Video Retrieval of Near–Duplicates using k–Nearest Neighbor Retrieval of Spatio–Temporal Descriptors

Daniel DeMenthon and David Doermann
Language and Media Processing (LAMP)
University of Maryland Institute for Advanced Computer Studies (UMIACS)
College Park, MD 20742, USA
daniel@cfar.umd.edu

January 10, 2004

Abstract.

This paper describes a novel methodology for implementing video search functions such as retrieval of near-duplicate videos and recognition of actions in surveillance video. Videos are divided into half-second clips whose stacked frames produce 3D space-time volumes of pixels. Pixel regions with consistent color and motion properties are extracted from these 3D volumes by a threshold-free hierarchical space-time segmentation technique. Each region is then described by a high-dimensional point whose components represent the position, orientation and, when possible, color of the region. In the indexing phase for a video database, these points are assigned labels that specify their video clip of origin. All the labeled points for all the clips are stored into a single binary tree for efficient k-nearest neighbor retrieval. The retrieval phase uses video segments as queries. Half-second clips of these queries are again segmented by space-time segmentation to produce sets of points, and for each point the labels of its nearest neighbors are retrieved. The labels that receive the largest numbers of votes correspond to the database clips that are the most similar to the query video segment. We illustrate this approach for video indexing and retrieval and for action recognition. First, we describe retrieval experiments for dynamic logos, and for video queries that differ from the indexed broadcasts by the addition of large overlays. Then we describe experiments in which office actions (such as pulling and closing drawers, taking and storing items, picking up and putting down a phone) are recognized. Color information is ignored to insure independence of action recognition to people's appearance. One of the distinct advantages of using this approach for action recognition is that there is no need for detection or recognition of body parts.

Keywords: Content-based indexing and retrieval, video retrieval of near-duplicates, action recognition, space-time segmentation, spatio-temporal descriptors, object motion.

1. Introduction

Human perception has a unique ability to judge similarity between video sequences at a variety of levels. We can recognize objects and people, observe their spatial relations and their temporal interactions. We can focus on detailed similarities, even if the common elements are only a small part of the scene. By contrast, current video indexing

and retrieval methods typically do not analyze video structure at the object level. Existing approaches primarily use text data from closed captions or speech transcription, image retrieval using keyframes, and specialized detectors for faces or vehicles. Color, shape and texture features are commonly used for indexing individual frames, but temporal features have often been ignored. A primary reason for this has been an inability to represent the temporal information compactly and to use it effectively in measuring similarity.

Recent efforts on developing video indexing and retrieval systems using visual content have mainly leveraged progress on analysis of individual frames, as well as progress on indexing and retrieval of static images. In the simplest approach, videos are divided into shots through scene transition detection, and representative keyframes are selected for each shot and indexed into an image database. These keyframes are used for retrieval. Examples include methods developed by IBM (Flickner et al., 1995) and Virage (Hampapur et al., 1997). There is no attempt to extract temporal relations between keyframes. In general, information about the temporal evolution of the video has been lost.

To obtain descriptions of videos that are characteristic of the stream of information as opposed to snapshots of information, features can be obtained for each frame, and video sequences can be represented as strings of features. Video sequences can then be retrieved using string matching techniques (Lienhart et al., 1998). A related approach consists of mapping features for each frame to a point in a feature space so that the set of feature points generates a curve in feature space (DeMenthon et al., 1998). Retrieval then involves comparing curves, possibly at hierarchical levels of detail, between stored video sequences and query video sequences (Dimitrova and Abdel-Mottaleb, 1997). However, the features extracted in each frame typically reflect global pixel statistics whose profile can be completely altered by the additions of banners and logos to the original videos. It is our experience that such systems have difficulty retrieving near-duplicates in these cases. By contrast, the voting mechanism of the retrieval system described in this paper would consider the banner descriptors as outliers that are factored out (Section 6).

There has been isolated interest in the use of motion information for video indexing and retrieval. In some of the earliest work, Dimitrova and Golshani extract coarse descriptions of object motion by detecting and grouping trajectories of macroblocks, obtained from the motion vectors of MPEG encoding (Dimitrova and Golshani, 1995). Bruno and Pellerin characterize the motion patterns in video by components of the Fourier transform (Bruno and Pellerin, 2000) or wavelet coef-

ficients (Bruno and Pellerin, 2002) of the optical flow field of frame pairs. Fablet and Bouthemy use as indexing information a global description of the motion obtained from the temporal cooccurrences of local motion (Fablet et al., 2002). See also (Sahouria and Zakhor, 1997; Syeda-Mahmood et al., 2001).

There are compelling advantages to analyzing color and motion of whole video sequences as opposed to processing individual frames or pairs of frames. From a larger number of frames, much more solid evidence about the salient motion of color regions can be accumulated than from the optical flow obtained from local differential analysis. In addition, spatio-temporal segmentation is a more direct and robust way of tracking moving regions than the classical tracking approaches that extend inferences from image pairs to multiple frames. Early progress in this domain, pioneered by Allmen (Allmen and Dyer, 1993) in the early 90s, was hindered by the heavy memory and processing requirements. However, these obstacles are subsiding (DeMenthon, 2002). Consequently, a few researchers have started to explore ways of applying such an approach to video indexing (Del Bimbo et al., 2000; Sun et al., 2000). For example, Del Bimbo et al. (Del Bimbo et al., 2000) summarize the spatio-temporal volume by using a 3D Haar wavelet transform. Video sequences are then compared by computing a distance between wavelet descriptors.

We propose a novel methodology for the compact representation of a video's spatio-temporal structure and describe a system that uses this representation to let users with access to large collections of videos take a short query video sequence and identify and rank all occurrences of "similar" sequences in the collection (Sections 3.2 and 5). Collections of videos are analyzed to produce spatio-temporal descriptors that summarize the location, color and dynamics of independently moving regions with only a small number of bytes. The similarities of sequences are defined using these descriptors.

In the second part of of this paper, we show that this approach can also be applied to action recognition in video sequences. The general action recognition problem is clearly hard; therefore researchers have addressed subsets of the problem produced by making simplifications, and the approaches can be classified by the simplifying assumptions that are made. One type of approach assumes that the segmentation and tracking problems have been somehow solved, and the focus is on the issues related to temporal segmentation (Ivanov and Bobick, 2000; Pinhanez and Bobick, 1998). Another type looks at scenes in which people are small blobs where their articulated structure is unimportant when seen from cameras mounted high on buildings (Oliver et al., 2000). Yet another type of approach looks at articulated structures

of people, but simplifies the detection of silhouettes by placing people in front of featureless background (Ricquebourg and Bouthemy, 2000). With the approach proposed here, we can analyze the video streams produced by fixed surveillance cameras. We can retrieve sequences in which people interact with objects that remain relatively fixed in the scene and are projected to relatively well defined regions of the camera image. Aside from this simplifying assumption, we do not impose any restrictions on the clutter level of the background, and we solve the action recognition problem all the way from low level analysis of the image stream to final action selection (Section 7). The main applications for our approach are in surveillance and security. People interact with shredding machines, cash registers, metal detector gates, safes, ID card swiping devices, etc. The proposed system can be used to recognize the components of these interactions. Extensions of the system could be used to detect unexpected patterns in the use of this equipment.

2. Motivation

A company has paid millions to have 15-second advertising spots broadcast day and night at several local stations. They want to make sure that the total number of broadcasts actually corresponds to the contract specifications, and that the spots were not mutilated. Manual verification would require that several people take turns watching all the channels around the clock, a tedious job at the mercy of 15 seconds of inattention. This company needs a system that can reliably recognize an advertising spot, even if it has been cut short or was framed with a dynamic banner warning about severe weather conditions.

A user has recorded a football game and has time to watch only the most exciting moments. These moments generally are repeated as slow-motion replays. To alert the viewer that there is a jump back in time, replays are bracketed between two dynamic logos (Fig. 9, right). This user needs to be able to select one of these dynamic logos and request all the places where a similar sequence occurs.

A surveillance camera is pointed toward a door controlled by a magnetic card reader accessing an airport runway. To verify a suspicion of unauthorized access by an outsider, several days of continuous recording must be reviewed to identify instances showing that the user is hesitant about the proper use of the door system. The retrieval task can been trained by manual selection of clips showing people accustomed to the system and accessing the door. The system must retrieve all instances of video sequences with spatio-temporal patterns that do not quite match the expected patterns.

