Multimedia Tools and Applications, 26, 327–344, 2005

Adaptive Video Fast Forward

NEMANJA PETROVIC nemanja@ifp.uiuc.edu
Beckman Institute, University of Illinois, 405 N Mathews Avenue, Urbana, IL 61801

NEBOJSA JOJIC jojic@microsoft.com
Researcher, Microsoft Research, One Microsoft Way, Redmond, WA 98052

THOMAS S. HUANG huang@ifp.uiuc.edu
Beckman Institute, University of Illinois, 405 N Mathews Avenue, Urbana, IL 61801

Abstract. We derive a statistical graphical model of video scenes with multiple, possibly occluded objects that can be efficiently used for tasks related to video search, browsing and retrieval. The model is trained on query (target) clip selected by the user. Shot retrieval process is based on the likelihood of a video frame under generative model. Instead of using a combination of weighted Euclidean distances as a shot similarity measure, the likelihood model automatically separates and balances various causes of variability in video, including occlusion, appearance change and motion. Thus, we overcome tedious and complex user interventions required in previous studies. We use the model in the adaptive video forward application that adapts video playback speed to the likelihood of the data. The similarity measure of each candidate clip to the target clip defines the playback speed. Given a query, the video is played at a higher speed as long as video content has low likelihood, and when frames similar to the query clip start to come in, the video playback rate drops. Set of experiments o12n typical home videos demonstrate performance, easiness and utility of our application.

Keywords: content-based retrieval, generative models, video fast forward

1. Introduction

Although the issues around image and video databases, including browsing, search and retrieval, have been studied extensively over the last few years, the area is being reinvigorated by the apparent shift in the targeted user category. Most of the previous work has, perhaps inadvertently, been oriented towards a professional user or an aficionado interested in searching over the specialized databases locally or over the web. Today it finally seems that every family is likely to soon start using media browsing and search tools either on their PC while consuming their own personal media (photos and videos) or on a digital TV while fast-forwarding live video, for example.

Early Content Based Image Retrieval (CBIR) systems invited the user into the loop by asking the user to provide a feature-weighting scheme for each retrieval task. This proved to be too technical and a formidable burden on the users side. A more natural and friendlier way of getting user in the loop is to ask the user to give feedbacks regarding the relevance of the current outputs of the system. This approach, while useful for browsing image databases, proved not mature enough for browsing large video content. Another drawback is that the amount of analyst's involvement is still excessive and the method is far from autonomous video content searching.

This changes slightly the design of user interfaces for the media browsing tools, and this in turns places new expectations on the media analysis technology, while perhaps alleviating some of the problems addressed in the past. For example, in a fairly elegant and comprehensive search engine described in [1], one of the components allows for a motion-based search based on a query consisting of region appearances and sketched motion patterns. While compelling and useful for a professional user searching over soccer games, for example, for a typical user of home DV cameras, this type of search seems to still require too much user input while the actual functionality may not even be required. Home videos tend to be organized around people and scenes and much less around actions (e.g., as in [14]), at least not to the extent that actions cannot be easily found by searching for scenes and people. The other issue for a home user is that the material being searched over is truly small, e.g., an hour of a professional baseball game provided through cable or, in the future, downloaded from the web, or an hour or two of video filmed during a family vacation. This introduces the need for fast querying and makes complex strategies that require entering a number of positive and negative examples as Diverse Density Algorithm [10] less desirable. On the positive side, the content variation within the piece of media being looked at a time is smaller, but on the negative side, the diversity of possible media that the tool should be useful for is huge.

The core idea behind adaptive video fast forward is depicted in figure 1. User is required to provide the query sequence of interest, usually a few second long video clip from the database where similar clips are searched. Based on the input sequence and the statistical generative scene model, model parameters are learned in an unsupervised fashion. In the testing phase, all frames in the candidate scene are tested against the model. Video playback rate speed is roughly inversely proportional to the frame likelihood under the generative model: high likelihood indicates high degree of similarity between the target sequence and the query sequence, and the playback rate drops. Low likelihood indicates dissimilarity between target and query sequence, and consequently the playback rate surges. In other words, playback speed adapts to the content of the video.

The rest of the paper is organized as follows. In Sec.řefsec:previouswork we address current state of the art methods and their limitations. In Sec.řefsec:genmodel we introduce graphical model for pixel generation in video scene that can be used as the statistical measure

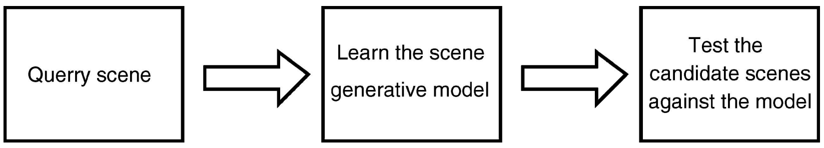

Querry scene
Learn the scene generative model
Test the candidate scenes against the model

Figure 1. Adaptive video fast forward. After the user specifies a query sequence, statistical generative scene model is learned in an unsupervised fashion. All frames from the candidate scenes are tested against the model. Video playback rate speed is controlled by the frame likelihood under the generative model. High likelihood indicates high degree of similarity between the target sequence and the query sequence, and the playback rate drops. Low likelihood indicates dissimilarity between target and query sequence, and consequently the playback rate surges. Playback speed adapts to the content of the video.

of video clip similarity. We show results in Sec.řefsec:experiments. Finally, we summarize and outline future directions in Sec.řefsec:conclusions.

2. Previous work

The most appealing in terms of simplicity and computational efficiency are the content-based search engines that use various types of aggregate statistics over visual features, such as color or texture elements (e.g. [18]). These tend to be sensitive to the quality of the data. A home video usually has bad color characteristics, especially around the edges of the frame, and could be blurry at places making it difficult to recognize texture. However, a more serious limitation is that the spatial configuration of the scene is not encoded in the scene description. In order to preserve some of the spatial information, a very interesting use of multiresolution color histograms has recently been proposed [4], although results are preliminary. Another approach to circumvent the lack of global spatial information in the representations based on local features is to work with a large number of features and select automatically the most discriminative ones [2, 19].

The approaches that attempt to model the spatial layout of the interesting regions, and not just the low-level region characteristics include [1, 5, 10, 12]. The limitations of these methods include the amount of user interaction in specifying positive and negative examples [10], the small size of foreground objects modelled [5] limiting the application domain, and the handcrafted cost functions that need to be weighted by hand [1, 12]. Recently, some interesting preliminary results were presented on modelling jointly motion and appearance by using derivatives in the space-time volume [21]. The applicability of their method pends, though, on motion compensation and background substraction routines.

