Issues in managing image and video data

Shawn Newsam*[a], Jelena Tešic[b], Lei Wang[b], and B. S. Manjunath[b]
[a]Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA, USA 94551
[b]Dept. of Elec. and Comp. Eng., University of California, Santa Barbara, CA, USA 93106-9560

ABSTRACT

This paper presents an overview of our recent work on managing image and video data. The first half of the paper describes a representation for the semantic spatial layout of video frames. In particular, Markov random fields are used to characterize the spatial arrangement of frame tiles that are labeled using support vector machine classifiers. The representation is shown to support similarity retrieval at the semantic level as demonstrated in a prototype video management system. The second half of the paper describes a method for efficiently computing nearest neighbor queries in high-dimensional feature spaces in a relevance feedback framework.

Keywords: Texture, spatial layout, semantic retrieval, nearest neighbor query, relevance feedback.

1. INTRODUCTION

This paper is an overview of our recent work on managing image and video data. Two challenges in particular are addressed: 1) how to represent the semantic spatial layout of frames in a video, and 2) how to efficiently compute nearest neighbor queries in high-dimensional spaces in a relevance feedback loop. To address the first challenge, we present an approach to modeling the common texture features and their spatial distribution in video frames. A semantic labeling of frame regions is achieved by using the class-conditioned texture feature likelihoods obtained from support vector machines to bias the Gibbs energy functions of a Markov random field. The region labeling is formulated as an optimization problem in which a causal greedy algorithm is used to make the assignments. To address the second challenge, we present a method to efficiently compute nearest neighbor queries in a relevance feedback loop. An adaptive algorithm computes the nearest neighbors at each query iteration based on the current set of nearest neighbors and the modified weight matrix of a quadratic distance measure.

2. SEMANTIC SPATIAL LAYOUT

This section describes a semantic/spatial analysis technique for video data at the frame level. There are two fundamental challenges to performing such an analysis: 1) semantically labeling the basic video frame elements, and 2) representing the spatial layout of these elements. Many techniques have been proposed to semantically labeling image content using extracted content features. Support vector machines (SVMs)[1] are one of the more successful of these techniques especially for complex feature distributions in high-dimensional spaces. However, such an application of SVMs does not consider the spatial distribution of the data. Alternately, Markov random fields (MRFs) are one of the more successful techniques for modeling the spatial distribution of image data[2,3,4,5]. The hidden random fields are used to assign semantic labels to lattice sites by considering both the spatial layout and content features of the data. The feature vectors corresponding to a label are commonly assumed to be clustered in the feature space. This is not always true, however, for real image data. For example, an empty parking lot and a parking lot that is full of cars are visually different and therefore unlikely to be close in the feature space. But they are both parking lots. General MRF models are thus inadequate for representing the spatial layout of semantic classes with complex feature distributions.

We propose an SVM-MRF model for analyzing the semantic spatial layout of video data at the frame level. This model combines an SVM's capacity to classify complex feature distributions with an MRF's capacity to represent complex spatial layouts. Video frame tiles are taken to be the basic elements. Texture features based on Gabor wavelets[6] are extracted since we are analyzing remote sensed aerial video that only contains a single infrared band. A manually labeled training set is used to train a tile level classifier in three steps. First, the training set is clustered in the texture feature space. Second, the training labels are used to train a set of SVMs for each cluster. Finally, the spatial layout of

*newsam1@llnl.gov; phone 925-422-7392

the training set is modeled as an MRF. After the classifier has been trained, novel video frames are labeled by using the MRF to refine the SVM tile classifications. This labeling can be used to provide semantic level browsing and similarity retrieval.

2.1. Feature clustering

The statistical distribution of texture features corresponding to different semantic classes, such as roads, buildings, fields, etc., can be learned using a labeled training set. In the proposed approach, the video frames are divided into 64×64 pixel non-overlapping tiles. A 30 dimension illumination normalized Gabor wavelet texture feature[5] is extracted for each tile. A training set is created by manually labeling video frame tiles using a predetermined set of semantic classes. Since the texture features tend to be clustered in the high-dimensional space, GMMs are used to model their statistical distribution. However, since semantically similar tiles are often visually quite different, it is unreasonable to expect that 1) the clusters are semantically homogeneous, or 2) that the features from a particular semantic class form a cluster. Therefore, the role of the GMMs is not to provide a semantic clustering of the features but to initialize the SVM stage of the classification. In particular, they allow a separate set of SVMs to semantically classify each of the clusters in the high-dimensional space.

The GMMs are applied in a lower-dimensional space since clustering high-dimensional spaces is a challenge. Principle component analysis (PCA) is used to reduce the dimension of the feature space by representing each normalized training feature vector using d eigenfeatures, where d<30. The transformed feature vectors $Y \in R^d$ are modeled as a mixture of K Gaussians. The probability density of these vectors, assuming a full covariance model, is therefore

$$f(y|q) = \sum_{j=1}^{K} a_j \frac{1}{\sqrt{(2p)^d |\Sigma_j|}} \exp\left\{-\frac{1}{2}(y-m_j)^T \Sigma_j^{-1}(y-m_j)\right\} \quad (1)$$

where $q = \{a_j, m_j, \Sigma_j\}_{j=1}^{K}$ are the GMM parameters. The prior probability of the j^{th} component is a_j with the constraints $a_j > 0$ and $\sum_{j=1}^{K} a_j = 1$. The mean vector of the j^{th} component is $m_j \in R^d$, and the covariance matrix of the j^{th} component is the d×d positive definite matrix Σ_j. The expectation maximization (EM) algorithm[7] is used to estimate the GMM parameters using the training set. The results at the end of this step are K Gaussian distributed feature clusters.

