Management and Rendering of Multimedia Views *

K. Selçuk Candan[1], Eric Lemar[2], and V.S. Subrahmanian[2]

[1] Computer Science and Engineering Department,
Arizona State University,
Tempe, AZ 85287-5406.
Phone: (602) 965-2770. Fax: (602) 965-5142. candan@asu.edu

[2] Department of Computer Science,
Institute
for Advanced Computer Studies &
Institute for Systems Research,
University of Maryland,
College Park, Maryland 20742.
Phone: (301) 405-2711. Fax: (301) 405-6707. {elemar,vs}@cs.umd.edu

Abstract. Though there has been extensive work on multimedia databases in the last few years, there is no prevailing notion of a multimedia view, nor there are techniques to create, manage, and maintain such views. In this paper, we present a formal definition of a multimedia view definition (MVD). Materializing an MVD corresponds to assembling and delivering an interactive multimedia presentation in accordance with the specification. We develop a probabilistic model of interactive multimedia presentations where the probabilities reflect the likelihood of certain user interactions. Based on this probabilistic model, we develop a prefetching algorithm that anticipate how users will interact with the presentation. These prefetch algorithms allow efficient delivery of these multimedia presentations in accordance with the underlying specification. We have built a prototype system that incorporates these algorithms. We report on the results of experiments conducted on top of this implementation.

1 Introduction

Over the last few years, there has been considerable interest in multimedia databases – most such research has been devoted to content management [29, 25, 15], design of hierarchical media storage servers [27, 21], and networked media servers [28, 30]. Excellent work on querying media databases has been done – ranging from querying image data [26, 24, 5, 15, 19], video data [26, 2], audio data [34] and mixes of all the above [22, 31]. However, there is no prevailing

* This work was supported by the Army Research Office under Grants DAAH-04-95-10174, DAAH-04-96-10297, DAAH04-96-1-0398 and DAAG559710047, by the Army Research Laboratory under contract number DAAL01-97-K0135, by an NSF Young Investigator award IRI-93-57756, and by an ASU-FGIA Grant F-014-98

notion of a *multimedia view*. In [32], Sheth and Kashyap discuss a declarative view management notion for multimedia information; yet, there is no research in the literature on how such views can be created, managed, and maintained. In contrast, in the setting of relational databases, view management techniques have been widely studied [6, 13], as well as in the case of file systems [1], object databases [14], and distributed mediated systems [3]. The benefits that views bring to query processing and optimization in databases are widely known [23]. The primary aim of this paper is to define a notion of *view* for multimedia databases, together with a notion of *materialization* (or rendering) the view. Such a notion would bring to the domain of multimedia databases, the ability to utilize views to enhance efficiency.

In this paper, we start by defining the concept of a multimedia view definition (MVD). MVDs consist of two parts:

- A specification stating what media objects are included in the view.
- A specification describing how these media objects will be presented to the user. *In other words, the results of a multimedia view materialization is an interactive media presentation that is delivered to the user.*

Informally, an MVD is a *declarative* specification of what is to be shown. In classical view materialization, once a view is materialized, *showing* the results is trivial. In contrast, in a multimedia view management system, this is not the case. Presenting the results of materializing an MVD is not trivial, and needs to take into account, the locations of the objects, the bandwidth offered by the network, the load on different servers, the buffer space offered, etc.

Finally, we will report on our **MediaView** system for the management of such multimedia views, and will include experimental results on the benefits of our probabilistic scheduling algorithms.

2 Multimedia View Definitions

A multimedia view contains not just a specification of the objects, but also a description of how these objects are presented to the user. James F. Allen proposed a formalism based on intervals [4]. Many researchers including [33, 20] use intervals for the scheduling of multimedia documents. Several other researchers, including Candan et.al. [10, 11], Buchanan and Zellweger [7–9], and Kim and Song [17, 18], on the other hand, proposed the use of a highly-structured class of linear constraints called *difference constraints* for this purpose. In this paper, we will use the framework introduced by Candan et.al. in [10, 11] as *it can handle inconsistencies in the specifications.* Note that in the absence of disjunctions, this framework is capable of expressing the thirteen temporal relationships introduced by Allen. Furthermore, unlike Allen's framework, it is capable to represent quantitative information.

Let MDB be any multimedia database, and let QL be any query language for querying this database. A *content framework*, CF, is a pair, (MDB, QL).

