In Proc. 10th International Symposium on Spatial and Temporal Databases (SSTD'07), Boston, U.S.A., 2007.

Querying Objects modeled by Arbitrary Probability Distributions

Christian Böhm, Peter Kunath, Alexey Pryakhin, and Matthias Schubert

Institute for Computer Science, Ludwig-Maximilians Universität München
{boehm,kunath,pryakhin,schubert}@dbs.ifi.lmu.de
WWW home page: http://www.dbs.ifi.lmu.de

Abstract. In many modern applications such as biometric identification systems, sensor networks, medical imaging, geology, and multimedia databases, the data objects are not described exactly. Therefore, recent solutions propose to model data objects by probability density functions(pdf). Since a pdf describing an uncertain object is often not explicitly known, approximation techniques like Gaussian mixture models(GMM) need to be employed. In this paper, we introduce a method for efficiently indexing and querying GMMs allowing fast object retrieval for arbitrary shaped pdf. We consider probability ranking queries which are very important for probabilistic similarity search. Our method stores the components and weighting functions of each GMM in an index structure. During query processing the mixture models are dynamically reconstructed whenever necessary. In an extensive experimental evaluation, we demonstrate that GMMs yield a compact and descriptive representation of video clips. Additionally, we show that our new query algorithm outperforms competitive approaches when answering the given probabilistic queries on a database of GMMs comprising about 100.000 single Gaussians.

1 Introduction

In recent years, a large variety of database applications has emerged where it is beneficial to consider the uncertainty of the given data. The object uncertainty might be caused by the inexactness of feature measurement like in biometrical or biological databases. Furthermore, object uncertainty is often introduced to obtain a compact description of very large or complex objects. For example, in sensor networks, it is not feasible to transmit an exact value at all points of time. Therefore, several points of time are aggregated into a single uncertain description. In order to model this uncertainty, the data objects are described by probability density functions (pdf) over a given feature space (cf. [1–6]).

In most applications, there is no explicitly known pdf describing the uncertain data object. Instead, the density function is given by a sample of feature values or in the case of data compression, the set of feature values being compressed. Thus, approximation techniques have to be employed in order to find an explicit density function. In other words, we need to describe each object as a function instead of set of sample points. A large number of database applications in all fields of science and industry approximate the underlying data distributions by a *Gaussian mixture model (GMM)*. The idea of a

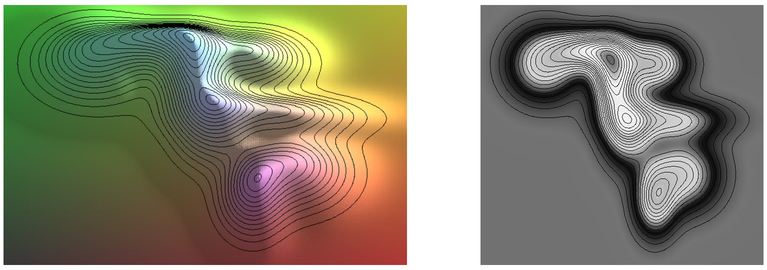

(a) 3D view. (b) 2D view.

Fig. 1. Approximation of an arbitrary complex bivariate probability density by a mixture of Gaussians with independent attributes.

GMM is to model a complex density function by a weighted sum of several Gaussian distributions. If a GMM is not exact enough, the quality can be arbitrarily increased by adding further Gaussians (cf. Figure 1). The popularity of GMMs can be explained by the following characteristics. A GMM has the ability *to approximate an arbitrary statistical data distribution* very closely [7, 8]. An example for such an approximation can be seen in Figure 1. In particular, GMMs are capable of *modeling arbitrarily shaped correlations* in data (cf. [8]) by a mixture containing distributions with independent attributes (cf. Figure 2). Moreover, GMMs can be efficiently derived by a variety of *mathematical methods* from any given sample set. The widespread use of these methods contributes to prevalent employment of GMMs in several applications.

In the following, we survey some of example application areas employing GMMs more closely. The first example for using objects represented by GMMs is managing *multimedia* data. An ordinary movie of about 90 minutes contains about 140.000 single images, called frames. Storing all these images for content-based video retrieval would require large storage capacities and an enormous computational effort for comparing videos. To avoid this problem, the multimedia community often employs summarization techniques representing videos as mixture models [9]. Thus, a video clip is described by a GMM over the feature space representing single frames. Storing the videos as a GMM dramatically reduces the resource consumption and still allows accurate content-based video retrieval. Moreover, the use of mixture models is justified by the demand to consider high level or semantic features (cf. [10–12] for details) in order to guarantee effectiveness in the retrieval of multimedia data. For instance, [13] describes the usage of face detection and recognition techniques in order to calculate a compact description of video data. Another example for the use of GMMs is *bioclimatic* research. For instance, the authors of [14] propose the use of GMMs to describe species ranges from mapped climatic variables. Moreover, authors of [15] employ a GMM in order to describe *drug dissolution* profiles. They also discuss implications of the GMMs for pharmaceutical product formulation tasks. Further application areas of GMMs are *sensor networks* (e.g. [16, 4]), *audio signal analysis* (e.g. [17]), *economics*

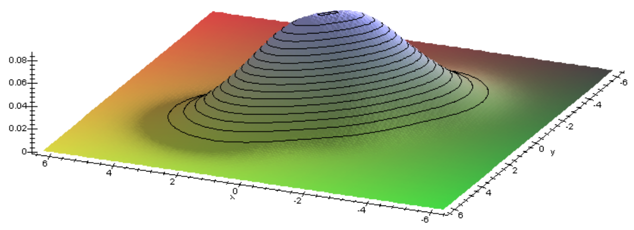


Fig. 2. A Gaussian mixture model representing correlated 2D data.

(e.g. [18]). Additionally, the authors of [7, 8] list several examples of the use of GMMs in the following application areas: *medicine*, *psychology*, *geology*, *agriculture*.

In his paper, we propose a novel method for query processing on arbitrarily shaped pdf modeling each data object as a GMM. Our new models aims to answer identification queries of the following form: Given a query GMM and a database of GMMs, return the k database objects which might describe the same data object with the highest probability. Unlike previous approaches, we additionally consider the case that no database object describes the same object as the query. Thus, we can handle the case that it is most likely that there is no object in the database resembling the given query. To efficiently handle this type of query, we propose a method for storing GMMs in a probabilistic index structure. The principal idea of this method is to store each component (i.e., each Gaussian of a GMM and its weighting) as a separate object. To calculate the probability that a given query object corresponds to a database object, the GMMs are reconstructed during query time. Therefore, we store the information about the possible candidates in a so-called candidate table. The candidate table is located in main memory and stores two values for each object that was touched but is not yet complete. Based on these two values, it is possible to calculate a conservative approximation of the probability for a database object (see Section 4 for details). When a component belonging to an object o is retrieved from the underlying index structure the information about o (i.e., two values in the candidate table) is updated.

