Semantic Queries with Pictures: The VIMSYS Model

Amarnath Gupta
Indian Statistical Institute
Calcutta, India

Terry Weymouth
Computer Science and Engineering
The University of Michigan
1101 Beal Avenue
Ann Arbor, MI 48109
U.S.A.

Ramesh Jain
Computer Science and Engineering
The University of Michigan
1101 Beal Avenue
Ann Arbor, MI 48109
U.S.A.

Abstract

The goal of the VIMSYS project is to create an information management system that can retrieve images or parts of images in addition to textual data. The system allows users to incrementally formulate a query starting from semantic pictorial objects. It allows incompletely specified and similarity-based queries. This paper presents a new layered data model and the design of the query processing unit for this system. The data model is a combination of object-oriented and functional models, and permits multiple representations of image entities. The knowledge module guides the user to progressively refine similarity-based queries and to query by defining new semantic objects and composing its attributes. Design considerations for the access structure of the system are also discussed, alongwith some of the problems of creating efficient mechanisms of access.

2 Introduction

Treatment of images as database items leads to several special complications compared to alphanumeric items. The aspect that has received most attention among the designers of current image databases, is the two-dimensional or spatial nature of the data. The fact that a data item can be a point or a region in two-dimensional space, has led to the development of newer data models with specialized operators for finding the location, extent, and sometimes shape of the data items. While this emphasis on spatial operations is justified in its own right, the semantics of the inherent information in the images is largely underutilized. We ascribe the reasons for this deficiency to a lack of a clear definition of image information and the principles of its organization in a large visual information management system. The image information in the present technology is stored mostly as record-based relational structures, or in some systems, as object-oriented entities [3, 6, 5, 7]. The fields of the records or the attributes of the objects are either spatial information, or properties considered significant for *all* images in the database. For both these cases, it is possible to forward a query, for which the appropriate response *exists* and can be extracted from the image, but the system would fail because there is *no mechanism to extract the answer* from the predesigned records or objects. For example, consider a geographic image database containing all topographical information. A naive user, in his attempt to find a specific region where "a major river makes a 90° turn in a plateau in U.S.A." expects to see or get an account of all image regions, but most of the present image databases would not be able to process the query. A more significant class of query that cannot be handled is one based on pictorial similarity. As an example, an user shows a part of the coast line in Florida, and asks "where is this place in North America?" Our emphasis is not on the *interpretation* of the images, but on the organization of images and the design of access methods to answer these queries.

The semantics of an image data item should be

expressible in terms of a set of basic image features, operations to compute these features, and operations to combine them to form more complex features. Once these are established one can organize the information on the basis of concepts pertaining to the domain of the image. Of course the "processed image data" must be closely integrated with the spatial data structures that relate these "features". These can also be supplemented by creating alphanumeric records or objects that relate domain-specific semantic concepts to the features derived from the images. This, we hold, is not the usual approach taken for the construction of image databases. Grosky and Mehrotra [1] for example, raise the issue of feature indexed image databases, but then uses the idea for object recognition. The GRIM_DBMS group [2] have developed a system with similar ideas, but have not formalized the ideas to a data model. We share their belief that query formulation is an incremental process based on system feedback, but they approach the problem from the viewpoint of image interpretation based on partial evidence. We have deliberately avoided this approach because it tends towards automation of the feature matching process. One basic question is, with such a wide and domain dependent variation of significant image features, how does the system determine which features are more significant for a specific image or for a specific query? We take the position that the system *does not* determine the significant features: that task is performed *initially by a system designer, and at query time, the end user* who, directly or indirectly interacts with the system to determine and compute the significant features. This human assisted index feature determination is, we believe, the key to success for an image-feature indexed, pictorially queried image database system.

In this paper, we present Visual Information Management System (VIMSYS) based upon these concepts. This is part of the InfoScope project at the University of Michigan. Our project is developing hetereogeneous databases for global change and other applications.

3 The VIMSYS Concepts

In VIMSYS framework, data consists of images, together with their headers in a prescribed format and a record of textual descriptor fields for each image. The VIMSYS project aims to

1. establish a relationship between non-computable 'semantic' attributes of images and computable features of these attributes,

2. develop a methodology to organize and index the images and image-related information by taking into account the semantic attributes through the association expressed in (1),

3. formulate a computable 'similarity measure' for quantitative comparison between the semantic attributes, and

4. design and develop a system which has an acceptable response time for user queries.