The target user for this type of systems is a casual user that browses generalized class of video databases. Therefore, learning has to be unsupervised, and it has to minimize the amount of user attention and intervention. Moreover, intuitive user interfaces are necessary to navigate the user and mimic the structure of the model. We believe that generative graphical models are well suited to reconcile these requirements. They are powerful in explaining complex process of video scene creation that involves multiple, mutually interacting video objects and scenes, further complicated by camera motion, changes in lighting, occlusion, etc. Bayesian networks [13] were widely used for different levels of video understanding tasks. However their primary role was to infer the relations between some low level features (color histogram, edge map, texture, audio spectral coefficients, etc.) and high level semantic concepts (indoor/outdoor scene, highlight of baseball game, explosion, helicopter scene, etc.).

The drawback of this approach is that it simply tries to infer which low level features give rise to certain semantic concepts. There is no clear picture what features are good and how many of them should be used. The number of high level concepts is huge, and it becomes excessive once we take into account symbolic meanings and context. And there is still ambiguity since mapping between features and semantics is not 1-1. The hope was that Bayesian network itself will detect relations between low level features and high level semantic concepts and achieve good generalization. These models are not able to capture all variations of the features since there is a number of causes why the same scene appears differently, and those causes tend to be concurrent making the video creation process even more difficult. Other graphical models, like Hidden Markov Models (HMM) or Dynamic Bayesian Networks (DBN) are often used to capture temporal aspect of video, i.e., coherence between neighboring frames. Still, it is very difficult to explore semantic content from the raw video data without proper scene modelling.

We propose different strategy. We use a Bayesian network, but its role in this work is to model video generation process, rather than trying to bridge the gap between low level features and high level semantics. Instead on focusing on features and trying to infer the distribution over features that corresponds to semantic concept, we are modelling the causes of variability that change the features and try to infer high level semantic concept invariant of changes in the low level features.

Generative models are flexible graphical models, that require little effort to design and understand. They are excellent in capturing and separate object properties, sources of variability, and sources of correlations in a joint manner. In the case of videos, it translates to easiness of explaining the scene in terms of its components, including appearance and motion of various objects. Minimal number of manually adjustable parameters makes the retrieval results reproducible. Finally, there is a number of tools for learning the parameters of models or inferring the values of hidden variables, including exact EM algorithm [3], approximate (variational) EM algorithm [9], loopy probability propagation, or Markov Chain Monte Carlo (Gibbs sampler) [16].

While we will use simple color features (R, G and B color channels), Stauffer et al. illustrate an interesting alternative approach to modelling appearance [17], although so far tested only on a limited domain. Our spatial layout models could in principle use such appearance models. All of these methods attempt to bypass a complete computer vision solution and try to work with low level features. What we mean is that techniques are not aware of the real structure of the scene, such as mutual occlusions, changes due to lighting, object deformations, etc. In this way it is attempted to make a considerably impoverished representation of the video signal, without truly trying to understand objects in the scene, variability in their appearance and relationship to one another.

We build this work on previous study [8]. In this paper, we will address the media browsing needs of a user and focus on fully adaptive content-based search, that does not require much user intervention, and tries to mimic the types of user interface the user is already used to, such as fast forwarding. At the core of our system is a scene generative model, figure 3, that describes the spatial layout of objects in a scene and can be used to compute the likelihood of the observed video as the cost on which we base our approach to video browsing, search and retrieval. The generative model explains the scene by separating it into detailed background model, and fuzzy foreground model represented as a mixture of Gaussians (blobs). The assumption is that short query segment is actually generated from a generative model with unknown parameters yet to be estimated from the query sequence.

3. Scene generative model

We define similarity measure between the frame and the model as the likelihood that given frame is generated from the statistical model, that is, in turn, learned from the query

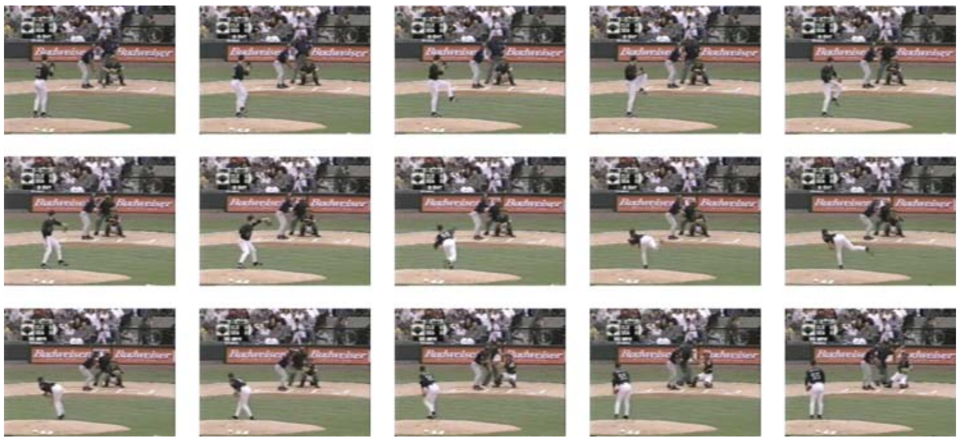


Figure 2. Fifteen frames from a five-second clip of pitching in a professional baseball game.

sequence. We model frame appearance as the composition of static detailed background image and blobs mimicking foreground motion. If for the moment we assume that the parameters of the model are given (background and blob images of figure 3, top row), then it is possible to sample (simulate) from this model. Typical image generated in this fashion (using exaggerated pixel sizes) looks like "artistic rendering" of baseball pitching scene (figure 3, down, center). Although not realistic, this synthetic image is generalized exemplar for the set of pitching frames (figure 2). It captures both the background scene, and the occlusion caused by the foreground objects.

We specify the generation of feature vectors $\mathbf{g}_c(i, j)$, where c is one of the C features modelled for each pixel at the location (i, j). These features, measured in the neighborhood around (i, j) could include color, texture, edge strengths or orientations. We will, initially, limit to R, G, and B color channels. The image features can be generated from several models indexed by s (see figure 3). One of these models ($s = 0$) generates each pixel with a separate mean and variance in each color channel, as in [6], while remaining models ($s > 0$) are blobs similar to [20]. The blobs are modelled as mixture of Gaussians with their spatial and color means and variances. We will refer to variable s as *segmentation*.

To put it simply, a pixel is modelled to either belong to the background (that is defined by its mean image and variance); or the pixel is sampled from the foreground blob mixture model (that is defined by spatial means and variances of blobs and theirs color means and variances). The pixel generation is assumed to start with the selection of the object model s (by drawing from the prior $p(s)$), followed by sampling from appropriate distributions over the space and color according to $p([i, j]|s)$ and $p(\mathbf{g}_c(i, j)|s, i, j)$.