The main contributions of this paper are: (1) A new probabilistic model for handling uncertain objects represented by GMMs. (2) A method to allow unsuccessful search for probabilistic queries. (3) A new query algorithm for efficiently answering probabilistic ranking queries that are based on decomposing the GMMs.

The rest of the paper is organized as follows. In Section 2, we survey the related work on managing and searching databases of uncertain objects. Section 3 introduces our new uncertainty model and specifies the examined types of queries. Section 4 introduces a new method for object indexing and query processing. Our experimental results are shown in Section 5 and the paper concludes with a summary in Section 6.

2 Related Work

Statistical Modeling of Data in Sensor Networks. In [1], a new probabilistic abstract datatype (ADT) based on a one-dimensional Gaussian is introduced. In order to handle large amounts of data, the authors store the Gaussian ADT data in an ORDBMS. For query processing, a two-stage process based on linear constraints is applied. However the GADT approach considers neither a mixture of probability density functions (pdf) nor multivariate probability distributions. The basic idea in [4] is to build a probabilistic model from both historical and current sensor readings and to query the model instead of the actual sensor network. The authors present an architecture, called BBQ, that realizes this idea by a model based on time-varying multivariate Gaussians. Users then submit one-shot queries to a relational database to request real-time information about the network. However, the authors do not discuss the usage of index structures for efficient query processing. The authors of [19] propose to use probabilistic models for acquisitional settings such as sensor networks and Internet monitoring. Such a probabilistic model enables them to define new query types, exploit correlations when answering queries and to provide probabilistic guarantees on the correctness of the answers. While it is possible to use many different types of models, the authors explicitly describe the time-varying multivariate Gaussian model as an example. However, the authors do not consider an index structure for query processing.

Spatial Uncertainty. Authors of [2, 3, 5, 20] deal with an uncertainty model for spatially uncertain objects and propose queries which are specified by intervals in the query space. In this setting a query retrieves uncertain objects w.r.t. the likelihood that the uncertain object is indeed placed in the given query interval. Let us consider the contributions more exactly. In [2] a new uncertainty model is introduced and several new types of queries are described that allow the handling of inexact data. [5] introduced the U-Tree for indexing uncertain 2D objects. Authors of [20] adapt the Gauss-tree proposed in [6] to spatial uncertainty model and discuss *probabilistic ranking queries*. The authors of [21] address the evaluation of probabilistic queries over uncertain data in constantly-evolving environments and discuss among others the so-called *probabilistic nearest neighbor query* that retrieves the set of objects being closest to a query object. All of these contributions employ the so-called interval uncertainty model and pose queries based on intervals in the data space. Thus, the mentioned approaches rather deal with spatial uncertainty (i.e., the location of an object is considered to be uncertain). Besides the mentioned methods for indexing spatially uncertain objects, [22] introduces *existential uncertainty*. The idea of this approach is that the existence of each data object is uncertain.

Uncertainty in Object Identification. In [6], the Gauss-tree is introduced which is an index structure for managing large amounts of Gaussian distribution functions. The proposed system aims at efficiently answering so-called identification queries. Additionally, [6] proposed *probabilistic identification queries* which are based on a Baysian setting (i.e., given a query pdf, retrieve those pdfs in the database that correspond to the query pdf with the highest probability). An example for this setting is: Given a facial image represented by a normal distribution, retrieve the person in the database whose face corresponds to the query image with the highest probability. However, the methods in [6] underly the limitation that each object can only be described by a single, axis-parallel Gaussian. In this paper, we approximate arbitrary probability distributions that are approximated by GMMs. Furthermore, unlike [6] our method facilitates un-

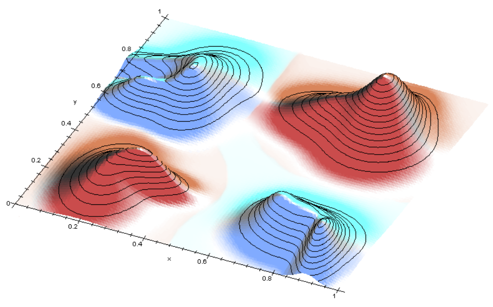


Fig. 3. Two uncertain objects (i.e., (dark) red object and (light) blue object) modeled by GMMs.

successful search which is an important feature for several application areas (e.g., the content-based retrieval of multimedia data).

Indexing Techniques for Uncertain Objects. The indexing methods introduced in [3, 5, 6] have serious problems that strongly limit their use for applications with objects modeled by GMMs. The first limitation of the introduced uncertainty models is that each object is represented as a single multivariate distribution. Though this type of pdf is rather common, it is for many of the above mentioned applications too simplifying. Figure 3 illustrates two uncertain objects modeled by GMMs (object A in blue and object B in red) in order to exemplify the problem. Each probabilistic object consists of two distinct sets of Gaussians that are placed on a diagonal in oppositional edges of feature space. The indexing techniques described in [3, 5, 6] would approximate each object with a node N that ranges over whole feature space (i.e., from 0 to 1 in each dimension). Therefore, the selectivity of query processing algorithms is very low (i.e., we should access all available nodes of an index structure while performing a probabilistic query). Additionally, several of the existing methods (e.g., [3, 6]) are based on distributions that do not consider any correlation between the dimensions of the underlying feature space which is a rather problematic assumption even for application were Gaussians are rather well-suited. Last, but not least, the models derive probabilities based on the assumption that the object which is searched for is always stored in the database. If the query object is unknown, the model does not necessarily calculate a small probability because the probability is relative to the total probability that the query belongs to at least one object in the database. Recently, [23]introduced a method and a corresponding index structure modeling pdfs using piecewise-linear approximations. This new approach also employs linear functions as the U-Tree but is more exact in its approximation.

GMMs in Multimedia Retrieval. In our experimental evaluation, we test our general approach of indexing arbitrary probability functions on a database of video clips being represented by GMMs. To demonstrate that this method is competitive with existing approaches to video retrieval, we will sketch some relevant work in this area. In [9], a summarization technique is presented which is based on GMMs. However, the authors do not propose any technique for speeding up the search on these summarizations. Furthermore, the named approach yields a specialized solution for video retrieval and is thus not concerned with the efficient search in general probabilistic data sets. The authors of [24] propose an approach for obtaining a compact representation of videos that computes the optimal representatives by minimizing the Hausdorff distance between the original video and its representation. If the Euclidian metric is used, k-means can be applied to summarize video clips [25] by a set of centroides. A randomized technique for summarizing videos, called video signature, is proposed in [26]. A video sequence in the database is described by selecting a number of its frames closest to a set of random vectors. The authors of [26] also propose a specialized distance function on the derived summarization vectors. However, to the best of our knowledge, none of these techniques uses an index structure to accelerate query processing.