2.2. SVM marginal distribution

In this step, SVMs are used to model the feature distributions of the semantic classes within each cluster. This is motivated by the observation that even though the textures in a cluster are visually similar in a global sense, they vary enough to belong to different semantic classes. The feature vectors in a cluster can be effectively classified using SVMs through the selection of suitable kernels.

The SVMs are not applied in the reduced dimension space, but in the original feature space. Since SVMs are binary classifiers, a set of "one-against-other"[8] SVMs are used to perform multi-class classification. In this approach, SVMs that separate one class from the others are constructed. Multi-class classification is enabled by arbitrating between these SVMs. A fundamental challenge in using SVMs in an MRF framework is that they do not provide a probabilistic classification. Platt[9] addresses this problem by introducing an additional sigmoid function that maps the SVM outputs to probabilities. Suppose that a feature vector y is to be classified into one of M semantic classes. Let the output of the m^{th} SVM be

$$f_m(y) = \sum_i x_{m,i} a_{m,i} k(y_i, y) + b \quad (2)$$

where $k(\cdot,\cdot)$ is the kernel function and

$$x_{m,i} = \begin{cases} -1, & \text{if the label of } y_i \text{ is } m \\ 1, & \text{otherwise} \end{cases} . \tag{3}$$

The SVM marginal distribution is then computed as

$$p(y|z) = \frac{1}{1+\exp(h f_z(y))} \tag{4}$$

where h is a constant. A maximum likelihood classifier can then be used to classify feature vector y

$$\begin{aligned} x &= \underset{1\le m\le M}{\operatorname{argmax}} P(m|y) \\ &= \underset{1\le m\le M}{\operatorname{argmax}} p(y|m)P(m) \\ &= \underset{1\le m\le M}{\operatorname{argmax}} \frac{1}{1+\exp(h f_m(y))} P(m) \end{aligned} . \tag{5}$$

This classification can result in an inconsistent labeling, however, since the spatial relationships of neighboring tiles are not considered. For example, it is possible that a tile labeled as "street" is surrounded by tiles labeled as "sky". The spatial distribution of the semantic labels must be considered in order to resolve this semantic conflict. The next section describes how MRFs are used to accomplish this.

2.3. Spatial Layout Consistency

An effective way of improving the SVM classification is to impose constraints on the spatial arrangement of the labels. Inconsistencies can be ruled-out, such as a tile labeled "parking lot" being surrounded by tiles labeled "water." The particular approach uses MRFs to model the spatial distribution of the class labels. The label of a tile at site s is modeled as a discrete-valued random variable X_s, taking values from the semantic label set $\mathsf{M} = \{1,2,...,M\}$, and the set of random variables $X = \{X_s, s \in S\}$ constitutes a random field where S is the lattice of image blocks. The random field X is modeled as an MRF with a Gibbs distribution,[8]

$$p(x) = \frac{1}{Z} e^{-U(x)} \tag{6}$$

where x is a realization of X. The Gibbs energy function $U(x)$ can be expressed as the sum of clique potential functions,

$$U(x) = \sum_{c\in Q} V_c(x) \tag{7}$$

where Q is the set of all cliques in a neighborhood. The MRF model is reinforced by incorporating the class-conditioned feature likelihoods into the energy function, as follows:

$$U(x) = \sum_{s\in S} \left(\sum_{s'\in N_s} -a\, LP_{s-s'} - b\, LP_s \right) \tag{8}$$

where $LP_{s-s'} = \log(p_{s-s'}(x_s, x_{s'}))$ and $LP_s = \log\left(p(y_s|q_{x_s})\right)$. N_s is the neighbors of the site s, and a and b are the weights of LP_s and $LP_{s-s'}$ respectively. $LP_{s-s'}$ represents the spatial relationship between neighboring sites s and s'

where *s*-*s*′ indicates the direction of neighborhood. LP_s represents the conditional probability density of feature vector y_s given the label x_s. $p_{s-s'}(x_s, x_{s'})$ is the joint probability of x_s and $x_{s'}$ along the direction *s*-*s*′ and can be approximated with a co-occurrence matrix computed using the labeled training set. For each type of clique *s*-*s*′, a co-occurrence matrix is constructed from the joint probabilities $P_r(i, j)$ between pairs of semantic labels *i* and *j* in a given direction *r*.

In order to simplify the model, a second order pair-site neighborhood system is used. Each site thus has eight neighbors. Four types of cliques are considered, wherein *s*-*s*′ makes angles of 0, 45, 90, and 135 degrees with respect to the x-axis. In any neighborhood, cliques along the same direction are considered equivalent. Four co-occurrence matrices are constructed along these four directions of label distribution.