The first goal is achieved through extensive user interaction. In fact, during query processing the user has to interact through a visual interface and a restricted English-like language to convey the 'semantics' of the features to the system. Our research aims to enrich the current status of the field [3, 4, 8] of visual information systems by incorporating pictorial semantics into the field. We are designing an integrated system using image feature based indexing and visual cues for query interface to elicit appropriate responses for the query classes outlined in the Introduction.

3.1 Data Model

The image database paradigm is insertion and search intensive but not update intensive, allowing complex structural representations. The choice of an appropriate data model for our application would be governed by the following factors:

- We should be able to access an image matrix completely or in partitions.

- The image features should be considered as both independent entities, and as related to the image.
- The image features should be ordered in a hierarchy such that more complex features can be constructed out of simpler ones.
- There should be several alternative methods to derive a specific semantic feature from image features.
- The data model should support spatial data and file structures which infer spatial parameters associated with images and image features.
- In the case of complicated image regions the image features should be represented as a sequence of nested or recursively defined entities.

With the above considerations, we have developed a data model in which the user can view the information entities in four different planes. These planes correspond to domain objects and relations (DO), domain events and relations (DE), image objects and relations (IO), and image representations and relations (IR) respectively. All **objects** have a set of **attributes** and **methods** associated with them. The attributes too have their own representations, and are connected in a **class attribute hierarchy** as described by [9]. The **relations**, as we shall explain shortly, may be spatial, functional or semantic. Figure 1 illustrates the basic concepts of the layered data model. In the following paragraphs we describe their properties in more detail. In all the object-oriented descriptions that follow, the leaf-level objects of a class hierarchy are called *prototypes*, while instances of prototypes and higher classes are called *instance images* and *instance features* depending on the nature of the entity.

Representations can be thought of as the underlying data types based upon which all the objects are constructed. The **base representations** are char, int, float, bool and string. The **constructors** are set_of, tuple_of, vector_of, matrix_of, graph_of [1], sequence_of, each having an appropriate set of arguments. A function forms a separate data type, and can serve as the *type* of any relation. An **image representation** is an abstract data type for each image object in the IO plane. The system allows multiple image representations for each image object, with the constraint that each image representation from a single image object *must be derivable* from any other image representation of the same image object. In general, all representations can be looked upon as **values**, where the term is used in a more generalized sense of [10]. The model for the entities in the IR plane is clearly functional; the functions may be multivalued and **composition** of functions is a permissible operation. The separation between image objects and their representations allows the users to define both new representations and new functions (or implementation of an existing function) without affecting the rest of the system.

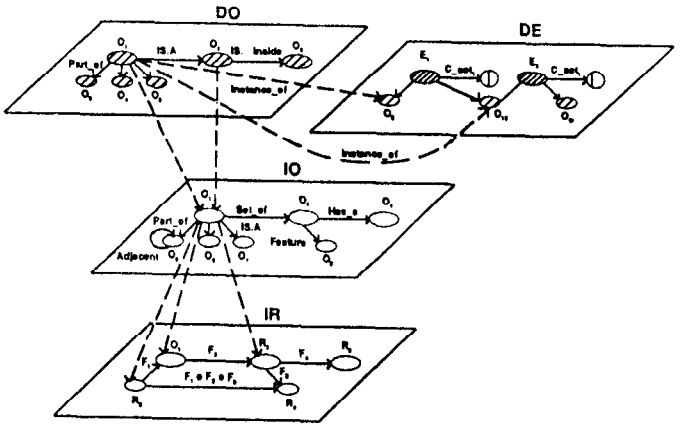

Figure 1: The IR, IO, DO, and DE planes in the layered data model of VIMSYS.

The IO plane has three basic classes of objects: images, image features, and feature organization. These objects are related to one another through set_of, generalization (is_a), and feature_of relations. Thus the elements in the IO plane are connected in an object graph. A partial view of the IO plane is shown in Figure 2. Note that **image_set** is chosen as a basic object, mainly because the domains that we have investigated show that often im-

[1] A graph is a two-row table of values

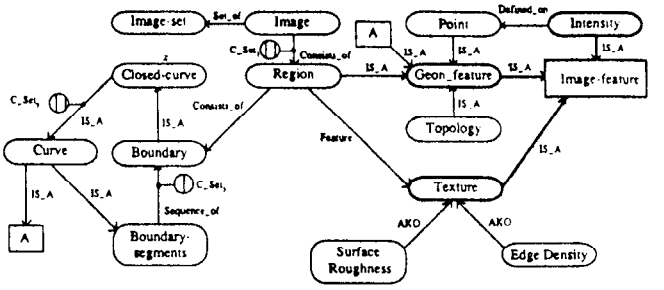

Figure 2: The Object_Attribute_Relationship Diagram of the Image Object plane.

