VIDEO RETRIEVAL OF NEAR–DUPLICATES USING K–NEAREST NEIGHBOR RETRIEVAL OF SPATIO–TEMPORAL DESCRIPTORS

Daniel DeMenthon

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

ABSTRACT

This paper describes a novel methodology for retrieval of near-duplicate videos. 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 point in a 7D space whose components represent the average color, position and motion of the region. In the indexing phase of a video database, the 7D points obtained by segmentation for each half-second clip of the videos are assigned labels that specify the origin of the video clip. 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 to produce sets of 7D 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 clips that are the most similar to the query video segment.

We describe experiments of retrieval for dynamic logos as well as for video queries that differ from the indexed broadcasts by the addition of large overlays.

1. INTRODUCTION

Human perception provides us with a unique ability to judge similarity between video sequences at a variety of levels. We can recognize objects, observe interactions, and focus on detailed similarities, even if the common components are only a small part of the scene. Unfortunately, current methodologies for feature extraction, indexing and retrieval in video typically do not capture the essence of the video structure at the object level. Existing approaches focus primarily on text data from closed captions or speech transcription, extensions of image retrieval techniques using keyframes, outputs from specialized detectors for faces or vehicles, and, in some focused domains, activity analysis. While color, shape and texture features are common for indexing at the frame level, temporal features are often ignored in general retrieval systems. A primary reason for this has been an inability to represent the temporal information compactly and to use it effectively in measuring similarity.

We propose a novel methodology for the compact representation of a video's spatio-temporal structure and a system that lets 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. 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.

2. MOTIVATION

A company has paid millions to have 15-second advertizing 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. Instead, the company can record the broadcasts, index them, and search for their advertising spots. The system must reliably recognize an advertising spot even if it has been cut short, or was framed with a dynamic banner warning about severe weather conditions or a schedule of upcoming shows.

A user has recorded a football game while he was out and has time to watch only the most exciting moments when he comes back. These moments generally are repeated as

The support of this research by the Department of Defense under contract MDA9040-2C-0406 is gratefully acknowledged.

slow-motion replays. To alert the viewer that there is a jump back in time, replays are bracketed between two dynamic logos (Fig. 7, right). The interface of the user's TV recording system allows our user to select a video sequence and say "find all the places where a sequence similar to this one occurs". So he selects the first occurrence of the dynamic logo introducing a replay, and his system then lets him jump to all the successive replays.

An intelligence analyst is tasked to evaluate the positions of foreign government-controlled TV broadcasts about the 9/11 attacks. Terabytes of foreign broadcasting have been recorded. An indexing system processes these data in the background for efficient retrieval. The analyst uses a video clip showing one of the airplane-tower collisions that newscasts around the world have been using. The system retrieves pointers to all the broadcasts that presented this clip, and sets them aside for further analysis of the commentators' positions during these broadcasts.

Although these scenarios represent real-world needs, solutions are not yet available. To our knowledge, existing commercial systems and laboratory prototypes do not come close to offering satisfactory solutions. Systems using keyframes would probably retrieve some of the relevant frames, but these would be buried among thousands of irrelevant candidates with similar histograms or correlograms. Other systems have drawbacks too, as discussed below. The dynamic structure of the video is an important feature for reducing the candidate set, and our ability to represent it compactly is essential. We describe a proof-of-concept system that could lead to a commercial application able to address the needs described by these scenarios.

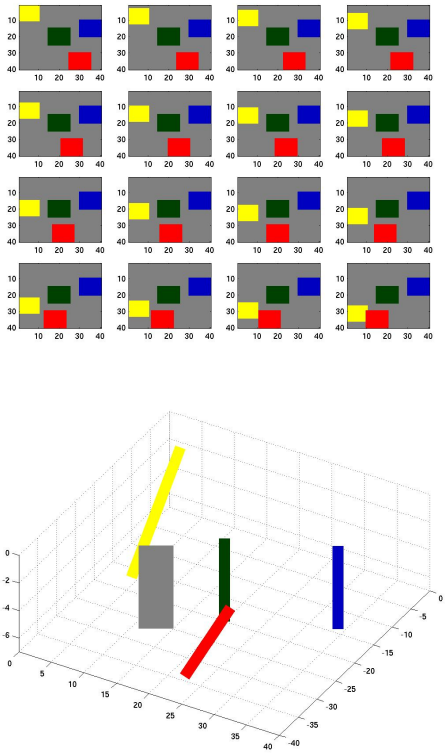

Fig. 1. Sequence of frames with moving colored square blocks (top) and strands extracted by space-time segmentation.

