scenes that our clustering heuristics can still figure out reasonable object extents and our correspondence heuristics can still link well across frames.

This simplified motion field is then clustered using the distance function induced by the l_1 norm on the space of motion vectors:

$$d(\mathbf{u}, \mathbf{v}) = |u_1 - v_1| + |u_2 - v_2|.$$

A grass-fire type clustering is used: an unclassified vector is assigned a new region number, and all adjacent vectors that lie within a threshold of it are assigned to the same region. The process is repeated recursively on these newly classified vectors, and iteratively on remaining unclassified vectors until all vectors in the field are assigned to a region. Regions with size smaller than the size of the kernel used in the morphological operations cannot represent real objects, since they would have been obliterated by the morphological opening. These artifacts are thus merged with other regions or deleted. Lastly, motion models are calculated for each region; they are simply the averages of the median filtered vectors in each region. Figure 2 illustrates the steps in segmenting a single motion vector frame.

The tracking process matches objects between two *motion field* frames based on the

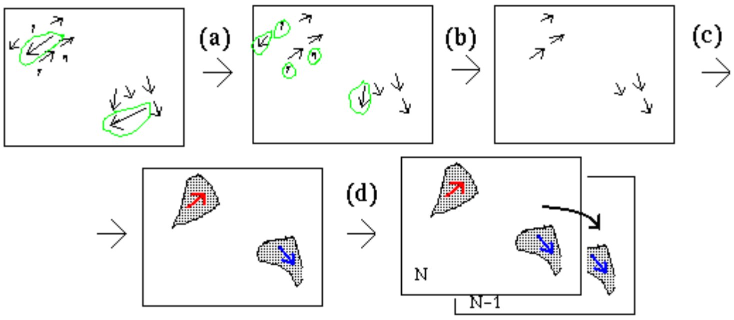

Figure 2: *The segmentation process. Steps shown: (a) impulsive noise removal, (b) median filtering and (c) clustering. The last box represents correspondences; it is further explained in the next figure.*

amount of overlap of their segmentation maps. For each object in the previous frame, each object in the current frame is projected backwards according to its motion model. The intersection sizes of these back-projected objects are measured, and the one with the largest intersection size is the match, as in Figure 3. The center of the the object in the current frame is finally recorded. The resulting of region centers linked across frames is the desired trajectory. A trajectory, then, is just a list of two-dimensional coordinates. Several simple heuristics actually moderate this correspondence process, in order to ensure a more accurate match:

- Object correspondence is only allowed if the directions implied by the motion models of the objects are consistent. Directions are "consistent" if the angle of the vectors differ by less than a given threshold. This allows adjacent or even occluded objects to be distinguished in many cases where the simple maximum overlap rule would assign incorrect correspondences.
- If two objects of the previous frame are in contention for the same current object, the one with the longer trajectory wins. This reduces the breakup of trajectories from inaccurate frame segmentation.

Once an object leaves the scene, or stops moving for longer than a specified time, its trajectory ends. "New" objects–those that do not correspond to existing trajectories–are assigned new trajectories.

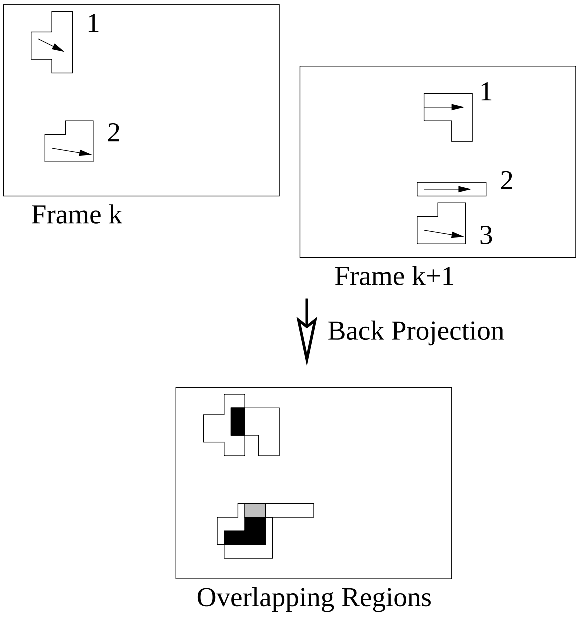

Figure 3: *A tracking example. The three objects in frame k+1, represented by their segmentation masks and motion models, are projected onto the previous frame according to their motion models. Object 1 of frame k is seen to overlap only with object 1 of frame $k+1$, establishing the correspondence between these objects. Objects 2 and 3 of frame $k+1$ overlap object 2 of frame k, but object 3 has the greater overlap. Thus object 3 is linked with object 2 of frame k, and object 2 of frame $k+1$ marks a new trajectory.*

