SPATIO-TEMPORAL SEGMENTATION OF VIDEO BY HIERARCHICAL MEAN SHIFT ANALYSIS

Daniel DeMenthon

Language and Media Processing (LAMP)
University of Maryland, College Park, MD 20742, USA
daniel@cfar.umd.edu

ABSTRACT

We describe a simple new technique for spatio-temporal segmentation of video sequences. Each pixel of a 3D space-time video stack is mapped to a 7D feature point whose coordinates include three color components, two motion angle components and two motion position components. The clustering of these feature points provides color segmentation and motion segmentation, as well as a consistent labeling of regions over time which amounts to region tracking. For this task we have adopted a hierarchical clustering method which operates by repeatedly applying mean shift analysis over increasing large ranges, using at each pass the cluster centers of the previous pass, with weights equal to the counts of the points that contributed to the clusters. This technique has lower complexity for large mean shift radii than regular mean shift analysis because it can use binary tree structures more efficiently during range search. In addition, it provides a hierarchical segmentation of the data. Applications include video compression and compact descriptions of video sequences for video indexing and retrieval applications.

1. INTRODUCTION AND RELATED WORK

One of the goals of video analysis is to find out as much as possible about what is going on in the scene from what was captured by the video. "Finding out what is going on" is more formally called "semantic interpretation". To interpret a scene, one first needs to label independent objects. Boundaries of objects typically correspond to boundaries of color patches in the video. (However, situations where a foreground object moves in front of a background of similar color are common.) In addition, when the camera translates or when objects move independently, boundaries of objects correspond to boundaries across which the optical flow changes in the video, and the patches inside these image boundaries display some consistency of optical flow. (However, optical flow is notoriously unreliable at object boundaries; also, moving objects could stand still during a shot, or first move then stand still.) To overcome the limitations of motion segmentation and color segmentation and to maximize their chance of extracting object information, researchers have been combining motion and color cues in various ways. The task of dividing video frames into patches that may correspond to objects in the scene is generally called object-based segmentation [15], layer extraction [14, 13, 24], sprite representation [14] or space-time segmentation [1, 3, 11, 19, 25] (there are arguably subtle differences between these concepts).

The support of this research by the Department of Defense under Grant MDA9049-6C-1250 is gratefully acknowledged.

There are two main categories of approaches for space-time segmentation of video sequences, (1) those that track regions from frame to frame, and (2) those that consider the whole 3D volume of pixels and attempt a segmentation of pixel volumes in that block. Our approach belongs to the second category. These two categories have parallels in the field of line detection in images: one can detect lines either by walking along a line and merging pixels for which edge directions are similar to previous edge directions [20]; or one can take decisions based on global information, for example by mapping the pixels to points in a feature space where edge directions and normal distances are components, and clustering these points (this is a Hough transform).

For the frame-by-frame tracking category of spatio-temporal methods, there are more variants than we can adequately review here. Typically, a new frame is divided into small patches (i.e, over-segmented). The patches of the previous frame are shifted to the new frame by use of optical flow information, and the decision of merging regions within frames and from frame pairs is taken according to a similarity measure reflecting their spatial proximity, color similarity and/or motion similarity [4, 8, 10, 19, 23, 25]. To verify motion similarity, parallelism of optical flow vectors can be verified. However, this tends to divide planar background surfaces with large depth variations, such as the ground plane, into several regions in cases of camera translation. Instead, affine motion models can been applied to patches, and patch merging can be based on a similarity measure between motion models, which produces larger regions at once [24]. Interesting new variations on this theme, using statistical modeling [14, 15] or subspace constraints [13], have been proposed recently.