The experiments of Sections 6 and 7 show that the needs examplified by these scenarios can be addressed by a system applying the principles described in Section 3.

3. Overview of Approach

3.1. Space-Time Descriptors

The features we use to characterize video sequences are *not* image-based. They are space-time descriptors that summarize half-second clips. These may be best understood with the example of a sequence of frames where colored blocks are moving in a linear fashion (Fig. 1). In the top of Fig. 1, a sequence of frames ordered from left to right and top to bottom shows a yellow block moving downwards along the left of the scene and a red block moving to the left along the bottom of the scene, while a green block and a blue block remain still over a stationary grey background. Consider the 3D volume of pixels created by stacking the frames of this sequence, with each frame horizontal and the first frame on top. In this *video stack*, the time axis is vertical and pointing down, while the row and column dimensions are horizontal. In this stack the blocks generate colored cylinders with rectangular sections. The spines of these cylinders are vertical for still blocks, and are slanted for moving blocks. We call these spines *video strands*. In the bottom of Fig. 1, these strands are shown in the position that they occupy in the video stack. Each strand is shown with a thickness proportional to the average pixel area of the strand (note that this area is not used in our retrieval algorithm because its computation is found to be less stable than the color, position and orientation components).

We have developed tools for the space-time segmentation of such regions in video stacks (see below). To summarize, at each pixel a feature vector with components describing its position, optical flow motion and color is computed and represents a point in 7D space. The points corresponding to the same cylinder form *clusters* that are extracted by a hierarchical mean shift technique. The most stable clusters across the multiple scales of clustering produced by this hierarchical technique are selected, and this removes the need for any predetermined scale, cluster count or threshold in this analysis. The seven dimensions of the found cluster centers describe the positions and orientations of the spines of the segmented regions in the video stack, as well as the average colors of these regions. The bottom of Fig. 1 shows the set of spines, i.e. video strands, corresponding to the cluster centers for the moving block sequence. The strands of this figure were obtained automatically

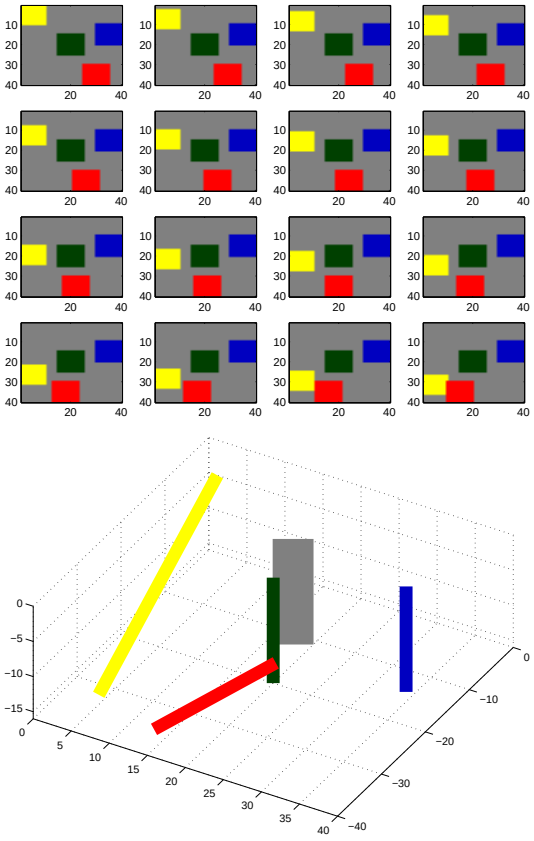

Figure 1. Top: Sequence of frames with colored square blocks; two blocks are fixed and two are moving in front of a grey background. Bottom: Strands extracted by space-time segmentation; the time axis is vertical with time increasing downward; the larger vertical strand is produced by the background.

by this space-time clustering technique from the sequence of synthetic images of moving blocks. The slanted strand in the front corresponds to the red block. The large vertical strand corresponds to the background.

In real life, things don't always move so linearly, they accelerate and make turns, and the camera that tracks them also introduces scene motion. But because of the inertia of objects and cameras, space-time regions produced over short durations by color patches in video stacks typically have fairly straight video strands.

This set of space-time descriptors is a very concise yet powerful description of a video clip. There is quite a lot of room in a high-dimensional space, thus the chances that several of the cluster centers from another sequence would *simultaneously* fall in the neighborhood

of the cluster centers of this sequence are remote. This is the reason for the observed resilience of the proposed video recognition approach, using space-time descriptors and nearest neighbors, to the addition of clutter video sequences in the database.

3.2. Training and Recognition

The video retrieval problem we are trying to solve can be stated as follows. We have a library of video sequences and a query video clip. We would like to retrieve the segments of video library sequences most similar to the query clip. Therefore we chop the library sequences into small clips, and would like to retrieve and rank the library clips most similar to the query clip.

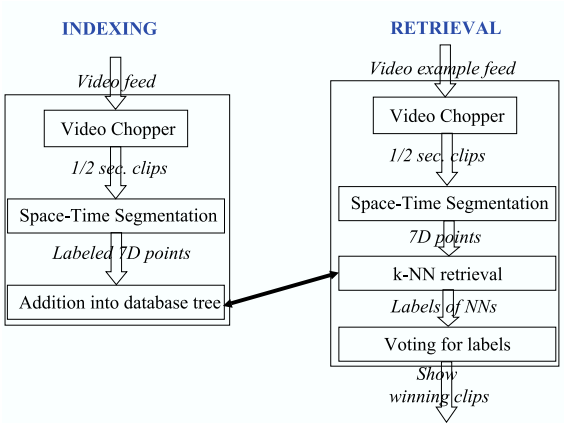

Figure 2. Anatomy of system for video indexing and retrieval.

The anatomy of our system is illustrated in Fig. 2. The video indexing module (left) and the retrieval module (right) use the same low level processing technique, which consists of chopping video into half-second clips and extracting video strands by the threshold-free space-time segmentation technique sketched above and described in more detail in the next section. In both modules, only the 7D points describing the colors, positions and orientations of the video strands (i.e., the spines of the spatio-temporal color cylinders of the video) are considered. In the indexing module, the points produced by video clips are stored in a point database along with the labels identifying the address or class of the clip that produced them, and organized into a tree structure for fast k-nearest neighbor retrieval (k-NN for short). In the retrieval module, the 7D points produced by space-time segmentation of the query video clip are used as k-NN queries to the

point database, and the labels of the retrieved points are tallied. The video clip whose largest number of labels were retrieved is considered to be the recognized clip. This approach is illustrated in Fig. 3. In this figure, the labeled points of the database are represented by the points inside the rectangle on the left, and their labels are illustrated by different colors; the query video clip is the rectangle on the left. The positions of the points of this query clip in the database are also shown in light gray in the database rectangle. The cumulated k-NN query with $k = 1$ for these points produces the winning count of four votes for Class 1, and this is the classification selected for this video clip. This approach is similar to the retrieval mechanism of gray level images from interest points described by Schmid and Mohr (Schmid and Mohr, 1997). It is also related to the cubist approach to object recognition advocated by Nelson (Nelson and Selinger, 1998).

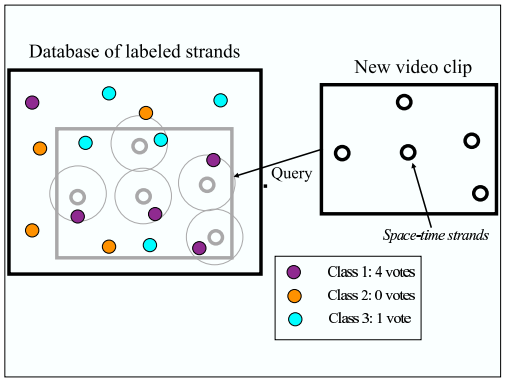

Figure 3. Principle of k-NN recognition of a video sequence represented by a set of video strands.