3 Probabilistic Similarity Search using a Mixture of Gaussians

3.1 Gaussian Mixture Model (GMM)

We assume that our objects are given by a probability density function (pdf) over a d-dimensional feature space $\mathbb{R}^d$. The pdf is defined in terms of a mixture of Gaussians, as specified in the following definition:

Definition 1 (Gaussian Mixture Model). *Let $x \in \mathbb{R}^d$ be a variable from a d-dimensional feature space. A Gaussian mixture model (GMM) G_O of an object O is the following probability density function corresponding to a weighted sum of k_O Gaussian functions:*

$$G_O(x) = \sum_{1 \leq i \leq k_O} W_{O,i} \cdot N_{O,i}(x) \tag{1}$$

where $W_{O,i}$ is a weight factor $0 \leq W_{O,i} \leq 1$ with

$$\sum_{1 \leq i \leq k_O} W_{O,i} = 1 \tag{2}$$

and $N_{O,i}(x)$ is the density of a multivariate normal distribution:

$$N_{O,i}(x) = \prod_{1 \leq l \leq d} N_{\mu_{O,i,l},\sigma^2_{O,i,l}}(x_l) = \prod_{1 \leq l \leq d} \frac{1}{\sqrt{2\pi\sigma^2_{O,i,l}}} \cdot \boldsymbol{e}^{\frac{-(x_l - \mu_{O,i,l})^2}{2\sigma^2_{O,i,l}}} . \tag{3}$$

Note that we assume the attribute independence of the *single* Gaussians involved in a GMM (i.e., axis-parallel Gaussians). This assumption does not limit the generality of our approach, because it can be shown that a mixture of axis-parallel Gaussians is able to model arbitrary pdfs including those distributions where some of the attributes are strongly correlated to each other [8, 7]. As an example, consider Figure 2 where a 2D Gaussian distribution with a strong correlation is modeled by a mixture of $k = 5$ axis-parallel Gaussians. Thus, the attribute independence assumption of the single Gaussian does not imply an attribute independence of the pdf modeled by the complete GMM.

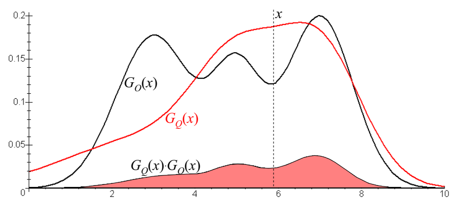


Fig. 4. An example of two GMM in univariate space.

3.2 Similarity of Objects Described by GMMs

After defining a GMM for describing an uncertain object, we now turn to the question how to determine the probability of an object O to be a *hit* for a query object Q. As a starting point, we assume that the query Q is provided as by an exact feature vector (i.e., a point $x \in \mathbb{R}^d$). In this case, Formula 1 can be used in combination with the Bayes theorem to determine this probability:

$$P(O|x) = \frac{P(O) \cdot G_O(x)}{\sum_{U \in DB} P(U) \cdot G_U(x)}.$$

Since it makes sense to assume that each database object is part of the answer with the same prior probability, $P(O) = P(U)$ for all $O, U \in DB$ and thus, we can cancel out the prior probability from the formula. Please note that the use of probability densities instead of probabilities is common practice in Bayes classification, as can be verified in [27, 28].

The situation becomes more complex if both data and query object (O and Q) are given as a GMM. We have to find a probability measure which indicates the probability that both GMMs O and Q correspond to the same unknown *true* object x (which is a traditional vector, and is used as a stochastic variable here). In general, we know nothing about x except that it is taken from $\mathbb{R}^d$. For every position of x in this data space, we can determine the probability density of O and Q, respectively. Both are given by a pdf modeled as a GMM. But for every position x, we can also determine the probability density with which x belongs to both O and Q. As belonging to O and belonging to Q are independent events, this is simply the product of the two corresponding pdfs. Since the true object x can be any point of $\mathbb{R}^d$, we have to form a d-dimensional volume integral over all possible positions of x to determine the overall probability that O and Q correspond to the same unknown x.

A one-dimensional example is visualized in Figure 4: Two objects, Q and O, each modeled as a mixture of $k = 3$ Gaussians are depicted. We have also indicated x, a *possible* position of the unknown true object. We can see from the diagram the probability density with which x belongs to O (approximately 0.12), to Q (0.18) and to both of them ($0.12 \cdot 0.18 \simeq 0.02$). Since all $x \in \mathbb{R}$ are possible, the overall probability of O and Q to belong to the same object corresponds to the integral of $G_Q(x) \cdot G_O(x)$ where $x = -\infty .. + \infty$ (i.e., the shaded area in Figure 4). We call this probability density

$p(Q|O)$ with

$$p(Q|O) = \iint_{-\infty}^{+\infty} G_O(x) \cdot G_Q(x) \mathrm{d}\boldsymbol{x}. \tag{4}$$

The theorem of Bayes can again be used to determine the result probability of every database object O like in the case of single-valued query objects (cf. Equation 3.2):

$$P(O|Q) = \frac{p(Q|O)}{\sum_{P \in DB} p(P|O)}. \tag{5}$$

We follow the convention to use the abbreviation $\iint_{-\infty}^{+\infty} f(x)\mathrm{d}\boldsymbol{x}$ for the so-called volume integral $\int_{-\infty}^{+\infty} \ldots \int_{-\infty}^{+\infty} f(x)\mathrm{d}x_1 \ldots \mathrm{d}x_d$. Note that, although only two integral signs are written, they actually stand for a d-fold integral. In the following, we show how the overall probability density $p(Q|O)$ defined in Formula 4 can be computed analytically:

$$\begin{aligned}
p(Q|O) &= \iint_{-\infty}^{+\infty} G_O(x) \cdot G_Q(x) \mathrm{d}\boldsymbol{x} \\
&= \iint_{-\infty}^{+\infty} \left(\sum_{1 \le i \le k_O} W_{O,i} N_{O,i}(x) \right) \left(\sum_{1 \le j \le k_Q} W_{Q,j} N_{Q,j}(x) \right) \mathrm{d}\boldsymbol{x} \\
&= \sum_{1 \le i \le k_O} \sum_{1 \le j \le k_Q} W_{O,i} \cdot W_{Q,j} \cdot \iint_{-\infty}^{+\infty} N_{O,i}(x) \cdot N_{Q,j}(x) \mathrm{d}\boldsymbol{x}.
\end{aligned}$$