Pixel position is chosen according to

$$\begin{aligned} &p([i, j]|s = 0) = \mathcal{U}(\text{uniform distribution}) \\ &p([i, j]|s > 0) = \mathcal{N}([i\ j]; \gamma_s, \Gamma_s), \end{aligned} \tag{1}$$

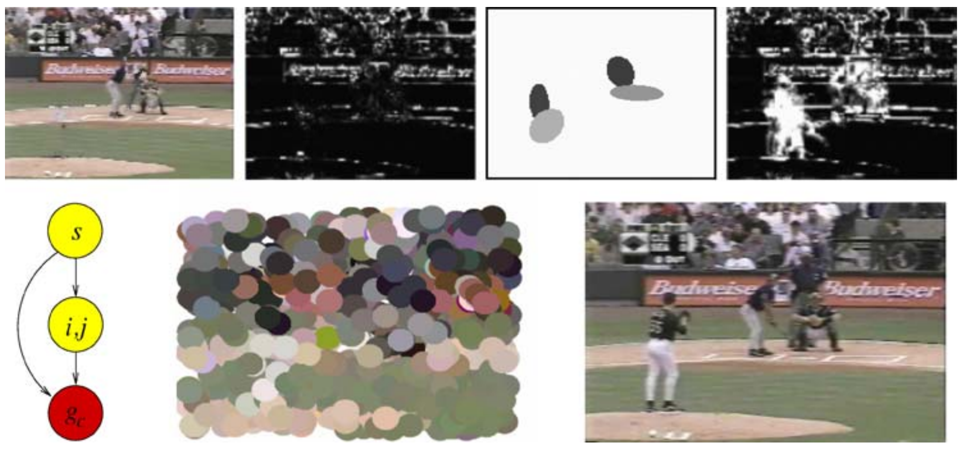


Figure 3. Top row: spatial layout of background and foreground objects learned from the clip. The blob model (second from the right) captures the foreground object and thus the pitcher is automatically removed from the background $\mu_{0,c}(i, j)$ (left). Note the difference in the variance of the learned background $\Phi_{0,c}(i, j)$ (second from the left) and the pixel variances in the clip (right). White denotes pixels with high variance of their features. Bottom row: generative model (left) consist of the class index s that indicates if the pixel belongs to background ($s = 0$) or sth blob ($s > 0$). Probability distribution of pixel position i, j depends on s (see text). Feature vector g_c for each pixel has probability distribution that depends both on class index and position. Unknown (hidden) variables are in yellow, actual pixel feature observed from a video frame are in red. This is a generative model for the generation of a single pixel in a single frame. This process repeats for all pixels and all frames in the query sequence. Typical result of sampling from given generative model (middle). This image is treated as the generalization of the whole class of natural images that depict a pitching scene (right).

where $\mathcal{N}$ denotes a Gaussian (normal) distribution. For the background model pixel position is arbitrarily chosen within the frame; for the foreground model, pixel position is more likely to be chosen around spatial center of the Gaussian blob with mean position γ_s, and position variance Γ_s.

Pixel feature vector (color) is sampled according to

$$\begin{aligned} p(\mathbf{g}_c|s = 0, i, j) &= \mathcal{N}(\mathbf{g}_c; \mu_{0,c}(i, j), \Phi_{0,c}(i, j)) \\ p(\mathbf{g}_c|s > 0) &= \mathcal{N}(\mathbf{g}_c; \mu_{s,c}, \Phi_{s,c}). \end{aligned} \tag{2}$$

Here, c indexes particular feature (color) in the color vector $\mathbf{g}_c$; $\mu_{0,c}(i, j)$ is mean of cth feature of the background pixel (i, j); $\Phi_{0,c}(i, j)$ is the variance of cth feature of the background pixel (i, j). cth feature of sth blob is denoted as $\mu_{s,c}$, and its variance as $\Phi_{s,c}$.

In figure 3 (top row), we show the mean and variance images $\mu_{0,c}(i, j)$, $\Phi_{0,c}(i, j)$ for the model $s = 0$, which usually captures the background of the scene. Variances are shown in gray intensities for easier viewing, although they are defined in color space. The blobs are illustrated by showing all pixels within a standard deviation along each principal component of Γ_s painted in the mean color $\mu_{s,c}$. Although not shown in the figure, the blobs also have their color covariance matrix $\Phi_{s,c}$. The images are assumed to be generated by repeating this sampling process K times, where K is the number of pixels, and a video sequence is generated by repeating the image generation T times, where T is the number of frames. The generative model creates a cloud of points in the space-time volume that in the case of the real video clips just happen to fill up.

Detailed pixel model $s = 0$ is focusing on the unchanging background captured in $\mu_{0,c}$. The changes in the appearance of the background can be well captured in the variance $\Phi_{0,c}$. However, the blob parameters γ_s, Γ_s can either be fixed throughout the sequence or allowed to change, thus tracking the objects not modelled by the overall scene model $\mu_{c,0}(i, j)$, $\Phi_{c,0}(i, j)$. A good strategy is to keep the blob spatial variances Γ_s fixed, regularizing the size of the object captured by the model, while letting γ_s vary through time, allowing the object to move without changing drastically its size. Note that due to the statistical nature of the blob's spatial model, it can always vary its size, but keeping the variances fixed limits the amount of the change. In this version of the model, blob centers γ_s become another set of hidden variables.

The joint likelihood over all observed and unobserved variables is

$$p(\{\{s_{k,t}, i_{k,t}, j_{k,t}\}_{k=1,\dots,K}, \gamma_{s,t}\}_{t=1,\dots,T})$$

where again, k is the index of one of K pixels per frame, t is the index of one of T frames, and c is the index of particular pixel feature. Joint likelihood can be expressed as the product of the appropriate terms in Eqs. (1) and (2) for all k, t. The joint likelihood is a function of the model parameters Θ that include the spatial covariance matrices of the blobs Γ_s, blob color distribution parameters $\mu_{s,c}$, $\Phi_{s,c}$, the scene background model $\mu_{0,c}(i, j)$, $\Phi_{0,c}(i, j)$ and blob means γ_s. The task is to maximize the likelihood of the data $\mathbf{g}(i, j)$, with respect to model parameters. This task can be efficiently solved using approximative learning techniques.