The second category of approaches attempts to directly segment the 3D spatio-temporal pixel volume obtained by piling up frames into a *video stack*. This category has received relatively little attention so far, partly because of its computational demands. Spatio-temporal analysis was pioneered by Adelson and Bergen [1], and Bolles et al. [5], and specifically applied to segmentation by Allmen [3] in the early 90s. Recently, Shi and Malik [21], followed by Fowlkes et al.[11] have obtained space-time segmentations by first constructing a graph in which the nodes are volume pixels and arcs are considered between all pixels, both within frames and between consecutive frames. The strengths of these arcs are decreasing functions of euclidean distances between feature vectors of the pixels, with feature components including color components, frame coordinates and optical flow components. Partitioning into space-time regions is performed by applying K-means clustering in a distance-preserving feature subpace. Finding the subspace involves solving a large system, a problem that requires

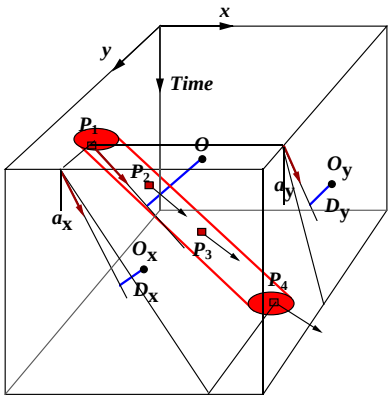


Fig. 1. A feature vector with seven components can be defined for each pixel, such that pixels along the trajectory of a color patch in the video stack all have feature vectors that are neighbors in feature space.

some simplifications, either by considering simply neighbor pixels in the graph [21], or by applying the Nystrom approximation while using only a sampling of the pixels [11].

Our algorithm belongs to the second category but significantly differs from this prior work. Color patches produce generalized cylinders in the pixel volume. Color flows [8], action cylinders [22] and feature trajectories [2] describe similar space-time entities in the literature. We refer to these as *video strands*. Our goals are to extract these strands, group them according to their motions, and characterize them by color, average radius, and position and orientation of their axis. Our algorithm consists first of mapping pixels from all the frames at once into a feature space such that pixels corresponding to the same color patch appearing in successive frames will be mapped to close neighbors in feature space *even if the patch moves rapidly*. Clustering is performed in feature space using a more efficient version of mean shift analysis, *hierarchical mean shift*, and the centers of the clusters directly define what colors and motion should be assigned to the pixels that contributed to the clusters. The result is a color segmentation of the video stack, and a *motion segmentation*.

In Section 2, we describe our technique for mapping pixels to points of a feature space. In Section 3, we provide an overview of the complete algorithm for clustering the mapped points, and for using the clusters to consistently label the pixels corresponding to moving patches. In Section 4, we introduce hierarchical mean shift analysis, and we present results that support our claim that this algorithm produces a significant computational improvement. Section 5 presents segmentation results and outlines how the space time segmentation can be used to extract concise representations suitable for video retrieval. Finally, Section 6 describes the application of hierarchical mean shift to hierarchical representations of video.

2. PIXEL MAPPING INTO FEATURE SPACE

Consider the pixel $P_t = (t, x, y)$ at frame t and position (x, y) belonging to a color patch (Fig. 1). 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). Therefore, 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 motion vectors $(1, u, v)$ for pixels of 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 for each projection we define an angle in degrees with respect to the plane (x, y). These angles are called α_x and α_y (see Fig. 1):

$$\alpha_x = 90 - \frac{180}{\pi} \arctan u; \quad \alpha_y = 90 - \frac{180}{\pi} \arctan v \qquad (1)$$

When the color patch does not move, both angles are 90 degrees. Angles approach 0 or 180 degrees only if motions are very fast. 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 patch trajectories over several frames.

Not only are the 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. 1). For each pixel P_i, we project the supporting line of its motion vector on planes (t, x) and (t, y) and obtain two lines L_x and L_y. Having defined the point O at the center of the video stack and the two projections O_x and O_y of O onto these planes, we can compute the distance D_x of 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 as follows

$$D_x = (x - x_{max}/2) \sin \alpha_x - (t - t_{max}/2) \cos \alpha_x \quad (2)$$
$$D_y = (y - y_{max}/2) \sin \alpha_y - (t - t_{max}/2) \cos \alpha_y \quad (3)$$

where t_{max}, x_{max} and y_{max} are the dimensions of the video stack. In the following, D_x and D_y are called the *motion distances* for pixel P. Motion distances present the advantage of defining the positions of pixels in the video stack in a way that is approximately invariant to the motion shifts of the pixels (using x and y as position features of pixels would provide less compact clusters for pixels of moving patches).