3. OVERVIEW OF APPROACH

3.1. Feature Extraction

The features we use 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 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 space-time regions across the multiple scales of segmentation produced by this technique are selected, and this removes the need for any predetermined scale 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 corresponding to the cluster centers for the moving block sequence, obtained automatically by the space-time clustering technique. We call these spines *video strands*. 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 in video stacks over short durations by color patches 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 seven-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 nearest neighbors and space-time descriptors to the addition of clutter video in the database.

3.2. Training and Recognition

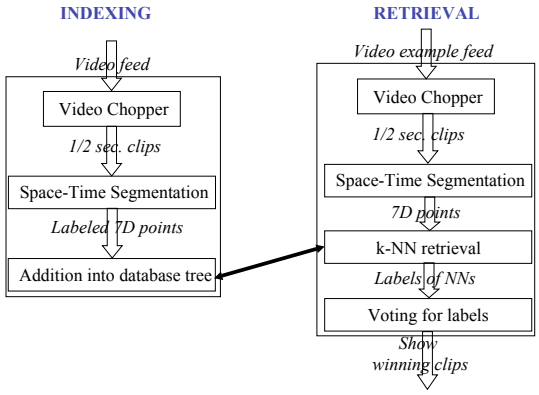

Fig. 2. Anatomy of system for video indexing and retrieval.

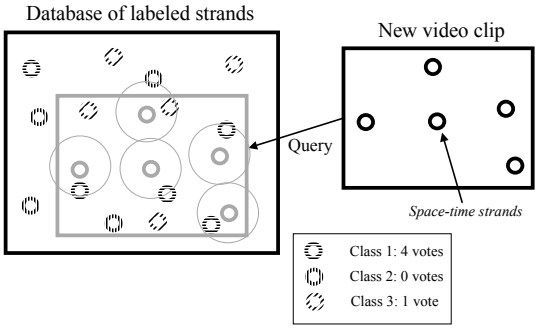

Fig. 3. Principle of k-NN recognition of a video sequence represented by a set of video strands.

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 described above. In both modules, only the 7D points corresponding to the colors, positions and orientations of the video strands 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 of the clip that produced it, and sorted into a binary tree for fast k-NN retrieval. 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 mechanism is illustrated in Fig. 3. It is similar to the retrieval mechanism of gray level images from interest points described by Schmid and Mohr [19]. It is also related to the "cubist approach to object recognition" advocated by Nelson [18].

4. SPACE-TIME SEGMENTATION

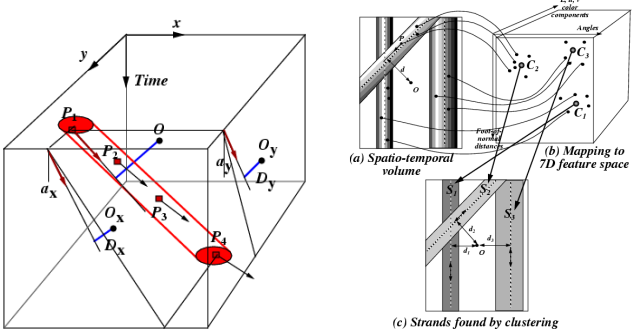

Fig. 4. Left: A feature vector with seven components can be defined for each pixel, such that pixels along the trajectory of a color patch in the video stack all have feature vectors that are neighbors in feature space. Right: Mapping process between pixels and feature space points, and inverse mapping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

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). 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.

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 on 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 lines of the motion vectors are approximately superposed (Fig. 4). 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. 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 D_x and D_y the *motion distances* for pixel P.

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 5), (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. For details, refer to [8].

5. CLUSTERING BY HIERARCHICAL MEAN SHIFT ANALYSIS

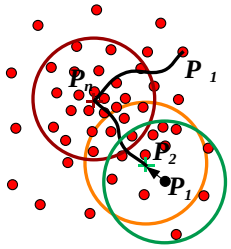

Fig. 5. Principle of mean shift analysis: To find the cluster center for point P_1, repeatedly find the centroid of points inside a sphere (initially at P_1) and recenter the sphere on the centroid, until the sphere is stationary (point P_n). (For Gaussian-kernel mean shift analysis, points further from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

Mean shift analysis is a clustering approach summarized in Fig. 5. For details, see [6, 5, 12]. 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. What is needed is an efficient *range searching* algorithm. For this task we use the powerful *TSTool* package created by Merkwirth et al. [17]. The auxiliary function *nn_prepare* arranges the background point set into a binary tree structure. A set of points is divided into two subsets by the hyperplane halfway between the two farthest points of the set. For each subset, the center C and the enclosing radius r are stored. This is done for each subset until the subset of a branch contains less than a preset number of points. During a range search at the radius R around a point A using the function *range_search*, branches for which the distance $AC-r$ is larger than R cannot contain points within distance R of A, and are pruned.