The volume integral can be solved by the following re-arrangement of terms:

$$\iint_{-\infty}^{+\infty} N_{O,i}(x) \cdot N_{Q,j}(x) \mathrm{d}\boldsymbol{x} = = \iint_{-\infty}^{+\infty} \prod_{1 \le l \le d} N_{\mu_{O,i,l}, \sigma_{O,i,l}}(x) \cdot N_{\mu_{Q,j,l}, \sigma_{Q,j,l}}(x) \mathrm{d}\boldsymbol{x}$$

because of attribute independence

$$\begin{aligned}
&= \prod_{1 \le l \le d} \int_{-\infty}^{+\infty} \frac{\mathrm{e}^{\frac{-(x_l - \mu_{O,i,l})^2}{2\sigma_{O,i,l}^2}} \cdot \mathrm{e}^{\frac{-(x_l - \mu_{Q,j,l})^2}{2\sigma_{Q,j,l}^2}}}{2\pi \sigma_{O,i,l} \sigma_{Q,j,l}} \mathrm{d}x_l \\
&= \prod_{1 \le l \le d} \frac{1}{\sqrt{2\pi(\sigma_{O,i,l}^2 + \sigma_{Q,j,l}^2)}} \cdot \mathrm{e}^{\frac{-(\mu_{Q,j,l} - \mu_{O,i,l})^2}{2(\sigma_{O,i,l}^2 + \sigma_{Q,j,l}^2)}} \\
&\quad \cdot \int_{-\infty}^{+\infty} N_{\frac{\mu_{O,i,l}\sigma_{Q,i,l}^2 + \mu_{Q,i,l}\sigma_{O,i,l}^2}{\sigma_{Q,i,l}^2 + \sigma_{O,i,l}^2}, \frac{\sigma_{O,i,l}^2 \sigma_{Q,i,l}^2}{\sigma_{O,i,l}^2 + \sigma_{Q,i,l}^2}}(x_l)\mathrm{d}x_l \\
&= \prod_{1 \le l \le d} \frac{1}{\sqrt{2\pi(\sigma_{O,i,l}^2 + \sigma_{Q,j,l}^2)}} \cdot \mathrm{e}^{\frac{-(\mu_{Q,j,l} - \mu_{O,i,l})^2}{2(\sigma_{O,i,l}^2 + \sigma_{Q,j,l}^2})} \cdot 1 \\
&= \prod_{1 \le l \le d} N_{\mu_{O,i,l}, \sigma_{O,i,l}^2 + \sigma_{Q,j,l}^2}(\mu_{Q,j,l}).
\end{aligned}$$

Therefore, the overall probability density $p(Q|O)$ corresponds to

$$p(Q|O) = \sum_{\substack{1 \le i \le k_O \\ 1 \le j \le k_Q}} W_{O,i} W_{Q,j} \prod_{1 \le l \le d} N_{\mu_{O,i,l}, \sigma^2_{O,i,l} + \sigma^2_{Q,j,l}}(\mu_{Q,j,l}).$$

3.3 Handling Unknown Objects

In many applications, we have to consider the case that the query object is not necessarily stored in the database. Thus, an algorithm should allow that the result of the query is empty (i.e., it is most likely that the given query does not fit to any of the database objects). In other words, no object can be considered as similar at all. Our solution to this problem is to introduce an additional probability distribution modeling the likelihood of an unknown data object.

When applying the Bayes theorem (cf. the Equation 5), we make the implicit assumption that the object which is searched for is always stored in the database, and that all stored database objects have the same *a priori* probability of being the result of the search. Therefore, the *sum* of all result probabilities of all database objects equals 1 for every query. If a query object is at a position of extremely low density , then some of the database objects (particularly those with high overall variance) still may obtain high result probabilities, which contradicts the intuition. In terms of the Bayesian probability model, the conditional probability might become rather large if the total probability in the denominator is very small. Since a conditional probability is normalized by the total density, a small probability density might lead to a large probability if the total density is very small.

This problem can be solved by modeling a placeholder for those objects which are not stored in the database. This placeholder should have a high variance to cover the complete data space and a prior probability P_{PH} corresponding to the expected rate of unsuccessful searches. For example, if 3 out of 4 search operations are not successful (the intended object is not found in the database), then $P_{PH} = 0.75$, and the remaining prior probability (0.25) is shared by the database objects. The value of P_{PH} can be set at the begin of database usage to a default value. The value of P_{PH} should be adapted when collecting exact statistics about unsuccessful queries during database deployment.

The probability distribution of the placeholder can e.g. be selected to be uniformly or normally distributed. Assuming that the data distribution of the unknown objects follows that of the known objects, we can use the mean and the variance of the object set. Let us not that these should not be confused with the intra-object variance describing the uncertainty of the feature value of a *single* object:

$$\mu_{PH,l} = \frac{1}{|DB|} \sum_{O \in DB} \sum_{1 \le i \le k_O} W_{O,i} \cdot \mu_{O,i,l}$$

$$\sigma^2_{PH,l} = \frac{1}{|DB| - 1} \sum_{O \in DB} \sum_{1 \le i \le k_O} W_{O,i} \cdot (\mu_{O,i,l} - \mu_{PH,l})^2.$$

To consider the placeholder, we have to extend our formula for calculating $P(O|Q)$ in the following way:

$$P(O|Q) = \frac{\frac{1-P_{PH}}{|DB|} \cdot p(Q|O)}{P_{PH} \cdot P(Q|PH) + \sum_{P \in DB} \frac{1-P_{PH}}{|DB|} \cdot p(P|O)}. \tag{6}$$

3.4 Probabilistic Ranking Query on GMMs

A method for deciding containment in the query result is to retrieve the k most likely results. An example for this type of query is: Retrieve the 5 objects from the database having the highest probability for corresponding to the query object. We will call this type of query *probabilistic ranking query (PRQ)*. In the following we will formalize PRQs:

Definition 2 (Probabilistic Ranking Query).
Let DB be a database of GMMs M, let Q be a query GMM and let $k \in \mathbb{N}$ be a natural number. Then, the answer to a probabilistic ranking query (PRQ) on DB is defined as the smallest set $RQ_k(Q) \subseteq DB$ with at least k elements fulfilling the following condition:

$$\forall M_a \in RQ_k(Q), \forall M_{db} \in DB \setminus RQ_k(Q) : P(M_a|Q) > P(M_{db}|Q)$$

4 Indexing Mixtures of Gaussians

4.1 General Idea

Our method is based on the idea of decomposing each GMM into its components (i.e., the single Gaussians, and index the components separately). For instance, in order to store a univariate GMM with $W_1 = 0.3, W_2 = 0.7, \mu_1 = 0, \mu_2 = 1, \sigma_1 = 2, \sigma_2 = 3$, we decompose the GMM in two components g_1 and g_2. The first component g_1 is described by $W_1 = 0.3, \mu_1 = 0, \sigma_1 = 2$. The second component g_2 is described by $W_2 = 0.7, \mu_2 = 1, \sigma_2 = 3$. After decomposition, we can store two components (i.e., g_1 and g_2) separately in an index that is appropriate for single pdfs. During query processing, the complete GMMs can be processed componentwise and thus, we can start excluding objects from the result set, even without retrieving the complete GMM.

