NPS52-88-024

NAVAL POSTGRADUATE SCHOOL
Monterey, California

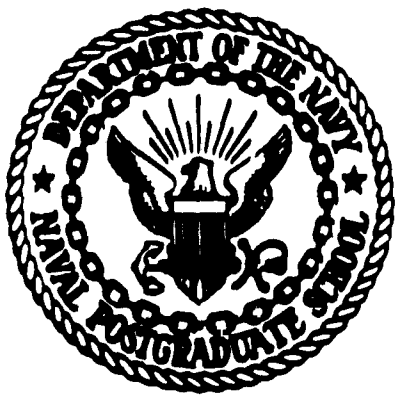

IMAGE DATABASE MANAGEMENT
IN A MULTIMEDIA SYSTEM

Klaus Meyer-Wegener

Vincent Y. Lum

C. Thomas Wu

August 1988

Prepared for:

Naval Postgraduate School
Monterey, CA 93943

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S): NPSL-88-024	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): 52	7a NAME OF MONITORING ORGANIZATION: Naval Ocean Systems Center
6c ADDRESS (City, State, and ZIP Code): Monterey, CA 93943	7b ADDRESS (City, State, and ZIP Code): San Diego, CA 92152
8a NAME OF FUNDING SPONSORING ORGANIZATION: Naval Postgraduate School; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: O&MN, Direct Funding
8c ADDRESS (City, State, and ZIP Code): Monterey, CA 93943	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO: RC 32510; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
IMAGE DATABASE MANAGEMENT IN A MULTIMEDIA SYSTEM (U)

12 PERSONAL AUTHOR(S)
Meyer-Wegener, Klaus, Lum, Vincent Y., Wu, C. Thomas

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
progress	FROM 88-3 TO 88-6	1988 August	

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	multimedia databases, image databases, theses, MW

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

A general concept for the representation of multimedia data by unformatted and formatted data is introduced. It leads to a basic-function approach to the design and development of multimedia database systems, which extends a relational database management system with new attribute types. In this paper, raster (or bitmap) images are used as an example. The structure of image values is defined, and a basic set of operations for access and manipulation is proposed. These operations can be integrated into a query language like SQL. To facilitate a contents-oriented search on multimedia data in general and on images in particular, text descriptions are introduced into the database that allow users to indicate the contents of an image. The well established techniques of information retrieval can be applied to search for these descriptions. The proposed system allows us to model images that are assigned to objects as well as stand-alone images. The paper finally sketches a prototype implementation on top of an existing relational database management system (Ingres).

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL: Vincent Y. Lum	22b TELEPHONE (Include Area Code): (408)646-2693; 22c OFFICE SYMBOL: 52Lu

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Image Database Management in a Multimedia System

Klaus Meyer-Wegener, Vincent Y. Lum, C. Thomas Wu

Naval Postgraduate School
Department of Computer Science
Code 52
Monterey, CA 93943
U.S.A.
Phone: (408) 646-2693
E-Mail: meyerweg@nps-cs.arpa

Abstract

A general concept for the representation of multimedia data by unformatted and formatted data is introduced. It leads to a basic-function approach to the design and development of multimedia database systems, which extends a relational database management system with new attribute types. In this paper, raster (or bitmap) images are used as an example. The structure of image values is defined, and a basic set of operations for access and manipulation is proposed. These operations can be integrated into a query language like SQL. To facilitate a contents-oriented search on multimedia data in general and on images in particular, text descriptions are introduced into the database that allow users to indicate the contents of an image. The well established techniques of information retrieval can be applied to search for these descriptions. The proposed system allows to model images that are assigned to objects as well as stand-alone images. The paper finally sketches a prototype implementation on top of an existing relational database management system (Ingres).

Keywords: multimedia databases, image databases

The work is sponsored by the Naval Oceans System Center (NOSC) as project no. RC32510.

1. Introduction

As database applications become more and more diversified, the capabilities of the current commercial database management systems (DBMS) developed on the basis of handling formatted data become less and less satisfactory. In many of the newer applications, handling of multimedia data such as text, graphics, images, voices, sound, and signal data is important and must be dealt with. Such are the cases of managing engineering and office data. However, storing data of this kind is one thing; organizing a large amount of them for efficient search and retrieval is quite another [LWH87]. Research to develop multimedia DBMS has been initiated few years ago [Ma87, Ch86, Gi87, WKL86]. Some prototypes have been implemented.

Unfortunately, because of the complexity in managing multimedia data, there are not generally accepted solutions at this time. In fact, it can be said that there is not yet a a good general solution. Most projects adopted the approach of developing a specialized system for a special application to reduce complexity (e.g. office environment or engineering environment). While this is definitely one approach we can try to solve our problems, one can also take a different direction as well.

The approach in this paper illustrates an alternative in finding a solution. Its approach is to develop a basic functional DBMS that can handle multimedia for any application, analogous to the way how one construct a normal DBMS for handling formatted data. That is to say, we shall concentrate on developing a DBMS with the basic functions for retrieving, searching, and managing multimedia data as we do in handling formatted data. Although there is the opinion that such a DBMS should be object-oriented, we think that we should start with a simple and well-established data model. i.e. the relational model, and concentrate on the multimedia data. However, in order for us to be successful with this approach, it is necessary for us to find a way to reduce the complexity of handling multimedia data. Thus, first we shall discuss a little on the complexity issue of multimedia data handling.

The fundamental difficulty in handling multimedia lies in the problem of handling the rich semantics that is contained in the multimedia data. In traditional DBMS, data is always formatted. The semantics that can be associated with the formatted data is very restrictive. For example, if the attribute is age with the unit to be year, then a storage of 34 in the data for this attribute can mean only 34 years of age, and nothing more. Further semantics in the interpretation of the data can be done, but would be at a different level. This, in fact, gives rise to the research in semantic data modeling, which after many years of research is still in its infantile stage. This problem is difficult and complex. No pat solution is expected in the near future.

Unfortunately multimedia data is intrinsically tied to a very rich semantics. Consequently, a simple extension from formatted data into textual data, for example, already brings us much difficulty. Information retrieval scientists have spent a number of years trying to solve this

problem with some good success. Extending into other kind of media such as image is much more difficult. To illustrate such a difficulty, one only need to look at a simple image of ships. Given such a picture, how are we to know what kind of ships are there? Are they destroyers? cruisers? aircraft carriers? passenger ships? freighters? oil tankers? or whatever? Or, if we are given a picture of a dog and a cat, both running, how are we to know if the dog is chasing the cat or vice versa? Or are they simply playing with each other?

To answer queries posed on images, a person must draw from a very rich experience one has encountered in life. Further, the person must also perform integration, analysis, synthesis, and even extrapolation of his or her knowledge to derive a good answer. One must have a very sophisticated technique to analyze the content of the images to get the semantics of many, many different things. This kind of capability is generally referred to as intelligence. As a result, persons with limited experience and knowledge, such as a child or some who has not been exposed to the various kind of ships, will not be able to give good answers to queries on multimedia data.