2.2 Indexing and Database Issues

For each trajectory, the set of tuples is reordered into two vectors, representing the projections of motion versus time onto the x- and y-projections: $\mathbf{X} = [x_1 x_2 ... x_N]^T$ and $\mathbf{Y} = [y_1 y_2 ... y_N]^T$ where N is the duration of the trajectory in frames. Several normalizations are now used to make the trajectory independent of spatial and temporal scale. This ensures that only the shape of the trajectory curve, and not its spatial or temporal extent, are analyzed in the what follows. Specifically, the magnitude of each vector is normalized by dividing by the absolute value of the largest component, and the dimension is normalized by a multirate sampling change, i.e., upsampling or downsampling. Of course, the spatial and temporal scale information is useful, so it is retained for future use. The velocity information is also extracted separately by measuring and storing an estimate for the average speed of an object, as described below.

A multiscale representation of a trajectory is desired, in order to emphasize the larger scale components of the trajectory and deemphasize the smaller variations. The Haar wavelet transform, chosen for simplicity, is computed separately for each of these two vectors. In order to keep the information in the large scales of the trajectory, only the first eight coefficients of each of the x and y projection transforms are stored as search keys in the index structure. In a query by example application, the user would enter a trajectory that resembles the motion of an object to be found in the database. The same normalizations and transforms are applied to this hand drawn trajectory, and the metric induced by the $\mathbf{R}^{16}$ Euclidean norm is used to compute distances from the database objects.

Velocity information is computed separately. In order to extract velocity information, several steps are first taken to smooth out the trajectory. This is necessary to overcome the imprecision and roughness of the tracking algorithm, which are due in part to the poor spatial resolution of the segmentation. The trajectory is low-pass filtered and then fit by a simple piece-wise linear function. Finally, the total spatial displacements in the horizontal and vertical directions are divided by the duration of the trajectory in units of number of frames. The result is reported as the velocity.

The representation (key) of a trajectory is now complete; it consists of the wavelet description of its shape and the velocity. The index structure is just a list of keys. Trajectory queries can be drawn on the graphical user interface described in the next section. The drawn query is simply a directed line representing the motion of an object. Simple linear scanning is used to search the list; i.e., the query's trajectory shape is checked against the trajectory shape part of each key[1]. A list of the "nearest" items in the database is returned, ranked in order of closeness to the query according to the trajectory shape distance function described earlier. The returned set can optionally be sorted further by velocity. A pointer to the MPEG-compressed video file and the location of the desired clip in that file are stored in the key, allowing the video corresponding to the matched object to be played back.

[1]Note that the user queries are merely represented by their trajectory shapes and not their velocities

2.3 The User Interface

A graphical user interface that is particularly designed for the surveillance application accepts hand-drawn queries and returns matches pictorially and in order of increasing distance from the query. Figure 4 shows the results of a typical query. The final set of returned records can also be sorted in order of velocity.