Definition 1 (Multimedia View Definition (MVD)). A *multimedia view definition (MVD)* consists of two parts:

- An content specification, CS, based on a content framework CF is a relation having the schema (`Num,ObjId,Type,Query,Constraint`), where
  - `Num` is either a non-negative integer, or the string `all`,
  - `ObjId` is a unique string that cannot be the same for two different tuples in CSP, unless the `Condition` fields of those two tuples are jointly inconsistent,
  - `Type` is either i (for action object) or q (for query object),
  - `Query` is a query in QL which returns only a set of media objects as output, and
  - `Constraint` is a set presentation constraints satisfying the following conditions: Given a row t of relation CF,
    1. if $t.Num \neq$ all and $t.ObjId = o_i$, then $t.Constraint$ is a set of constraints over objects $\{o_{i,1}, \ldots, o_{i,t.Num}\}$ and other objects(including $\texttt{start}, \texttt{end}\}$).
    2. if $t.Num =$ all, then:
       (a) $t.Constraint$ contains a special constraint of the form $cnum = c$ for some constant integer $c > 0$ and
       (b) all other constraints in $t.Constraint$ are over objects $\{o_{i,1}, \ldots, o_{i,h}\}$ and other objects (including two special objects, $\texttt{start}, \texttt{end}\}$). Here, $h = \lceil \frac{k}{cnum} \rceil$, where k is the total number of answers to $t.Query$. $o_{i,j}$ refers to the $((cnum \times (j-1)) + 1)$'th object in the answer through the $(cnum \times j)$'th object of the answer to the query. We use specialized techniques to handle the case where $o_{i,j}$ does not exist.
  - An `action` relation specifying, for each action object a_i, four sets: (`ObjId, InObjects, OutObjects, InConstraints, OutConstraints`). This relation describes the objects and presentation specifications associated with each object. $\emptyset$

Example 1. An example content specification is now given below.

Num	ObjId	Type	Query	Constraint
all	o_1	q	**SELECT** Map **FROM** Public_WeatherDB **WHERE** date = Today **AND** place = N.Korea.	`cnum = 3`
1	o_2	i	-	`cnum = 3`
1	o_3	q	**SELECT** Video-clip **FROM** Public_WeatherVideos **WHERE** date = Today **AND** place = N.Korea.	
1	o_4	q	-	-
1	o_5	q	-	-

A quick explanation of this table is in order.

- The first tuple describes the nature of object o_1.
- The second tuple says that object o_2 is an interaction object.
- The third tuple shows how a video object o_3 is instantiated.
- Objects o_4 and o_5 are explicitly specified objects.

The constraint fields associated with objects o_1 and o_2 set to $\texttt{cnum} = 3$. What this means is that the result of the query defining object o_1 will be grouped into three groups g_1, g_2, g_3. $\emptyset$

The following is an example `action` relation:

ActId	InObjects	OutObjects	InConstraints	OutConstraints
a_1	$\{o_5\}$	$\{o_{20}, o_{21}\}$	C_{in}	C_{out}

Here, the sets C_{in} and C_{out} describe the sets of constraints that are inserted to and removed from the original presentation constraints.

3 Interactive Media Presentation Specification

Suppose CSP is a content specification, and $\mathcal{O}_i$ is the set of all interaction objects occurring in the `ObjId` field of CSP, while $\mathcal{O}_p^\star = \{o_1, \ldots, o_n\}$ is the set of all presentation objects (i.e. objects that are not interaction objects) occurring in the `ObjId` field of CSP). We define $\mathcal{O}_p = \mathcal{O}_p^\star \cup \{start, end\}$ where `start` and `end` are special objects denoting the start and end of the presentation. $\mathcal{O}_p$ is the set of all objects to be shown to the user, based on the content specification, CSP. The reader is reminded that the o_i's may themselves be sets of objects (e.g. the answers to a query). A *qo*-subset, $\mathcal{QO}(\mathcal{O}_p)$, is any subset of $\mathcal{O}_p$. Let $\mathcal{Q}$ be the power set of $\mathcal{O}_p$.

Presentation and retrieval schedule generation for non-interactive presentations has already been studied in [10] and [16]. Section 4.2 will include a host of *Probabilistic Prefetching and Scheduling Methods* that will support distributed interactive multimedia presentations, building on top of the previous two bodies of work.