To index the components of the stored GMMs, our method can employ any hierarchically organized index structure Δ that is capable to store single pdfs (e.g., [3, 5, 6] that are based on the R-Tree [29] principle). Since Δ is designed to handle single pdf, we have to extend it to additionally store the weights for each component. Thus, an entry which corresponds to a GMM component O_i, consists of the parameters of the underlying Gaussian $\mu_{O,l}$ and $\sigma^2_{O,l}$ $(1 \leq l \leq d)$ and the weight of the component $W_{O,i}$. This way, it is possible to reconstruct the complete GMM after retrieving all of its components from the index structure Δ. In addition to extending the entries in the leaf nodes, we have to extend the node descriptions by the maximum weight W^p_{max} of any Gaussian being stored in the corresponding subtree.

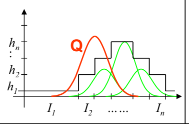
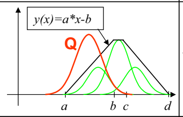
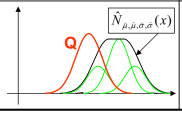
Index Structure	Idea of Approximation (P - Directory Node)	Sketch of Formula for N_{LB}(P,Q, j)
Approach of [R. Cheng, Y. Xia, S. Prabhakar et al.]	 Q h_n h_2 h_1 I_1 I_2 …… I_n	$N_{LB}(P,Q,j)=\sum_{k=1}^{n}\int_{I_{k-1}}^{I_k}(N_{Q,j}(x)\cdot h_k)dx$
U-tree		$N_{LB}(P,Q,j)=\int_a^b(N_{Q,j}(x)\cdot y(x))dx+\int_b^c...+..$
Gauss-tree	 $\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$ Q	$N_{LB}(P,Q,j)=\int_{-\infty}^{+\infty}(N_{Q,j}(x)\cdot\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x))dx$

Fig. 5. Idea of the approximation and sketch of the conservative approximation calculation ($N_{LB}(P,Q,j)$) for different index structures that handle single pdfs.

The key idea of our proposed query algorithms is that it is possible to calculate the probability $P(O|Q)$ componentwise. Thus, if we can calculate a conservative approximation of the components of $P(O|Q)$,

$$comp(O,i,Q,j) = W_{O,i}W_{Q,j}\prod_{1\leq l\leq d} N_{\mu_{O,i,l},\sigma^2_{O,i,l}+\sigma^2_{Q,j,l}}(\mu_{Q,j,l})$$

it is possible to calculate conservative approximations for completely and partially retrieved GMMs during query processing.

Therefore, we define a conservative approximation for any component Q_i and a given page P in the index structure Δ which maximizes $comp(O,i,Q,j)$ over all O_i that are stored in the subtree corresponding to P:

$$comp^P_{max}(Q,j) = W^P_{max}W_{Q,j}\prod_{1\leq l\leq d} N_{LB}(P,Q,j)$$
$$\geq \max_{O,i\in P} W_{O,i}W_{Q,j}\prod_{1\leq l\leq d} N_{\mu_{O,i,l},\sigma^2_{O,i,l}+\sigma^2_{Q,j,l}}(\mu_{Q,j,l})$$

where $N_{LB}(P,Q,j)$ is lower bound or maximum of probability density that can be achieved in a node or subtree of the underlying index structure and W^P_{max} is the maximum of the weights of all components stored in P. In particular, $N_{LB}(P,Q,j)$ can be calculated by:

- evaluation of one or several entries in so-called *ratio table* w.r.t. *x-bounds* if employing the indexing technique described in [3],
- using the so-called *U-Catalog* and the function *e.MBR(.)* that allows pruning subtrees if employing the U-tree [5],
- using the density of the *hull function* $\hat{N}_{...}$ if employing the Gauss-tree [6].

Figure 5 depicts the general idea of the computation of the conservative approximation $N_{LB}(P, Q, j)$ for the above mentioned index structures.

Furthermore, we define $comp^P_{max}(Q)$ approximating the probability for the complete query GMM Q and an arbitrary component stored in P as:

$$comp^P_{max}(Q) = |Q| \cdot \max_{1 \leq j \leq |Q|} comp^P_{max}(Q, j)$$

Thus, we have found a way to estimate the maximum influence of any GMM component stored in page P on the probability $P(O|Q)$ for any object O in the database that has not yet been retrieved completely. For query processing, $comp^P_{max}(Q)$ is useful for several tasks. First of all, we can use $comp^P_{max}(Q)$ to estimate the maximum probability $P(O'|Q)$ for the GMM O' which is stored in P and there is no component of O' that has been retrieved during query processing yet.

$$P(O'|Q) = k_{max} \cdot comp^P_{max}(Q)$$

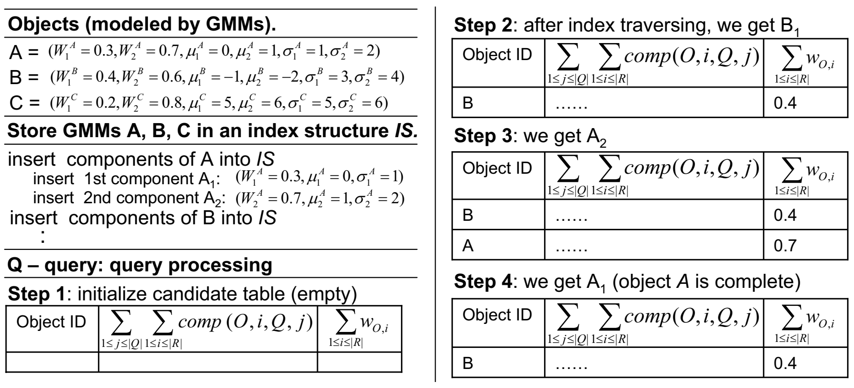
where k_{max} denotes the maximum number of components in any GMM in the database. Furthermore, we can approximate the maximal probability of a partially retrieved GMM O to match Q. This is important for pruning candidate GMMs as early as possible:

$$P(O|Q) = \sum_{1 \leq j \leq |Q|} \sum_{1 \leq i \leq |R|} comp(O, i, Q, j) + (1.0 - \sum_{1 \leq i \leq |R|} W_{O,i}) \cdot comp^P_{max}(Q)$$

where R is the set of already retrieved components of O and P is the page having the maximum value of $comp^P_{max}(Q)$. Note, that the use of $1.0 - \sum_{1 \leq i \leq |R|} w_{O,i}$ for all remaining weights is justified by Formula 2. The estimation of the sum of all densities is necessary to apply Formula 6. We can sum up the densities or their conservative approximations as described above during query processing. Thus, it is possible to determine a conservative approximation of the complete density of a candidate GMM at all times.