However, there is a major obstacle to using a tree structure efficiently 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 search that utilizes a tree structure such as the binary tree just described or a K-d tree, the cost for N points is $O(N \log N)$ *only for small radii*; for large radii it is closer to $O(N^2)$,

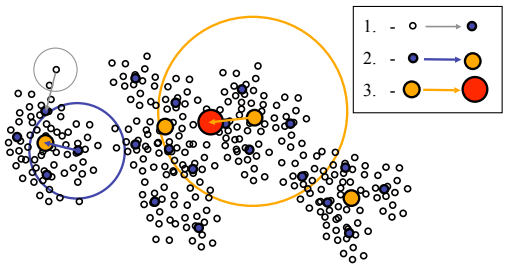

Fig. 6. Principle of hierarchical mean shift analysis. The original point set is composed of the small hollow points. Mean shift analysis with a small radius produces a set of small clusters. The cluster centers (larger blue points) are used as a new data set, which is again clustered using a larger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

because most of the branches of the tree must then be explored. We have adopted a hierarchical mean shift approach 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.

Essentially the same method was discovered by Leung et al. in their *clustering by scale-space filtering* approach (see [15, p. 1400, Eq. 22]).

Qualitatively, the segmentation obtained by this technique looks as good as or better than that obtained by standard mean shift. Equally important, a significant speedup is achieved with this method. The reason is that the initial tree handles a very large number N of points, but allows efficient range searching because the radius of the range search is small. At subsequent steps, the points are clusters 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 the range search contains N' points, with N' much smaller than N when the radius is large. The complexity of the range search then deteriorates toward $O(N'^2)$, but this is not costly because N' is already quite small when this occurs. Experiments confirming this analysis can be found in [8].

6. THRESHOLD-FREE SEGMENTATION FROM HIERARCHICAL MEAN SHIFT

Hierarchical mean shift produces a hierarchical segmentation that can be represented as a tree structure. 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. 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 several scales of the coarse-to-fine segmentation. This requirement, inspired by the scale-space school of thought and suggested by [15] for mean shift clustering, frees the segmentation computation of any threshold setup. To achieve this, we maintain two 3D segmentation arrays ; the first array, in which the regions are merged in the coarse-to-fine analysis, is eventually discarded; the second array contains at the end of the process the desired labeling corresponding to the most stable regions. Both arrays are initialized with the pixel labeling corresponding to the finest segmentation. For both arrays we keep track of the age of each region, defined as the number of steps during which a region survives from being merged into other regions. Every time a region of the first array becomes older, we compare its age with the weighted average age of the regions that it overlaps in the second array (region weights are their pixel size). These overlapped regions correspond

to children or children's children that were promoted from the first array to the second. As soon as a region becomes older than the average age of these overlapped regions, it has proven to be more stable than those regions and therefore its label is promoted to replace the labels of those regions in the second array. On the other hand, regions of the second array may survive if the overlapping region gets merged before it reaches the right age. In other words, in the second array, the surviving regions are those that are more stable than all their children and more stable than all their ancestors in the tree structure that represents the hierarchical segmentation. Observe that with the described method we do not need to actually build this tree structure.

7. EXPERIMENTS

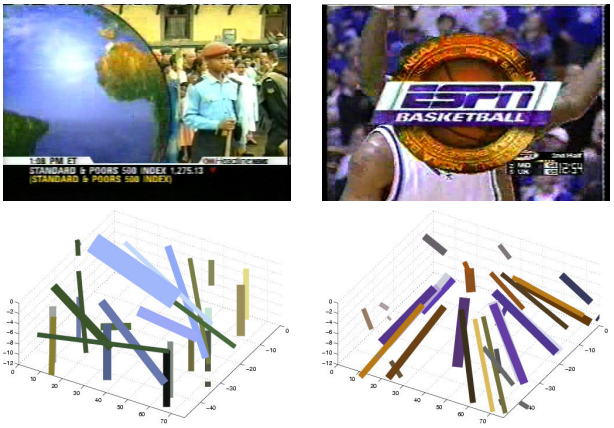

Fig. 7. Sample frames of dynamic logos (top), and corresponding video strands (bottom). The left logo is a globe that sweeps from left to right over the background crowd. It produces highly slanted blue strands (bottom left), while the background strands are vertical. The right logo expands quickly over the basketball background to indicate the beginning of a replay. It produces a characteristic conical pattern of video strands (bottom right).