To expect systems to have this kind of capability to answer multimedia query is definitely not possible in today's systems. Technology has not been developed to this level thus far. Hence, we cannot develop a DBMS to be able to handle the multimedia data to the same extent we know how to handle formatted data.

We can, however, do the next best thing. As the proverb says, "a picture is worth ten *thousand words*". *This means that we can describe a picture or an image by ten thousand words,* although one would never have exactly the same thing, feeling- or meaning-wise. Ten thousand words, more or less, is not so important. What is important is that we can abstract the content of the image data, sound data, or other forms into words or text. Once we have the text description, we can say that we have the "equivalent" of the original multimedia data, at least for searching and analysis purposes. We can then use the techniques developed in information retrieval and the formatted data to process these multimedia data since we know how to handle these kind of data fairly well. This is the principle we shall use in developing a DBMS to handle multimedia data for different applications.

The basic concept is that, for each piece of multimedia, it will be represented by three parts: registration data, description data, and raw data. Raw data is a bit string of the data. For example, in image data, it can be the bitmap of the image. Registration data is the data related to the physical aspect of the raw data for the device to display the raw data. For example, it includes the color intensity and the colormap for an image. Description data relates to the content of the multimedia data entered by the users. It is in the form of natural language description. For example, the image may contain "a battleship docked at the San Diego harbor". This part of the data will be used for content search for multimedia data in the system.

A-1

As far we the authors know, the use of such technique to represent multimedia data has not been proposed before, although registration data and raw data have been used. It is the definition and integration of the description data that allow us to do the complicated and complex content search of multimedia data that has been elusive to this date. By using the techniques of database and information retrieval disciplines, we will be able to handle multimedia data in similar ways as one does in handling formatted data. We can extend the relational structure and the query interface to allow us to construct a broadly capable multimedia database system for various applications. Operations for such a system will be described. However, the internal structure of the system goes beyond the scope of this paper and will not be discussed. Readers of this paper should have no problem to see that there are many alternatives for the internal structure.

In section 2 we introduce a general concept of multimedia data management that can be supported by such a DBMS. Section 3 concentrates on images that are used as a representative type of multimedia data during prototype development. This makes it necessary to review image databases briefly. In section 4 three different relation schemas for the modelling of images and their related data are discussed, and the details of the attribute type image are presented. Section 5 finally sketches the architecture of the prototype being developed.

2. Data Organization for Multimedia

Multimedia data are also referred to as unformatted data. More precisely this means that their values consist of a variable-length list of many small items the meaning of which is not associated with database processing: characters in the case of text, pixels in images, line segments and areas in graphics, and so on. There are usually higher-level structures as well (sentences, paragraphs, 2D objects, scenes), but again they may not be known to the DBMS when the data are stored. Invariably, multimedia data are accompanied by some standard formatted data called *registration data*. For text this could be something like document number, name and affiliation of the author, the wordprocessor used etc. For images it could be resolution, pixel depth, source, date of capture, and colormap. The important issue of the registration data is that they are required if anything is to be done with the multimedia data at all, either to interpret them for replay or display, or to identify them and distinguish them from others. Registration data can easily be stored in the attributes and tuples of standard relational database systems, thus making the full power of query languages available to retrieve and manipulate them.

While the registration is indispensable, other formatted (or unformatted) data describing the contents of multimedia data generally are not on hand. This so-called *description data* is per se redundant, because it repeats information already present in the image, text, or sound. However, because of the complexity and the depth of its information content, there is hardly any chance to

perform efficiently a contents-oriented search on the unformatted raw data themselves. It is much easier to use the description that is often structured as formatted data, so that the power of a query language can be applied, as suggested in the introduction section of this paper. It is very difficult and time-consuming to derive the description automatically (this is called *feature and content extraction*), although the areas of natural language understanding, image analysis, and pattern recognition have developed a number of techniques and algorithms. With these techniques, we have limited success in feature extraction. But we are nowhere near the success of achieving automatic information content extraction. As mentioned in the introduction, such kind of work requires much too much intelligence in a system than we know how to provide today. Thus, it is much easier and more effective to let a human user provide the description, just as an author provides abstract and keywords with an article. In either case the database should hold the result of the extraction, i.e. the description, and link it to the multimedia data. It is the purpose of a multimedia database system to provide long-term storage for the multimedia data as well as their description.

The description can be fairly rich and complicated, due to the amount of information embodied in an image or a signal. New modelling tools like semantic and object-oriented data models or knowledge representation methods could help to organize them, but are still in an experimental stage. None of the many different proposals has proven to be clearly superior over the others. In contrast, the relational model is well established now and has a significant modelling potential that should be exploited. In cases where it does not suffice, attachment of plain text to multimedia data offers great improvement at limited cost. It can be entered by users without special skills, and it can be used to search for multimedia data: All the well-known techniques of information retrieval can be applied [Sh64, LF73, SM83]. In doing so, one type of multimedia data (e.g. image) is in fact described with the help of another type of multimedia data (text) that is easier to handle. This is not unusual: graphics can be used to describe aspects of an image, and voice can partly be represented by text. However, it should be noted that this is almost always accompanied by a loss of information.

Multimedia data, their registrations and their descriptions can be used in various ways, as sketched in fig. 1. Any access to the raw data must go "through" the registration data to make sure that the raw data are interpreted correctly. *Editing* operations on the raw data including filtering, clipping, bitmap operations for images, stripping of layout commands and control characters for text, etc. are permitted. *Special operators* that are applied to the description data can be distance and volume calculations on geometric data [CF80], or the addition of synonyms in the case of keywords. These operators can actually do a lot of processing without ever touching the raw data. In fact, it is expected that most of the processing, except the editing of the raw data, will be done outside the raw data. Some of these operators cannot be implemented with