ages are treated in pairs, as a time sequence, or as a set of entities, where each member exhibits a different property. A typical example is the political, terrain, and vegetation map of the same geographical area. Instead of creating a dynamic stack as in PICDMS [4], we choose to perform set-like or sequence-like operations on such images. It is imperative that as images are composed of **regions**, they too would be subjected to the same operators. It is a general rule that if an operator is defined on an image entity, then it is automatically defined on all its decompositions. An additional attribute of regions is that each region instance directly contains a reference to its spatial identity in the parent image instance. The line features, in turn, contain a reference of the region instances that they form a part of. The localization of a region in an image has not been discussed in this context, because localization is a process, not a part of data model. In the experimental stages of our design, we do not concern ourselves with finding an appropriate spatial data structure, and are using R^+ trees [5]. Other data structures such as the Buddy-Tree [11], the LSD tree [12], and z-indexing [6] are under consideration for performance evaluation in future.

In Figure 2, it can also be observed that all image features are members of the generic class **image_feature**, further classified into **texture, color, intensity** and **geometric** features. **Point, line** and **region** are subclasses of geometric features. **Surface_roughness, edge_density** and **orientedness** are prototypes of texture features. A new constructor defined in this plane is **append**, which merely combines two existing features to create a third composite feature. **Point_intensity** is a result of an append operation between point and intensity. New feature types can also be created by taking any ordinary or composite feature and applying any of the representational constructors. A **histogram** is thus defined by graph_of(intensity,int), while a **texture field** is defined by matrix_of(append(orientedness,point)). When one feature-entity can be related to another feature-entity with a **feature_of** link, the two features have a functional relationship in the IR plane.

A complex feature defined in the IO plane, a **topology_network**, is a connected network of regions. The semantics of the network is as follows. Assuming a well-defined convention of direction, each arc connecting two regions in the network has a set of regions to its north, south, east and west quadrants. At the representation-level, therefore, a network is set of involved regions, coupled with a 4-column matrix of region sets. Thus, representationally, the topology network is a complex object. At the IO level, an **arrangement** is a topology network with a name and a set of necessary conditions. These conditions would imply, for example, that **adjacency** requires a symmetric distribution of the matrix elements. **Containment** is also specified in a similar way.

The DO plane basically consists of a semantic level specification of domain entities, built upon the two previous levels. The objects are connected through an object-relation graph. Any object in the DO plane may be a subclass or prototype of one or more objects in the IO plane. In our example domain of SAR images of the Arctic ice floe, an **ice_image_set** is a subclass of image_set, while the semantic feature object **open_water** inherits from both surface_roughness and color. Just as in the case of objects in the IR and DO planes, if objects DO_1 and DO_2 are subclasses or prototypes of objects IO_1

and IO_2 respectively, then the former pair are constrained to reflect all the relations between the latter pair. The definition of an **ice_region** is recursive through the relation composed_of, because any ice region may contain other ice regions. For each object of the DO plane, an emanating relation arrow implies that in the methods section of the source object, there is a function that generates the destination object. The destination object only transmits a request message with the appropriate parameters. Thus, if **ice_image** has_a **geo_code**, then the latter sends a request to **ice_image**, which invokes an internal method to find its own latitude and longitude limits. In this case, it is performed by sending a secondary message to the image header, where this information is retained as a local attribute.

The DE plane has been included in the data model to accommodate for events defined over image sequences. For the applications we have considered so far, we did not need to model an explicit time parameter [13, 14], nor an explicit temporal relation such as happened_before. The concept of event in our domain is an ensemble of features collected over a sequence of images. For example, a rotation in the domain of ice images is an event associated with at least a pair of images in which two corresponding ice regions are rotated with respect to each other. The event rotation is actually a set of the type image region, together with an attribute angle_of_rotation, a set of constraints, and methods for finding regions that satisfy the constraints. We choose to separate the constraints from the methods because the constraints are declarative units compared to the methods of implementing them, which may change with advances in a domain. An integrity rule imposed naturally is that a constraint insertion is not complete unless a method is associated with it. The implementation of the integrity rules for various classes of data and knowledge forms a separate design topic and is not covered here.