To compute the likelihood of the data $g(i, j)$, all other hidden variables $\mathbf{h} = (s_{k,t}, i_{k,t}, j_{k,t}, \gamma_{s,t})$ need to be integrated out, which can be efficiently done with the help of an auxiliary function $q(\mathbf{h})$, that plays the role of an approximate or an exact posterior:

$$\begin{aligned}\log p(\mathbf{g}) &= \int_{\mathbf{h}} p(\mathbf{g}, \mathbf{h})\, \mathbf{dh} = \log \int_{\mathbf{h}} \mathbf{q(h)\; p(g, h)/q(h)} \\ &\geq \int_{\mathbf{h}} q(\mathbf{h})\, [\log p(\mathbf{g}, \mathbf{h}) - \log q(\mathbf{h})] = B(\Psi, \Theta)\end{aligned} \tag{3}$$

where Θ are the model parameters from Eqs. (1) and (2) and Ψ are the parameters of the auxiliary function q. The above bound is derived directly from Jensen's inequality. When q has the same form as the exact posterior $q(\mathbf{h}|\mathbf{g})$, the above inequality becomes equality and optimizing the bound B with respect to Ψ is equivalent to Bayesian inference. If a simplified form of the posterior q is used, then Ψ can still be optimized for, thus getting q as close to the true posterior as possible. In particular, we assume a factorized posterior with simple multinomial distributions. Thus,

$$q = \prod_t q(\gamma_{s,t}) \prod_{i,j} q(s_t) \tag{4}$$

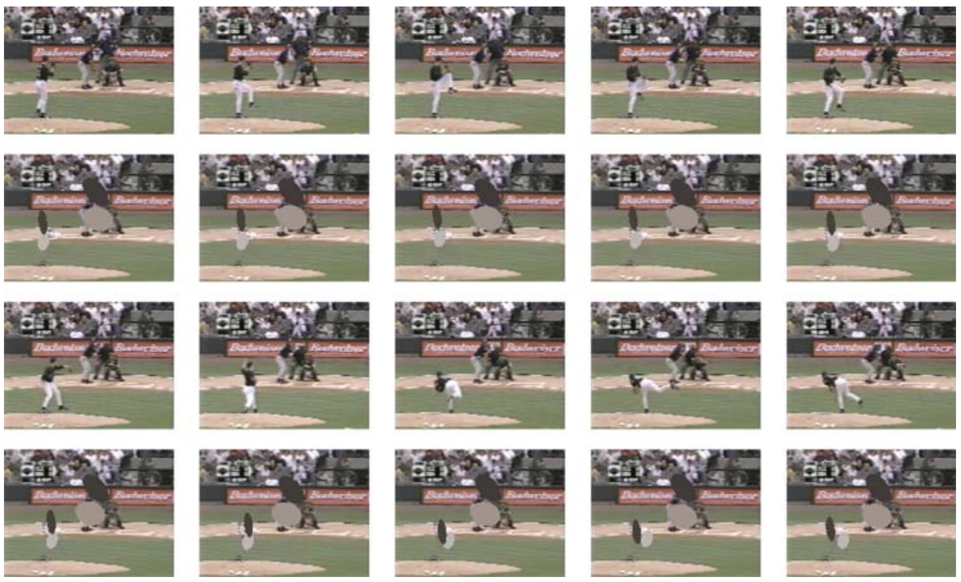


Figure 4. Tracking pitcher through frames. Each of 10 frames from original sequence is showed against synthetic image obtained by superimposing learned background image and inferred blob position $\gamma_{s,t}$ for blob s at time t. Note that our goal is not to obtain perfect tracking or segmentation, but rather model to design likelihood measure that is useful for detecting similar frames.

As already noted, to perform inference, we solve $\partial B/\partial\Psi = 0$, where Ψ are the parameters of the auxiliary function q and it includes the values of the discrete distributions $q(s_t(i, j))$ and $q(\gamma_{s,t})$. Note that set of parameters Ψ depends on t i.e. new set has to be calculated for each frame t. For example, in figure 4, we illustrate the results of the inference on $\gamma_{s,t}$ (tracking the mean of the blobs) using the model shown in figure 3.

To perform learning from a clip, we can alternate the bound optimizations with respect to the inference parameters and model parameters:

1. Initialize parameters randomly
2. Solve $\partial B/\partial\Psi = 0$, keeping Θ, t fixed; repeat for all frames t
3. Solve $\partial B/\partial\Theta = 0$, keeping Ψ fixed
4. Loop until convergence

For the model presented here, this procedure (the variational EM algorithm) is very efficient and usually converges in 10–20 iterations, while 1. and 2. above reduce to solving linear equations. In our experiments, model parameter for a 150-frame sequence, are learned in a few seconds. An example of a learned model is given in figure 3.

3.1. *Adaptive fast forward*

Our approach to intelligent video forwarding is based on using the frame or shot likelihood to control the playback speed. In less interesting parts of the video, the fast forward mechanism increases the speed, while slowing down when the likelihood of finding a clip of interest increases. This has the advantage of using a familiar interface to searching through video—the fast forward button—while the danger of fast forwarding over interesting content is reduced. In such a system, the user can still have the control over the fast forward speed, thus reducing the dependence on the automatic media analysis (see figure 10). If the fast forward speed at time t is denoted by V_t, interesting functional relationships between this speed and the likelihood include:

$$\begin{aligned} V_t &= r(\log p(f_t)) \\ V_t &= r(\log p(\{f_u\}_{u=t,\ldots,t+\Delta t})), \end{aligned} \tag{5}$$

where r is a monotone non-increasing function. The second form is useful when the likelihood model is defined on a shot and not on the clip as in the previous section, but it is also generally preferable because of its power to anticipate a change and gently change the speed around the boundaries of the interesting shots. This is especially useful if the user is trying to extract a clip from a larger video, in order to compile a shot collection of shots as a summary, for instance. Such a need is typical in personal media usage.

4. Experiments

4.1. *Scene generative model as a structured similarity measure*

The two extremes in the design of similarity measures for media search and retrieval are the use of very simple aggregate feature statistics and the use of complex, manually defined measures involving appearance, spatial layout and motion. Using a generative model that explains the scene in terms of its components, including appearance and motion of various objects has the advantage that while it stays simple to use it can still capture the various concurrent causes of variability in the sequence presented as a query. Both of these nice properties come from prescribing to the machine learning paradigm, in which the model adapts to the data, automatically balancing various causes of variability that the multimedia system designers are tempted to balance by presetting the weights for various parts of the cost function and specializing the system for a certain use or by shifting this burden to the user. In the process of integrating all hidden variables in order to come up with a single likelihood number, the frame is automatically broken into components and the similarity to the model is computed according to the learned amounts of variability in various parts of the model. However, the ultimate cost depends on how likely is the learned generative model to generate the new observed frame, and thus any of the multiple possible ways to explain the training data is usually satisfactory, as long as the structure of the model and the number of parameters are limited to avoid overtraining. In our case, the model structure mimics the structure of real world in terms of the existence of multiple objects possibly occluding each other and slightly changing shape, position and appearance.