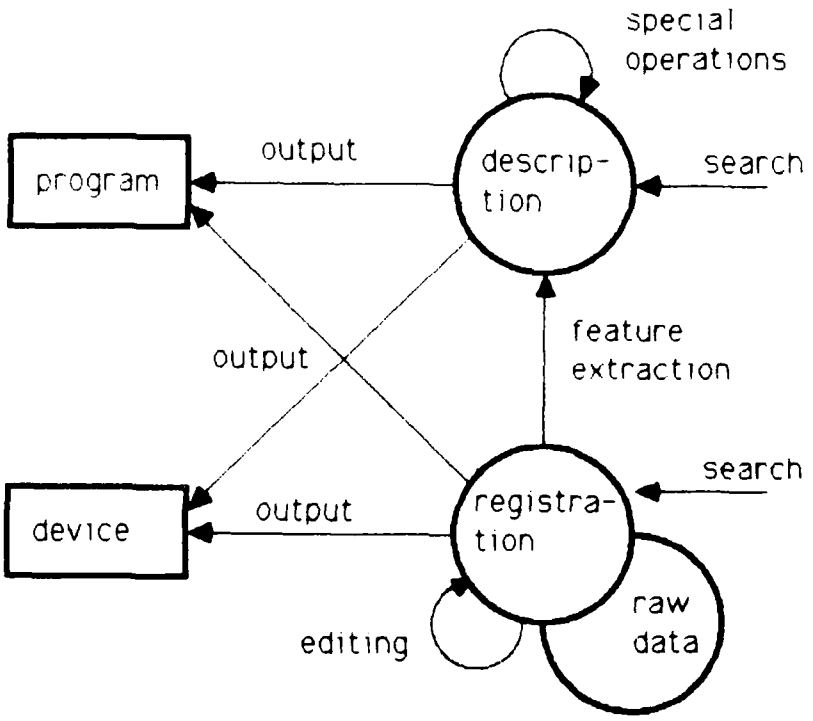

Figure 1: Groups of Operations on Multimedia Data and on the Associated Formatted Data

commands of the query language only. They need the features of a general-purpose programming language. New data models will allow them to be incorporated into the database as "procedures" or "methods".

To make the following discussion more explicit, we shall concentrate on images as a representative type of multimedia data. This allows us to define registrations, descriptions, and operations in detail. We plan to do similar things for the other types of multimedia data as well.

3. Image Database Systems

There is quite a tradition of database support for image management and image analysis [CK81, TY84]. Some of the approaches concentrate on the description data, while others address the raw data and registration data first. None has been found to address raw data, registration data, and description data in any thorough fashion.

Raw image data consist of a matrix of *pixels* (picture elements). Each pixel indicates the color or greyness of a small (atomic) portion of the image. It can be encoded by a single bit to indicate black or white. Alternatively, several bits can be used to encode a pixel, e.g. 8 or 24. The number of bits per pixel is called the *pixel depth*. As the size of the image (the number of pixels in rows and columns) as well as the depth can vary, the raw data appear just as a string of bits that can only be interpreted if the size and the depth are known. Hence, size (also called resolution) and depth are first examples of registration data.

Pixels may either define a color/greyness value directly or index a so-called *colormap*. A typical colormap contains 256 entries each of which specifies the particular intensities of the three basic colors red, green, and blue, or defines a certain color in another way. To display an image on a particular device, special storage segments or registers assigned to that device must be loaded with the colormap. The colormap can have a variable length, thus it is debatable whether it belongs to the raw data or to the registration data. Because it is needed to interpret and reproduce the image and because its size is rather limited, we classify it as registration data.

If the pixels consist of 8 bits each, up to 256 colors can be used in that image. If there are 24 bits per pixel, each 8-bit portion addresses a different entry in the colormap: The first one is used to obtain the intensity of red only, the second and third are used for green and blue respectively. Thus 2^{24} colors can be used in the image.

The use of a colormap primarily saves storage. Instead of repeating the definition of a color in thousands of pixels, it is done only once in the table entry where it can occupy several bytes. However, this indirection has more advantages: Instead of using the basic colors red, green, and blue (RGB) the encoding in the colormap could as well be done in terms of "intensity, hue, and saturation" (IHS) or the "YIQ" defined by the National Television Systems Committee. This can be required for the output on certain types of monitors. Formulae are available to calculate one color definition from the other [Ni86, BB82]. The translation is restricted to the 256 entries of the colormap and does not touch the 10000 or more pixels of the image. Finally, modifying the colors of an image can be used to highlight minimal color changes and thus to make visible hidden shapes on an image, or to perform some simple animations.

Some image *identification* should also be part of the registration data to be able to distinguish images properly. Depending on the application this could be merely an arbitrary number, a combination of source (camera, satellite) and time, or other similar schemes.

How are raw data and registration data integrated into a database system? Some systems simply put them in files, e.g. EIDES [TM77, Ta80b] and IMDB [LU77, LH80]. This means that they do not offer a data model, but only a set of operations (subroutines) to access and manipulate the image files. Others have moved the registration data to a relational database system and linked them to the raw data in the files, e.g. REDI/IMAID [CF79, CF81] and GRAIN [CRM77,

LC80]. They use special relations in which each tuple stands for one image. Display and editing operations can be applied to the tuples of these relations. However, as G.Y. Tang pointed out in [Ta80a], it is not clear what the semantics of the standard relational operators should be when they are applied to those image relations. Especially when two image tuples are joined, which of the images is represented by the result? Both of them?

For this reason, Tang proposed that the raw data should be conceptually represented in the data model as attribute values. This does not imply anything for the storage structures. Internally, images can still be kept in separate files, but they are now accessible through the query language. The display and editing operators are applied to the attribute, not to the tuple. Joining two tuples with image attributes yields a tuple with more than one image attributes which can be handled easily.

Tang himself and Grosky [Gr84] have designed data models based on this approach, but neither of them has reported a successful implementation. The IBM Tokyo Scientific Center has in fact implemented a system called ADM (Aggregate Data Manager) that is based on System R and uses SQL as a query language [TI179]. Some of the registration data are handled in the form of type information, i.e. there are different domains used for binary and grey-tone images. Using SQL queries, images can be retrieved as attributes in relations and tuples and can then be moved to a workspace, where a variety of editing operations can be applied to them. The resulting image can be reinserted into the database. Unfortunately, the program interface is not explained in the paper; it is expected to be some modification of the SQL embedding. However, this approach seems more appropriate than that of [Ta80a]. We shall adopt the ADM concept as a starting point for our system and develop it to more detail. The authors of the ADM model themselves have suggested the extension of their system to other types of multimedia data [TI179], but we could not find out whether they have actually pursued that goal.

Other image DBMS like IMAID and GRAIN have put much more emphasis on the image description data. They are stored in relations with a special structure (e.g. attributes holding geometric coordinates) that can be used as input to pictorial operators. It should be noted that this always implies a slight restriction towards a specific domain, in this case Landsat photographs. Lines detected almost immediately resemble objects like highways, rivers, or city boundaries. This is different from analyzing arbitrary photographs of three-dimensional objects, where it is much harder to relate a line to an object. Hence, we propose to built applications like that on top of a database system and use it to hold the images as well as the descriptions.

4. Extending the Relational Model with the Data Type Image

In this section we shall discuss the data type IMAGE in more detail. We begin with a look at some modelling issues of assigning images to objects and vice versa, which have not been addressed by the papers cited in the last section.