4. Details of Feature Extraction

4.1. Space-Time Segmentation

In this section, we provide more detail about our space-time segmentation technique. Consider a pixel $P_t = (t, x, y)$ at frame t and position (x, y) that belongs to a color patch (Fig. 4, left). In frame $t + 1$, the patch has moved by incremental displacements u in the x direction and v in the y direction, and the pixel P_t of frame t has moved to $P_{t+1} = (t + 1, x + u, y + v)$ (u and v can of course be equal to zero).

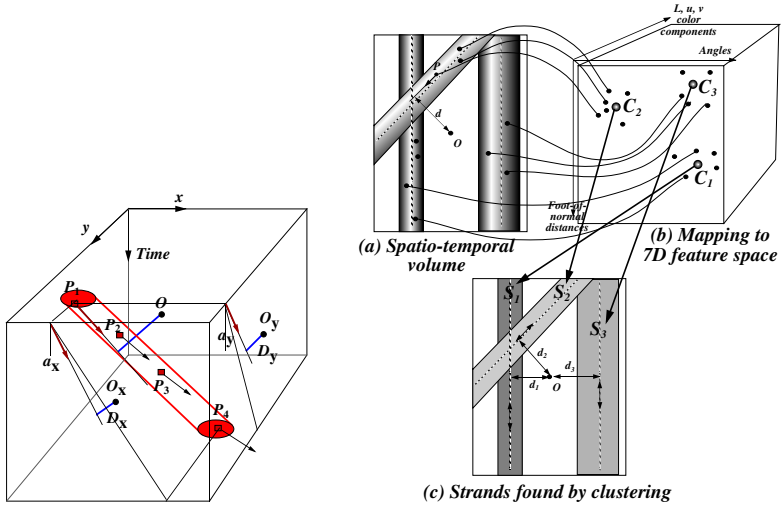

Figure 4. Left: A feature vector with seven components can be defined for each pixel in such a way that pixels along the trajectory of a color patch in the video stack all have feature vectors that are neighbors in feature space. Right: Mapping process from pixels to points feature space, and inverse mapping using the features and labels of cluster centers to obtain segmented regions and video strands.

The 3D direction $(1, u, v)$ is the direction of motion of the patch in the video stack. The motion vector $(1, u, v)$ can be found by optical flow computation (we use a Lucas-Kanade implementation (Lucas and Kanade, 1985) with adaptive filter sizes).

Images of color patches in a video stack produce trajectories along which color components tend to remain stable over a few frames, and the motion vectors $(1, u, v)$ of pixels in these patches tend to remain parallel.

Instead of directly using u and v to characterize the motions of color patches, we project the motion vectors onto the planes (t, x) and (t, y) of the video stack, and let α_x and α_y be the angles of these projections with respect to the plane (x, y) When the color patch does not move, both angles are 90^0. Angles approach 0^0 or 180^0 only for very rapid motions. While the angles obtained from the local optical flow computation characterize *local* orientation trends of trajectories, after the clustering process (described below) the angles of the cluster centers characterize the *global* orientations of the patch trajectories over several frames.

Not only are color and direction of motion approximately constant for a color patch, but in addition the motion vectors are aligned, i.e. the supporting 3D lines of the motion vectors are approximately superposed

(Fig. 4, left) and the parametric representations of these supporting lines should have similar parameters. One way to represent a 3D line is to project the line on the two planes (t, x) and (t, y), and to use the parametric representations of each of the projections. As parametric representation of a line projection, we use a pair (angle, distance).

Therefore, for each pixel P_i, we project the supporting line of its motion vector on the planes (t, x) and (t, y) and obtain two lines L_x and L_y. These lines respectively make angles α_x and α_y (defined above) with the plane (x, y). Let the point O be at the center of the video stack and let O_x and O_y be the two projections of O onto these planes. We consider the distance D_x from O_x to L_x and the distance D_y from O_y to L_y for a pixel located at t, x, y in the video stack. We call the position components D_x and D_y of the motion vectors the *motion distances* for pixel P. Note that each set of four parameters $(\alpha_x, D_x, \alpha_y, D_y)$ uniquely defines a 3D line in the video stack as the intersection of two planes perpendicular to the planes (t, x) and (t, y) and passing through the projections (α_x, D_x) and (α_y, D_y). This unique parameterization is just one among several possible representations of 3D lines. The clustering process discussed next uses the property that similar supporting lines of motion vectors from successive instances of the same patch inside successive frames of the video stack will produce similar sets of parameters.

At each pixel of a video stack, seven components are defined: two motion angles, two motion distances, and three color parameters. We can interpret these quantities as feature components; they define a feature vector which can be represented as a point in feature space. The components are approximate invariants in the color patches; *the points of the feature space that represent pixels of the same color patch moving through time tend to be close together and to form a cluster.* Therefore cluster analysis in this feature space allows us to detect and segment pixels that belong to color patches evolving through time.

Our approach to space-time segmentation is illustrated in Fig. 4 (right): (1) map pixels to points in feature space, (2) determine clusters in feature space (Section 4.2), (3) assign to each point the index of the cluster to which it belongs, and assign to each pixel of the video stack the index of its mapped point. Since each pixel of a color patch tends to be mapped to the same neighborhood and to belong to the same feature space cluster, color patch pixels tend to be assigned the same index across all the frames of the video stack. Therefore they are *tracked* from frame to frame, in the sense that given indexed color patches in one frame, we can find them in the next frames as the patches with the same indices.

We also find the centers of the clusters in feature space. Since the feature space has seven dimensions, these centers have seven components, which together characterize average values of the motion angles, motion distances, and colors of the patches through time. We can concisely describe a video clip by its set of cluster centers, whose components describe average characteristics of the video strands. These components correspond to the spines of the cylinders generated by the moving color patches, and can be drawn as colored lines in video stacks. See also (DeMenthon, 2002).

4.2. Hierarchical Mean Shift Analysis

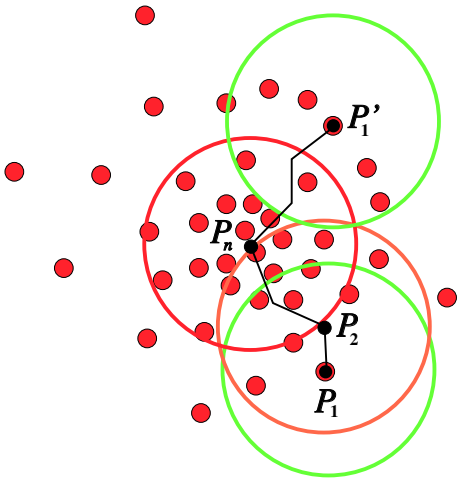

Figure 5. Principle of mean shift analysis: To find the cluster center for point P_1, repeatedly find the centroid of points inside a sphere (initially at P_1) and recenter the sphere on the centroid, until the sphere is stationary (point P_n). (For Gaussian-kernel mean shift analysis, points further from sphere centers are given exponentially decreasing weights in the centroid calculation.) This produces an adaptive gradient ascent mechanism in the space of point densities.

We now turn to the clustering technique that we use to group video stack pixels when they are mapped to feature space. We call it *hierarchical mean shift analysis*. Mean shift analysis is a clustering approach summarized in Fig. 5. For details, see (Comaniciu and Meer, 2002; Cheng, 1995; Fukunaga, 1990). Leung et al. elegantly prove (Leung et al., 2000, p. 1359, Section 3.3) that performing mean shift analysis with a Gaussian kernel is equivalent to performing the following two steps: (1) find a Parzen density estimation of the data set, and (2) find cluster memberships of individual data points by gradient

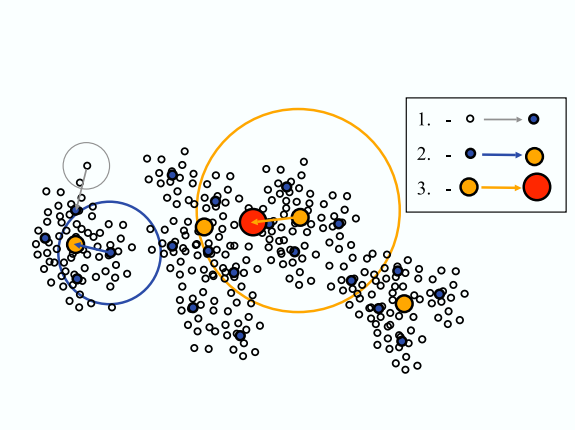

Figure 6. Principle of hierarchical mean shift analysis. The original point set is composed of the small hollow points. Mean shift analysis with a small radius produces a set of small clusters. The cluster centers (larger blue points) are used as a new data set, which is again clustered using a larger radius (blue circle). This process can be repeated (orange points and circles) until a single cluster center remains (large red point).

ascent on the density estimation. By contrast with the classical k-means approach, the number of clusters does not need to be specified. Furthermore, with k-means the clusters are artificially separated by hyperplanes at equal distance between the cluster centers, while clusters that are found with mean shift are separated by valleys of point densities. Finding the natural borders of clusters is important, because such borders in feature space are mapped back to more natural segmentation borders.