Definition 2 (Interactive Multimedia Presentation (IMP) Space). The *Interactive Multimedia Presentation (IMP) Space* w.r.t. a content specification CSP involving objects $\{o_1, \ldots, o_n\}$ (together with `start` and `end`), consists of a labeled tree Υ where

- Each node N is labeled by a triple $(\mathcal{QO}_N, \mathcal{CONS}_N, \mathcal{ACT}_N)$, where $\mathcal{QO} \in \mathcal{Q}$, $\mathcal{CONS}_n$ is a set of *presentation* constraints (to be described below) on the objects in $\mathcal{QO}_i$, and $\mathcal{ACT}_N$ is a set of interaction objects.
- The root of Υ; is labeled by $(\{\texttt{start}\}, \emptyset, \emptyset)$ (and no other node in Υ is labeled by $(\{\texttt{start}\}, \emptyset, \emptyset)$;
- Each leaf in Υ is labeled by the triple $(\{\texttt{end}\}, \emptyset, \emptyset)$;
- Each edge (from node N to node N') in Υ is labeled by a seven-tuple (a, $\mathcal{QO}^+$, $\mathcal{QO}^-$, $\mathcal{CONS}^+$, $\mathcal{CONS}^-$, $\mathcal{ACT}^+$, $\mathcal{ACT}^-$), where $\mathcal{QO}^+$ and $\mathcal{QO}^-$ are sets of objects, $\mathcal{CONS}^+$ and $\mathcal{CONS}^-$ are sets of constraints (to be described below) on the objects in $\mathcal{QO}_N \bigcup \mathcal{QO}^+$, $\mathcal{ACT}^+$ and $\mathcal{ACT}^-$ are sets of action objects, and a is an interaction object or $\perp$. If a is an action object, then
  - $\mathcal{QO}_{N'} = (\mathcal{QO}_N \bigcup \mathcal{QO}^+) - \mathcal{QO}^-$,
  - $\mathcal{CONS}_{N'} = (\mathcal{CONS}_N \bigcup \mathcal{CONS}^+) - \mathcal{CONS}^-$, and
  - $\mathcal{ACT}_{N'} = (\mathcal{ACT}_N \bigcup \mathcal{ACT}^+) - \mathcal{ACT}^-$.

  If, on the other hand, a is $\perp$, then
  - $\mathcal{QO}_{N'} = \mathcal{QO}_N$,
  - $\mathcal{CONS}_{N'} = \mathcal{CONS}_N$, and
  - $\mathcal{ACT}_{N'} = (\mathcal{ACT}_N \bigcup \mathcal{ACT}^+) - \mathcal{ACT}^-$. $\emptyset$

Intuitively, each node in the IMP-space specifies a *non-interactive* presentation that is delivered to the user. The transitions between nodes occur if there is a user interaction or if the action objects on the screen are modified without the user action. If N is a node labeled by the triple $(\mathcal{QO}_N, \mathcal{CONS}_N, \mathcal{ACT}_N)$, then this node presents all objects in $\mathcal{QO}_N$ by generating a presentation schedule using constraints $\mathcal{CONS}_N$,

Given any IMP-Space, when the presentation is viewed by a specific user, s/he follows one path in the IMP-Space from the root to a leaf. This is termed an interactive multimedia presentation (IMP) path, and is described below.

Definition 3 (Interactive Multimedia Presentation (IMP) Path). An *Interactive Multimedia Presentation (IMP) Path* w.r.t. a content specification CSP involving objects $\{o_1, \ldots, o_n\}$ (together with `start` and `end`) is a finite path from the root of the IMP-Space (i.e., $(\{start\}, \emptyset, \emptyset)$) to a leaf node (i.e., $(\{end\}, \emptyset, \emptyset)$). ø

Note that IMP-spaces may be infinite. For example, the bold-dotted line in Fig. 1(a) shows an infinite sequence of nodes in the IMP-space.