4.1. The Relationship of Objects and Images

IMAGE is a new attribute domain, i.e. an image is supposed to be an attribute of some object or entity (a ship or an aircraft, for instance). Usually it is an attribute of the object shown on the picture, but that need not be the case. Making image an attribute does not prevent the treatment of pictures as stand-alone objects (see relation schema type 3 below). The simplest way of assigning an image to an object leads to a relation schema like this:

OBJECT (O-ID , ... , O-IMAGE)

OBJECT is the name of the relation such as SHIP, CAR, or PERSON, followed by a list of attributes. The object identifier O-ID is underlined to indicate that it is the primary key. We denote this as the *relation schema type 1*. Its advantage is that access to the tuple describing an object fetches the image, too. More than one attributes of type IMAGE can be defined for a relation. However, it may often be the case that the number of images per object varies. If first normal form is required, such repeating groups can only be modelled by a separate relation. Hence, there is a *relation schema type 2*:

OBJECT (O-ID , ...)
OBJECT-IMAGE (O-ID, O-IMAGE)

In the relation OBJECT-IMAGE the O-ID alone cannot serve as a key, because there may be several images of one object, leading to several tuples with the same O-ID. Thus O-IMAGE has to be included to make the key unique. The fact that an attribute of type IMAGE is part of the primary key might lead to severe implementation problems, but we do not consider them here (introducing an image identifier can help). Access to an image is not as simple as it was with schema type 1, for a natural or outer join is required. If the tuple of the object is available, a selection on the OBJECT-IMAGE relation must be performed, using the given object identifier.

Another problem with the two approaches discussed so far is that a picture showing several objects must be stored redundantly, i.e. the same image is repeated in the relation for the number of different objects "having" (shown on) this image. The database system treats the copies as different images. To avoid this, a *relation schema type 3* has to be used:

OBJECT (O-ID , ...)
IMAGE-OBJECT (I-ID , I-IMAGE)

IS-SHOWN-ON (O-ID, I-ID , COORDINATES, ...)

The COORDINATES can be used to give the approximate position of the object on the image. Please note that we do not distinguish the statement "object x has an image y" from "object x is shown on image y", but represent both by the same modeling concept. Now it becomes even more complicated to find the images of an object:

NATJOIN (SELECT $_{\text{O-ID=object1}}$ (IS-SHOWN-ON), IMAGE-OBJECT)

NATJOIN stands for the natural join of two relations, i.e. the equi-join on the attributes with the same name (IS-SHOWN-ON.I-ID = IMAGE-OBJECT.I-ID). Each image is stored only once, regardless of how many objects it shows. It is possible now to start with an image and to retrieve the depicted objects:

NATJOIN (OBJECT, SELECT $_{\text{I-ID=image1}}$ (IS-SHOWN-ON))

One could even define a window on the image, use it to restrict the coordinates, and thus retrieve only the objects shown in the window. Hence, the third type of relation schema is a little bit unwieldy, but it provides the highest degree of freedom in modelling and processing (even images with unknown contents can be stored).

The three schema types are depicted in fig. 2. The dotted line indicates a primary-key-foreign-key relationship (one-to-many). A relational database system extended by image attributes supports all of them. The choice depends on the application. If there is at most one image per object and each image shows only one object (e.g. a database of employees), then type 1 is *most appropriate*.

There is one problem with schema type 3 that has not been mentioned yet: There may be different types of objects, e.g. ships, aircrafts, and submarines, each represented by a different relation. In this case different IS-SHOWN-ON relations are needed as well, for the domain of the O-ID part of the key cannot be the union of the domains of all the object identifiers. This makes the path from a picture to the shown objects really awkward. The introduction of a generalization hierarchy with a superclass 'object' is a solution, but that goes beyond the relational model.

4.2. The IMAGE Data Type

As indicated earlier, not all the operations of the relational algebra can be performed directly on the data type IMAGE. They treat an IMAGE value as a whole, i.e. projection either drops it completely or keeps it in the result. The comparisons needed in selections and joins cannot be performed on the whole image. Even the definition of equality is rather complex for images, whereas it is easy to see what "pixel depth = 8" means. Hence, IMAGE should be regarded as an *abstract data type* with its own set of operators or functions, some of which map

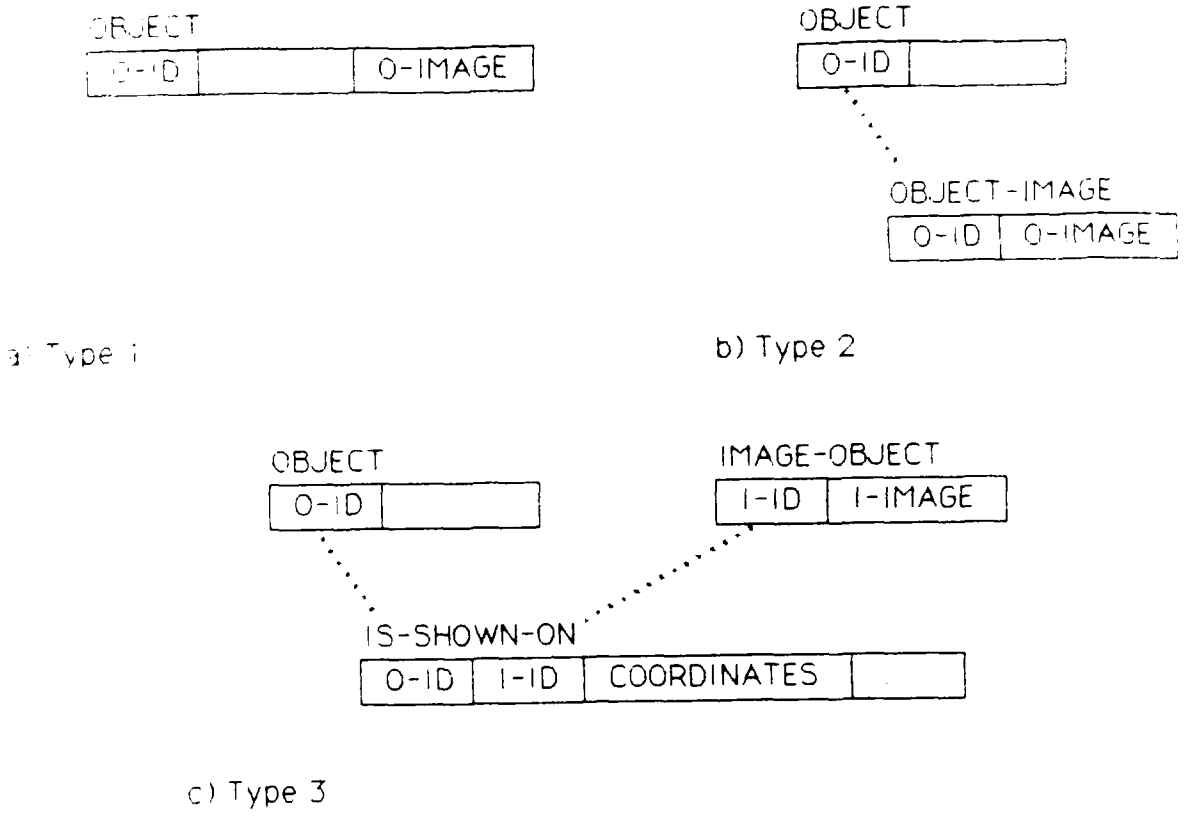

Figure 2. The Three Relation Schema Types for Storing Images

the complex domain IMAGE to standard domains like number or string. The result of these functions can be used in selections and joins without problems. To identify the functions, we have to take a closer look at the structure of an IMAGE value. It will have the three parts introduced before, namely raw data, registration, and description. Raw data and the registration are intrinsically tied together, so they will both be covered in the next subsection, while description data are discussed separately after that.