Mean shift clustering takes a set of background points and a set of starting points, and requires finding centroids of background points contained in spheres of a given radius R centered on starting points, or centered on centroids found at the previous step. Finding points within spheres requires finding points within distance R of the sphere centers. Therefore, what is needed is an efficient *range searching* algorithm. Range searching is actually one of the internal components of k-NN algorithms discussed above. Therefore, its complexity is also generally reduced by the sorting of the data in a tree structure.

However, there is a major obstacle to blindly using a tree structure in mean shift analysis in high dimensions: in order to produce a small number of clusters in a 7D space, mean shift has to be run with a *large radius*, typically around 1/5 of the span of the largest feature component. For range searching with a tree structure, the cost for each

query is $O(\log N)$ *only for small radii*; for large radii it is closer to $O(N)$, the complexity of brute force search, because most of the branches of the tree must then be explored. We have adopted a hierarchical approach, first sketched in (Leung et al., 2000, p. 1400, Eq. 22) to circumvent this problem (Fig. 6):

- We first run standard mean shift to completion with a very small radius, starting from all points of the data set and shifting the spheres over the static background of points to reach cluster centers that are local maxima of point densities. In the centroid computations used to compute the shifts, each point is assigned a weight equal to 1. Spheres from several starting points typically converge to the same cluster center, and these points are considered to be members of the corresponding cluster.
- We assign *weights* to these cluster centers, equal to the sums of the weights of the member points.
- We consider the set of cluster centers as a new cloud of points, and recompute a new binary tree. We run mean shift using range searching with a larger radius that is a small multiple of the previous radius (we have used a multiplying factor of 1.25 or 1.5). In the centroid computations, the weight of each point is used.
- We repeat the previous two steps until the desired radius size (or the desired number of large regions) is reached.

A significant speedup is achieved with this method. The reason is that although the initial tree must handle a very large number N of points, it allows efficient range searching because the radius used for range searching is very small. At subsequent steps, the points are cluster centers from the previous passes, and their number N' gets smaller at every pass as the radius gets larger; therefore the new tree structure generated for range searching contains N' points, with N' much smaller than N when the radius is large. The complexity of range searching per query then deteriorates toward $O(N')$, but this is not costly because N' is already quite small when this occurs. Experiments confirming a reduction of empirical complexity from $O(N)$ for standard mean shift to $O(\mathrm{Log}N)$ for hierarchical mean shift can be found in (DeMenthon, 2002).

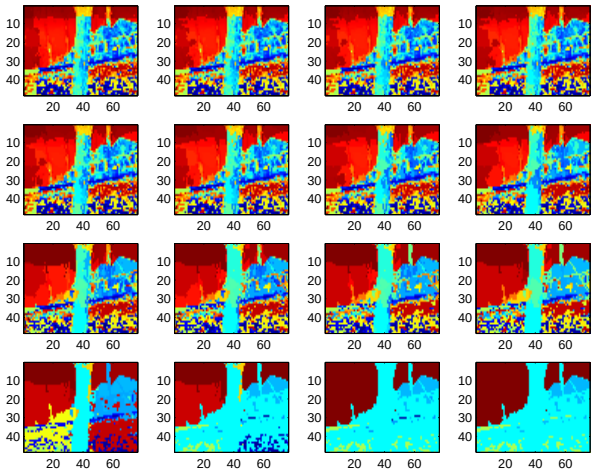

Figure 7. Labeling of segmented regions (represented in pseudo-color) at increasing scales of hierarchical mean shift analysis for the Flower Garden sequence.

4.3. Threshold-Free Segmentation

Hierarchical mean shift produces a hierarchical segmentation that can be represented as a tree: at each step of the procedure, clusters are merged into new clusters; each cluster represents a region of the video stack, and regions corresponding to new clusters are groupings of regions corresponding to clusters of the previous step. The *scale* of the clustering at each step is given by the mean shift radius for the step. A fine-to-coarse evolution of the segmentation occurs from step to step. With a large enough radius, we would obtain a single region corresponding to the whole set of pixels (Fig. 7 shows several steps of this evolution for the Flower Garden sequence). However, our goal is to obtain a segmentation computation that is stable in the presence of lighting variations or compression artifacts. To achieve this, we have to give the preference to regions that remain immune to merging across the largest range of scales of the fine-to-coarse segmentation. Applying this rule, which is inspired by the scale-space school of thought and suggested by (Leung et al., 2000) for mean shift clustering, frees the segmentation computation of any threshold or cluster count setup. In Fig. 8, an example of clustering tree is shown for a simple three-pixel region, along with the evolution of segmentation for the region (left column) and the segmentation produced by considering the most stable clusters (right column).

To find the regions that are the most stable during the hierarchical clustering process, we need to count the numbers of steps of hierarchi-

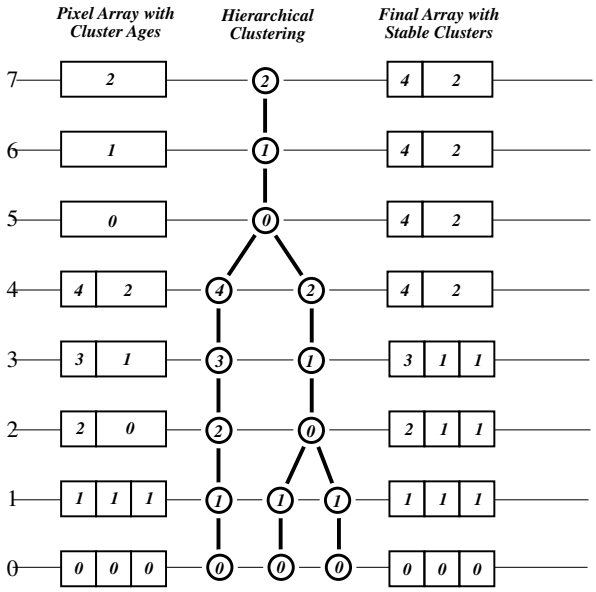

Figure 8. Example of hierarchical clustering (center column) for a three-pixel image and corresponding evolution of regions of image (left column) from bottom to top. On the right column, a segmented image using only the most stable regions is constructed from bottom to top. There, a cluster is allowed to replace an existing cluster only if its age is higher than the weighted average age of its children, with weights equal to the relative pixel proportions of the children. The numbers along the left edge are the clustering steps, used to compute the ages of the clusters. Cluster ages are shown in the tree nodes and in the regions of the three-pixel images.

cal clustering during which each cluster center survives intact without being merged to another region. We call this step count the *age* of a cluster. Life for a cluster begins when it was created by the merging of several children clusters and ends when it gets merged to other clusters into a parent cluster (Fig. 8). We keep track of the cluster centers that are older than the average age of their children (we use a weighted average in which each child's age is multiplied by its relative size in pixels, one of the many schemes listed in (Leung et al., 2000)). Once these are merged and die, they will be forgotten and superseded only if a parent or grandparent that contains them is itself able to become older than the average age of its children. Unless this happens they are at each clustering step remembered and used to define the segmentation borders for the pixels that they comprise.

To implement this mechanism, we maintain two 3D segmentation arrays containing cluster ages and cluster labels for each pixel; the first array is temporary and at each clustering step contains for each pixel

the label and age of the cluster to which it belongs (Fig. 8, left column); in the second array, at each clustering step we write for each pixel the label and age of the oldest cluster containing this pixel so far (Fig. 8, right column). As a result, in the second array, the surviving regions at the end of the clustering process are those that are more stable than all their children and more stable than all their ancestors in the tree that represents the hierarchical segmentation (Fig. 8, middle column).