In figure 3 (top row) we show the results of learning algorithm for the pitching sequence: mean and variance images $\mu_{0,c}(i, j)$, $\Phi_{0,c}(i, j)$ for the background model $s = 0$. The blobs are illustrated by showing all pixels within a standard deviation along each principal component of s painted in the mean color $\mu_{s,c}$ where c is in this case the color channel (R, G or B). Variances are shown in gray intensities for easier viewing, although they are defined in color space. Although not shown in the figure, the blobs also have their color covariance matrix $\Phi_{s,c}$.

Note that background subtraction was successful even for this simple model. Note also that blobs have position, orientation and color that corresponds to the foreground objects in the query clip (moving pitcher and batter). Blob position accurately tracks the positions of legs and torso of the player at the left. Although the model captures various causes of variability, the model's purpose is *not* to perform perfect segmentation or tracking, but rather to define likelihood useful for a search engine. Due to the structure of the model that describes various objects, their changing appearance, positions and shapes as well as the camera shake, the training usually results in a reasonable explanation of the scene that is useful for detecting similar scenes.

The performance of the system directly depends on the choice of threshold that disseminates similar from dissimilar clips. As a measure of accuracy of searching for similar clips we take type I and type II errors. We define type I error as the probability that the similar clip is played with the increased frame rate, i.e. likelihood falls below threshold. This corresponds to the case that the true hypothesis is rejected (false negative). Type II error is the probability that dissimilar clip is played with normal frame rate (clip has unusually high likelihood, although it is dissimilar to query). This error corresponds to the case when false hypothesis is accepted (false positive). Likelihood thresholds that are set high reject all clips, and vice versa. For certain threshold values both type I and II errors could be acceptably small.

Note that even when a clip is erroneously classified the consequences are controllable. One of the advantages is that even though the algorithm can sometimes miss the scene of interest, or focus on scene dissimilar to query the user has always the full control of the player and can change the playback speed. The user can also move threshold slider thus getting more or less, as desired, similar clips.

In the testing phase, we tracked the likelihood of the training sequence under the model. In figure 8 (left), we show how the retrieval error varies for various thresholds on the likelihood. Usually it was chosen correctly by the system by setting the threshold to be 20% below the minimum frame likelihood of the training set. Admittedly, this is not an optimal solution and some burden has to be shifted to the user. When the likelihood of current frame is higher than the threshold, frame is classified as being similar to query clip. Note that simplified model that uses only background image to explain query clip has much worse performance than our more sophisticated background/foreground model. Our model has higher discrimination power and allows for large range of thresholds that still yield small type I and II errors. Setting the probabilities of blobs to zero would reduce our model to background only model. In figure 8 (right) we show the ground truth and variations of the likelihood of this model over several minutes of the baseball video. The sudden changes in variable playback speed correctly correspond to onsets of similar ir dissimilar video segments.

Initially, background is set to the background of the data set. Blob spatial means are randomly chosen in the area of the high variance of the data. Blob color means are set to the grayscale value of 0.5 (on the 0-1 basis). Since the model is initialized randomly, or using the first order fit to the data perturbed by some noise, and since the above procedure guarantees improvement of the bound in each step (and thus convergence), but not he global optimality, a natural issue to raise is the one of sensitivity to initial conditions.

To illustrate various ways the model can balance the causes of variability to design a good likelihood-based measure of similarity, in figure 5 we show two models of boogie-boarding learned from different initial conditions. The first model uses the detailed pixel-wise model $s = 0$ to capture the wave's foam and two blue blobs to adjust the appearance of the ocean in the absence of the foam. The third blob is modelling the boogie-boarder. On the other

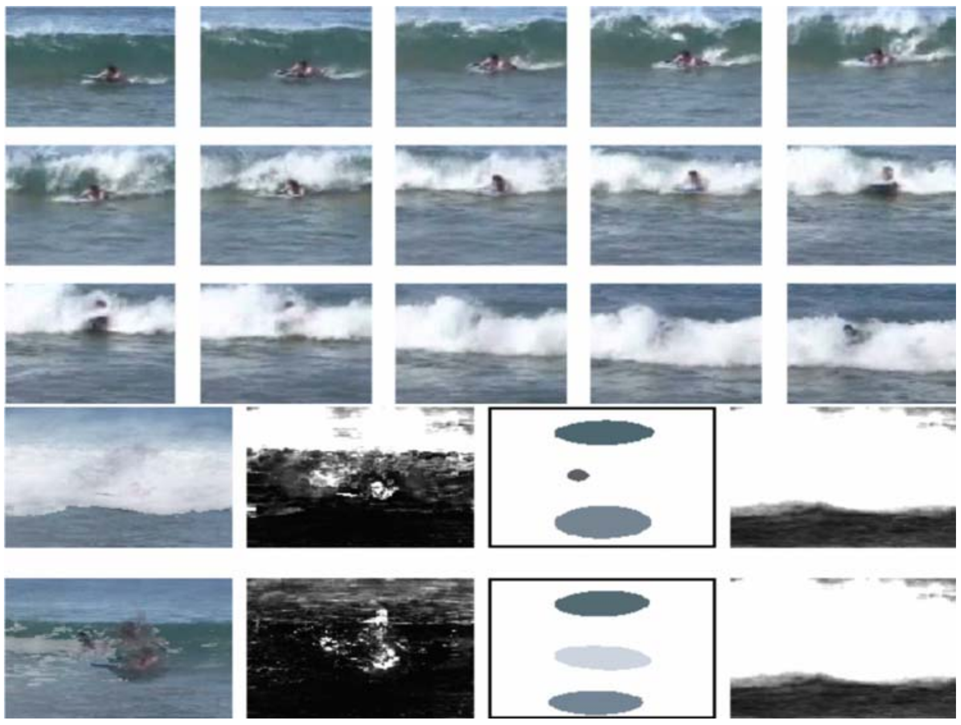


Figure 5. Robustness to local maxima issues: For a surfing shot, whose some frames are shown above, two different initialization conditions resulted in two different models. However, both models capture the variability in the data in a reasonable way and have created similar generalizations. They are also equally effective in search for similar sequences. The first model uses a blob to represent a surfer, while the second treats the surfer as a part of the more detailed model for $s = 0$, while a large white blob is used to model the foam of the crashing wave. See figures 6 and 7 for inference examples.