2.4. Spatial layout retrieval

In this step, a representation termed semantic layout is used to characterize an image or video frame based on the spatial arrangement of its labeled tiles. For retrieval purposes, the similarity between the query image and each stored image can be determined from their semantic layouts. Let the semantic layout of the query image be X^q and that of the stored image be X^I. In order to improve the retrieval performance, a soft classification scheme is adopted. For a given image tile, the labels with the three largest local conditional probabilities are selected to represent this tile. All candidate labels are stored along with the feature vectors for future retrieval. The modified semantic layout similarity between the query image and each stored image is given by:

$$\begin{aligned} S_3 = & \sum_{s\in S}\left(\sum_{j=1}^{3} a_1 a_j d(x_{s,j}^q, x_{s,1}^I)\right) \\ & + \sum_{s\in S}\left(\sum_{j=1}^{3} a_2 a_j d(x_{s,j}^q, x_{s,2}^I)\right)\left(1-\sum_{i=1}^{3} d(x_{s,i}^q, x_{s,1}^I)\right) \\ & + \sum_{s\in S}\left(\sum_{j=1}^{3} a_3 a_j d(x_{s,j}^q, x_{s,3}^I)\right)\left(1-\sum_{i=1}^{3} d(x_{s,i}^q, x_{s,1}^I)\right)\left(1-\sum_{i=1}^{3} d(x_{s,i}^q, x_{s,2}^I)\right) \end{aligned} \tag{9}$$

where $a_i = 1/2^{i-1}, i = 1,2,3$ are the weights for different label similarities, and $x_{s,j}^q$ is the *j*th candidate label of the query tile at site *s*. The similarity measure shown in Eq. (9) is computed by comparing each candidate label of a query tile to all the candidate labels of the corresponding target tile. This approach to similarity retrieval is expected to perform better than methods that do not consider the underlying semantics. The similarity measure in Eq. (9) is effective only when all images are of the same size. In order to compare the layouts of images of any size, a semantic histogram can be computed for each image. This is similar to the image histogram where the inputs are semantic labels instead of image intensities. As long as the semantic label set is the same, two different sized images can be compared using their semantic histograms.

Examples of semantic spatial layout retrieval in a prototype video management system are shown in Fig.1. The top frame is the query and remaining frames are the results. The semantic labeling of the query is shown. Observe that the retrieved frames can be visually quite different from the query even though their semantic layouts are similar. These results could not be achieved using a low-level description alone. The combination of SVM and MRF models is thus shown to capture the semantic spatial layout of the frames. Fig. 2 shows an example of retrieval based on semantic histograms. Fig. 2(a) shows how a query with 50% freeway and 50% is specified in the prototype system. Fig. 2(b) shows the top four retrievals for such a query. These results demonstrate how the proposed SVM-MRF method supports frame-level semantic spatial layout similarity retrieval. Users can either supply a query image or construct one using a simple interface. In either case, the user does not need to be familiar with the low-level features but can interact with the system at a semantic level.

3. ADAPTIVE NEAREST NEIGHBOR SEARCH

In database research, indexing structures are used to prune the search space. Recently, there has been much work on indexing structures to support high-dimensional feature spaces However, as reported in [10], the effectiveness of many of these indexing structures is highly data dependent and, in general, difficult to predict. Often a simple linear scan of the database items is cheaper than using an index based search in high dimensions.

An alternative to high-dimensional index structures is sequential search over a compressed representation of the database items. The vector approximation file (VA-file) [11] is one such representation. In the VA-file architecture, the feature space is quantized and each feature vector in the database is encoded with a compressed representation. Sequential search over quantized candidates reduces the search complexity. Only a fraction of the actual feature vectors is accessed, thus reducing the number of page accesses.

Consider a database f of N elements F_i, where F_i is an M-dimensional feature vector. Let Q be the query object from the database f. Define the quadratic distance metric $d(Q, F_i, W)$ between query Q and a database object F_i as:

$$d^2(Q, F_i, W_t) = (Q-F_i)^T W_t (Q-F_i) \tag{10}$$

where W_t is a symmetric, real and positive definite matrix. In the VA-file representation, each of the feature space dimensions is partitioned into non-overlapping segments. Generally, the number of segments is 2^{Bj}, $j = 1,\ldots,M$. B_j is the number of bits allocated to dimension j. For a feature vector F_i, its approximation $C(F_i)$ is an index to the cell containing F_i. If F_i is in partition p, $p=0,1,2,\ldots,2^{Bj}-1$ along the j^{th} dimension, the boundary points that determine the p^{th} partition are b_{pj} and b_{p+1j} where $b_{pj} = f_{ij} < b_{p+1j}$.

3.1. Nearest neighbor (NN) search

VA-file based nearest neighbor search can be considered as a two phase filtering process [10]. In Phase I, the set of all vector approximations is scanned sequentially and lower and upper bounds on the distances of each object in the database to the query object are computed. Let $?$ be the K^{th} largest upper bound found from the scanned approximations. During the scan, a buffer is used to keep track of $?$. If an approximation is encountered such that its lower bound is larger than $?$, the corresponding feature vector can be skipped since at least K better candidates exist. Otherwise, the approximation will be selected as a candidate and its upper bound will be used to update the buffer, if necessary. The resulting set of candidate objects at this stage is $N1(Q,W)$, and the number of elements in the set is $|N1(Q,W)|$. Phase II finds K nearest neighbors from the feature vectors contained in the approximations filtered in Phase I. The actual feature vectors, whose approximations belong to a candidate set $N1(Q,W)$, are accessed. The feature vectors are visited in increasing order of their lower bounds and the exact distances to the query vector are computed. If a lower bound is reached that is larger than the K^{th} actual nearest neighbor distance encountered so far, there is no need to visit the remaining candidates. Let $N2(Q,W)$ be the set of objects visited before the lower bound threshold is encountered. The K nearest neighbors are found by sorting the $|N2(Q,W)|$ distances. In database searches, the disk/page access is an expensive process. The number of candidates from Phase I filtering determines the cost of disk access/page access. Our focus is on improving Phase I filtering in the presence of relevance feedback.