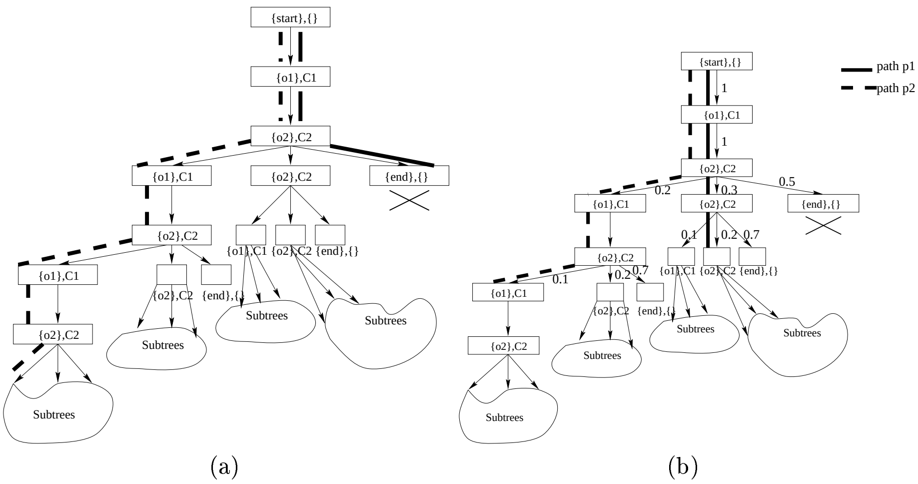

Fig. 1. (a) Example of infinite IMP-Space; (b) example of probabilistic IMP-Space

Suppose now that we consider a specific MVD μ. μ can easily be seen to generate an IMP-Space. This is because given μ, we can construct an IMP-space by taking the root of the IMP-space to be $(\{\texttt{start}\}, \emptyset, \emptyset,)$. If we have a partially constructed IMP-space Γ, we can expand any leaf in it (which is not labeled with $(\{\texttt{end}\}, \emptyset, , \emptyset))$ by finding all actions that are *applicable* in it. The applicable actions can be readily found from a multimedia view definition. Each of those actions is used to generate a new child node IMP-state. This process is repeated till all leaves are labeled with $(\{\texttt{end}\}, \emptyset, , \emptyset)$ and the expansion is no longer possible.

4 Rendering and Delivering Multimedia Views

In classical relational databases, views define "derived" relations that are computed from underlying "base" relations and stored. The resulting "materialized" views are then often used to rewrite queries, resulting in optimized execution times. In the case of an MVD, this task is much more complex. A materialized view in relational DBMSs is static. In contrast, an MVD must be *presented* to the user and the user has the opportunity to interact with the presentation, making choices on what he wants to see. In this section, we will provide an object-based approach to prefetching objects so as to facilitate the delivery of interactive multimedia presentations generated by a multimedia view.

4.1 Probabilistic Spaces

The prefetch approach depend upon a concept called a *probabilistic IMP-space.*

Definition 4 (Probabilistic IMP-Spaces). Suppose we consider an IMP-space Υ. A *probabilistic extension of* Υ is a pair $(\Upsilon, \wp)$ where $\wp$ is a mapping that takes edges in Υ as input and produces real numbers in the $[0,1]$ interval as output such that for each non-leaf node N in Υ,

$$\sum_{N' \text{ is a child of } N} \wp(N, N') = 1.$$

∅

Intuitively, whenever the user is presented with an interactive option, he chooses at most one of the options. $\wp$ measures the probability of these choices. For example, Fig. 1(b) provides an example of how the IMP-space of Fig. 1(a) can be extended to a probabilistic IMP-space.

Definition 5 (Path Probabilities in Probabilistic IMP-Spaces). Suppose $(\Upsilon, \wp)$ is a probabilistic IMP-space, and N is a node in Υ. Let Υ_N^k denote the subtree rooted at node N k levels down. For any path p in this subtree consisting of at most k edges $(N_0, N_1), (N_1, N_2), \ldots, (N_{k-1}, N_k)$ where $N_0 = N$, the *probability of path p being taken, given that node N has been reached in the playout* is given by $\mathbf{P}(p \mid N) = \Pi_{i=0}^{k-1} \wp(N_i, N_{i+1})$. ∅

The above probability is a *conditional probability* that assumes that node N has been achieved via the playout described by the path from the root to node N. This definition associates a probability with each path of length k. As we will see in Section 5, choosing a large k causes the probabilities of most paths to become very small, making it difficult to identify "highly" likely paths. Choosing a k that is too small is not helpful either, it provides few opportunities for prefetching.