Note that with this segmentation method (as with plain mean shift), patches that are disconnected in the video stack can produce feature vectors that are very close to each other in feature space because, even though they are some distance apart, they move identically and have the same color. Being in the same feature space cluster they are given the same label. For example in the Flower Garden sequence, the pixels of the red flowers of the garden receive the same segmentation label over large areas, because their color and motion components bring them together in feature space. This is not an issue in classification, as sets of disjoint patches are summarized by single video strands both in the training and query videos. For segmentation tasks requiring separate labeling for disjoint regions, we have adopted the straightforward solution of considering in turn each label and its corresponding set of pixels, performing a 3D connected component analysis for this set, and relabeling each subset of connected pixels in the 3D video stack with a distinct label.

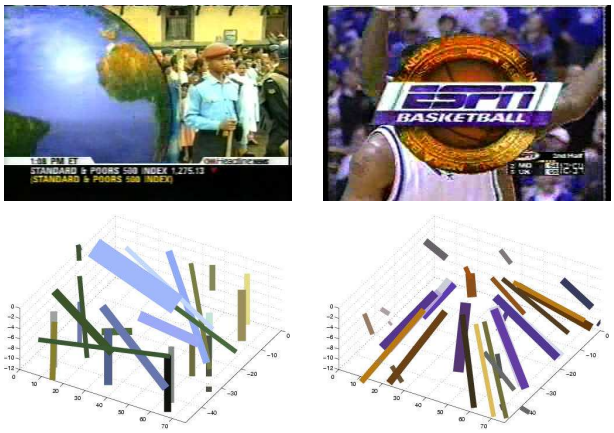

Figure 9. Sample frames of dynamic logos (top), and corresponding video strands (bottom). The left logo is a globe that sweeps from left to right over the background crowd. It produces highly slanted blue strands (bottom left), while the background strands are vertical. The right logo expands quickly over the basketball court background to indicate the beginning of a replay. It produces a characteristic conical pattern of video strands (bottom right).

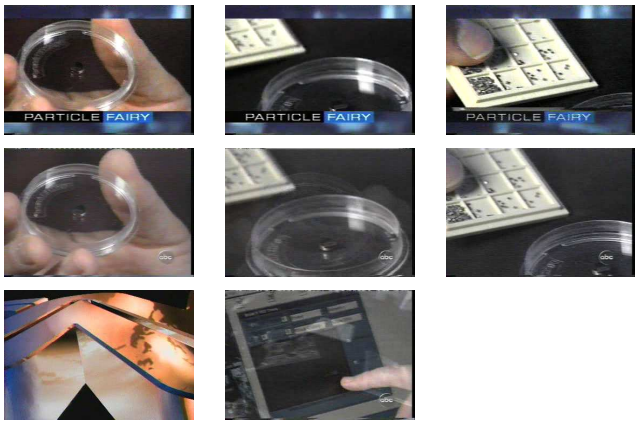

Figure 10. Frames from short clips of a news summary used as queries (top row), and frames from clips of the full story that appeared several minutes later (second row). The locations of these clips were retrieved by the system. The two frames of the bottom row belong to the two incorrect clips that were returned for two of the eight query clips.

5. Detailed Training and Recognition Approach

Consider the task of retrieving from a video library the segments that are most similar to a query clip; one simple approach that first comes to mind is to represent each clip as a *single point* in a vector space with components that describe clip features and labels that describe their class (e.g., video source or category). The query clip would also be represented by a single point. Then we would turn to one of the classic nonparametric classification techniques to tell which class the query clip is most likely to belong to, using for example the k-nearest neighbor rule (more details below). The major problem with this approach is that it is difficult to produce a rich description of a clip with a single point in a vector space. This single point would have to summarize the structure of the clip with a set of components of fixed dimensionality, regardless of its internal structure and whether it contains a single object or a dozen. Such a representation is likely to oversimplify the structure of clips and to lose information about individual objects. Furthermore, some ordering would have to be chosen for the feature components. A change of ordering produces a completely different point, and therefore should only occur for completely different object configurations in the clip. To our knowledge, there is no satisfactory solution to finding an ordering of feature components that would reflect individual object

characteristics and would be stable against small changes of object configuration.

To circumvent these problems, we are representing each video clip by a variable number of points instead of a single point. Each point describes one video strand of the clip, so that the set of points represents the set of video strands. The number of points varies from one to several dozens. Then the retrieval approach just suggested can be edited to use the representation of clips by sets of points instead of single points in the following way:

- Represent each clip generated from the library videos by a set of points in a feature vector space, each with the same label describing the clip class (video source or category).
- For the query stage, represent the query clip by a set of n query points
- For each query point, perform a k-nearest neighbor search to find k neighbor points and their labels showing their class. For the query clip, a total of kn labels will be found.
- Tally the classes. Select the class with the highest vote.

These steps are illustrated in Fig. 2 and Fig. 3, and were also described in general terms in the overview of our approach in Section 3.2. As shown in Fig. 2, videos are chopped into subclips of around 1/2 second, whether they are used for indexing or for retrieval. The merit of this operation is that, over this short duration, the motion of objects in the world and cameras can be considered linear; also, the clusters that represent moving color patches of the world are more compact than over extended periods, and the centers of these clusters (found by the hierarchical mean shift described in Section 4) are better summaries for compact clusters than for elongated clusters.

Objects moving over extended periods generate moving patches inside several subclips, and since each subclip is segmented into video strands, the same color patch is represented by several segments of video strands representing the same trajectory across several subclips and consequently generates several points. This is not a concern for the nearest-neighbor classification; indeed, in the training database, these points are all given the same label which identifies the address or class of the larger video of origin (and not the small 1/2 second subclip).

Among the queries we tested, some dynamic logos used to bracket slow motion replay in a sports broadcast would seem particularly challenging because they are zoomed (in or out) on top of random backgrounds of live frames; therefore many of the video strands generated

in subclips during the training or query phases come from the random backgrounds. However, this is not a concern for this approach, as those strands simply don't get matched during the nearest-neighbor search, while strands from the foreground logos are matched and contribute to the classification voting.

A more subtle issue arises because of the use of position components for the points describing video strands. Position components carry important information about the spatial arrangement of strands in each clip, and retrieval performance deteriorates without their use. However, for a color patch, the position components of video strand segments at successive subclips will be all the more different that the strand is more slanted. If we limit subclip duration to 1/2 second, the subclips generated when chopping query clips have a maximum offset of 1/4 second with respect to corresponding training subclips, which limits the magnitude of position shifts, so that even slanted video strands in the query will typically still be in the neighborhood of corresponding training video strands. The requirement of limiting deviations in position components of video strands with large slants due to time offsets between training and query video subclips is an additional reason why we are selecting short subclips. Even so, we find that giving the position components less weight than to the other components in the distance computations tends to improve performance of the k-nearest neighbor retrieval process (we do find position components to be useful, so we do not set these weights to zero). Also note that slanted video strands are less common and therefore more discriminative than video strands of static patches. This seems to contribute in offsetting the negative impact of position deviations. More details are given in the experimental sections.

The nearest-neighbor rule of pattern classification is a nonparametric classification method that bypasses probability estimations and directly yields a decision function. The nearest-neighbor rule consists of assigning a query point $\mathbf{x}$ the label of the point $\mathbf{x}'$ nearest to $\mathbf{x}$. When the number of labeled points is very large, one can show that the error rate for this decision is at most twice that of the minimum possible error rate obtained from the Bayes decision rule (Duda–Hart–Stork, 2001). With the k-nearest neighbor rule, a decision is made by examining the labels of the k nearest points and selecting the label that received the most votes. In the multi-point method just described, each query is represented by a set of points $\mathbf{x_i}$. Pooling the votes provided by the nearest neighbors of each point is the simplest approach. It is the approach taken by people facing similar issues in image retrieval, for example for the retrieval of gray level images described by a variable number of interest points (Schmid and Mohr, 1997). Note also that even when

the classification task does allow for the definition of single feature points to describe the entities that have to be classified, the "curse of dimensionality" for these points or the need to deal with missing feature components may force researchers to replace these points by sets of points of lower dimension, either by projection into several subspaces or by random selection of subsets of components (Bay, 1999). They then end up with the same representation of their entities with sets of points as we do and also turn to the cumulative voting of the nearest neighbor labels of each point. These researchers report increased classification speed with acceptable increase of misclassification. Looking at a more formal analysis of this multi-point method could provide insights on asymptotic upper bounds in comparison to the Bayes error, and could point us to refinements leading to improved classification performance. This is left to future work.