3.2. Relevance feedback

In the context of content-based image retrieval, relevance feedback has attracted considerable attention. In a typical scenario, given a set of retrievals for an image query, the user may identify some relevant and some non-relevant examples. Based on this, the similarity metric is modified to compute the next set of retrievals. Modification to the similarity metric should help provide better matches to a given query and meet the user's expectations.

Let W_t be the weight matrix used in iteration t, and R_t be the set of K nearest neighbors to the query object Q. At iteration t, define the k^{th} positive example vector, $k=\{1,\ldots,K'\}$, as:

$$X_k^{(t)} = [\, x_{k1}^{(t)}, x_{k2}^{(t)} \ldots x_{kM}^{(t)} \,]^T,\ X_k^{(t)} \in R^M . \tag{11}$$

K' is the number of relevant objects identified by the user. These K' examples are used to modify the weight matrix W_t to W_{t+1}. We consider an optimized learning technique that merges two existing well known updating schemes. The first scheme termed MARS[12] restricts W_t to be a diagonal matrix. The weight matrix is modified using the standard deviation s_m of $x_{km}^{(t)}$, $m=\{1,...,M\}$. The weight matrix is normalized after every update and the result is:

$$(W_{t+1})_m = \frac{1}{s_m^2}\left(\prod_{i=1}^{M} s_i^2\right)^{\frac{1}{M}}. \tag{12}$$

The second scheme termed MindReader[13] updates the full weight distance matrix W_t, by minimizing the distances between the query and all positive feedback examples. In this scheme, the user picks K' positive examples, and assigns a degree of relevance $p_k^{(t)}$ to the k^{th} positive example $X_k^{(t)}$. The optimal solution for W_t is equivalent to the Mahalanobis distance if we assume the Gaussianity of positive examples, i.e:

$$W_{t+1} = \det(C_t)^{\frac{1}{M}} (C_t)^{-1} \quad . \tag{13}$$

The elements of the covariance matrix C_t are defined as:

$$(C_t)_{ij} = \frac{\sum_{k=1}^{K'} p_k^{(t)} (x_{ki}^{(t)} - q_i)(x_{kj}^{(t)} - q_j)}{\sum_{i=1}^{K'} p_k^{(t)}}. \tag{14}$$

For $K'=M$, matrices C_t and W_t are symmetric, real and positive definite, and can be factorized as:

$$\begin{aligned} C_t &= P'^{T}_t l'_t P'_t, \;\; P'^{T}_t P'_t = I, \; l'_t = \mathrm{diag}(l'_1,...,l'_M), \\ W_t &= P_t^T l_t P_t, \;\; P_t^T P_t = I, \; l_t = \mathrm{diag}(l_1,...,l_M), \\ P_t &= P'_t, \;\; l_i = \frac{\left(\prod_{k=1}^{M} l'_k\right)^{\frac{1}{M}}}{l'_i}. \end{aligned} \tag{15}$$

The full matrix update approach better captures the dependencies among feature dimensions, thus reducing redundancies in high-dimensional feature space and allowing us to filter out more false candidates. The downside of this full matrix update approach is that the inverse covariance matrix exists only if number of positive examples K' is larger than or equal to the number of feature dimensions M. If $K' < M$, we adopt the MARS approach.

With this brief introduction, we can now formulate the nearest neighbor search problem as follows: given R_t, W_t, and K', the weight matrix W_{t+1} is computed from W_t using the above schemes. The challenge is to efficiently compute the next set of K nearest neighbors R_{t+1}, using the current nearest neighbors R_t and the new weight matrix W_{t+1}.

3.3. Bound computation

The nearest neighbor filtering process in the vector approximation approach (Phase I) uses information on lower and upper bound of the distances between a query point Q and a feature vector F_i. Given a query Q and a feature vector F_i, lower and upper bound of distance of $d(Q, F_i, W_t)$ are defined as $L_i(Q, W_t)$ and $U_i(Q, W_t)$ so that the following inequality holds:

$$L_i(Q, W_t) \le d(Q, F_i, W_t) \le U_i(Q, W_t). \tag{16}$$

The computation of lower and upper bound of distance $d(Q, F_i, W_t)$ using the VA-file index is straightforward for a diagonal W_t. Bounds are constructed based on hyper rectangular approximations. For the case of general quadratic distance metric, a nearest neighbor query becomes an ellipsoid query. Points F_i that have the same distance $d(Q, F_i, W_t)$ from a query point Q form an ellipsoid centered around query point Q. Lower and upper bound computations in the

cases of weighted Euclidean and quadratic distance metrics are illustrated in Fig. 3(a). It is computationally expensive to determine whether a general ellipsoid intersects a cell in the original feature space. For the quadratic metric, exact distance computation between the query object Q and a rectangle $C(F_i)$ requires numerically expensive quadratic programming. This undermines the advantages of using a vector approximation indexing structure. Our solution to this problem is to approximate the upper and lower bound on a cell, based on approximation techniques for ellipsoid queries.

The cell $C(D)$ that approximates feature point D is transformed into the hyper parallelogram $C(D')$ in the mapped space, as illustrated in Fig. 3(b). The parallelogram $C(D')$ can be approximated with the bounding hyper rectangular cell $C'(D')$. The weight matrix in the mapped space is $?_t$, and the quadratic distance becomes a weighted Euclidean distance.