In all our experiments, we scaled down the frames by a factor of 8, to thumbnails of size 44×30, in order to speed up the computation of the video strands, with the consequence that only the large-scale color regions were considered.

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. 7. 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 the label of its dynamic logo of origin. 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. 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. As described above, recognizing a clip consists of using the complete set of 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 for which the highest number of its labels was retrieved. 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 orientation components of the points have three times the weight of the position components. 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 training points in the database. All queries, except one, were correctly assigned to their logo of origin, a 97% success rate.

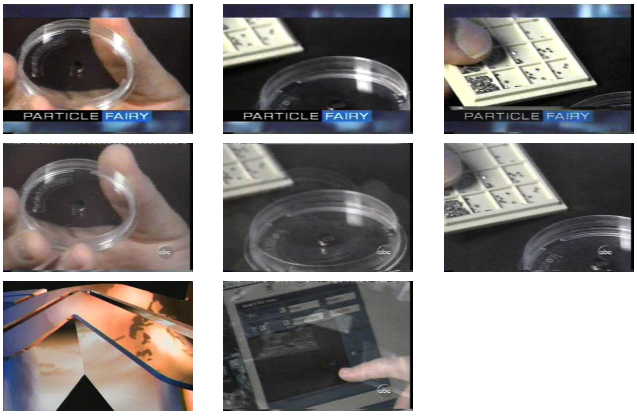

Fig. 8. Frames from short clips of a news summary used as queries (top row), and frames from clips of the full story that appeared several minutes later (second row). The locations of these clips were retrieved by the system. The two frames of the bottom row belong to the two incorrect clips that were returned for two of the eight query clips.

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. 8 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. 8 are mid-frames of the clips that were correctly retrieved. In the example of Fig. 8, 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. Six out of eight correct locations received the highest vote. The two frames of the bottom row of Fig. 8 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. These results are encouraging. We are in the process of performing an extensive formal performance evaluation of our system using several hours of video.

8. RELATED WORK

Recent efforts on developing video indexing and retrieval systems using visual content have leveraged primarily 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 [11] and Virage [13]. There is no attempt to extract temporal relations between keyframes. The visual features used to characterize the videos in commercially available products (histograms-correlograms) are less meaningful than those defined with the proposed approach (moving regions). 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 [16]. 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. Retrieval then involves comparing curves, possibly at hierarchical levels of detail, between stored video sequences and query video sequences [9]. However, the features extracted in each frame typically reflect global pixel statistics whose profile can be completely altered for example by the additions of banners and logos to the original videos. It is our experience that such systems have difficulty retrieving near-duplicates in such cases. By contrast, with our space-time segmentation descriptors, additions of banners would add descriptors without altering descriptors of the main content, and the voting mechanism used by our proposed retrieval system would consider the banner descriptors as outliers that are factored out.

There has been a growing interest in the use of motion information for video indexing and retrieval. For example, Bruno and Pellerin characterize the motion patterns in video by components of the Fourier transform [3] or wavelet coefficients [4] 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.

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. Also, spatio-temporal segmentation is a more direct way of tracking moving regions than the classical approaches extending inferences from image pairs. Early progress in this domain, pioneered by Allmen [1] in the early 90s, was hindered by the heavy memory and processing requirements. However, these obstacles are subsiding. Consequently, a few researchers have started to explore ways of applying such an approach to video indexing [7, 20]. For example, Del Bimbo et al. [7] 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 believe that a characterization of videos in terms of interaction of color regions is an important step toward further refinements in retrieval. In particular, it provides a building block for queries about the occurrence of dynamic events in videos, such as “Show me the sequence where the red car collides with a black car”.

9. CONCLUSIONS

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

1. Space-time segmentation transforms the dynamic con-

tent of video clips into simple purely geometric patterns.

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 dissected into sets of 7D points, an approach using k-NN and voting on the retrieved labels allows for a high level of resilience against video clip variability caused by editing and overlays.

4. Once video strands are obtained for video clips from the runtime video stream, the k-NN voting step is very fast.

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.

10. REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

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

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

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

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

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

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

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

[16] 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.

[17] 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/

[18] R.C. Nelson and A. Selinger, "A cubist approach to object recognition" , Proc. ICCV, Bombay, India, January 1998, pp. 614–621.

[19] C. Schmid, R. Mohr, "Local grayvalue invariants for image retrieval", IEEE Trans. PAMI, vol. 19 (5), 1997, pp. 530–535.

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