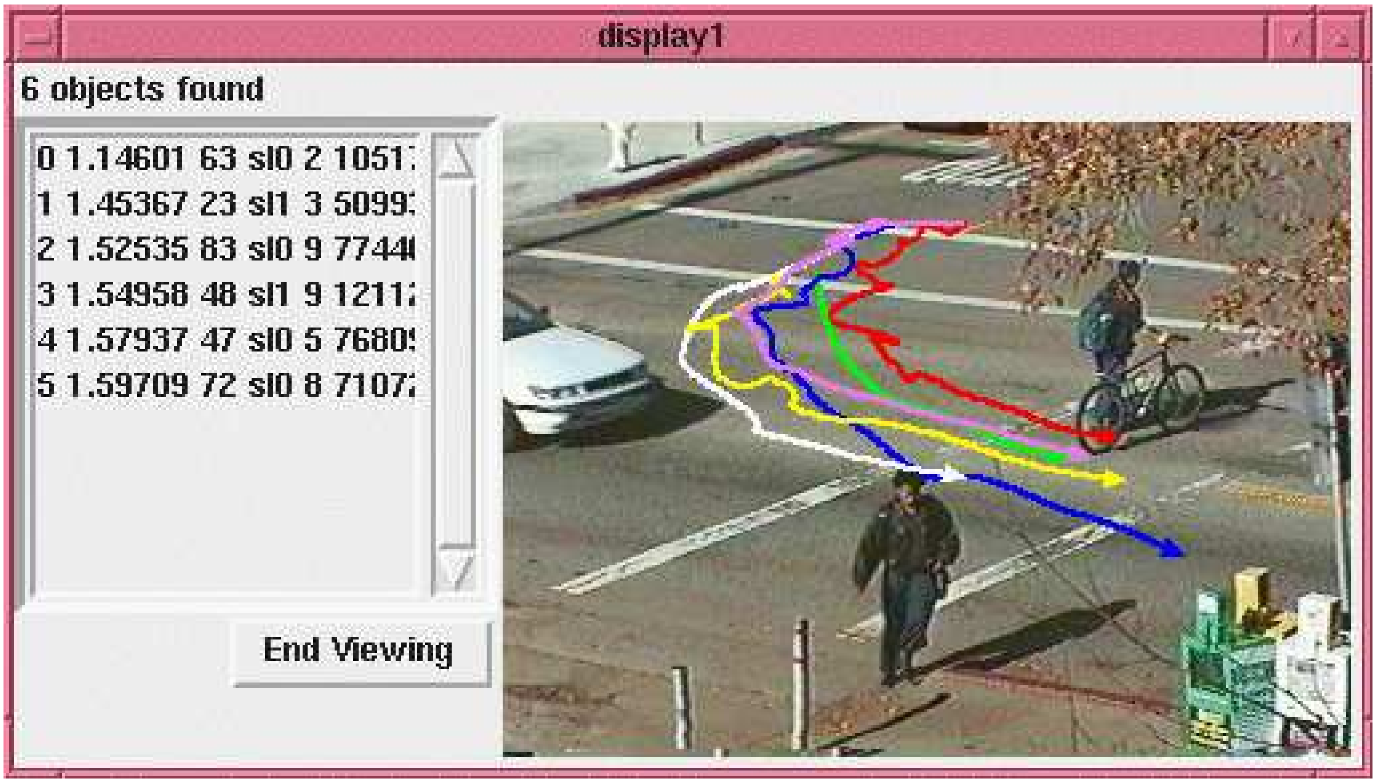

Figure 4: *Example query results. The query was for left-hand turns. The six curved lines are the trajectories of cars which moved similar to the way specified by the query. The coloration of the lines is artificial; it makes viewing easier when lines overlap. .*

3 Experimental Evaluation

The efficacies of several parts and algorithms of the system were tested separately. The video footage used in the testing was obtained from a camera filming a busy intersection, through which vehicles, people and bicycles passed. The camera resided in a nearby build-

Object Type	Manual Detections	Algorithm Detections	Rate
People	209	163	78%
Bicycles	6	6	100
Motor Vehicles	194	169	87

Table 1: *Segmentation results.*

ing, approximately 40 feet above and 70 feet distant from the intersection. This distance made the gross motion in the scene approximately affine. The following sections describe the evaluation of the trajectory extraction and trajectory search algorithms.

3.1 Trajectory extraction

A 20 minute length of video was analyzed both manually and by the proposed segmentation and tracking algorithm for trajectories. The number of correctly extracted trajectories was determined by comparing a graph of extracted trajectories to those observed by a human viewing the video. 409 objects were detected manually, of which the algorithm found 338. The tracking program also produced 141 false detections, representing noise and short fragments of actual trajectories. The fragments of trajectories result when tracking temporarily fails, breaking a trajectory into pieces, or when different parts of the same object are temporarily tracked, resulting in duplicates of short pieces of the full length trajectory. Neither type of fragment is useful for matching trajectories; hence, they are classified as false positives.

The results in Table 1 show the rates of missed detection for three categories of objects. As seen, the algorithm is more successful in tracking vehicles than people. This is due to problems in segmentation resulting from the more complex motions of people in the scene, and to their smaller size. The imprecision and noisiness of the MPEG motion estimates is simply inadequate to describe these motions correctly, or to recognize them at all. Tracking fails as well when motions are too similar, or occlusions too great. The segmentation and tracking algorithms work easily in real time.

3.2 Velocity Estimation

The velocity estimator was tested as well. Recall that a user can request that queries be sorted according to velocity. Various trajectory queries were posed to the database with this ordering requested. The queries were made intentionally vague so that several types of objects – people, cars, etc. – would be included in the returned records. The average velocity of the objects was estimated by the human operator by measuring the distance traveled by an object, and the duration of its travel. Theses manual estimates are compared to the algorithm estimates in several ways in Table 2. The units of the standard deviation and mean are pixels per frame; since the sequence was filmed at 10 frames per second, the units are may also be expressed as $10 * pixels/s$. The mean and standard errors compare individual measurements, while the order distance compares whole rankings. Before describing the order distance, we note that ratio of standard error to mean velocity is much lower for larger objects (cars) than for smaller ones (people), implying that the estimator works better for these larger objects. This is mainly due to the greater difficulty in tracking smaller objects described in the previous section.