Before we explain our probabilistic query processing on GMMs in detail, we take a closer look at the data structure which manages the result candidates. This candidate table can either be located in main memory or on secondary storage, depending on the size of a particular dataset. Every component of a GMM which is fetched from the underlying index structure is stored in this table. An object is removed from the table under two conditions. Either all components of a particular object are in the candidate table (cf. Figure 5) or the object is pruned based on the conservative approximation of the object probability.

In the following we will introduce our algorithm for processing PRQs. For this type of query, the minimum probability for a result depends on the object having the k-highest probability for corresponding to the query GMM Q. Therefore, we employ two priority queues: The first priority queue is used for traversing the probabilistic index structure storing the components of the GMMs. We will refer to this queue as the traversal queue. The second priority keeps those k GMMs which currently have the largest probabilities for corresponding to Q. We will sort this second priority queue in ascending order and refer to it as the result queue. The pseudo code for the algorithm is



Objects (modeled by GMMs).

A = $(W_1^A = 0.3, W_2^A = 0.7, \mu_1^A = 0, \mu_2^A = 1, \sigma_1^A = 1, \sigma_2^A = 2)$

B = $(W_1^B = 0.4, W_2^B = 0.6, \mu_1^B = -1, \mu_2^B = -2, \sigma_1^B = 3, \sigma_2^B = 4)$

C = $(W_1^C = 0.2, W_2^C = 0.8, \mu_1^C = 5, \mu_2^C = 6, \sigma_1^C = 5, \sigma_2^C = 6)$

Store GMMs A, B, C in an index structure *IS*.

insert components of A into *IS*

insert 1st component A_1: $(W_1^A = 0.3, \mu_1^A = 0, \sigma_1^A = 1)$

insert 2nd component A_2: $(W_2^A = 0.7, \mu_2^A = 1, \sigma_2^A = 2)$

insert components of B into *IS*

:

Q – query: query processing

Step 1: initialize candidate table (empty)

Object ID	$\sum_{1\le j\le\|Q\|}\sum_{1\le i\le\|R\|} comp(O,i,Q,j)$	$\sum_{1\le i\le\|R\|} w_{O,i}$

Step 2: after index traversing, we get B_1

Object ID	$\sum_{1\le j\le\|Q\|}\sum_{1\le i\le\|R\|} comp(O,i,Q,j)$	$\sum_{1\le i\le\|R\|} w_{O,i}$
B	……	0.4

Step 3: we get A_2

Object ID	$\sum_{1\le j\le\|Q\|}\sum_{1\le i\le\|R\|} comp(O,i,Q,j)$	$\sum_{1\le i\le\|R\|} w_{O,i}$
B	……	0.4
A	……	0.7

Step 4: we get A_1 (object *A* is complete)

Object ID	$\sum_{1\le j\le\|Q\|}\sum_{1\le i\le\|R\|} comp(O,i,Q,j)$	$\sum_{1\le i\le\|R\|} w_{O,i}$
B	……	0.4

Fig. 6. General idea of the candidate table used in our query algorithms on GMMs.

displayed in Figure 7. We start by ordering the descendant nodes of the root page w.r.t. $comp^p_{max}(Q)$. Afterwards we enter the main loop of the algorithm and remove the top element of the traversal queue. If this element is a node, we load its child nodes. If these child nodes are nodes themselves, we determine $comp^p_{max}(Q)$ and update the traversal queue. If the child nodes are components of GMMs, we check the candidate table for a corresponding GMM M and insert a new descriptor, in the case that there is not already a descriptor for M. Afterwards, we can update the candidate table as mentioned before. If a GMM M has been read completely, we can delete it from the candidate table and compare its probability $P(M|Q)$ to the probability of the top element of the result queue (i.e., the GMM encountered so far having the k highest probabilities). If the probability of M is higher than that of the top element, we need to add M to the queue. However, to make sure that we do not retrieve more than k elements, we have to check the size of the result queue. If there are already k elements, we have to remove the previous top element before inserting M. In the case, that the entry in the candidate table does not contain the complete information about M yet, we still can calculate a probability estimation and compare it to the top element of the result queue. If $P(M|Q)$ is smaller than the k highest probability in the result queue, we can guarantee that M is not a potential result. Thus, M is deleted from the candidate table and stored in our list for excluded GMMs. The algorithm terminates if the top of the traversal queue provides a lower value than the top of the result queue and the candidate table is empty. Please note that it is not necessary to calculate any complex integral functions during query processing. As shown in Section 3.2 it is sufficient to compute a weighted product of probability densities during query execution.

5 Experimental Evaluation

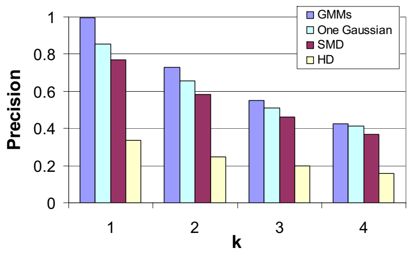
To demonstrate the effectiveness and efficiency of our new approach, we implemented the proposed methods in Java 1.5. All experiments were performed on a workstation equipped with a 2.2 GHz Opteron CPU and 8GB RAM. In order to store components

```
ALGORITHM PRQ(Query Q, integer $k$)
BEGIN
  $hits$ := new PriorityQueue(ascending)
  $drops$ := new List()
  $candidates$ := new List()
  $APL$ := new PriorityQueue(descending)
  $APL$.insert(root,$1 - P_{PH}$)
  DO
    $currNode$ := $APL$.removeFirst()
    IF $currNode$.isDirectoryNode() = '$TRUE$' THEN
      FOR EACH $node \in currNode$ DO
        APL.insert($node$, $comp^{node}_{max}(Q)$)
      END FOR
    IF $currNode$.isDataNode() = '$TRUE$' THEN
      FOR EACH $c \in currNode$ DO
        IF $drops$.contains(c.ObjectID) = '$TRUE$' THEN
          CONTINUE
        END IF
        $candidates$.update($c$.ObjectID, $c$, Q)
        $entry$ := $candidates$.get($c$.ObjectID)
        IF $entry$.isComplete() = '$TRUE$' THEN
        $prob$ := $entry$.probability(Q)
          IF $prob \geq hits$.topProbability THEN
            IF $hits$.Size() = $k$ THEN
              $hits$.removeFirst()
            END IF
            $hits$.add($c$.ObjectID, $prob$)
          END IF
          $candidates$.delete($c$.ObjectID)
        ELSE
          IF $entry$.approximation(Q) $\leq$
            $hits$.topProbability THEN
            $drops$.add($c$.ObjectID)
            $candidates$.delete($c$.ObjectID)
          END IF
        END IF
      END FOR
    END IF
  WHILE(($candidates$.Size() $>$ 0
    or $APL$.topProbability $>$ $hits$.topProbability)
    and $APL$.Size() $>$ 0)
  report $hits$;
END
```

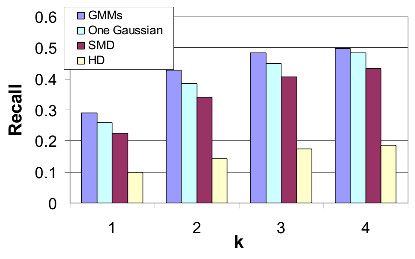
Fig. 7. Pseudo code: Probabilistic Ranking Query on GMMs.

of GMMs, we implemented an X-Tree [30] variant that uses split and insert methods as proposed in [6], and the technique for calculation of conservative approximations on Gaussians (i.e., for calculation of N_{LB} in Formula 7 of Section 4) as described in [6].