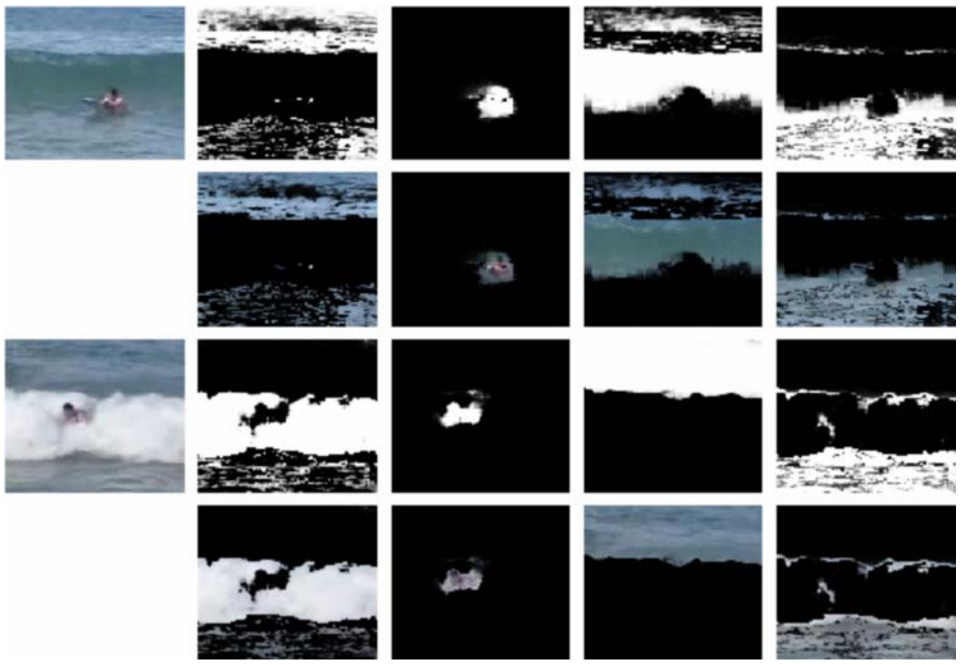


Figure 6. Inference using model 1 from figure 5 on two very different frames shown in the first column. The rest of the columns show segmentation, i.e., the posterior probability maps $q(s(i, j))$. The color images in every other row show the segmentation $q(s)$ multiplied with the frame to better show which regions were segmented.

hand, the second model has placed the boogie boarder against a foam-free ocean into the detailed appearance model $s = 0$, and uses the three blobs to model the foam as well as the darker and lighter blue regions of the ocean. In figures 6 and 7 we show how these two models understand two very different frames from the sequence. While the two models place objects into different components s, they are both coming up with a good fit to the data and a high likelihood for both frames. This indifference to a particular explanation of the data is in contrast to a more bottom-up approaches as in [12], where in order to compare two video segments, the system has to compare the two extracted structures, thus potentially failing if the system happens to extract the components in an inconsistent fashion.

4.2. Adaptive fast forward experiments

To test our models in realistic situations, we built a video browsing tool that we call Adaptive Fast-Forward, or AFF (figure 10). The application can play back video in the window and allows the user to move through video in several ways:

– by moving the time line slider
– by clicking on an index derived by standard shot detection techniques

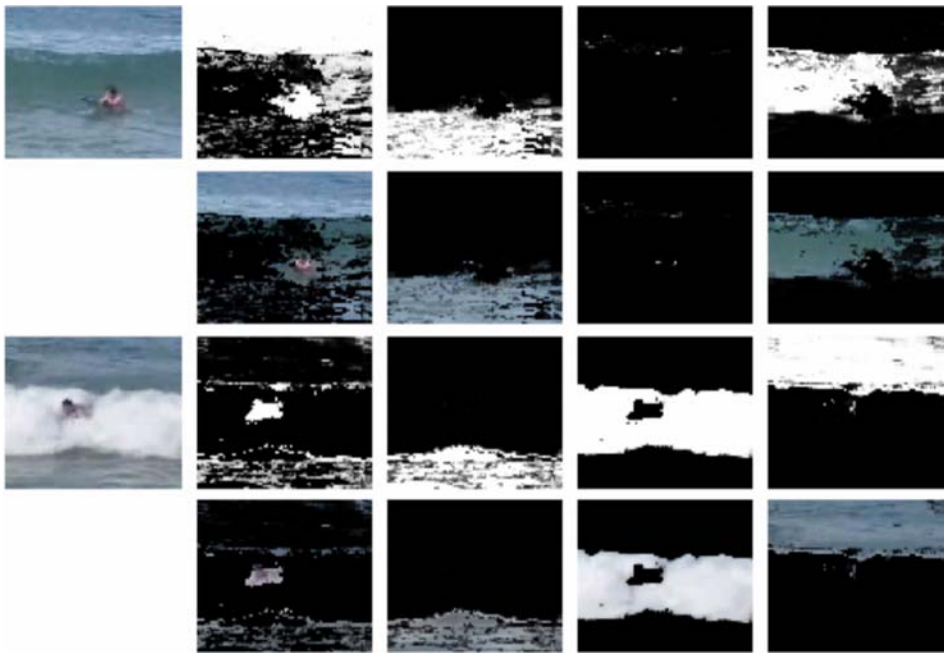
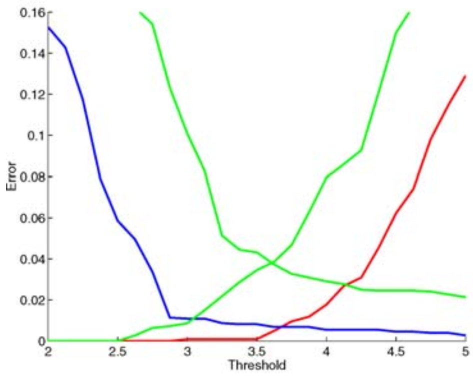


Figure 7. Inference using model 2 from figure 5 on two very different frames shown in the first column. The rest of the columns show segmentation, i.e., the posterior probability maps $q(s(i, j))$. The color images in every other row show the segmentation $q(s)$ multiplied with the frame to better show which regions were segmented.



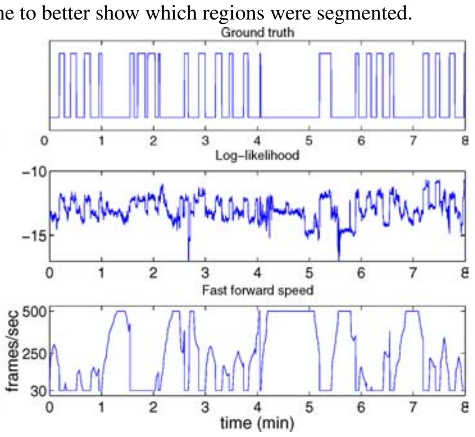


Figure 8. Left: Error plot for background only *vs*. background and blobs model for pitching sequence. Type I and II errors for background only model are marked in green. There is only a narrow range of threshold values that guarantees both errors less than acceptable 5%. Type I error for composite (background and blobs) model is marked in red, whereas type II error for the same model is marked in blue. This model allows comfortable range of threshold values that guarantees good performance. Threshold value is preadjusted by the system, and later, if necessary, fine tuned by the user. Right: Playback speed as the function of frame likelihood under the learned model. Note how playback speed mimics ground truth (presence of pitching frames) for baseball pitching sequence.