Definition 6 (j-bounded Object Appearance Probability). Suppose $PS = (\Upsilon, \wp)$ is a probabilistic IMP-space, N is a node in Υ, Υ_N^j denotes the subtree rooted at node N, j levels down and o is an object in the presentation. Then, the *j-bounded appearance probability of o given* $Hist(t)$ is

$$\mathbf{P}_{j-bounded}(o,N) = \sum_{0 \le i \le j} \sum_{\forall path_m \in \Upsilon_N^i \text{ with } o \text{ in it}} \mathbf{P}(path_m).$$

$\emptyset$

Intuitively, $\mathbf{P}_{j-bounded}(o,N))$, describes the probability of object o's occurrence within the presentation subtree of depth j with root N. If we "look ahead" j levels in the tree, then the probability that object o is needed in those j levels is the sum of the probabilities of the paths containing j. We will use this to prefetch objects for efficient materialization.

4.2 Object-based Prefetching Strategy

Suppose now that our prefetch buffer is of size B bytes. In this strategy, we fill the prefetch buffer by first inserting the most probable object, then the second most probable object, and so on till the buffer is filled. This strategy requires an efficient way of identifying the most probably needed objects in decreasing order of probabilities.

Definition 7 (j-bounded Object Retrieval Profit). Suppose $PS = (\Upsilon, \wp)$ is a probabilistic IMP-space, N is a node in Υ, and o is an object in the presentation. Then, the *j-bounded object retrieval profit* of o is $profit_{j-bounded}(o,N) = \mathbf{P}_{j-bounded}(o,N) \times gain(o,N) - cost(o,N)$, where *gain* is a function which evaluates how much system gains by retrieving object o at time t (considering only the next j units of time), and *cost* is a function which evaluates the cost of the retrieval of the object at node N – and hence at the time node N was reached in the playout. $\emptyset$

Note that $\mathbf{P}_{j-bounded}(o,N) \times gain(o,N)$ is the expected gain of the system from the retrieval of object o at the time node N is reached. Hence, the term $profit_{j-bounded}(o,N)$ describes how desirable is the retrieval of object o at node N if one considers a future of j time units.

Definition 8. Suppose $PS = (\Upsilon, \wp)$ is a probabilistic IMP-space, N is a node in Υ, and o is an object in the presentation. Then, a *(k,j,c)-retrieval policy* (w.r.t. N) is a set O of objects, such that

- $\forall_{o_i \in O}(profit_{j-bounded}(o_i,N) \ge k)$.
- $\sum_{o_i \in O}(cost(o,N)) \le c$. $\emptyset$

Intuitively, a (k,j,c)-retrieval policy dictates that the system chooses (after considering a future of j time units) to retrieve a set of objects such that (1)

Algorithm 1. (Object retrieval-caching algorithm ($ORCA$))
Input:

- A probabilistic IMP-space, PS,
- A presentation state, N,
- A set of objects OBJ,
- A $(k, j, c) - retrieval$ policy,
- A bound j on the future.

1. For each object $o_i \in OBJ$, calculate $gain(o_i, N)$ and $cost(o_i, N)$.
2. For each object $o_i \in OBJ$, calculate (incrementally) $\mathbf{P}_{j-bounded}(o, N)$.
3. For each object $o_i \in OBJ$, calculate $profit_{j-bounded}(o_i, N)$, eliminate all objects whose profits are smaller than k,
4. Solve the *0-1 knapsack algorithm* with the knapsack size c, object weights $cost(o_i, N)$, and object values $profit_{j-bounded}(o_i, N)$).
5. Return the set of objects returned by the *0-1 knapsack algorithm*

Fig. 2. Object Retrieval-Caching Algorithm ($ORCA$)

each of these objects has a profit greater than or equal to k, and (2) the sum of the retrieval costs of these objects is less than or equal to c.

Figure 2 shows the object retrieval-caching algorithm which uses the *optimal (k,j,c)-retrieval policy.* Since, object caching algorithm is expensive ($O(2^j + c \times number\_of\_objects)$, in Fig. 3, we present a heuristic which prunes tree, reducing the execution cost to $O(number\_of\_objects \times c + number\_of\_actionobjects^s)$.

Theorem 1. Let $m = number\_of\_actionobjects$ and let s be an integer such that $0 < s << j$). The total number of nodes created by the heuristic is

$$\sum_{i=1}^{s}(\frac{s!}{(s-i)!} \times m^i + \frac{s!}{(s-i+1)!} \times m^{i-1} \times 2)$$

or, equivalently, it is $O(m^s)$.