Testbed. In order to demonstrate, that our new uncertainty model yields an competitive solution for content-based object retrieval in multimedia databases, we tested our uncertainty model based on GMMs on a dataset of 1,050 music video clips. The average length of a video is 4 minutes 14 seconds. To transform the videos into GMMs, we first of all extracted the set of all frames contained in a video and transformed each frame into a color histogram. Thus, each video was represented by a set of 3,000 up to 10,000 single images. To model these set of images as a GMM, we applied Expectation Maximization clustering. Afterwards each video was represented by a GMM consisting of 100 Gaussians. Let us note that some of the videos are described by a smaller number of Gaussians because the clustering generated empty clusters without any weight in the GMM. Thus, the testbed consists of about 100,000 16-dimensional Gaussians which

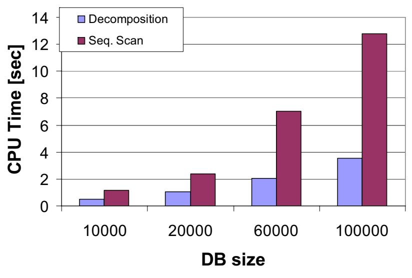


(a) Precision.

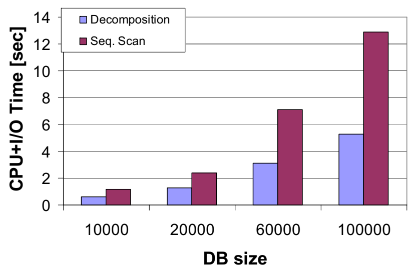


(b) Recall.

Fig. 8. Precision and recall for PRQ on GMMs and comparison partners.



(a) CPU-time.



(b) CPU+IO time

Fig. 9. Efficiency for PRQ (k=3) on synthetic data for varying database size.

correspond to approximately 150 GB of video data. To demonstrate that our new uncertainty model is competitive to other methods for content-based video retrieval, we additionally generated a database representing each video as a set of color histograms. However, since using thousands of feature vectors for representing a video clip of 3 to 5 minutes is not feasible, we had to reduce the number of employed feature vectors considerably. Thus, each video is described by selecting every 50th frame in chronological order. To query a database describing a video as set of feature vectors, we employ well known distance functions for set-valued objects like the sum of minimum distances (SMD) and the Hausdorff distance (HD) [31]. Furthermore, we generated a third database representing each data object as a single Gaussian, instead of a GMM.

Effectiveness. We compare the results of our new method to the mentioned comparison partners for video retrieval w.r.t. precision and recall. Therefore, we posed ranking queries, subsequently retrieving the 4 objects in the database that most likely match the query object. The queries consist of 40 videos for which there exist multiple versions in the database corresponding to different versions or having a different recording quality. Thus, we can measure precision and recall for these query videos. For each size

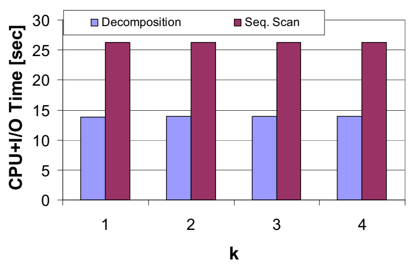


Fig. 10. CPU+IO time for PRQ on the video dataset.

of the result set $(k = 1, \ldots, 4)$, we measured precision and recall. Figure 8 displays the precision and the recall achieved by all 4 compared methods. Our new approach for modeling uncertain objects as GMMs outperformed all other methods w.r.t. precision as well as recall. For example, the GMM based model displayed a more than 20 % better precision for $k = 1$ than the best of its comparison partners. Thus, we can state that modeling a video as a GMM instead of a single Gaussian significantly increased the quality of object representation and therefore, the precision and the recall of the result sets improved as well. Furthermore, the result indicates that modeling videos as GMMs is superior to modeling videos as sets of feature vectors.

We additionally tested the capability, of our new method to cope with query objects that are not stored in the database. The result indicates a probability for unsuccessful queries that is less than 6%. Since such a small probability value is not observed for any result of a successful search, we assume an unsuccessful search for result probabilities of less then 10 %.

Efficiency. After demonstrating the usefulness of our new approach for uncertain objects, we now examine the efficiency of our new method to index GMMs. To the best of our knowledge, the method proposed in this paper is the first approach for efficiently managing large sets of GMMs. Thus, we compare our method to the basic approach that no index is available and the result set has to be determined by a sequential scan over the database. Since measuring real disc accesses is often not significant due to the existence of disc caches, we measure the IO performance in this section by counting IO operations and afterwards add up the cost for the counted disk accesses based on a transfer rate of 50 MB/s and an access time of 6 ms. We evaluated the efficiency of our new approach on the real world video data set. The result is displayed in Figure 10. For all query parameters our new PRQ method was capable to speed up query processing to a factor of approximately 2. Furthermore, the query time was comparably stable for the given query parameters, e.g. the query time did not significantly increase for larger values of k. Since the video data set is rather small, we generated an artificial data set of up to 100,000 GMMs to examine the scalability of our approach. These GMMs consisted of up to 10 Gaussians over an 2D data space. The results are displayed in Figure 9. In this figure, we distinguish between the CPU query time (cf. Figure 9(a)) and the complete query time (cf. Figure 9(b)). The speedup of our new approach increased with the size of the database. Thus, using our approach is more beneficial for larger data sets.