3. ALGORITHM FOR SPACE-TIME SEGMENTATION

At each pixel of a video stack, seven components are defined, two motion angles, two motion distances, and three color parameters. The CIE L*u*v* color space is used for the color components so that small distances along the color dimensions tend to correspond to perceptually similar colors [6]. We can interpret these quantities as feature components; they define a feature vector which can be represented as a point in feature space. Since the components are approximate invariants for pixels of 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 will allow us to detect and segment pixels belonging to color patches evolving through time.

Our approach to space-time segmentation is illustrated by Fig. 2: (1) map pixels to points in feature space, (2) determine the clusters in feature space (Section 4), (3) assign to each point the label of the cluster to which belongs, and assign to each pixel of the

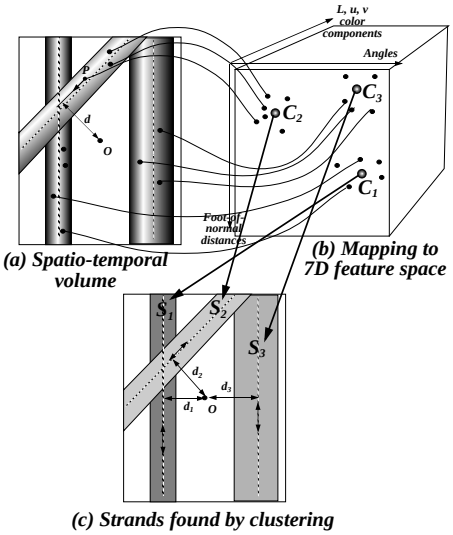


Fig. 2. Mapping process between pixels and feature space points, and inverse mapping to obtain segmented regions and video strands.

video stack the label 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 label across all the frames of a video stack. Therefore they are *tracked* from frame to frame, in the sense that given one frame and color patches, these patches have labels in that frame, and we can find their positions in the next frames as the patches with the same labels. See for example Fig. 6 showing color patches being assigned the same label from frame to frame.

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 obtain a color segmentation of the video stack by replacing the color of each pixel by the color of the cluster to which it is assigned (Fig. 2(c)). Similarly, we can obtain a motion segmentation by assigning to each pixel the motion of its cluster. In addition, we can concisely describe a video clip by its set of cluster centers, whose components describe average characteristics of the video strands.

4. CLUSTERING BY HIERARCHICAL MEAN SHIFT ANALYSIS

Mean shift analysis is a relatively new clustering approach originally advocated by Fukunaga [12], and recently extended and brought to the attention of the image analysis community by Yizong Cheng [9], and then by Comaniciu and Meer [6, 7] who convincingly applied it to image segmentation and frame-by-frame tracking. Refer to these references for details, and to Fig. 3 for a reminder of the principle of the method. Leung et al. elegantly prove ([16], 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 ascent on the density estimation. By contrast with the classical K-means approach, the clusters that are found are separated by

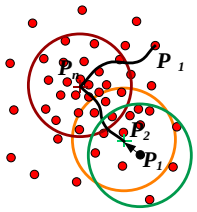


Fig. 3. Principle of mean shift analysis: To find cluster center for point P_1, repeatedly find centroid of points inside a sphere (initially at P_1) and recenter sphere on centroid, until sphere is stationary. (For Gaussian-kernel mean shift analysis, points further from sphere centers are given exponentially decreasing weights in the centroid calculation.) This is an adaptive gradient ascent in the space of point densities.

valleys of point densities, not by artificially defined hyperplanes at equal distance between the cluster centers. 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 consists of finding points within range R of sphere centers. What is needed is an efficient *range search* algorithm. For this task we use two functions, *nn_prepare* and *range_search*, from the well-written *TSTool* package created by Merkwirth et al. [18]. The auxiliary function *nn_prepare* arranges the background point set into a binary tree structure called an *ATRIA tree*. A set of points is divided into two subsets by the hyperplane half way between the two furthest points of the set. For each subset, the center C and the enclosing radius r are stored. This is repeated recursively for each subset until the subset of a branch contains less than a preset number of points. During a range search of 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 range R of A, and are pruned.