All of the object descriptions in the above planes are for classes and prototypes. The instance images consist of the header records for the images, along with a pointer to the images themselves, and some presentation information which we shall discuss in relation to the User Interface. The instance features just contain specific values for the attributes, and a reference of the instance image. Of course, the instance features also reflect the same relations with other features by maintaining references to them. For the purpose of access and referential integrity, references to each image instance is also maintained in each of the classes and prototypes of which it is an instance. So each individual image instance has a multithread access to it. This achieves the essential benefit of our design. An image instance can be accessed from objects specified any combination of planes. If an access needs a feature that is not precomputed for an image but the necessary representation and the method for computing it does exist, then such a feature instance can be created on the fly.

3.2 Query and Knowledge Model

The multiplane multithreaded access to each image instance allows any query from any object set in one or more planes. We also allow the following imprecisions in formulating the query:

1. The attribute of an object is mentioned but its value is left underconstrained.
2. The user uses a semantic object name that is not defined in the system.
3. A similarity-based query is forwarded, where the user presents an exemplar image instance, but only incompletely specifies the feature attributes that are important for conducting the search. Wang [15] characterizes similarity based queries in the following way: If **D** is a database of objects and **T** is a target, then the query classes are
   - Find k objects for some k in **D** that are closest to **T**.
   - Find the closest object of **T** in **D**.
   - Find the objects in **D** that are within some distance ϵ of **T**.
   - Find the k objects in **D** that are farthest from **T**.

- Find the farthest object from T in D.
- Find the objects in that are beyond a distance of ϵ from T in D.

In general, every similarity query is a range query given an arbitrarily specified range (we shall introduce one more element of complexity later). The three issues here are the identification of the necessary attributes, the choice of the range bounds, and the specification of a distance measure that quantifies similarity. These issues are discussed later, here, we continue with a general model of query processing.

Our query model differs from other object-oriented query models (e.g., [9, 16]) in that the user does not explicitly specify a complete **select** operation in one formal statement, but the specification is performed incrementally. An important part of our task is to formulate the select operation as precisely as possible to minimize access efforts. We use a graphical user interface as a query language, a Knowledge Module to direct the user refine a query, and a data dictionary to store some coarse statistics on the database. The basic style of query formulation is influenced by [7]. Once the user specifies the **domain** of interest, the user is requested to specify an **aspect** of interest in the domain. An aspect is a *predefined view* of the Object_Attribute_Relation (OAR) schema with a unique name. Given the domain-aspect combination, the Knowledge Module searches a table to determine a corresponding subgraph of the OAR structure in the DO and DE. As in [7], the attributes are not displayed, only the objects and their relations are. The user then chooses as many of the objects (and relations) that are necessary in his or her view of the query. The Knowledge Module then analyzes the objects with respect to the domain-aspect context desired, and for each entity selected, displays the relevant attribute set. Again, the user chooses a subset of these attributes for query. For each of the selected attributes the user specifies, textually or otherwise, the restrictive predicate on that attribute. [2] This process completes the user's initial query formulation. Internally, the query specification fills out a form that has the data structure W given below:

```
domain
aspect
record {plane object attribute restriction}
```

The next stage of processing is to complete a query if required. Our definition of completeness is to rewrite terms in the query until every terms refers to objects in the IO plane only. This is performed by the Knowledge Module using the Data Dictionary as a supportive collection of facts. The basic idea behind the decision making is to check if the user's specification of the required object, attribute and values lead to a small enough instance set. In the next three subsections we discuss the three components of the Query Processing System.

3.2.1 User Interface

In the user interface windows are categorized into menus, forms, selection buttons, text windows (including system responses and error messages), image windows, system view windows, color palette with scroll bars, icons and sketch pads. With the menus the user is helped to traverse down a decision tree which specifies the domain of interest, the aspect of interest within the domain, and the attributes of interest for each feature. The forms aid to fill in attribute values that may be specified textually by prompting the user. In the domain of ice images, the aspect of interest may be motion, and the features of interest may be the regions greater than 200 sq. km., with pressure ridges longer than 10 km., that rotate by more than 30° over a period of two days. Figure 3 shows how it would be presented with forms and menus. In the same figure note the use of the term OTHER. This choice implies that the user would like to specify an attribute not defined within the database. Then the user is asked to *compose* the attribute by using lower level features. This can be done either using the internal constructors in a text window, or in the graphic sketch pad by dragging feature and relation

[2] The term predicate is to be taken in a broader sense because it would often evaluate to a non-Boolean.