6 Conclusions

In this paper, a new uncertainty model is proposed which represents uncertain objects by Gaussian mixture models (GMM). A GMM can approximate arbitrary multivariate probability distribution and thus, GMMs are applicable in a wide variety of new applications like multimedia retrieval, sensor networks, medicine, geology. To use GMMs for query processing, we demonstrate that the probability that a query GMM corresponds to a database GMM can be calculated analytically. Furthermore, our model is capable of handling unsuccessful queries (i.e., if the database does not contain a similar data object the results will have a very low result probability). Thus, we can recognize that there is most likely no corresponding object in the database. After introducing the uncertainty model, we examine probabilistic ranking queries as an important query type. To speed up these queries, we propose a new indexing technique which is based on object decomposition. The key idea of this method is to store each component of each GMM separately in an index structure for single pdfs like [3, 5, 6]. During query processing, we can prune certain GMMs based on their already retrieved components. Additionally, the result GMMs are reconstructed while searching the database. Thus, our new method is capable of efficiently retrieving the GMMs in the database that resemble a query GMM with maximum likelihood. In our experimental evaluation, we demonstrate the effectiveness of our new approach for content-based multimedia retrieval on a data set of 1,050 video clips. Additionally, we show that the new query algorithm employing object decomposition yields a significant speed up compared to sequential query processing on a synthetic data set and the named data set of video clips.

References

1. Faradjian, A., Gehrke, J., Bonnet, P.: ”GADT: A Probability Space ADT For Representing and Querying the Physical World”. In: Proc. 18th Int. Conf. on Data Engineering (ICDE’02), San Jose, CA, USA. (2002) 201
2. Cheng, R., Kalashnikov, D., Prabhakar, S.: ”Evaluating Probabilistic Queries over Imprecise Data”. In: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD’03), San Diego, CA, USA). (2003) 551–562
3. Cheng, R., Xia, Y., Prabhakar, S., Shah, R., Vitter, J.: ”Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data”. In: Proc. 30th Int. Conf. on Very Large Data Bases (VLDB’04), Toronto, Cananda. (2004) 876–887
4. Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., Hong, W.: ”Model-driven data acquisition in sensor networks”. In: Proc. 30th Int. Conf. on Very Large Data Bases (VLDB’04), Toronto, Cananda. (2004)
5. Tao, Y., Cheng, R., Xiao, X., Ngai, W., Kao, B., Prabhakar, S.: ”Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions”. In: Proc. 30th Int. Conf. on Very Large Data Bases (VLDB’05), Trondheim, Norway. (2005) 922–933
6. Böhm, C., Pryakhin, A., Schubert, M.: ”The Gauss-Tree: Efficient Object Identification of Probabilistic Feature Vectors”. In: Proc. 22nd Int. Conf. on Data Engineering (ICDE’06)),Atlanta,GA,US. (2006) 9
7. Titterington, D.M., Smith, A.F.M., Makov, U.E.: Statistical analysis of finite mixture distribution. new york: Wiley (1985)
8. Lindsay, B.G.: Mixture models: Theory, geometry, and applications (1995)

9. Greenspan, H., Goldberger, J., Mayer, A.: A probabilistic framework for spatio-temporal video representation & indexing. In: ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV, London, UK, Springer-Verlag (2002) 461–475
10. Yang, M., Ahuja, N.: Gaussian mixture model for human skin color and its application in image and video databases. In: Proc. of the Conf. on Storage and Retrieval for Image and Video Databases (SPIE 99). Volume 3656. (1999) 458–466
11. Chen, S.C., Kashyap, R., Ghafoor, A.: "Semantic Models for Multimedia Database Searching and Browsing". Kluwer Academic Publishers (2002)
12. Srinivasan, U., Nepal, N.: Managing Multimedia Semantics. IRM Press (2005)
13. Deb, S.: Video Data Management and Information Retrieval. Idea Group Publishing (2005)
14. Gavin, D.G., Hu, F.S.: Bioclimatic modelling using gaussian mixture distributions and multiscale segmentation. Global Ecology and Biogeography **14** (2005) 491
15. Lim, P., Quek, S., Peh, K.: Application of the gaussian mixture model to drug dissolution profiles prediction. Neural Comput. Appl. **14**(4) (2005) 345–352
16. Zajdel, W., Kröse, B.: Gaussian mixture model for multi-sensor tracking. In: Proc. of the 15th Dutch-Belgian Artificial Intelligence Conference (BNAIC'03). (2003) 371–378
17. Reynolds, D., Quatieri, T., Dunn, R.: Speaker verification using adapted gaussian mixture models. Digital Signal Processing **10**(1) (2000) 19–41
18. Yoo, S.H.: Application of a mixture model to approximate bottled water consumption distribution. Applied Economics Letters **10**(3) (2003) 181–184
19. Deshpande, A., Guestrin, C., Madden, S.R.: "Using Probabilistic Models for Data Management in Acquisitional Environments". In: Proc. CIDR. (2005)
20. Böhm, C., Pryakhin, A., Schubert, M.: "Probabilistic Ranking Queries on Gaussians". In: Proc. of the 18th Int. Conf. on Scientific and Statistical Database Management (SSDBM'06). (2006) 169–178
21. Cheng, R., Kalashnikov, D.V., Prabhakar, S.: "Evaluation of Probabilistic Queries over Imprecise Data in Constantly-Evolving Environments". **32**(1) (2007) 104–130
22. Dai, X., Yiu, M.L., Mamoulis, N., Tao, Y., Vaitis, M.: "Probabilistic Spatial Queries on Existentially Uncertain Data". In: Proc. 9th Int. Symposium on Spatial and Temporal Databases (SSTD'05), Angra dos Reis, Brazil. (2005) 400–417
23. Ljosa, V., Singh, A.K.: APLA: Indexing arbitrary probability distributions. In: Proc. of the 23rd Int. Conf. on Data Engineering (ICDE 2007). (2007)
24. Chang, H.S., Sull, S., Lee, S.U.: "Efficient Video Indexing Scheme for Content-Based Retrieval". In: IEEE Transactions on Circuits and Systems for Video Technology. Volume 9. (1999) 1269–1279
25. Zhuang, Y., Rui, Y., Huang, T.S., Mehrotra, S.: Adaptive key frame extraction using unsupervised clustering. In: ICIP (1). (1998) 866–870
26. Cheung, S., Zakhor, A.: "Efficient video similarity measurement with video signature". In: IEEE International Conference on Image Processing (ICIP 02). Volume 1. (2002) 621–624
27. Han, J., M., K.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2006)
28. Witten, I.H., E., F.: Data Mining. Practical Machine Learning Tools and Techniques. Morgan Kaufmann (2005)
29. Guttman, A.: "R-trees: A Dynamic Index Structure for Spatial Searching". In: Proc. ACM SIGMOD Int. Conf. on Management of Data. (1984) 47–57
30. Berchtold, S., Keim, D.A., Kriegel, H.P.: "The X-Tree: An Index Structure for High-Dimensional Data". In: Proc. 22nd Int. Conf. on Very Large Data Bases (VLDB'96), Bombay, India, pp. 28-39. (1996)
31. Eiter, T., Mannila, H.: Distance measures for point sets and their computation. Acta Informatica **34**(2) (1997) 103–133