There are essentially three methods for reducing the complexity of k-NN search, (1) pruning using partial distances computed from a subset of the point dimensions, (2) organizing the database points in a tree structure and tree pruning based on the triangle inequality, and (3) generation of a Voronoi diagram of the class separations in the database and discarding of the points that don't contribute to the diagram (Duda–Hart–Stork, 2001). Currently, our system relies only on the tree structure method. This method reduces the average complexity of a single query from $O(N)$ to $O(\mathrm{Log}N)$. With this method, a tree structure is used to partition the set of data points into smaller subsets that are contained in (hyper)spheres. Each partition is represented by its sphere center and radius. We use the gh-tree (generalized hyperplane tree) introduced in (Uhlmann, 1991) in which two points are picked and the remaining points are divided into two groups depending on which of these two points they are closer to. This partitioning is repeated recursively to create a binary tree structure. Our system runs Merkwirth's implementation of this technique (Merkwirth et al., 2000), in which the two points are selected as far from each other as possible, to reduce sphere overlaps (at the expense of larger sphere radii). The reason for the reduced query complexity is that, once k candidate neighbors to a query point have been found, there is no need to descend a branch of the tree if the corresponding center and radius of the branch tell us that all the points inside the sphere are further from the query point than these candidates. Then the whole branch can be pruned from the search. Note that this pruning assumes that the triangle inequality holds, i.e. that the points are in a metric space. In our case, the components of each point have independent distributions (the color components are in the LUV space and the other components are also uncorrelated), thus we

can represent the video strand points in a Euclidean space and choose the L_2 norm in the tree construction and search.

Finally, it should be noted that in practice, while the segmentation used to obtain the video strands is run without requirements for parameter setup, the choice of k in the k-nearest neighbor retrieval as well as the weighting of the strand components in the distance calculations benefit from some tuning for optimal performance in our experiments. Instead, it would be useful to automatically adjust k as a function of N the number of database points (Duda–Hart–Stork, 2001), and to automatically adjust the weights of the components as a function of their variance within each class in the training set.

6. Retrieval of Near-Duplicates

Our first set of experiments focused on dynamic logos. Dynamic logos are graphic animations used by broadcasters during the introduction of a show, or to separate sections of a show. The ability to recognize dynamic logos is useful, because it can let the user search for specific shows in a large database by searching for the dynamic logos that are used to introduce them. In sports, specific dynamic logos are also used to separate a replay from the live broadcast. Examples are shown in Fig. 9. We used 19 dynamic logos, each composed of 15 to 100 frames. These logos were chopped into 12-frame clips; video strands were extracted from these clips, and the corresponding 7D points were stored in a database, each with a label specifying its dynamic logo of origin. Note that the components of all the database points are normalized so that all the points fit in a hypercube of size 1. In other words, the range of each component is mapped to a range $[0, 1]$. Each clip produced 11 points in average. A total of 430 other short clips from news broadcasts were also used to produce other 7D points in the database, so that 90% of the points were not from logos. A total of 5300 points were created in the database. These points are organized in a tree structure which gives the k-nearest neighbor operation used for retrieval an average cost of $O(\log N)$ in the number of database points for each query.

As described in Section 5, recognizing a clip consists of using the complete set of video strands for the clip as a set of 7D points for query to the database of labeled points, applying k-NN for each 7D point to retrieve the labels of neighbor labeled points, and selecting the dynamic logo with the label that received the highest label count in the retrieval. In the calculations of the Euclidean distances between runtime points and training points in k-NN, we find that the best results are obtained when normalized orientation and color components

of the points have three times the weight of the normalized position components. As discussed in the previous section, a likely reason for this observation is that query clips can be offset by up to 6 frames with respect to training clips, which does not affect orientation components but introduces deviations in the position components of slanted video strands. Also, $k = 3$ in k-NN seemed to work best for our data, but the optimal number may be a function of the average density of the points, i.e. of the total number of training points in the database.

To test the ability of the system for logo recognition, the goal was to identify unlabeled 12-frame clips coming from the same pool of logos, but with the worst possible offset (6 frames) compared to the offset used for the generation of labeled points. Around 45 clips were used as queries. All queries, except one, were correctly assigned to their logo of origin, a 97% success rate.

In a second set of experiments, we used, as queries, video sequences that were not exact duplicates of those used to generate the database of labeled points. Extended news broadcasts are often introduced by short summaries advertising the full stories to come; these summaries are framed by large banners containing text that describes their content. For example, Fig. 10 shows, in the top row, frames from the news summary, and, in the second row, frames from the full story that was broadcast several minutes later. Our goal was to find the location of the full story when using query clips from the summary. The frames of the second row of Fig. 10 are mid-frames of the clips that were correctly retrieved. In the example of Fig. 10, eight successive clips from a single summary were used as queries. The full story contained around 5300 frames, and was chopped into 430 short clips which produced around 5000 points labeled by the index of the mid-frames of the clips. Retrieval again used k-NN and voting. The correct locations were always found among the three highest votes. In the example of Fig. 10, six out of eight correct locations received the highest vote. The two frames of the bottom row of Fig. 10 are frames of clips that were incorrectly located. They contain motions similar to the query clip. Finally, if the sequence of eight clips is used as a single query, the location of a middle clip within the correct sequence wins by a large margin. Such near-duplicate video search would allow the user of a video recorder to easily jump from a news summary item that interests him to the full story.

7. Action Recognition

We also applied our system to the retrieval of actions captured from a fixed camera. Applications include tools in support of the review of

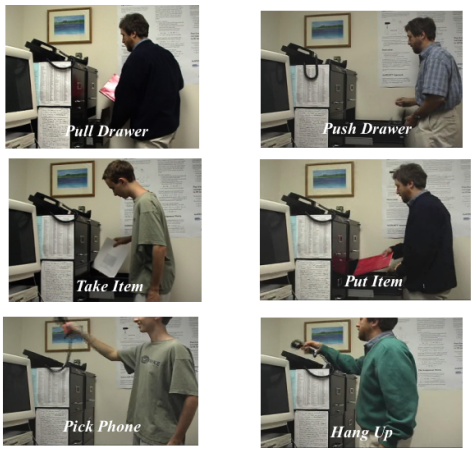

Figure 11. Representative frames for each of the actions for which the system was trained.

surveillance video, as described in the last scenario of Section 2. We used video sequences from a camera observing a cabinet at a corner of an office. Subjects come to the cabinet to open a drawer, put or take folders or books, close the drawer, then leave. Or they pick up, then put down, a phone receiver located on top of the cabinet. Six action parts were considered, pulling drawer, pushing drawer, taking item from drawer, putting item in drawer, picking up phone, and hanging up. Representative frames for these actions are shown in Fig. 11. Although the figures show only two subjects, five subjects were involved on different days, and contributed to several video takes, wearing different clothes. A total of 85 action-specific clips were obtained from the video data, and 14 frames of each clip were used by the space-time segmentation module for the extraction of video strands. An example of space-time segmentation for the action of putting a folder in a drawer is shown in Fig. 12. Examples of sets of video strands obtained as a result of space-time segmentation for each of the six actions are shown in Fig. 14. The colors of the cluster centers are shown for the corresponding video strands in this figure, but they were not used for recognition because we wanted the approach to be independent of people's appearance. We also ignore all cluster centers corresponding to stationary regions with vertical spines. Here, the loss of color specificity is partially compensated by the unlikely geometric distribution of regions with high slants composing active scenes. In Fig. 14 the time axis is vertical pointing downward. The strands with higher slants correspond to color patches with high motions. Drawer actions incline the strands toward