The "order" distance measures the amount of work needed to correctly order the algorithm ranking. Specifically, suppose $[v_1 \ldots v_N]$ is the ranking of velocities produced by the algorithm, and $[v_{k_1} \ldots v_{k_N}]$ is a permutation of the N velocities representing the correct order. Let $f : \{1, \ldots, N\} \mapsto \{1, \ldots, N\}$ be this permutation ($k_i = f(i)$). Then let

$$D'(\mathbf{v_{alg}}, \mathbf{v_{manual}}) = \sum_{i=1}^{N} |i - f(i)|. \tag{1}$$

This measures the sum of the distances each number in the velocities ranking must be moved to put the list in the correct order. D' is zero when the order is correct, and is maximum when the list is reversed: $D'_{max} = 2 * \lfloor N/2 \rfloor * (\lfloor N/2 \rfloor + 1)$. D'_{max} is used to normalize the distance measure shown in equation 1 to arrive at the order distance given by

$$D_{order}(\mathbf{v_{alg}}, \mathbf{v_{manual}}) = \frac{\sum_{i=1}^{N} |i - f(i)|}{D'_{max}}.$$

D_{order} has a range of $[0, 1]$. Table 2 shows that the queries had an order distance between 0.15 and 0.25; we conclude that the velocity estimator chooses a ranking that is fairly close to the correct order in all the test cases. This is demonstrated graphically in Figure 5 for the

Query description	Object type	Mean velocity	Mean error	Std. error	Order distance
Objects moving E $\rightarrow$ W	people	6.8	-0.19	3.19	0.15
	cars	15.76	3.13	1.8	
Objects moving W $\rightarrow$ E	people	5.14	-0.27	1.66	0.15
	cars	14.11	3.11	2.08	
Objects "turning left"	cars	2.77	-0.75	0.76	0.25

Table 2: *Velocity estimation results.*

first query in Table 2, i.e. people moving east to west. Recall that the original purpose of the velocity feature is to provide an ordering of returned records, not to provide very precise estimates of velocity. In this light, the measure of errors in individual velocity estimates is less important than the measure of how close the final ordering is to the correct ordering, and according to this measure the velocity estimation algorithm is acceptable.

Various inaccuracies in the tracking algorithm explain the errors in the velocity estimation. While capturing the gross trends in the motion of an object, the local roughness of the trajectory – a result of the coarseness of the frame segmentation – can artificially inflate the measure of the distance traveled by an object. Worse estimates occur when different parts of an object are tracked during the course of a trajectory; for instance, the low-texture side of a large moving object, like a bus, can be easily mistaken for a much more slowly moving, or stationary object. Similarly, different people in a moving group may be tracked as part of the same trajectory, causing the position to "jump" artificially between frames.

3.3 Trajectory Search

The recall precision of the trajectory search was tested next. The precision for a given recall rate r is defined as

$$precision = \frac{r * N_{objects}}{r * N_{objects} + N_{false\_detections}}$$

where $N_{objects}$ is the number of objects in the database determined by a human to match the query, and $N_{false\_detections}$, the number of false detections, is the number of objects not matching the query activity that are found before a total of $r * N_{objects}$ matching objects are found. Table 3 shows the precision rates for the 90% recall rate for various hand-drawn

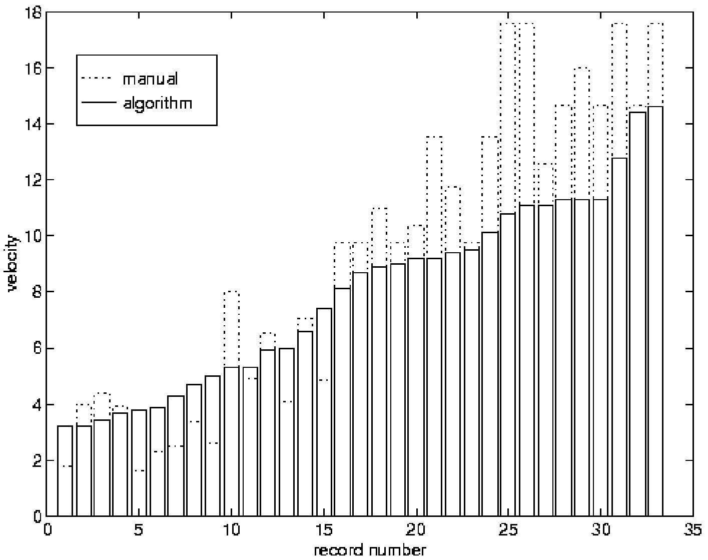

Figure 5: *Comparison of manual and algorithm velocity orderings for the first test query.*