$L_i(Q, W_t)$ and $U_i(Q, W_t)$ are approximated in the mapped space with $L_i(Q', ?_t)$ and $U_i(Q', ?_t)$. Conservative bounds on rectangular approximations introduced in [14] allow us to avoid the exact distance computation for query object Q and every approximation cell $C(F_i)$. However, for restrictive bounds, the distance computation stays quadratic with number of feature dimensions. The approximation $C(F_i)$ only specifies the bounding rectangle position in the mapped space. Note that the size of relative bounding rectangle depends only on the cell size in the original space, and the rotation matrix P_t. The rotational matrix P_t is computed prior to a new search. Also, the computational complexity of the distance is further reduced using the fact that the weighted Euclidean distance has the same computational complexity as Euclidean distance. Define a rotational mapping for matrix P_t as $Q' = P_t Q$. All quadratic distances in the original space transform to weighted Euclidean distances in the mapped space, i.e.:

$$d^2(Q, F_i, W_t) = (Q - F_i)^T W_t (Q - F_i) = (P_t (Q - F_i))^T l_t (P_t (Q - F_i)) = (Q' - F_i')^T l_t (Q' - F_i'). \quad (17)$$

3.4. Adaptive nearest neighbor search for relevance feedback

In database searches, the disk/page access is an expensive process, directly proportional to the number of approximations determined in Phase I filtering. Phase I filtering determines a subset of approximations from which the K nearest neighbors can be retrieved. Let $N1^{opt}(Q, W_t)$ be the minimal set of approximations that contain K nearest neighbors. The best case scenario for Phase I filtering is to exactly identify this subset $N1(Q,W_t)=N1^{opt}(Q,W_t)$. However, Phase I filtering introduces false candidates. First, the number of false candidates depends on how firm the filtering bound is throughout the sequential scan. Second, the approximate lower bound can be much smaller than the real lower bound and can introduce many false candidates. The smaller this candidate set is, the better is the indexing and search performance.

Let $?$ be the K^{th} largest upper bound encountered so far during a sequential scan of approximations. In the standard approach, the approximation $C(F_i)$ is included in $N1(Q,W_t)$ only if $L_i(Q,W_t) < ?$, and the $?$ is updated if $U_i(Q,W_t) < ?$. Only the K^{th} smallest upper bound from the scanned approximations is available and used for filtering.

Let $R_{t-1}=\{F_k^{(t-1)}\}$ be the set of K nearest neighbors of query Q at iteration $t-1$ under weight matrix W_{t-1}. Define $r_t^u(Q)$ as: $r_t^u(Q)= max\{d(Q, F_k^{(t-1)}, W_t)\}$. Let $R_t=\{F_k^{(t)}\}$ be the set of K nearest neighbors from f to Q under W_t. Define $r_t(Q)$ as the maximum distance between Q and the items in R_t: $r_t(Q)=max\{d(Q, F_k^{(t)}, W_t)\}$. When W_{t-1} is updated to W_t, we can establish an upper bound on $r_t(Q)$ as: $r_t(Q) = r_t^u(Q)$[15]. For approximation $C(F_i)$ to be a qualified one in $N1^{opt}(Q,W_t)$, its lower bound $L_i(Q,W_t)$ must satisfy: $L_i(Q,W_t) = r_t^u(Q)$. Let $R_1=\{E, H\}$ be a user's answer set for a query Q in a feature space illustrated in Fig. 3(c). When W_1 is updated to W_2, $r_t^u(Q)= d(Q, E, W_2)$. The answer set offered to a user will be limited to points inside radius $r_t^u(Q)= d(Q, E, W_2)$, as marked in Fig. 3(c).

3.5. Adaptive nearest neighbor algorithm

For $t=1$, the K-NN search Phase I filtering reduces to standard Phase I filtering[10]. Note that, in the standard approach, a buffer is used to keep track of the value of $?$, the K^{th} largest upper bound found so far during a scan. In practice, this buffer is used only when we do not have user feedback. In the presence of relevance feedback, we avoid the buffer update, and use a new filtering bound defined as $l_t^u(Q)$. The data filtering (Phase II) step results in a set of K nearest neighbors R_t. Then, the algorithm starts a new iteration and increments t. The user identifies positive examples in R_{t-1}. This information is used to update W_{t-1} to W_t.

Note that W_t is updated (P_t and $?_t$ are computed in the process) and $r_t^u(Q)$ is computed, before the Phase I filtering is started. We decide to choose an approximation $C(F_i)$ as a Phase I candidate if its lower bound is smaller than $r_t^u(Q)$. The number of such false candidates resulting from Phase I filtering depends on the convergence rate of $?$ to its final value. Using firmer filtering bounds reduces the number of false candidates collected during the sequential scan of the approximations[14]. Note that $r_t^u(Q)$ is computed before Phase I filtering. Also, keeping the standard $?$ updated requires an upper bound computation for every candidate[10]. Calculation of upper bounds and the K^{th} smallest upper bound is expensive under quadratic distance metric. However, if $r_t^u(Q)$ is a Phase I filtering bound for lower bound on distance between the query vector and database objects, upper bound computation for each approximation encountered is entirely avoided.