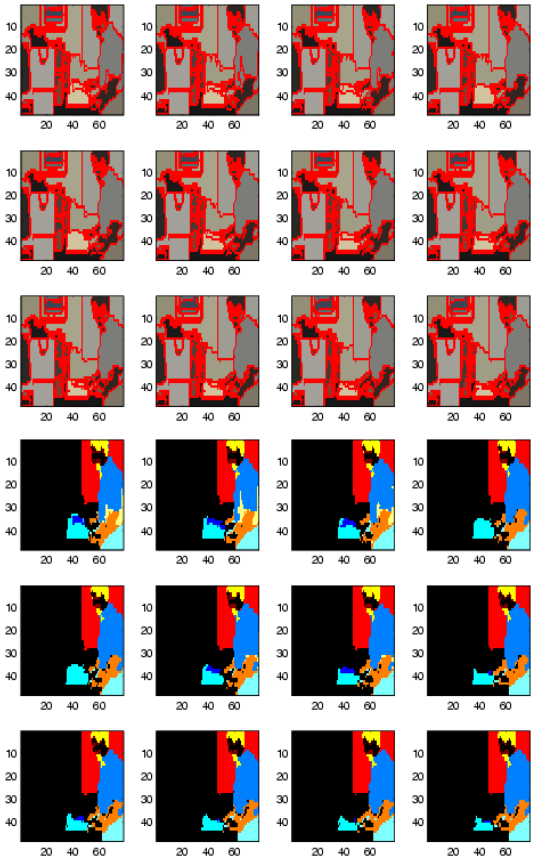

Figure 12. Segmentation obtained by hierarchical mean shift without thresholding. Top: Regions are colored with colors of cluster centers. Bottom: False colors are used to indicate distinct labels. Note the consistent labeling of corresponding regions from frame to frame. All regions for which the cluster centers indicated near-zero motion are colored in black.

the direction of frame rows, while item handling inclines them more toward frame columns.

There were 1919 points in our database for these action recognition experiments. Each point is labeled by a number from 1 to 6 indicating which of the six actions produced it. As a refinement, specific regions of interest can be defined for each action during training (Fig. 13). They are the regions where the action is expected to occur, and to which the focus of attention is limited for the considered training action. These are shown as space-time rectangular volumes in the sets of strands of Fig. 14. For each action, only strands inside its region of interest were considered as training data. This tended to improve the performance

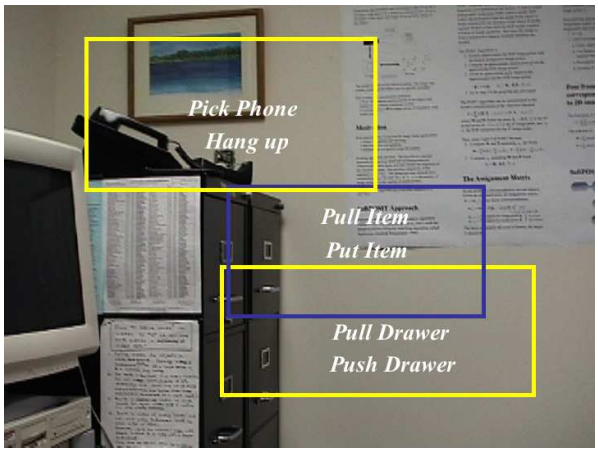

Figure 13. Regions of interest can be defined to limit the focus of attention during the training (i.e., indexing) phase of the system

of the system. The following evaluation made use of this improvement. Of course, during runtime action recognition, all strands produced by the video have to be considered as parts of the query since there is typically no knowledge at that time about which action is taking place or where in space it is more likely to occur.

For performance evaluation, we applied the leave-one-out approach, taking each action clip one after the other, removing all trace of its contribution from the training database, finding its action label, and comparing the result to its actual label. Again, finding the action label consists of using the complete set of strands for the clip as a set of points for query to the training database, applying k-NN ($k = 3$) for each point to retrieve the labels of neighbor training points, and selecting the action with the highest number of labels. Orientation components of the points were given three times the weight of the position components. (As already mentioned, the color components are not used so that people's appearances are factored out).

The results of this evaluation are presented as a confusion matrix in Fig. 15. The matrix has one row per action and also one column per action. Each row of the table corresponds to the performance evaluation for one specific action and each column cell along that row indicates the percentage of votes that the action of that column received. The number in the diagonal cell is the number of times this action was correctly recognized, as a percentage over the total number of times that this action was presented to the system. For example, the action "Put item" was correctly recognized as a "Put item" action 86 % of the time, while the rest of the time it was recognized as a "Push drawer" action. It is

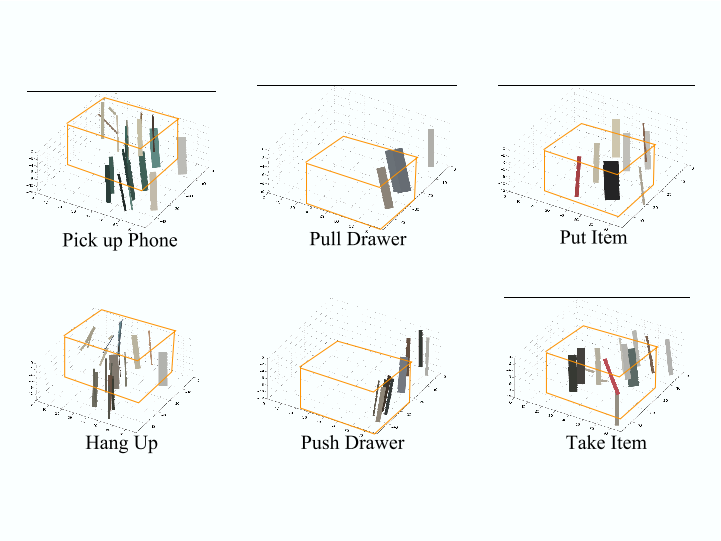

Figure 14. Examples of video strands obtained for office actions. The rectangular volumes correspond to the predefined regions of interest in which specific actions are expected to occur and are used to prune the strands during the indexing phase.

interesting to note that subjects tended to start closing the drawer before they were done dropping the item in it, so there was indeed

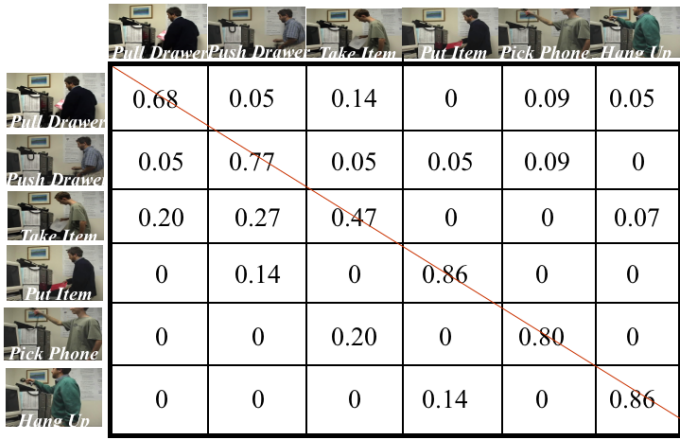

Figure 15. Confusion matrix for six actions.

some drawer pushing ingredients in this action. The phone "Hang up" action in the last row was sometimes confused with a "Put item" action 14% of the time. The lowest results were for the "Take item" action recognized 47% of the time, with 20% or more votes going to drawer pulling and pushing. Again, users tended to seamlessly combine drawer and item handling, so there was no true pure action and the response of the system reflected this fact.

The percentages of correct action recognition given by the performance evaluation are very encouraging, given the fact that we went directly from low level processing to action recognition. We have avoided the detection of bodies, limbs, hands, folders, drawers. The main reasons for this success seem to be the discriminative power of sets of video strands for summarizing actions in video clips, and the stable computation of these video strands by our hierarchical mean shift technique.

8. Conclusions

We have described a combination of techniques that allow retrieval of near-duplicate videos and recognition of actions in surveillance video. Videos are divided into short clips, and the spines of the cylinders generated by color patches in the space-time pixel blocks of these clips, the *video strands*, are extracted by a hierarchical segmentation technique that efficiently finds the strands that are most stable at multiple grouping scale, without preset threshold or predefined tube count. These strands are colored lines that can be represented as points in a high-dimensional feature space and are assigned labels that specify their video clip of origin or the category of the clip. All the labeled points for all the clips are organized in a tree structure to reduce k-nearest neighbor retrieval complexity. The retrieval phase uses video segments as queries. Short clips of these queries are again segmented to produce sets of points. The labels of their nearest neighbors are tallied, and the cumulated votes are used to rank the results. We have illustrated this approach for video indexing and retrieval of dynamic logos and near-duplicates, as well as for action recognition when people interact with a fixed environment and are being observed by a fixed surveillance camera.