queries. The numbers show that the indexing and retrieval strategy is significantly more successful for vehicles than for pedestrians. Once again, the nature of the objects explains these results. Vehicles are rigid objects moving along relatively simple paths; these smooth, simple trajectories are more easily matched to queries. People not only follow less regular paths, but the complex motions associated with walking introduce small-scale fluctuations in the trajectories. Using only the coarse scale coefficients of the wavelet transform helps solve the latter problem, but the greater variety of trajectories presented by pedestrians necessarily makes matching less successful.

4 Conclusion

The potential for using motion to both index video and aid in the estimation of other parameters has been demonstrated. While the segmentation method used does not provide extremely accurate results, the imprecise matching afforded by the wavelet representation allows one to overcome these inaccuracies as well to provide a matching strategy which agrees well with the human perception of similarity. Further, the problems of feature extraction and feature representation are separable: a better segmentation and tracking

Query	Tracking Detections	Precision at 90% Recall
People, cross near $\rightarrow$ far	35	86%
People, cross far $\rightarrow$ near	30	75
People, near side, E $\rightarrow$ W	32	69
People, near side, W $\rightarrow$ E	29	78
People, far side, W $\rightarrow$ E	11	65
Car Left Turns	6	100
Cars E $\rightarrow$ W	68	93
Cars W $\rightarrow$ E	72	80

Table 3: *Precision for 90% recall rate.*

algorithm may always be implemented in order to raise the whole system performance without requiring changes to any other part of the database. A straightforward extension in this respect is the consideration of global camera motion. Global motion could be removed using any number of current techniques, and then the algorithms in the present system could be applied. Segmentation and tracking themselves are continually developing fields [11].

Several more fundamental issues have been left unanswered by this effort. One is the important question of how to integrate features, rather than using the simple successive filtering method outlined in section 2. Similar issues are receiving attention in the still image database community. Another unaddressed issue is that of indexing. It may be possible to exploit the multiresolution nature of the wavelet basis functions to provide a more efficient index structure than the current list structure. Another potential solution is to classify the keys of the database into groups, and to first determine the closest such group to a query. This idea of a feature "thesaurus" has also found application in image database research [8]. Efficient indexing will of course become increasingly important as the size of the database grows.

One may also ponder the applicability of this technique to more general classes of video. Clearly, most video does not conform to the model assumed in surveillance video; that is, long trajectories corresponding to translating objects. As is often the case, trajectory-based description will almost certainly need to be used in some intelligent combination with other techniques to achieve a suitably general way of using motion to describe video.

References

[1] http://www.mediastra.com.

[2] E. Ardizzone and M. La Cascia. Video indexing using optical flow field. In *ICIP*, volume 3, pages 831–34, 1996.

[3] F. Bartolini, V. Capellini, and C. Giani. Motion estimation and tracking for urban traffic monitoring. In *ICIP*, volume 3, pages 787–90, 1996.

[4] N. Dimitrova and F. Golshani. Motion recovery for video content classification. *ACM Transactions on Information Systems*, 13(4):408–439, 11 1995.

[5] C. Gu and M. Kunt. Morphological motion segementation and tracking. Technical report, MPEG-4, 1994.

[6] M. Ioka and M. Kurokawa. A method for retrieving sequences of image son the basis of motion analysis. In *SPIE Image Storage and Retrieval Systems*, volume 1662, pages 35–46, 1994.

[7] G. Iyengar and A. B. Lippman. Videobook: An experiment in characterization of video. In *ICIP*, volume 3, pages 855–58, 1996.

[8] B. S. Manjunath and W. Y. Ma. Browsing large satellite and aerial photographs. In *ICIP*, volume 3, 1996.

[9] J. Meng and S.-F. Chang. Tools for compressed-domain video indexing and editing. In *SPIE Proceedings*, volume 2670, pages 180–191. SPIE, 1996.

[10] G. Rigoll, A. Kosmala, and M. Schuster. A new approach to video sequence recognition based on statistical methods. In *ICIP*, volume 3, pages 839–42, 1996.

[11] S. Sclaroff and J. Isidoro. Active blobs. In *Sixth International Conference on Computer Vision*, pages 1146–53, 1998.

List of Figures

List of Tables