3.6. Experiments

We compare the standard VA approach to computing the K nearest neighbors to our proposed adaptive method for different resolutions S and different filtering bounds as described in the previous sections. We demonstrate the efficiency of the proposed approach on a dataset of $N=90774$ texture feature vectors. 60-dimensional feature vectors are formed using the first- and second-order moments of Gabor filter outputs[16]. The approximations are constructed using the standard VA index. Experiments are carried out for different numbers of bits assigned to every dimension, S={2,...,8}. A larger value S corresponds to an approximation constructed at a finer resolution. For each query, $K=M+10$ nearest neighbors are retrieved during each iteration. The feedback from the user is based on texture relevance only. For a specific query, the user selects K' relevant nearest neighbors to update the distance measure. The distance metric is updated before every iteration. Queries Q_i are selected from the dataset to cover both dense cluster representatives and outliers in the feature space. For a given resolution S, and query vector Q_i, let the number of candidates from Phase I standard filtering approach be $|N1^{(s)}(Q_i)|$ for the standard approach. Let the corresponding number for the proposed adaptive method with filtering bound $r_t^u(Q)$ be $|N1^{(r)}(Q_i)|$. Define the average numbers of Phase I candidates over the example queries and the corresponding effectiveness measure as:

$$N1^{(s)} = \frac{1}{I}\sum_{i=1}^{I} |N1^{(s)}(Q_i)|,\ N1^{(r)} = \frac{1}{I}\sum_{i=1}^{I} |N1^{(r)}(Q_i)|,\ a^{(r)} = \frac{1}{I}\sum_{i=1}^{I} \frac{|N1^{(s)}(Q_i)|}{|N1^{(r)}(Q_i)|}. \quad (18)$$

For the Weighted Euclidean distance, we average over $I=20$ query vectors, for $K=20$ nearest neighbors and $K'=15$ relevant features for weight matrix update. The number of candidates resulting from the standard and adaptive Phase I filtering (with different filtering bounds) is shown in Fig. 4(b). $r_t^u(Q)$ restricts the search space even more for finer resolutions. Filter bounds at all resolutions are shown in Fig. 4(a). The average gain of the proposed method is not monotonic over S, since the results are strongly correlated with distribution of the feature points in the high-dimensional space. However, the minimum gain of the proposed adaptive filtering is still significant at as demonstrated in Fig. 4(c). When S=8, $?$ converges to a value that is smaller than $r_t^u(Q)$, but very close to it, as shown in Fig. 4(a).

For the quadratic distance, we average over $I=20$ query vectors, for $K=M+10$ nearest neighbors and $K'=M+5$ relevant features for weight matrix update. Filter bound $?$ rapidly increases when the resolution is smaller, due to the larger sizes of the hyper rectangles used in the corresponding approximations. A larger difference between $r_t^u(Q)$ and $?$ should impose a significant improvement for the proposed method. Thus, in the presence of relevance feedback we can either save some memory for approximation storage or reduce the number of disk accesses for the same resolution. Fig. 5(a) shows that the difference between $r_t^u(Q)$ and $?$ increases for lower values of S. At coarser approximation levels, $r_t^u(Q)$ restricts the search space significantly more than $?$ (Fig. 5(a)) and further reduces the number of false candidates (Fig. 5(b)). Results show that the efficiency gain is significant and even more correlated with the distribution of the feature points, as demonstrated in Fig. 5(c).

4. CONCLUSION

This paper presents some of our recent work on managing image and video data. Two challenges in particular are addressed: 1) representing the semantic spatial layout of video frames, and 2) efficiently computing nearest neighbor queries in high-dimensional spaces in a relevance feedback loop. These challenges can be viewed as two of the many

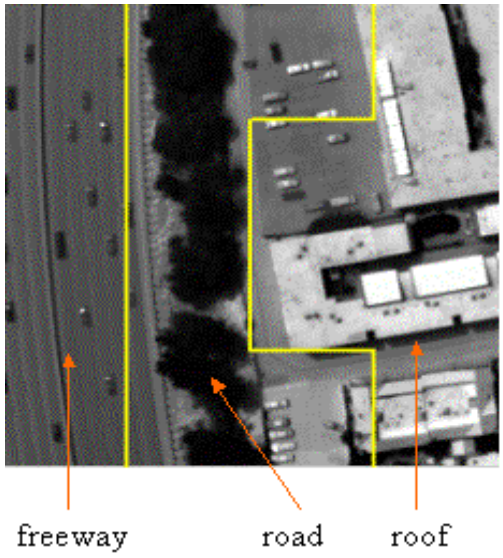
fundamental sub-problems of the larger challenge of providing efficient and effective access to the growing repositories of multimedia data. The temporal dimension makes linear navigation unwieldy for sizable video datasets. Non-linear or random access is needed if data consumers are to fully realize the information contained in the videos. Techniques for browsing and retrieving video frames based on their semantic spatial layout, as described and demonstrated in this paper, can provide such access. And, as the methods for accessing multimedia data mature, being able to provide personalized access will become increasingly important. Relevance feedback is fast becoming one of the more attractive approaches since it allows the user interface to remain simple and intuitive. The real challenge is in designing index structures that support dynamic queries. The nearest neighbor search scheme presented in this paper is one possible solution for dynamic queries in high-dimensional feature spaces.

ACKNOWLEDGEMENTS

UCRL-CONF-200471: This work was performed in part under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48. It was supported in part by an award from the Institute of Scientific Computing Research (ISCR) at LLNL, and the following grants: ONR# N00014-01-1-0391, NSF Instrumentation #EIA-9986057, and NSF Infrastructure #EIA-0080134. The authors would like to thank S. Chandrasekaran at UCSB and C. Kamath at LLNL for many fruitful discussions.