4.2.1. Raw Data and Registration Data

The registration data could be stored in normal attributes next to the IMAGE attribute, but then it would be the user's responsibility to define them, and the display of an image could be impossible, if the user forgot some of them. Hence, to make sure that they are available for every IMAGE attribute, those required to interpret the pixel matrix are made part of the IMAGE

value, as shown in fig. 3. They can be seen as internal or hidden attributes, and they are accessed through operators of the IMAGE data type. This is almost as easy as the access to the other attributes. The registration data identifying an image are application-dependent and thus are kept in ordinary attributes (compare the O-ID and the I-ID in the schema examples of the last section).

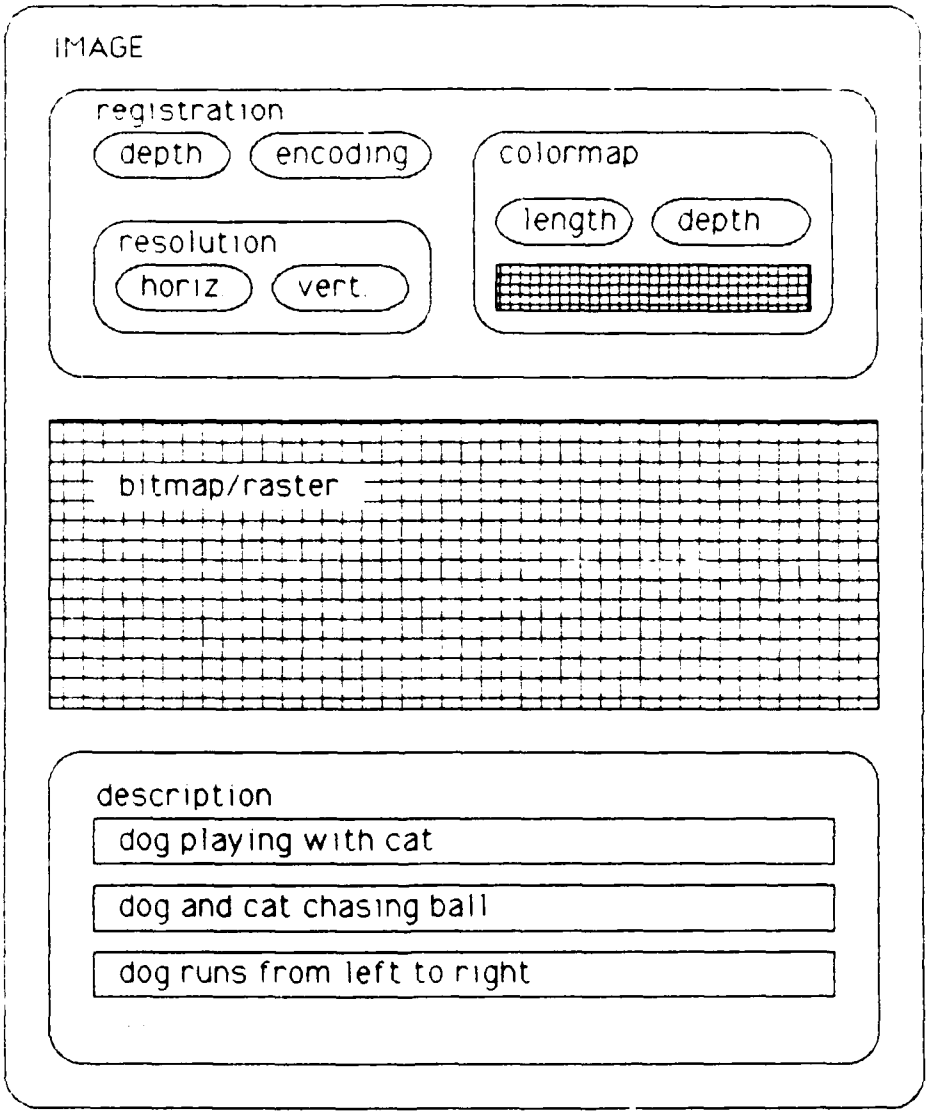

Figure 3: Conceptual View of an Instance or Value of the Abstract Data Type IMAGE

The internal attribute named "encoding" specifies the way the colors are defined in the colormap, or in the pixels, if no colormap is available. Possible values may be "RGB" or "IHS",

but it must also indicate how the values of the three components are encoded, i.e. integer or real, and how many bits they use (8, 24, 32). This, of course, must be consistent with the depth of the colormap or the depth of the pixels. If a colormap is used, its size must further be consistent with the depth of the pixels. However, the depth may sometimes be set to 8 bits, although less than 256 colors are used, in which case the pixel values have to be consistent with the size of the colormap.

To read an attribute of type IMAGE from the database into a program, one could use a very complex, variable-length record structure in the program. It seems more convenient to make the components of an IMAGE value accessible only through *functions*. This has the additional advantage that the program is even more independent of the storage structures and data encodings used by the DBMS. For instance, the function

```
CONSTRUCT_IMAGE (resolution, pixel_depth, encoding, colormap_size,
colormap_depth, colormap, pixel_matrix)
```

produces a (transient) value of type IMAGE that cannot be assigned to program variables, but can only be used in *INSERT* and *UPDATE statements of the query language*. It reads a number of input parameters (variables or results of other functions) and combines their values into a single value of type IMAGE. To illustrate this, we show how the CONSTRUCT_IMAGE function could be used in the query language SQL [Ch76] (cf. relation schema type 3 above):

```
UPDATE IMAGE-OBJECT
SET I-IMAGE = CONSTRUCT_IMAGE ($resolution, $depth, RGB_REAL_32, 256, ... )
WHERE I-ID = 1234;

INSERT (4567, CONSTRUCT_IMAGE ($resolution, 24, IHS_INT_8, 0, ... ))
INTO IMAGE-OBJECT;
```

Identifiers with a leading dollar sign represent program variables, whereas parameters with capital letters only indicate named constants.

It should be clear at this point that this kind of command notation may be appropriate for the programmer, but not for the end-user. The interface for the latter should offer menus and icons to specify the source of the image to be stored. Even if only text input is possible, functions like READ_CAMERA (device_id) should be used in place of CONSTRUCT_IMAGE. The program that actually implements the user interface with the help of the query language could utilize this to avoid unnecessary copying of the large pixel matrix: It replaces the parameter variable in the internal CONSTRUCT_IMAGE call by a function call that reads the camera input (usually part of the driver software that is delivered with a camera):

CONSTRUCT_IMAGE (..., READ_RGB_CAMERA ($camera_id), ...)