However, there is a major obstacle to using a tree structure efficiently with standard mean shift analysis: in order to produce a small number of clusters, mean shift has to be run with a *large radius*, typically up to 1/5 of the span of the largest feature component. However, for range search utilizing a tree structure such as the binary ATRIA 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 $O(N^2)$, because most of the branches of the tree must then be explored. We have adopted a hierarchical mean shift approach to circumvent this problem:

- We first run a standard mean shift to completion with a very small sphere radius, starting from all points of the data set and shifting 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 one. Spheres from several starting points typically converge to the same cluster center, and these points are considered members of the corresponding cluster.
- We assign to these cluster centers *weights* equal to the sum of the weights of the member points.

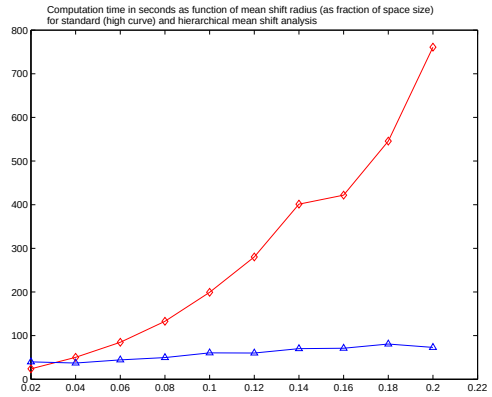


Fig. 4. Computation time in seconds as a function of mean shift radius for standard mean shift (higher curve, diamond plots) and for hierarchical mean shift (lower curve, triangle plots). The radius is expressed as a fraction of the feature space size, and is increased to a maximum of 0.2

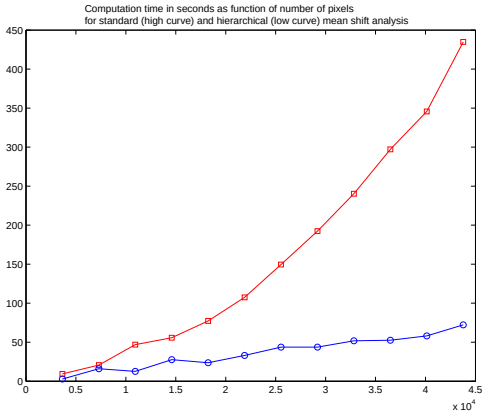


Fig. 5. Computation time in seconds as a function of number of pixels for standard mean shift (higher curve, square plots) and for hierarchical mean shift (lower curve, round plots).

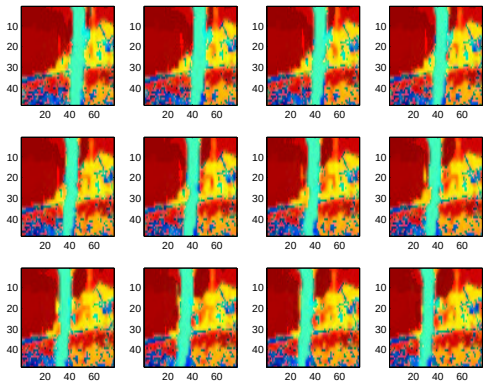
- We consider the new cloud of points composed of cluster centers. We recompute a new binary tree. We run mean shift using range search with a larger radius that is a small multiple of the previous radius (we have used a multiplying factor of 1.25 or 1.5). In the centroid computations, the weight of each point is used.
- We repeat the previous two steps until the desired radius size (or the desired number of large regions) is reached.

It turns out that essentially the same method was discovered by Leung et al. with their *clustering by scale-space filtering* approach (see [16], p. 1400, Eq. 22).

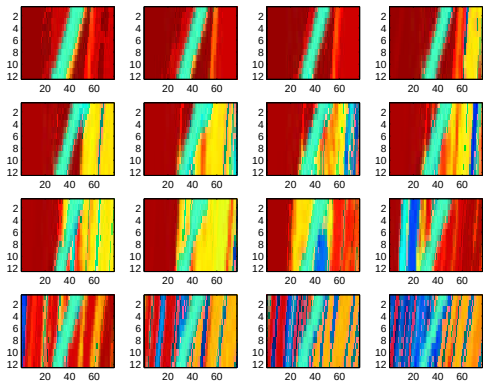
Qualitatively, the segmentation obtained by this technique looks as good or better than with the standard mean shift. As importantly, a significant speedup is achieved with this method. The reason is that the initial tree handles a very large number N of points, but allows efficient range search because the radius of the range search is small. At subsequent passes, 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.