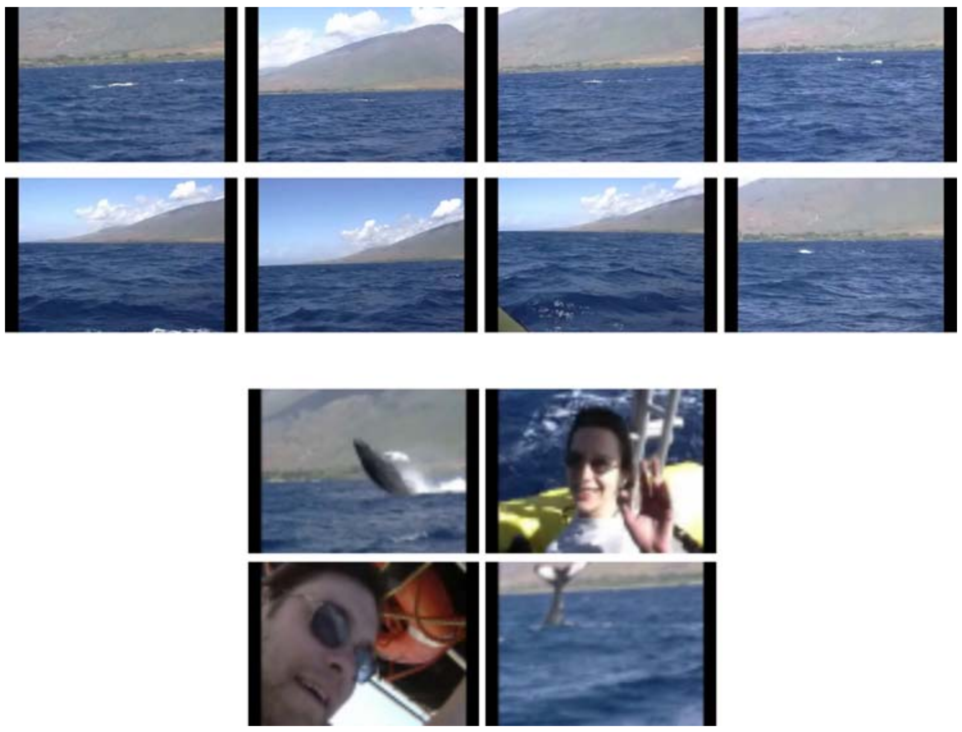


Figure 9. Top: Eight randomly sampled frames from the training clip (whale-watching sequence). Bottom: Representative frames from four short clips found unusual, i.e., different from the training set. The same user interface from figure 10 is used to search for rare events. In this case the logic is inverted: the system plays with high frame rate clips that are *similar* to query. User interface still provides control over playback speed and sensitivity threshold.

– by changing the speed of playback (speeding up) using the playback speed slider
– by specifying an interesting piece of video and letting the system do the variable speed fast-forward searching for similar sequences.

The first three ways are standard and most users are comfortable with them, but they are inefficient, for several reasons. Big jumps on the time-line are risky, since interesting content is easily skipped. The index is very useful but it typically reflects general categories in video and does not discriminate, for example, among various scenes in about 20 minutes of beach shots that include waves, sky, sand but also the boogie boarding shots that the user is really looking for. The index can be used to jump into the “beach part” of the two-hour vacation video, but the further search requires fast forwarding. Fast forwarding requires constant attention from the users, unless they can instruct the forwarding engine on what they are looking for. Thus, we allow the users to select a clip from the video as a query. The system then trains a scene generative model on the query and starts searching. During the search, the playback speed is increased well beyond 30 frames per second in parts that

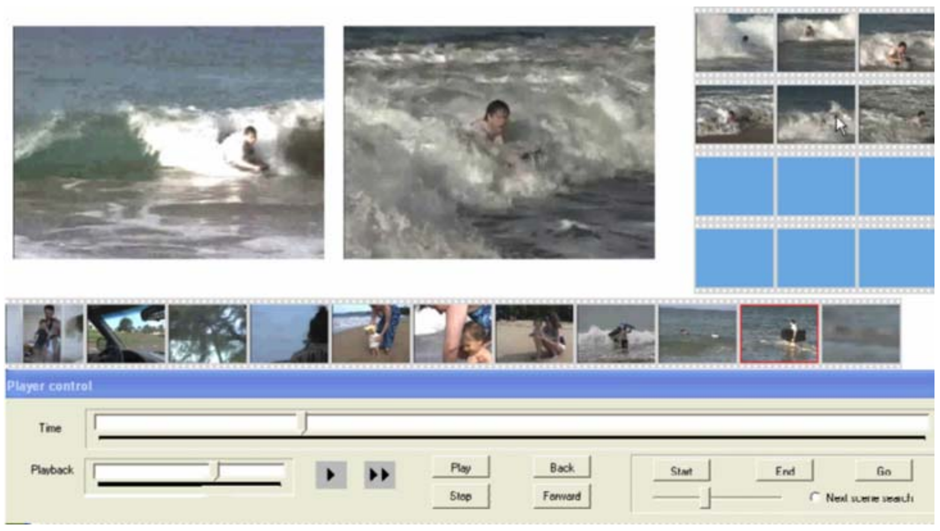


Figure 10. Adaptive fast forward application. Video is played in the window (upper right), and a typical frame from the query is displayed in the window (center, up). The set of thumbnails (upper right) point to the results of the search—clips similar to the query clip. The set of clickable thumbnails in the row below show a part of the index. The user has control over the time-line slider (‘Time’) and the playback speed (‘Playback’), as well as the threshold slider for tightening and loosening the search criteria (lower left).

are most dissimilar to the query and is dropped down to 30 frames when the part of the video is the most similar to the query (see also figure 8, right). When a new video segment similar to the query is found, its thumbnail is placed on the right part of the screen into the bag of found shots. The user can later watch these parts of the video again by clicking on the thumbnails. While the system is searching, the user still has the control over all sliders including the likelihood threshold slider that is used to tighten or loosen the search. The system automatically computes the threshold based on the least likely frame in the training sequence, but on very short queries the scene model can be over-trained in which case loosening the search helps.

We informally tested our system on several dozen users previously unfamiliar with the video content. The first task we tested our system on was browsing baseball games. For example, the user can effortlessly select any pitching example and let the system skip through all audience shots, coach close-ups, scoreboards and commercials and create the thumbnail library of pitching scenes. This is achieved with very high accuracy (figure 8). In the second study, we used AFF to browse through a two-hour vacation video. Every author of personal media needs a tool to quickly extract interesting content and create short clips convenient for sharing with others. Thus, we paid special attention to speeding up this process. We found the “thumbnailed” index of the time-line very useful for global navigation in video, but the greatest advantage of our tool for the users was in the case of looking for repetitive scenes, such as the dozen boogie boarding shots (figure 10). Creating a short, thirty second video of the beach was created in just a couple of minutes, although the beach material was 20 minutes long.