Avoiding intermediate storage and unnecessary copying is a very important design issue in multimedia databases. We shall return to this in section 5.

Retrieving attribute values of type IMAGE from the database into program variables uses another set of functions like:

GET_RESOLUTION (IMAGE attribute) : resolution_type;
GET_DEPTH (IMAGE attribute) : integer;
GET_ENCODING (IMAGE attribute) : encoding_type;
GET_8BIT_COLORMAP (IMAGE attribute) : array [0:255] of record ...;
GET_8BIT_RASTER (IMAGE attribute) : array [0:n, 0:m] of 8bit_int;
etc.

Each function has a specific output type. Different functions can be defined to produce different output types for the same component of an IMAGE attribute. A query may look like this:

SELECT GET_8BIT_RASTER (I-IMAGE), GET_8BIT_COLORMAP (I-IMAGE)
INTO $rgb_screen, $rgb_colormap
FROM IMAGE-OBJECT
WHERE I-ID = 35;

Instead of copying the image into program variables, it should again be possible to send it to an output device directly. To do so, the DBMS might be required to perform some transformation on the colormap and the pixel matrix, e.g. change the RGB encoding to IHS. It has not been decided yet how the syntax for that should look like. One could also think of many other access functions like GET_WINDOW, GET_ZOOMED_IMAGE, etc. [LH80, TI179]. The system is planned in a way that it is easy to add those functions when it seems appropriate.

4.2.2. Description Data

Some contents of an image can be represented by linking it to the objects it shows (fig. 2). That is not enough if we want to use the description data instead of the raw data whenever possible, especially in search. It does not say anything about *how* the objects are shown on the image For instance, a ship could be shown in a harbor, out on the sea, in a storm, or in a convoy. Neither does it say anything about the relation or interaction of the objects shown on the same image.

We have already pointed out why a text description seems to be most appropriate. In general, text can also cause some problems: it can be imprecise, it depends on the capabilities of the

author, and it can be ambiguous. In this context however, we need a special type of text that differs from the intended general multimedia data type TEXT in several aspects:

- it is not self-content, but refers to other data
- it has a very simple structure (one paragraph)
- it is explicitly determined to support search.

Therefore, we have decided to tie the image and its description together, that is, we enhance the IMAGE type with a description part. It consists of a *set of phrases or sentences* that characterize the contents of the image (fig. 3). The set notion implies that each phrase or sentence is independent of all the others, which necessarily leads to some repetitions (of nouns), but makes it much easier for the search mechanisms to grasp the meaning - or at least the important phrases. A typical example would be:

dog chases cat;
cat is running from left to right;
a house in the background;
front door of house is open;

This is still easy to enter and easy to read for human beings, but it also gives the system much more opportunity to distinguish images and to locate the ones that fit to a query.

The description part can be empty, if an image is entered into the database that nobody has looked at yet. To add the description later, a function will be provided that takes an IMAGE value as input, expands the set of description phrases by the given new ones, and produces a new IMAGE value that can be assigned to an IMAGE attribute:

```
UPDATE IMAGE-OBJECT
SET I-IMAGE = ADD_DESCRIPTION (I-IMAGE,
                    { dog playing with cat,
                      dog and cat chasing ball,
                      dog runs from left to right,
                      cat runs from right to left,
                      ball is between dog and cat,
                      ball bounces up in the air,
                      dog and cat are in the backyard of a house } )
WHERE I-ID = 1122;
```

This suits particularly well to situations where someone examines an image and adds his or her observations to those that others have entered before, thereby sharing the new knowledge with all the users of the system.

In case the contents of an image are already known when it is stored, the two functions can be combined in the INSERT command:

```
INSERT (4387, ADD_DESCRIPTION (CONSTRUCT_IMAGE ( ... ),
        | USS Enterprise in the South Pacific on the way to ... | ) )
INTO IMAGE-OBJECT;
```

Other functions can be defined to read the set of descriptions or to delete elements from it. The most important operator is the one used in search: CONTAINS (IMAGE attribute, template). The simplest form of a template is just a word, and CONTAINS yields true, if any of the phrases in the description contains that word. The template may be more complicated, containing several words that must appear in an arbitrary or given order, or specifying "wild cards" for unknown parts of a word. Many access paths and indexing methods are available to support this kind of retrieval [Fa85, KW81, KSW79].

To give an example of how the description can be used in the retrieval of images, consider a relation schema type 2 with ships as objects. The following query tries to find out whether the database holds some photos of a sinking destroyer:

```
SELECT SHIP.S-ID, GET_RESOLUTION (S-IMAGE), ....
INTO $ship_id, $resolution, ....
FROM SHIP, SHIP-IMAGE
WHERE SHIP.CLASS = "destroyer"
AND SHIP.S-ID = SHIP-IMAGE.S-ID
AND CONTAINS (SHIP-IMAGE.S-IMAGE, "sinking");
```

As another example consider the image of a cat and a dog playing as given above, and a relation schema of type 3. If we want to find all the pictures that show a dog playing with a cat, and both are chasing after a ball, we can a query like the following:

```
SELECT GET_RESOLUTION (I-IMAGE), ...
INTO $resolution, ...
FROM IMAGE-OBJECT
WHERE CONTAINS (I-IMAGE,
                "cat | play* | dog",
                "dog & chas* & ball",
                "cat & chas* & ball" );
```

The | symbol between two given words requests that both words appear in the same sentence, but in an arbitrary sequence. The & symbol means that between the two words there may be other words that are ignored in the selection process. The * symbol finally matches strings of arbitrary

length (not containing spaces), usually part of a single word like prefix or suffix. Hence, if the sentence "a black cat plays with a brown dog running in the backyard" is used to describe the contents of a picture, this satisfies the first search pattern in the SELECT query, and so does the phrase "dog playing with cat" entered in the example UPDATE operation above.

The details of the syntax for search patterns or templates are still to be determined. It is desirable to have Boolean operations. For instance, the given example assumes that a logical conjunction (and) between the the three search patterns. That means, each search pattern must be satisfied by at least one of the phrases, for the image to be selected. Naturally, one would also like to specify a disjunction (or), a negation (not), or combinations of all.

Apart from registration data and description data there are some other issues about images that a multimedia DBMS should support, e.g. the management of *subimages* [Ta80a, Gr84]. The relation schema type 3 with the coordinates in the IS-SHOWN-ON relation already provides a way to define subimages, but it is rather cumbersome to extract them for display (GET_WINDOW function). Instead we want to access a subimage as easily as the full image - without storing the pixels redundantly. There are several ways to do this, and we have not yet decided about it. However, it seems clear that the system has to support two different concepts: First, subimages can be derived from other images by selecting rectangular subsections. Second, several images can be combined to form a larger image. The latter is particularly useful for Landsat photographs. Ideally, both concepts can be handled by the same mechanism. We plan to further extend the data type IMAGE by some kind of reference to other images.