Our contributions to video indexing and retrieval can be summarized as follows:

1. Space-time segmentation is a useful tool for transforming the dynamic content of video clips into simple, purely geometric configurations of video strands.

2. With hierarchical mean shift, these geometric patterns capture enough information about the dynamic interaction of color regions in videos to allow for the successful use of pattern recognition techniques.

3. If these patterns are represented as sets of high-dimensional points, an approach combining k-nearest neighbor search and voting on the retrieved labels can provide fast and efficient retrieval.

4. For the retrieval of near-duplicates in video, this approach allows for a high level of resilience against video clip variability caused by editing and overlays.

5. For the recognition of actions occurring at fixed places in the field of view of fixed surveillance cameras, this approach provides good discriminative power while avoiding the difficult intermediary steps of body part and object recognition and tracking that are plaguing other approaches.

The time spent extracting video strands from video is not an issue during video indexing, but it would be the main bottleneck at run time if the video clips used as queries have not been preprocessed. We are working on accelerating our hierarchical mean shift analysis. Other research areas of interest include a formal performance evaluation of the video retrieval performance with very large data sets, the adjustment of k in the k-nearest neighbor retrieval as a function of N the number of database points (for example with $k = k_0\sqrt{N}$ with k_0 constant, as suggested in (Duda–Hart–Stork, 2001)), the acceleration of k-nearest neighbor retrieval by combining the tree pruning technique with other known techniques, and the use of texture descriptors (Martin et al., 2004) as pixel components to obtain video strands of consistent texture.

Acknowledgments

Support of this research by the Department of Defense under Contract MDA 9049-6C-1250 is gratefully acknowledged. The authors would also like to thank the anonymous reviewers for their helpful comments and suggestions.

References

A. Akkusand H.A. Guvenir, "K Nearest Neighbor Classification on Feature Projections", Proceedings of ICML, pp. 12–19, 1996.

M. Allmen and C.R. Dyer, "Computing Spatiotemporal Relations for Dynamic Perceptual Organization", CVGIP: Image Understanding, vol. 58, pp. 338-351, 1993.

S. D. Bay, "Nearest Neighbor Classification from Multiple Feature Subsets", Intell. Data Anal. 3(3), pp.191–209, 1999.

R.C. Bolles, H.H. Baker and D.H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion", Int. J. of Computer Vision, 1(1), pp. 7–55, 1987.

E. Bruno and D. Pellerin, "Global Motion Fourier Series Expansion for Video Indexing and Retrieval", Advances in Visual Information System, VISUAL, Lyon, pp. 327–337, 2000.

E. Bruno and D. Pellerin, "Video Structuring, Indexing and Retrieval based on Global Motion Wavelet Coefficients", Proc. Int. Conf. of Pattern Recognition (ICPR), Quebec City, Canada, 2002.

Y. Cheng, "Mean Shift, Mode Seeking, and Clustering", IEEE Trans. on PAMI, vol. 17, pp. 790–799, 1995.

D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis", IEEE Trans. on PAMI, vol. 24, pp. 603–619, 2002.

A. Del Bimbo, P. Pala and L. Tanganelli, "Video Retrieval based on Dynamics of Color Flows", ICPR 2000, vol. 1, pp. 851–854.

D. DeMenthon, V. Kobla and D. S. Doermann, "Video Summarization by Curve Simplification", ACM Multimedia, pp. 211–218, 1998.

D. DeMenthon, "Spatio-Temporal Segmentation of Video by Hierarchical Mean Shift Analysis", SMVP 2002 (Statistical Methods in Video Processing Workshop), Copenhagen, Denmark, 2002.

N. Dimitrova and F. Golshani, "Motion Recovery for Video Content Classification", ACM Transactions on Information Systems, vol. 13, (4), pp. 408–439, 1995.

N. Dimitrova and M. Abdel-Mottaleb, "Content-based Video Retrieval by Example Video Clip", Proc. SPIE vol. 3022, Storage and Retrieval for Image and Video Databases, pp. 59–70, 1997.

R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern Classification", John Wiley and Sons, Inc., New York, 2001.

R. Fablet, P. Bouthemy and P. Perez, "Non-parametric Motion Characterization using Causal Probabilistic Models for Video Indexing and Retrieval", IEEE Trans. on Image Processing, vol. 11(4), pp. 393–407, 2002.

M. Flickner, H. Sawhney, W. Niblack, J.Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele and P. Yanker, "Query by Image and Video Content: the QBIC System", Computer, vol. 28, no. 9, pp. 23–32, 1995.

K. Fukunaga, "Introduction to Statistical Pattern Recognition" (2nd ed.), Academic Press, 1990.

A. Hampapur, A. Gupta, B. Horowitz, C-F. Shu, C. Fuller, J. Bach, M. Gorkani and R. Jain, "Virage Video Engine", Proc. SPIE vol. 3022, Storage and Retrieval for Image and Video Databases, pp. 188–198, 1997.

Y. Ivanov and A. Bobick, Recognition of Visual Activities and Interactions by Stochastic Parsing, IEEE Trans. PAMI, vol. 22 (8), pp. 852–872, 2000.

V. Kobla and D. Doermann, "Indexing and Retrieval of MPEG-compressed Video", Journal of Electronic Imaging, pp. 294-307, 1998.

Y. Leung, J-S. Zhang and Z-B. Xu, "Clustering by Scale-Space Filtering", IEEE Trans. on PAMI, vol. 22, pp. 1396–1410, 2000.

R. Lienhart, W. Effelsberg and R. Jain, "Visual GREP: A Systematic Method to Compare and Retrieve Video Sequences", Proc. SPIE vol. 3312, Storage and Retrieval for Image and Video Databases, pp. 271–282, 1998.

B. D. Lucas and T. Kanade, "Optical Navigation by the Method of Differences", IJCAI, pp. 981–984, 1985.

D. Martin, C. Fowlkes and J. Malik, "Learning to Detect Natural Image Boundaries Using Local Brightness, Color and Texture Cues", IEEE Trans. PAMI, vol. 26 (5), pp. 530–549, 2004.

C. Merkwirth, U. Parlitz and W. Lautherborn, "Fast Nearest-Neighbor Searching for Nonlinear Signal Processing", Phys. Review E., vol. 62, pp. 2089–2097, 2000. TSTool package available at http://www.physik3.gwdg.de/tstool/

R.C. Nelson and A. Selinger, "A Cubist Approach to Object Recognition" , Proc. ICCV, Bombay, India, pp. 614–621, 1998.

N. Oliver, B. Rosario, A. Pentland, A Bayesian Computer Vision System for Modeling Human Interactions, IEEE Trans. PAMI, vol. 22 (8), pp. 831–843, 2000.

C. Pinhanez and A. Bobick, Human Action Detection using PNF Propagation of Temporal Constraints, Proc. CVPR, pp. 898 –904, 1998.

Y. Ricquebourg and P. Bouthemy, "Real-time Tracking of Moving Persons by Exploiting Spatio-temporal Image Slices", IEEE Trans. PAMI, vol. 22 (8), pp. 797–808, 2000.

C. Schmid, R. Mohr, "Local Gray-Value Invariants for Image Retrieval", IEEE Trans. PAMI, vol. 19 (5), pp. 530–535, 1997.

H. Sun, T. Feng and T. Tan, "Spatio-Temporal Segmentation for Video Surveillance", ICPR 2000, vol. 1, pp. 843–846, 2000.

E. Sahouria, A. Zakhor, "Motion Indexing of Video", ICIP, vol. 2, pp. 526-529, 1997.

T. F. Syeda-Mahmood, A. Vasilescu and S. Sethi, "Recognizing Action Events in Video", IEEE Workshop on Event Detection and Recognition in Video, pp. 64–72, 2001.

J. K. Uhlmann, "Satisfying General Proximity/Similarity Queries with Metric Trees", Information Processing Letters, vol 40, pages 175-179, 1991.