REFERENCES

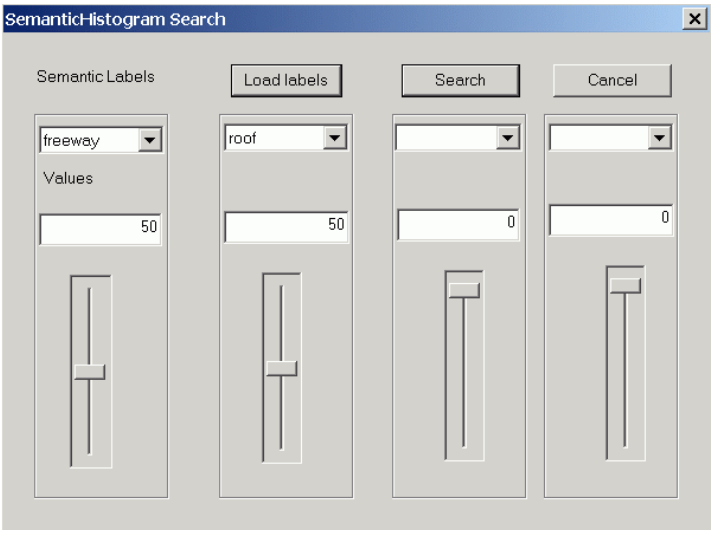
1. C. Cortes, V. Vapnik, "Support-vector networks," *Machine Learning*, Vol.20, No. 3, pp. 273-297, 1995.
2. J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Statistical Society*, Vol.B 48, pp. 259-279, 1986.
3. S. Li, *Markov Random Field Modeling in Computer Vision*, Springer-Verlag, 1995.
4. L. Wang and J. Liu, "Texture classification using multiresolution Markov random field models," *Pattern Recognition Letters*, Vol.20, No.2, pp. 171-182, 1999.
5. R. Chellappa and A. Jain, *Markov Random Fields: Theory and Applications*, Academic Press, 1993.
6. B. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG-7: Multimedia Content Description Interface*, Wiley, 2002.
7. A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, Vol.B 39, No. 1, pp. 1-38, 1977.
8. K. Kim, K. Jung, S. Park, and H. Kim, "Support vector machines for texture classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No.11, pp. 1542-1550, 2002.
9. J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, pp. 61-74, MIT Press, 1999.
10. C. Bohm, S. Berchtold, and D. A. Keim, "Searching in high-dimensional spaces: index structures for improving the performance of multimedia databases," *ACM Computing Surveys*, Vol. 33, No. 3, pp. 322-373, September 2001.
11. R. Weber, H. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," In *Proceedings of the International Conference on Very Large Data Bases*, pp. 194-205, August 1998.
12. Y. Rui and T. Huang, "Optimizing learning in image retrieval," In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 236-243, June 2000.
13. Y. Ishikawa, R. Subramanya, and C. Faloutsos, "Mindreader: querying databases through multiple examples," In *Proceedings of the International Conference on Very Large Data Bases*, pp. 218-227, August 1998.
14. M. Ankerst, B. BraunmÄuller, H. Kriegel, and T. Seidl, "Improving adaptable similarity query processing by using approximations," In *Proceedings of the International Conference on Very Large Data Bases*, pp. 206-217, August 1998.
15. J. Tešic and B. S. Manjunath, "Nearest neighbor search for relevance feedback," In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 643-648, June 2003.
16. B. Manjunath and W. Ma, "Texture features for browsing and retrieval of image data", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.18, No.8, pp. 837-842, 1996.



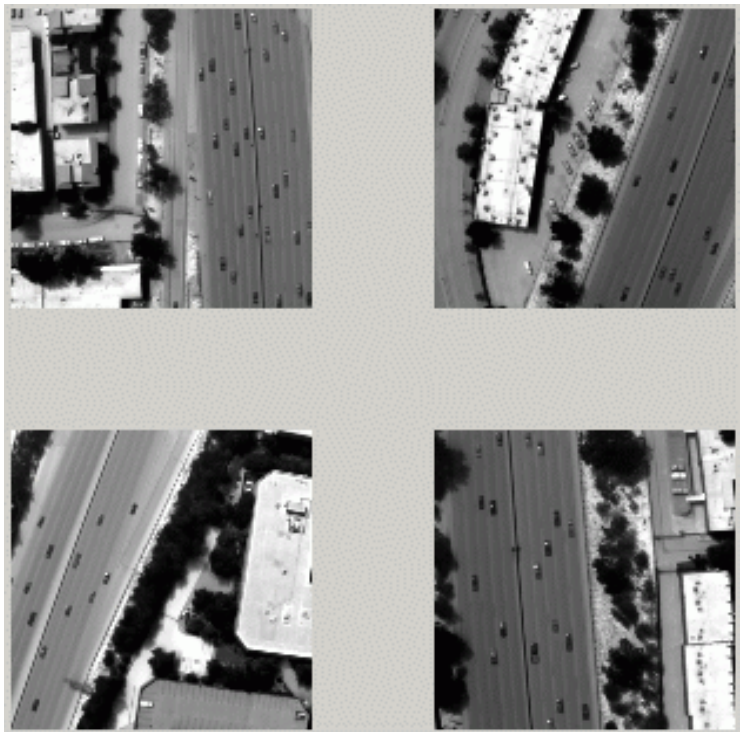
(a) Query image and its semantic layout.