5 Implementation and Experiments

We implemented *MediaView*, a multimedia view management system, based on the algorithms described in this paper.

We implemented the object-based prefetching algorithm described in this paper using C on Unix/Solaris platform. We have also implemented a view-definition/case generator to create the view definitions on which these object prefetching algorithms will be used. The view-definition/case generator uses 2000 objects of varying size (approximately 50% of the objects average 500,000 bytes each, 30% of the objects average 5 MBs each, and the rest average 12 MBs each) to create a document, where there are at most 50 objects visible at a time and there are at most 10 choices that a user can have at a time. The document/case generator also generates user profiles to attach probabilities to each choice.

Algorithm 2. **(Tree Creation Algorithm (TCA))**
Input:

- A probabilistic IMP-space, PS,
- A presentation node, N,
- A set of objects,
- A bound, j, on the future,
- A limit, s, on the time points per action object.

1. If $s = 0$ then stop
2. At node N (and current time t) calculate the *j-bounded future* (assuming no user interaction).
3. For each action object a_i, where $t < et(a_i) \leq t + j$, choose s time points $t_{i,b}$, where $1 \leq b \leq s$, $t < t_{i,b} < t_{i+1,b} < t + j + 1$), and $t_{a,b} \neq t_{c,d}$ if $a \neq c$ or $b \neq d$. Here, $et(a_i)$ denotes the end time of action object a_i.
4. Let $t_{\perp,\perp}$ correspond to time t and let $t_{\top,\top}$ correspond to time $t + j + 1$.
5. Let the probability that object a_i is chosen at time $t_{i,b}$ be equal to
$$p(t_{i,b}) = \sum_{t'=t_{i,b}}^{t_{i+1,b}} (\mathbf{P}(a_i, t'|Hist(t_{i,b})))$$
Let also $p(t_{\perp,\perp})$ be 1.
6. For each pair of time points $(t_{a,b} < t_{c,d})$ such that there exists no other time point $t_{e,f}$ satisfying $t_{a,b} < t_{e,f} < t_{c,d}$, create an edge with probability $1-p(t_{a,b})$.
7. For each time point $t_{a,b}$ (not $\in \{t, t+j\}$) execute this algorithm recursively (with initial time set to $t_{a,b}$, the bound, j, set to $t + j - t_{a,b}$, and the limit set to $s - 1$) to get a tree $T_{a,b}$. Connect $t_{a,b}$ with the root of $T_{a,b}$ with an edge of probability $p(t_{a,b})$.

Fig. 3. Tree creation algorithm (TCA)

In addition to the view-definition/case generator, we developed a viewer/case generator which uses document information and user profiles to generate a walk-through (i.e., presentation realization) of the document. The walk-through uses a parameter called *randomness* to decide how much it follows the profiles. If this parameter is equal to 0.0, then the viewer is assumed to follow the user profiles, whereas, if the randomness value is equal to 1.0, then the user behaves completely randomly.

We varied the local buffer size between 5MBs and 50MBs and the randomness between 0.0 and 1.0. In each execution, we looked 5 steps ahead for prefetching. We ran each experiment 80 times.

Effectiveness of the Prefetch Strategy (Hit-Ratios): Figure 4 shows the results of the experiments we ran to evaluate the effectiveness of object-based prefetching algorithm. The algorithm proved to be robust against user randomness. It showed an increase in the hit ratio as the local buffer size is increased and reached a plateau around 70%.