The poor performance of standard mean shift with large radii and the effectiveness of the hierarchical mean shift solution are illustrated in Fig. 4. In this figure, computation time is plotted as a function of the mean shift radius, expressed as a fraction of the data hypercube size, for standard mean shift (diamond plots) and for hierarchical mean shift (triangle plots), for the space-time segmentation of 12 frames of a video, corresponding to 45,000 pixels. The computing time is almost independent of the mean shift radius for hierarchical mean shift, whereas it grows at a rate faster than polynomial for standard mean shift.

Next, we kept the mean shift radius (i.e., the final radius in the hierarchical mean shift algorithm) constant and equal to 1/6 of the data hypercube size. We repeated space-time segmentation experiments for a video stack containing an increasing number of frames, from 1 to 12, with the corresponding number of pixels increas-



(a)



(b)

Fig. 6. Consistent labeling of regions, shown with false colors on 12 consecutive frames (a), and shown in 12 cross-sections of the video stack along row and time dimensions (b). Note that most of the pixels of the tree trunk have been given a single label.

ing from 4,000 to 45,000. Results for standard mean shift (square plots) and for hierarchical mean shift (round plots) are plotted in Fig. 5. For the largest number of pixels, standard mean shift is almost 10 times slower than hierarchical mean shift. An estimate of the slopes of the log-log versions of the two curves shows that for standard mean shift the computation time increases as a power 1.8 of the number of pixels, i.e. almost quadratically, while for hierarchical mean shift, computation time increases as a power 1.1 of the number of pixel, i.e. almost linearly. The computing time *per frame* for hierarchical mean shift remains *sensibly constant* when the number of frames is increased (around 6 seconds per frame in the experiments shown in Fig. 5), whereas standard mean shift becomes impractically slow.

5. RESULTS

We provide examples of video segmentation for the flower garden sequence. A list of 7D feature vectors was generated, one per pixel of a video stack of twelve 88×60 frames, with two motion angles computed from optical flow over 20 frames by the Lucas-Kanade method [17], two motion distances computed by the method of Section 2, and three L*u*v color components. The motion components were offset and scaled to ranges 0-0.8, and distance components to ranges 0-0.6, while the color components were scaled to ranges 0-1: instead of using ellipsoids in mean shift, we always use spheres in order to take advantage of the range search machinery, but squeeze the data world along the components that are less important. Motion importance is decreased because it is less reliable than color, and distance importance is decreased even more to avoid subdividing spatially elongated regions. The final radius in the hierarchical mean shift analysis was 1/6, the initial radius was 1/60, and the radius multiplying factor was 1.5 at each pass. Furthermore, we run the segmentation twice, in order to use in the second pass the cleaner segmented optical flow obtained by the first pass. This leads to qualitative improvements in segmentation results. The code is written in Matlab, with bottlenecks rewritten as C functions with Matlab interfaces. The TSTool functions *nn_prepare* and *range_search* used for computing range search [18] are also written in C with Matlab interfaces. The total processing time per frame was around 15 seconds.

Fig. 6 shows space-time volumes labeled by this algorithm in frames and cross-sections of the video stack in false color. These volumes are obtained by giving a unique label to each cluster obtained by hierarchical mean shift, and assigning these labels to the pixels that were mapped to these clusters. Note that the tree trunk is consistently labeled across the stack, and that the house regions have the same label on the left and right of the tree; this segmentation can accommodate to a certain extent both spatial and temporal occlusions, because of a "Hough transform effect": regions that are disjoint in pixel space can be neighbors in feature space.