5. Architecture of a Prototype

The prototype is intended to cope only with the management of images, including storage organization, query and browsing facilities, and presentation issues. To keep the effort limited and to make the functionality of the envisioned system available as soon as possible, it is being built around an existing relational DBMS (Ingres [RTI85, RTI87]). That implies that performance will not be an issue in the first version. The high-level architecture is shown in fig. 4.

The *dialogue manager* can be regarded as the main program. It calls the device manager to perform the exchange of the data with the user, employing a variety of input/output devices. It also calls the DBMS interface to store and retrieve the data, and maintains the state of the dialogue with the user. The *device manager* is to hide the specific details of the different I/O devices (cameras, monitors, VCRs) and to provide the dialogue manager with a more abstract view on their capabilities (comparable to the HIOMM in [WLK87]). The *DBMS interface* implements the query language sketched in section 4.2. It gives the dialogue manager (and other applications) the illusion of using a DBMS with integrated image management facilities. In fact it

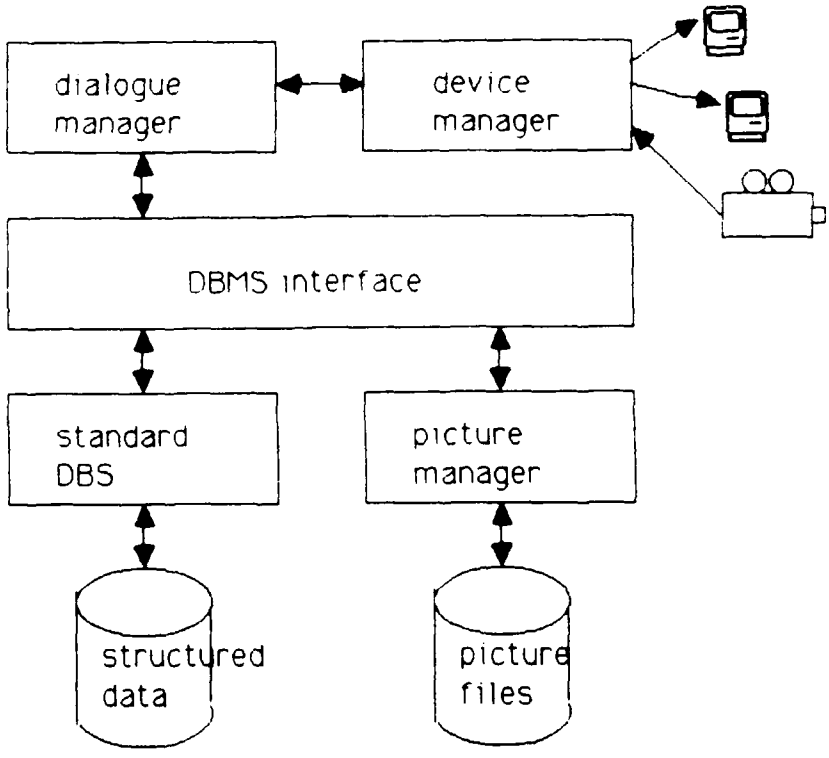

Figure 4: Architecture of the Prototype

engages two different systems, a standard relational DBMS for the structured data and a picture manager. The *picture manager* is responsible for storing all the images in standard files. Each image will be given a unique identifier (e.g. the file name) that is used by the relational database to refer to the image.

All the interfaces must be defined in detail. To do so, we have to investigate the different ways to encode images and the transformations required during input (capture) and output (display). How does the actual signal read from a camera through a video board look like? What has to be sent to the various types of monitors? And what is a suitable standard format that can cover both and thus avoid redundant storage of pictures?

Implementing the interfaces should take into account that we have to avoid copying the whole picture whenever possible. A good idea might be to *pipe* the data read from the database through the transformation process into the monitor driver. To do this the dialogue manager should be written in the style of functional programming:

```
DISPLAY (TRANSFORM (DB-ACCESS (attribute,
                               db-access-parameters)
                    transformation-parameters)
          display-parameters)
```

The analogous solution can be used for data capture. This gives the implementation the freedom not to copy the data but to hand over pointers instead. The only main memory copy of an image will then probably reside in the DBMS interface.

6. Outlook and Future Work

Design and implementation along the presented line have begun. A simple version of the prototype handling images is expected to be operational at the end of 1988. There are four major areas of continuing development:

- investigation in the various search issues on description data
- other attribute domains like text and sound
- integration with an object-oriented data model
- user interfaces and applications

The management of the description data is a central issue in our proposal, and syntax and semantics of the search expressions as well as the internal organization (indexing) have to be designed carefully. To support the full range of multimedia applications, the database system must offer data types like text and sound as well. They will be included in the query language in a way similar to that of IMAGE. However, their access functions will be different. Especially the proper treatment of sound relies on some real-time features of the DBMS: When a recorded sound sequence is to be heard through a speaker, the DBMS must deliver the data fast enough to guarantee uninterrupted and timely replay. At the same time, the volume of data increases drastically. Investigations about this special data type are about to begin.

Once all the new data types and their access functions have been tested thoroughly, it is time to think about their integration into an object-oriented data model. This should be much easier than with the relational model. It should also be easier to design and implement new applications using the object-oriented DBMS. The user interface can be enhanced with sophisticated query and browsing facilities. In addition to that applications can be built to provide higher-level objects like documents or hypertext to the user - including the operations to access and manipulate them. Enough experience should be available then to discuss new storage methods and devices, such as optical disk, that may be more appropriate for multimedia data than the standard magnetic disk, for their integration into the new DBMS.

Literature

BB82 Ballard, D.H., and Brown, C.M., *Computer Vision*, Prentice-Hall, Englewood Cliffs, 1982.

CF79 Chang, N.S., and Fu, K.S., "Query-by-pictorial-example," in *Proc. COMPSAC 79* (Chicago, IL, 1979), pp. 325-330, also *IEEE Trans. on Software Engineering*, vol. SE-6, 1980, pp. 519-524.

CF80 Chang, N.S., and Fu, K.S., "A Query Language for Relational Image Database Systems," in *Proc. IEEE Workshop on Picture Data Description and Management* (Asilomar, CA, Aug. 1980), IEEE Computer Society, catalog no. 80CH1530-5, pp. 68-73.

CF81 Chang, N.S., and Fu, K.S., "Picture Query Languages for Pictorial Information Systems," *IEEE Computer*, vol. 14, no. 11, Nov. 1981, pp. 23-33.

Ch76 Chamberlin, D.D., et al., "SEQUEL2: A Unified Approach to Data Definition, Manipulation and Control," *IBM Journal of Research and Development*, vol. 20, 1976, pp. 560-575.

Ch86 Christodoulakis, S., Theodoridou, M., Ho, F., Papa, M., and Pathria, A., "Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System," *ACM Trans. on Office Information Systems*, vol. 4, no. 4, Oct. 1986, pp. 345-383.