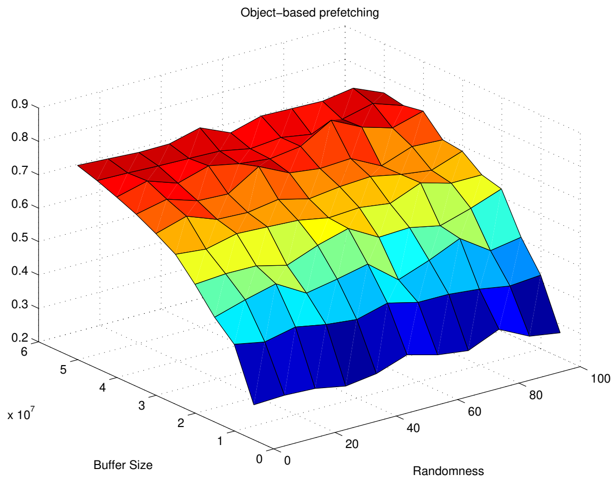

Fig. 4. *Hit-ratio* of object-based prefetching algorithm

Execution Times: Last, we report on the execution times of the prefetch algorithm. For the configuration described above, the object-based algorithm (without the use of the TCA heuristic) takes approximately half a second to execute. This suggests that, despite of its advantages, unless a heuristic (such as TCA) is used to reduce the number of nodes on the tree or the incremental algorithms are used, the object-based algorithm may be too costly to use for larger prefetch depths.

6 Conclusions

Though multimedia database systems have been extensively developed over the last few years, the concept of a multimedia view has remained undeveloped. In this paper, we have taken the first steps towards the development of the concept of *view* for multimedia database systems. We have stated that a multimedia view definition consists of a content specification, specified through queries, and a presentation component, that specifies in what form the results of materialization are to be presented. The crux of our idea is that multimedia views must produce multimedia presentations when they are materialized.

Presenting the results of materializing a multimedia view is not trivial, and it needs to take into account the compute and communication resources. We have presented an object-based prefetching technique that minimize the response time of interactive materialization.

We have implemented *MediaView* based on the algorithm described in this paper. The experiments we ran using this system showed that the prefetching algorithm can be effectively used in improving the efficiency of rendering of multimedia views. Our results show that the strategy may be efficiently implemented in large scale systems, and that on the average, they achieve about 63% correctness in their guess.

References

1. S. Abiteboul, S. Cluet, and T. Milo. (1993) *Querying and Updating the File.* Proc. Int. Conf. on Very Large Data Bases (VLDB), pp. 73-84, 1993.
2. S. Adali, K.S. Candan, S.-S. Chen, K. Erol, and V.S.Subrahmanian. (1996) *Advanced Video Information Systems*, ACM Multimedia Systems Journal, 4, pps 172–186, 1996.
3. S. Adali, K.S.Candan, Y. Papakonstantinou and V.S.Subrahmanian. (1996) *Query Processing in Distributed Mediated Systems*, *Proc. 1996 ACM SIGMOD Conf.*, Montreal, Canada, June 1996.
4. J.F. Allen (1983) *Maintaining Knowledge about Temporal Intervals*, Communications of the ACM, vol. 26, no. 11, pp. 832-843, November 1983.
5. M. Arya, W. Cody, C. Faloutsos, J. Richardson and A. Toga. (1995) *Design and Implementation of QBISM, a 3D Medical Image Database System*, in: (V.S. Subrahmanian and S. Jajodia, eds.) "Multimedia Database Systems: Issues and Research Directions", Springer 1995.
6. J. Blakeley, N. Coburn, and P.-A. Larson. (1989) *Updating Derived Relations: DEtecting Irrelevant and Autonomously Computable Updates. ACM Trans. on Database Systems*, 14(3):369–400, 1989.
7. M.C. Buchanan and P.T. Zellweger (1993) *Automatic Temporal Layout Mechanisms* ACM Multimedia 93, pp. 341-350, 1993.
8. M.C. Buchanan and P.T. Zellweger (1993) *Scheduling Multimedia Documents Using Temporal Constraints*, Network and Operating System Support for Digital Audio and Video, P. Venkat Rangan (ed.), Springer-Verlag, Berlin Heidelberg New York, pp. 237-249, 1993.
9. M.C. Buchanan and P.T. Zellweger (1993) *Automatically Generating Consistent Schedules for Multimedia Documents*, ACM/Springer-Verlag Journal of Multimedia Systems, vol. 1, no. 2, 1993.
10. K.S. Candan, B. Prabhakaran and V.S. Subrahmanian. (1996) *CHIMP: A Framework for Supporting Multimedia Document Authoring and Presentation*, Proc. 1996 ACM Multimedia 1996 Conference, Boston, MA, Nov. 1996.
11. K.S. Candan, B. Prabhakaran, and V.S. Subrahmanian (1996) *Retrieval Schedules Based on Resource Availability and Flexible Presentation Specifications,* Accepted for publication in ACM-Springer Multimedia Systems Journal.
12. K.S. Candan (1997) *A Framework for Distributed Multimedia Collaborations.* Ph.D. Thesis, University of Maryland at College Park, College Park, Maryland, 1997.
13. Stefano Ceri and Jennifer Widom. Deriving Production Rules for Incremental View Maintenance. In *17th VLDB*, 1991.
14. U. Dayal. Queries and views in a object-oriented databases. In *Int. Workshop on Database Programming Languages*, 1989.
15. C. Faloutsos. (1996) *Searching Multimedia Databases by Content*, Kluwer Academic Publishers.
16. E. Hwang, B. Prabhakaran and V.S.Subrahmanian. (1998) *Distributed Video Presentations.* accepted for publication in IEEE intl. Conf. on Data Engineering'98.