(b) Top four retrieval results.

Figure 1. An example of semantic spatial layout retrieval.

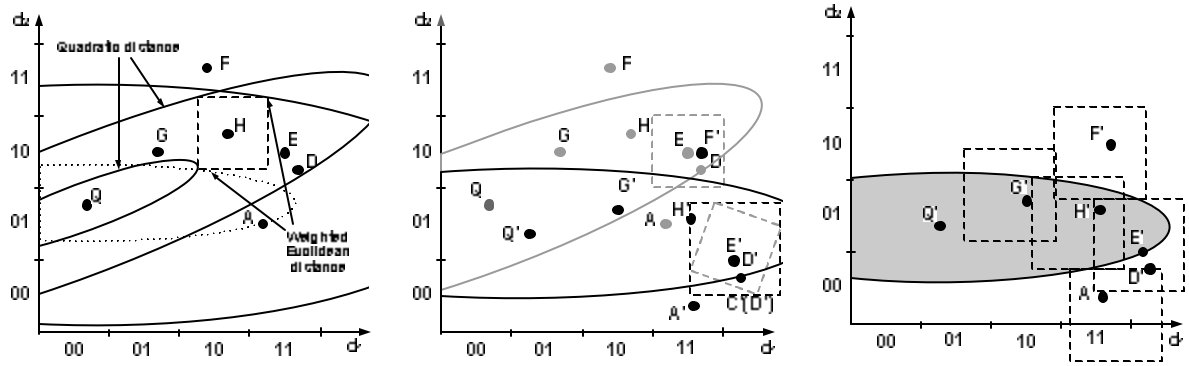


(a) Specifying a semantic histogram query.



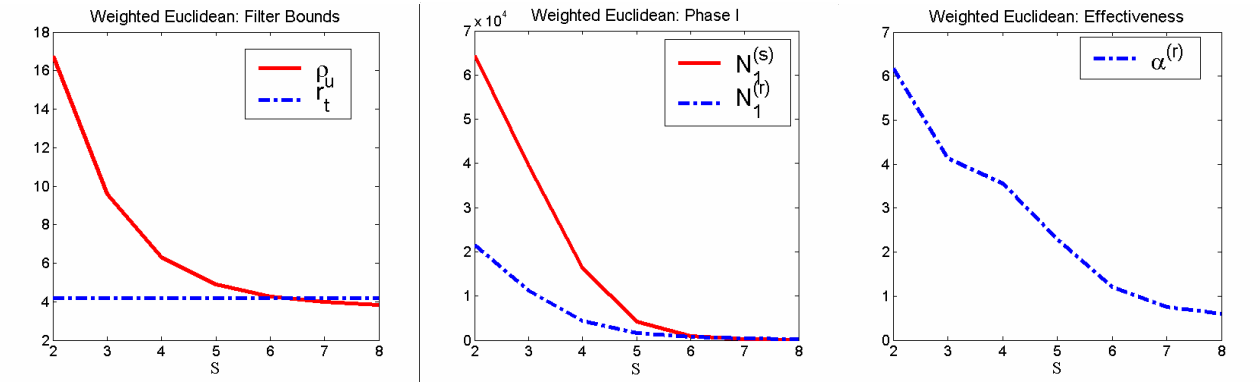
(b) Semantic histogram retrieval results.

Figure 2. An example of semantic histogram retrieval for a query with 50% freeway and 50% roof.



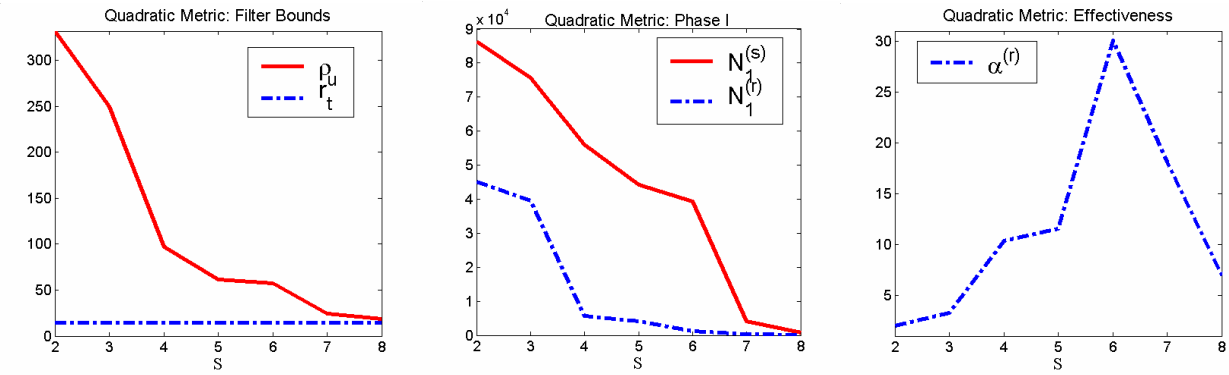
(a) (b) (c)

Figure 3. a) Bound computations for (1) weighted Euclidean and (2) quadratic distance metrics. b) Rotational mapping of feature space and approximation cells: D'=PD. c) Adaptive search space: illustration of using r_t^u to limit the search space in Phase I adaptive filtering (shaded space).



(a) (b) (c)

Figure 4. Adaptive nearest neighbor search for weighted Euclidean metric.



(a) (b) (c)

Figure 5. Adaptive nearest neighbor search for quadratic distance metric.