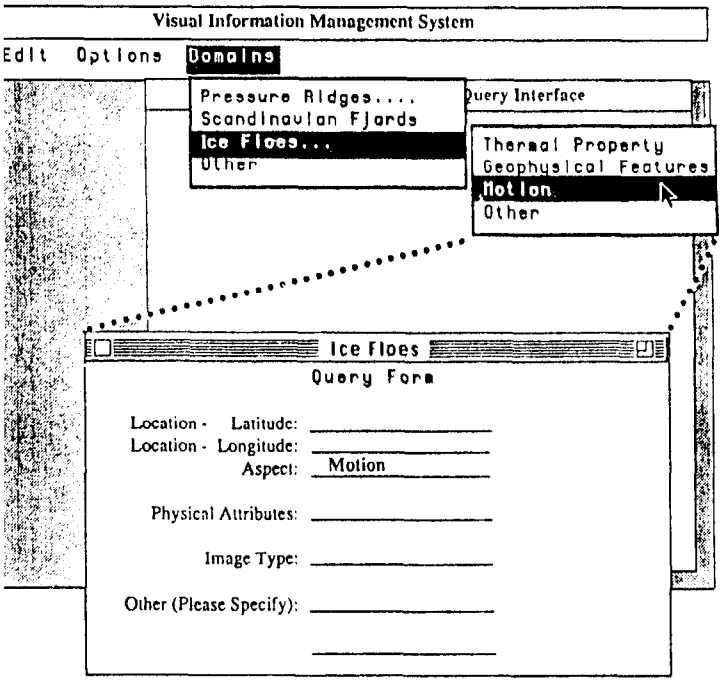

Figure 3: A Display from the ice image domain showing how user interaction formulates the query.

icons. The image window is used for displaying the scanned and retrieved images. The image window allows specialized operations such as choosing subimages, regions, point and line features by button and mouse operations. Moreover an output image window has internal operations such as pan, zoom, superpose text and highlight feature. The system classifies icons into feature icons and image icons. We have already described a use of feature icons; an image icon is a miniaturized representation of the image retrieved. The case where multiple objects are retrieved and the user is expected to choose from among them, the icon is presented instead of the image to save expensive I/O operation of actual image retrieval. The iconized image is a bitmap, as opposed to full gray-tone or color image. As every instance feature directly or indirectly contains a localization information within an image, parts of the bitmap icon is marked with the location of the features used to retrieve it. The system view window is to show the advanced user an Object-Attribute-Relation diagram of the system at each level, so that the user can formulate the query, by choosing not only objects and attribute values but access paths that would reach the desired instance images. The result of the query specified this way generates a well-formed path expression to reduce the system's effort to create an efficient access pathway from a relatively ill-specified query. For example, the color for a "reddish" rug may be made more specific by letting the user start on an average red color on a window and manipulate the hue, saturation and lightness scrollbars until he chooses a few shades of red that he wishes to further explore. The cases where the user makes multiple choices of feature vectors forms, they are combined to form an OR-ed query in the IR plane.

3.2.2 Knowledge Module

The principal role of the Knowledge Module is to control the navigation using as much domain information as possible. The role of the domain information becomes clear when the system has to decide from user responses whether a query has been adequately specified. It is imperative therefore, that the Knowledge Module also plays a role in determining the access path for a query.

Ee use the object oriented paradigm in the following way. The KM consists of an OAR graph of the four-plane data model, complete with the structure of each object. In this graph, called **d-graph** (for decision graph), we associate with each object, a set of methods to determine if that object was adequately specified by the user. This has the advantage of distributing the knowledge among objects. A controlling program passes to each referred object, a message, giving a copy of the initially completed query form. The objects respond by returning a message that specifies what *more* information is necessary to adequately specify itself. The methods section of each object contains a set of rules to make this decision. Any defaults on attribute values are obtained through inheritance. The Data Dictionary is taken as a read-only information structure, that can be accessed by all objects. However, sometimes the rules for any one object depends on the results of other computations. For a totally distributed processing system, communication between objects leads to the problems of message queueing

and associated network management. Therefore, we combine the distributed OAR system with a centralized Query Manager. The inclusion of a centralized control at once raises some special problems. One problem is that the object responses are asynchronous; secondly, often a decision would have to depend on a combination of object responses. For the second problem, the theoretical worst case solution is combinatorial with the number of objects involved. We have circumvented the problems by taking some engineering decisions. During system development a domain expert has been consulted to outline and prioritize attributes - the object response includes the priority of each missing attribute. The objects are served in the order of the highest priority missing attribute, and are processed only after the responses from all the concerned objects are received. The Query Manager operates on the basis of a set of rules and a work space that retains the total pathway traversed upto the current point of time. The rules are used to trigger an action which can be any of