Figure 7 demonstrates motion segmentation. Each cluster center has seven dimensions, including two motion angle components. In the figure, the motion angle of each cluster center in the x direction was assigned to all the pixels that contributed to the corresponding cluster. Faster lateral motion is coded with a lighter color. Because the camera is translating, the images are in fact depth maps of the scene. Note that the patch of sky on the top left corner of the frames was coded closer than the rest of the background because texture was generated by hanging branches of the tree. Color segmentation (not shown here) is similarly obtained by transferring color components of cluster centers to their contributing pixels.

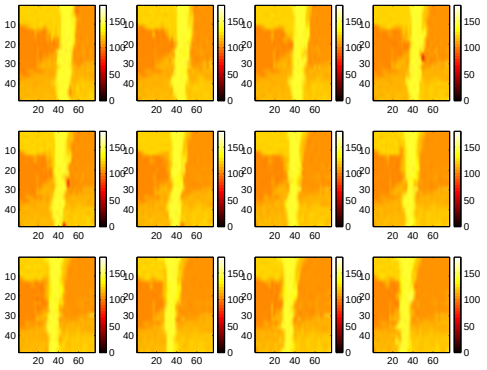


Fig. 7. Motion segmentation of video. Color coding showing values of angles of patch trajectories in the x direction from 0 to 180. These angles increase as distance decreases because the camera translates in the x direction, so lighter colors correspond to closer range.

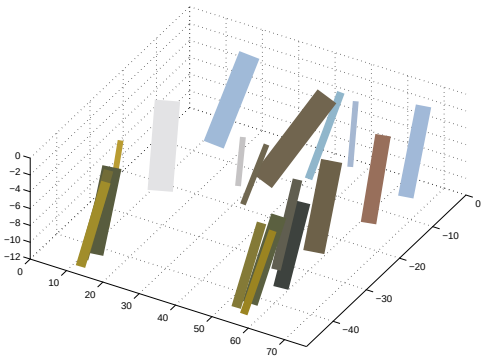


Fig. 8. Representation of flower garden sequence as a set of lines with specific thicknesses (video strands). The vertical dimension is time. The thick brown line with the highest slope corresponds to the tree trunk. The blue line on the top left corresponds to a patch of blue sky. It is tilted as it contains tree branches that seem to move because of camera translation. Further background lines are almost vertical.

The seven dimensions of each cluster center in feature space describe the average color, position and orientation of a color region. These components describe the geometry and color of a straight line in the video stack. The set of lines corresponding to regions with more than 20 pixels in average in the frames that they occupy is represented in Fig. 8. This is what we call a video braid representation. This representation can describe one MB of a half-second video with 200 bytes, which can be used as indexing description. We are investigating retrieval techniques using this description.

6. HIERARCHICAL SEGMENTATION FROM HIERARCHICAL MEAN SHIFT

Hierarchical mean shift produces a hierarchical segmentation that can be represented as a tree structure. At each pass of the procedure, clusters are merged into new clusters. Each cluster represents a region of the video block, and regions corresponding to

new clusters are groupings of regions corresponding to clusters of the previous pass. There is a fine-to-coarse evolution of the segmentation occurring from pass to pass. With a large enough radius, we obtain a single region corresponding to the whole set of pixels. We can generate tree representations of videos by examining the sequence of events in the reverse order from which it was produced, i.e. from coarse to fine. We are developing applications of this representation to video compression.

7. CONCLUSIONS

We have described a general algorithm for the space-time segmentation of video sequences. Our contributions to the problem of space-time segmentation are the following

1. Taking cues from the Hough transform, we define pixel positions by normal distances from a fixed point to lines of motions, which are approximately invariant for pixels belonging to the same moving patch.
2. Hierarchical mean shift analysis is of lower empirical complexity than standard mean shift analysis for useful radii when range search is optimized with a binary tree structure.
3. Regions are simultaneously segmented and tracked by the same mechanism; a single parameter is specified by the user.
4. Cluster centers are compact descriptors characterizing the video strands and are useful for video indexing and retrieval.