CK81 Chang, S.-K., and Kunii, T.K., "Pictorial Data-Base Systems," *IEEE Computer*, vol. 14, no. 11, Nov. 1981, pp. 13-19.

CRM77 Chang, S.K., Reuss, J., and McCormick, B.H., "An Integrated Relational Database System for Pictures," in *Proc. IEEE Workshop on Picture Data Description and Management*, (Chicago, IL, Apr. 1977), IEEE Computer Society, catalog no. 77CH1187-4C, pp. 49-60.

Fa85 Faloutsos, C., "Access Methods for Text," *ACM Computing Surveys*, vol. 17, no. 1, March 1985, pp. 49-74.

Gi87 Gibbs, S., Tsichritzis, D., Fitas, A., Konstantas, D., and Yeorgaroudakis, Y., "Muse: A Multimedia Filing System," *IEEE Software*, vol. 4, no. 2, March 1987, pp. 4-15.

Gr84 Grosky, W.I., "Toward a Data Model for Integrated Pictorial Databases," *Computer Vision, Graphics, and Image Processing*, vol. 25, no. 3, March 1984, pp. 371-382.

KSW79 Kropp, D., Schek, H.-J., and Walch, G., "Text Field Indexing," in *Datenbanktechnologie*, ed. J Niedereichholz, Teubner, Berichte des German Chapter of the ACM, vol. 2, Stuttgart 1979, pp. 101-115.

KW81 Kropp, D., and Walch, G., "A Graph Structured Text Index," *Information Processing and Management*, vol. 17, no. 6, 1981, pp. 363-376.

LC80 Lin, B.S., and Chang, S.K., "GRAIN - A Pictorial Database Interface," in *Proc. IEEE Workshop on Picture Data Description and Management* (Asilomar, CA, Aug. 1980), IEEE Computer Society, catalog no. 80CH1530-5, pp. 83-88.

LF73 Lancaster, F.W., and Fayen, E.G., *Information Retrieval On-Line*, Melville Publ. Comp., Los Angeles, CA, 1973.

LH80 Lien, Y.E., and Harris, S.K., "Structured Implementation of an Image Query Language," in *Pictorial Information Systems*, eds. S.K. Chang and K.S. Fu, Springer-Verlag, Lecture Notes in Computer Science, vol. 80, pp. 416-430.

LU77 Lien, Y.E., and Utter, D.F., "Design of an Image Database," in *Proc. IEEE Workshop on Picture Data Description and Management* (Chicago, IL, Apr. 1977), IEEE Computer Society, catalog no. 77CH1187-4C, pp. 131-136.

LWH87 Lum, V.Y., Wu, C.T., and Hsiao, D.K., "Integrating Advanced Techniques into Multimedia DBMS," report no. NPS52-87-050, Naval Postgraduate School, Monterey, CA, Nov. 1987.

Ma87 Masunaga, Y., "Multimedia Databases: A Formal Framework," in *Proc. IEEE CS Office Automation Symposium* (Gaithersburg, MD, April 1987), IEEE CS Press, Washington, pp. 36-45.

Ni86 Niblack, W., *An Introduction to Digital Image Processing*, Prentice/Hall Intern., Englewood Cliffs, 1986.

RTI85 Relational Technology Inc., *INGRES Reference Manual*, Version 3.0, UNIX, October, 1985.

RTI87 Relational Technology Inc., *INGRES/Embedded SQL User's Guide*, Release 5.0, UNIX, 1987

Sh64 Sharp, H.S. (ed.), *Readings in Information Retrieval*, The Scarecrow Press, New York & London 1964

SM83 Salton, G., and McGill, M.J., *Introduction to Modern Information Retrieval*, McGraw-Hill, New York 1983.

Ta80a Tang, G.Y., "A Logical Data Organization for the Integrated Database of Pictures and Alphanumerical Data," in *Proc. IEEE Workshop on Picture Data Description and Management* (Asilomar, CA, Aug 1980), IEEE Computer Society, catalog no. 80CH1530-5, pp. 158-166.

Ta80b Tamura, H., "Image Database Management for Pattern Information Processing Studies," in *Pictorial Information Systems*, eds. S.K. Chang and K.S. Fu, Springer-Verlag, Lecture Notes on Computer Science, vol. 80, Berlin 1980, pp. 198-227.

TII79 Takao, Y., Itoh, S., and Iisak, J., "An Image-Oriented Database System," in *Database Techniques for Pictorial Applications*, ed. A. Blaser, Springer-Verlag, Lecture Notes in Computer Science, vol. 81, Berlin 1979, pp. 527-538.

TM77 Tamura, H., and Mori, S., "A Data Management System for Manipulating Large Images," in *Proc. IEEE Workshop on Picture Data Description and Management* (Chicago, IL, April 1977), IEEE Computer Society, catalog no. 77CH1187-4C, pp. 45-54, extended version in *Int. Journal on Policy Analysis and Information Systems*, vol. 1, no. 2, Jan. 1978.

TY84 Tamura, H., and Yokoya, N., "Image Database Systems: A Survey," *Pattern Recognition*, vol. 17, no. 1, 1984, pp. 29-44.

WKL86 Woelk, D., Kim, W., and Luther, W., "An Object-Oriented Approach to Multimedia Databases," in *Proc. ACM SIGMOD '86 Int. Conf. on Management of Data* (Washington, D.C., May 1986), ed. C. Zaniolo, *ACM SIGMOD Record*, vol. 15, no. 2, pp. 311-325.

WLK87 Woelk, D., Luther, W., and Kim, W., "Multimedia Applications and Database Requirements," in *Proc. IEEE CS Office Automation Symposium* (Gaithersburg, MD, Apr. 1987), IEEE CS Press, order no. 770, Washington, 1987, pp. 180-189.

Distribution List

SPAWAR-3242 Attn: Phil Andrews Washington, DC 20363-5100	1
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Dudley Knox Library, Code 0142 Naval Postgraduate School Monterey, CA 93943	2
Center for Naval Analyses 4401 Ford Ave. Alexandria, VA 22302-0268	1
Director of Research Administration Code 012 Naval Postgraduate School Monterey, CA 93943	1
John Maynard Code 402 Command and Control Departments Naval Ocean Systems Center San Diego, CA 92152	1 1
Dr. Sherman Gee ONT-221 Chief of Naval Research 800 N. Quincy Street Arlington, VA 22217-5000	1
Leah Wong Code 443 Command and Control Department Naval Ocean Systems Center San Diego, CA 92152	1
Klaus Meyer-Wegener Code 52Mw Naval Postgraduate School Monterey, CA 93943	1

Vincent Y. Lum
Code 52Lu
Naval Postgraduate School
Monterey, CA 93943

50

C. Thomas Wu
Code 52Wq
Naval Postgraduate School
Monterey, CA 93943

1