4.3. Detecting unusual events in video

If an event of interest is buried in a long segment of uninteresting material, the search strategy can be reversed and a single or mixed scene model can be trained on the typical video and the search criteria can then be defined in terms of finding video segments that are unlikely under the model.

In the whale-watching clip, for example, some great shots of whale breaching are buried in 13 minutes of boring jerky video of the ocean and the mountains, simply because the cameraman adopted the strategy of keeping the camera rolling in order not to miss the rare event. We trained a shot mixture model on a typical scene shown in the first two rows of figure 9 and used the learned model to find unusual segments that have low likelihood under the model of a typical scene. Again, within a couple of minutes, we were able create a new 30-second clip, much richer in content (two members of the family, whale, ocean).

5. Conclusions

We have developed and tested computationally efficient generative models of scenes in video. The models are based on the spatial layout of objects. Likelihood function is used as a cost function for multimedia retrieval. Segmentation, tracking, and balancing the complex similarity measure are all a part of one unified process that requires minimal number of predefined parameters. The expressive power of the model allows it to be “creative” in the ways it understands the data while still being useful as a similarity measure for search and retrieval.

In order to include the user in the loop in an unimposing way, we have connected the video fast forward speed with the likelihood under the model trained on a query. The query is simply a clip from the video, and this simple user interface has proven sufficient for fast video browsing, clip extraction and editing in various domains. In addition to the illustrated examples, we were able to search for people and nature scenes using the same tool.

In the future, we plan to investigate more elaborate features than simply RGB colors (e.g., features used in [2, 15]). We also plan to add translation invariance to this model (see [7]) and thus make it less sensitive to variability introduced by camera translations.

References

1. S.-F. Chang, W. Chen, H.J. Meng, H. Sundaram, and D. Zhong, “A fully automated content-based video search engine supporting spatiotemporal queries,” IEEE Trans. On Circuits and Systems for Video Technology, Vol. 8, No. 5, pp. 602–615, 1998.
2. J.S. De Bonet and P. Viola, “Structure driven image database retrieval,” in Advances in Neural Information Processing Systems, No. 10, 1997.
3. A.P. Dempster, N.M. Laird, and D.B. Rubin, “Maximum Likelihood from incomplete data via the EM algorithm, Journal of the Royal statistical Society, Series B, Vol. 39, No. 1, pp. 1–38, 1977.

4. E. Hadjidemetriou, M.D. Grossberg, and S.K. Nayar, "Spatial Information in Multiresolution Histograms," in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Vol. I, pp. 702–709, 2001.
5. M. Irani and P. Anandan, "Video indexing based on mosaic representation," IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 86, No. 5, May 1998.
6. N. Jojic and B. Frey, "Learning flexible sprites in video layers," in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2001.
7. N. Jojic, N. Petrovic, B. Frey, and T.S. Huang, "Transformed hidden markov models: Estimating mixture models and inferring spatial transformations in video sequences," in Proceedings IEEE Conf. Comp. Vis. Pattern Recogn., 2000.
8. N. Jojic, N. Petrovic, and T.S. Huang, "Scene generative models for adaptive video fast forward," Accepted, IEEE Int. Conf. Image Proc., Barcelona, Spain, 2003.
9. M.I. Jordan, Z. Ghahramani, T. Jaakkola, and K. Saul, "An introduction to variational methods for graphical models," Machine Learning, Vol. 37, No. 2, pp. 183–233, 1999.
10. O. Maron and A.L. Ratan, "Multiple-instance learning for natural scene classification," in Proc. 15th Int. Conf. on Machine Learning, 1998, pp. 341–349.
11. M.R. Naphade, T. Kristjansson, B. Frey, and T.S. Huang, "Probabilistic multimedia objects (multijects): A novel approach to video indexing and retrieval in multimedia systems," in Proceedings of ICIP, Vol. 3, 1998, pp. 536–540.
12. C.W. Ngo, T.C. Pong, and H.J. Zhang, "On clustering and retrieval of video shots," in Proceedings of ACM Multimedia, 2001.
13. J. Pearl, "Probabilistic Reasoning in Intelligent Systems Morgan Kaufmann Publishers: San Mateo, CA, 1988.
14. G. Pingali, G.A. Opalach, and I. Carlbom, "Multimedia retrieval through spatio-temporal activity maps," in Proceedings of ACM Multimedia, 2001.
15. C. Schmid, "Constructing models for content-based image retrieval," in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2001.
16. D. Spiegelhalter, A. Thomas, and W. Gilks, "BUGS, Bayesian inference using Gibbs sampling," Technical Report, MRC Biostatistics Unit, Cambridge, UK, 1993.
17. C. Stauffer, E.G. Miller, and K. Tieu, "Transform-invariant image decomposition with similarity templates," in Adv. in Neural Inf. Proc. Systems, 2001.
18. M.J. Swain and D.H. Ballard, "Color Indexing," Int. J. Comp. Vis, Vol. 7, No. 1, 1991.
19. K. Tieu and P. Viola, "Boosting image retrieval," CVPR, pp. 228–235, 2000.
20. C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, pp. 780–785, 1997.
21. L. Zelnik-Manor, and M. Irani, "Event-based video analysis," CVPR, 2001.

Nemanja Petrovic received his Ph.D. from University of Illinois in 2004. He is currently the member of research staff at Siemens Corporate Research at Princeton, New Jersey. His professional interests are computer vision and machine learning. Dr. Petrovic has published 20 papers in the area of video understanding, data clustering and enhancement.

Nebojsa Jojic received his Ph.D. from University of Illinois at 2001. His currently a researcher at Microsoft Research at Redmond, Washington. His professional interest include computer vision and machine learning. Dr. Jojic has published over 40 papers in the area of computer vision, bioinformatics and graphical models.

Thomas Huang received his Sc.D. from MIT in 1963. He is William L. Everitt Distinguished Professor in the University of Illinois, Department of Electrical and Computer Engineering and a full-time faculty member in the Beckman Institute Image Formation and Processing and Artificial Intelligence groups. Professor Huang has published over 600 papers in the area of computer vision, image compression and enhancement, pattern recognition, and multimodal signal processing. He is a Member of the National Academy of Engineering, Foreign Member of the Chinese Academy of Engineering and Chinese Academy of Science, and recipient of IEEE Jack S. Kilby Signal Processing Medal.