We have several areas of further study in mind. First, the boundaries of moving regions are jagged, and arguably do not look as good as those produced by, e.g., color-texture segmentation [10]. (Note, however, that segmentation code generally includes a post-processing phase which merges small regions to larger neighbors and smoothes boundaries, while we wanted to show how far our hierarchical mean shift approach can go all by itself.) Cleaner boundaries could be obtained if motion components of the feature vector are given lower weight with respect to color components in low-confidence motion field regions. Second, in the spirit of scale-space analysis, we can analyze our hierarchical segmentation to discover which regions remain stable through increasing mean shift radii and give preference to these regions in the segmented output, as suggested by Leung et al. [16]; then there would be no parameter left to specify. Third, we need to develop efficient ways of using video strands and hierarchical segmentation for indexing and retrieval of large video data sets and for compression.

8. REFERENCES

[1] E.H. Adelson and J.R. Bergen, "Spatiotemporal Energy Models for the Perception of Motion", J. Opt. Soc. Am. A. 2(2), pp. 284–299, 1985.

[2] Z. Aghbari, K. Kaneko and A. Makinouchi, "Modeling and Querying Videos by Content Trajectories", IEEE Int. Conference on Multimedia and Expo (ICME2000), New York, July 2000.

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

[4] M. Black, "Combining Intensity and Motion for Incremental Segmentation and Tracking over Long Image Sequences", ECCV 1992, pp. 485–493.

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

[6] D. Comaniciu and P. Meer, "Mean Shift Analysis and Applications", IEEE Int. Conf. Computer Vision, Kerkyra, Greece, pp. 1197-1203, 1999.

[7] D. Comaniciu and P. Meer, Distribution Free Decomposition of Multivariate Data, Pattern Analysis and Applications, Vol. 2, pp. 22-30, 1999.

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

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

[10] Y. Deng and B.S. Manjunath, "Unsupervised Segmentation of Color-Texture Regions in Images and Video", IEEE Trans. on PAMI, vol. 23, no. 8, pp. 800–810, 2001.

[11] C. Fowlkes, S. Bellongie and J. Malik, "Efficient Spatiotemporal Grouping using the Nystrom Method", CVPR 2001, Kaui, pp. I-231–238 , December 2001.

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

[13] Q. Ke and T. Kanade, "A Subspace Approach to Layer Extraction", CVPR 2001, Kaui, pp. I-255–262 , December 2001.

[14] N. Jojic and B. Frey, "Learning Flexible Sprites in Video Layers", CVPR 2001, Kaui, pp. I-199–206 , December 2001.

[15] S. Khan and M. Shah, "Object Based Segmentation of Video Using Color, Motion and Spatial Information", CVPR 2001, Kaui, pp. II-746–751, December 2001.

[16] Y. Leung, J-S. Zhang and Z-B. Xu, "Clustering by Scale-Space Filtering", IEEE Trans. on PAMI, vol. 22, no. 12, pp. 1396–1410, 2000.

[17] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", Proc. DARPA Image Understanding Workshop, pp. 121–130, 1981.

[18] 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/

[19] F. Moscheni, S. Bhattacharjee and M. Kunt, "Spatiotemporal Segmentation Based on Region Merging", IEEE Trans. on PAMI, vol. 20, no. 9, pp. 897–915, 1998.

[20] R. C. Nelson, "Finding Line Segments by Stick Growing", IEEE Transactions on PAMI, Vol. 16, 5, May 1994, 519-523.

[21] J. Shi and J. Malik, "Motion Segmentation and Tracking using Normalized Cuts", IJCV 98, Bombay, India, January 1998.

[22] T. Syeda-Mahmood, A. Vasilescu and S. Sethi, "Recognizing Action Events from Multiple Viewpoints", Proc. IEEE Workshop on Detection and Recognition of Events in Video, Vancouver, Canada, July 2001.

[23] W.B. Thompson, "Combining Motion and Contrast for Segmentation", IEEE Trans. on PAMI, pp. 543–549, 1980.

[24] J. Y. A. Wang and E. A. Adelson, "Representing Moving Images with Layers", IEEE Trans. on Image Processing, 3, pp. 625-638, Sept. 1994.

[25] D. Zhong and S-F. Chang, "Spatio-Temporal Video Search Using the Object Based Video Representation", Proc. ICIP, Santa Barbara, CA, October 1997.