17. M.Y. Kim and J. Song (1993) *Hyperstories: Combining Time, Space and Asynchrony in Multimedia Documents*, RC19277(83726) (revised 1995) IBM Computer Science/Mathematics Research Report, 1993.
18. M.Y. Kim and J. Song (1995) *Multimedia Documents with Elastic Time*, ACM Multimedia Conference '95, 1995.
19. W.-S. Li and K.S. Candan. (1998) *SEMCOG: A Hybrid Object-based Image Database System and Its Modeling, Language, and Query Processin* to appear in Proceedings of The 14th International Conference on Data Engineering, Orlando, Florida, USA, February 23 - 27, 1998.
20. T.D.C. Little and A. Ghafoor (1990) *Synchronization and Storage Models for Multimedia Objects*, IEEE J. on Selected Areas of Communications, vol. 8, no. 3, April 1990, pp. 413-427. April 1990.
21. D. Makaroff and R. T. Ng. (1994) *Schemes for Implementing Buffer Sharing in Continuous Media Systems*, Information Systems, 19, 4, pps 33–54.
22. S. Marcus and V.S. Subrahmanian. *Foundations of Multimedia Database Systems*, Journal of the ACM, Vol. 43, 3, pps 474–523, 1996.
23. I. S. Mumick. (1991) *Query Optimization in Deductive and Relational Databases.* Ph.D. Thesis, Stanford University, CA 94305, 1991.
24. W. Niblack, et. al. (1993) *The QBIC Project: Querying Images by Content Using Color, Texture and Shape*, IBM Research Report, Feb. 1993.
25. V.E. Ogle and M. Stonebraker. (1995) *Chabot: Retrieval from a Relational Database of Images*, IEEE Computer, Sep. 1995.
26. E. Oomoto and K. Tanaka. (1993) *OVID: Design and Implementation of a Video-Object Database System*, IEEE Trans. on Knowledge and Data Engineering, 5, 4, pps 629–643.
27. B. Ozden, R. Rastogi and A. Silberschatz. (1995) *Research Issues in Multimedia Storage Servers*, CM Computing Surveys , December 1995.
28. S.V. Raghavan, B. Prabhakaran and Satish K. Tripathi (1994) *Synchronization Representation and Traffic Source Modeling in Orchestrated Presentation*, Special issue on Multimedia Synchronization, IEEE Journal on Selected Areas in Communication, January 1995.
29. P. Venkat Rangan and H. Vin. (1991) *Designing File Systems for Digital Audio and Video*, Proc, 13th ACM Symp. on Operation Systems Principles, pps 69–79.
30. P. Venkat Rangan, H. Vin and S. Ramanathan. (1992) *Designing an On-Demand Multimedia Service*, IEEE Communications Magazine, pps 56–64, July 1992.
31. V.S. Subrahmanian. (1997) *Principles of Multimedia Database Systems*, Morgan Kaufman Press, Dec. 1997.
32. A. Sheth and V. Kashya. (1996) *Media-independent Correlation of Information: What? How?* Proceedings of First IEEE Metadata Conference, April 1996.
33. T.K. Shih, S.K.C. Lo, S.-J. Fu, and J.B. Chang (1996) *Using Interval Logic and Inference Rules for the Automatic Generation of Multimedia Presentations*, IEEE International Conference on Multimedia Computing and Systems '96, pp. 425-428, Hiroshima, June 1996.
34. E. Wold, T. Blum, D. Keislar, and J. Wheaton. (1996) *Content-based Classification, Search, and Retrieval of Audio*, IEEE Multimedia Magazine 1996.