- requery: which specifies the content of a message to the user requesting him or her to provide more information on any attribute. The format of the request shares the same data structure *W* as the internal query form, except that the restriction clause is changed into a specific requirement clause. An example would be

  object: pressure ridge **attribute:** ?ice-type
  requirement: threshold on region histogram

- display: which specifies the form of window to be triggered and a pointer to the data structure *W* that the user interface knows how to interpret.

- addobject: which is used if the query presents a new semantic object that is definable through a composition of existing objects and attribute at other planes. At first the addition of the node and the composition links are volatile. The d-graph and the Data Dictionary, get updated *persistently* iff the addobject operation produces a non-null output. The rules for the new object are introduced by a separate interaction.

- navigate: which sends messages to other objects, thus propagating the frontier of activated objects in the d-graph. The results of the previous response updates the contents of *W*.

- subquery: which uses the object response to formulate a partial query that can be submitted independent of other parts of the query. Typically, the response to such a subquery would fetch the instance identifiers for objects in the IO plane.

- terminate: which denotes that a complete query can now be formed using the information obtained.

A similarity-based query is considered unspecified unless we have the attributes on which the similarity is to be calculated, the distance function which will determine the degree of similarity, and a threshold which limits relevant choices. Initially the user is requested to choose the attributes of comparison. If the user considers that the set of choices offered are inappropriate, the user is asked to compose the attribute as mentioned in the previous section. As the user makes the choice of attributes, known feature extraction methods stored in a library are invoked on the scanned input image. As for objects, the result of the composition for similarity is also stored as a *persistent* entry in the d-graph if the user makes such a request. The ultimate purpose of the decision is to develop clear query classes so that with each query class, we can associate some cluster of features and attributes. This enables us to take anticipatory memory management decisions when the system grows larger. The system allows the reuse of an already retrieved picture, to specify a further query. This is performed by creating a separate workspace when an image is actually retrieved (in contrast with just its iconized bitmap). The workspace contains all the precomputed entities, related to the retrieved image in all the four different planes. The access path of a semantically

'similar' instance image along any set of precomputed attribute set would branch off from the retrieved access path only by a few levels of abstraction. The system does not have to start afresh to construct the entire access path for the newly requested image.

The next step in a similarity-based retrieval is to select thresholds on attribute values, and a distance norm to combine them into a composite measure. For an attribute whose domain is a single number on an integer or real scale (such as brightness), selection of a threshold is implemented through a user manipulated palette. The attributes whose domain is set-valued such as ice-class regions present in the query image (it can be one of five distinguishable categories), the sets may have been created either by some domain specific classification algorithm or by simple enumeration. In the first case, thresholds can be set on the underlying numerical representations. For example, if texture and average intensity are the basis of ice classification, then the similarity can be considered as a range setting operation around these parameters. For enumerated sets, the similarity is provided just by a weighting factor to each of the set members. weights are adjusted by the user as before. Thresholds on vector and matrix valued attribute are set on each element, and in case of matrices both row and columnwise. Once the thresholds are set, the composite similarity measure should be determined. We are currently experimenting with three common measures, namely, weighted average, Euclidean norm, and city block distance. The present design would use the Euclidean measure as a default for ordinary users, while the expert user can choose between the norms. The part of the KM dealing with similarity-based retrieval is mostly procedural. The fact-base with which the procedures work are maintained in the Data Dictionary.

3.2.3 Data Dictionary

The Data Dictionary maintains information about instances of all the objects, features and attributes. It is a collection of tables, primarily organized by the planes. For each plane the table maintains, the number of objects, a list of objects alongwith the subordinate attributes for each, and the number of instances of each object. For each object-plane combination it also maintains a table for the each attribute, the number of instances of each attribute, the domain of the attribute, its priority with respect to the object as designated by the expert, and a default threshold setting method. Most of the action of the KM performs a search operation of specific entries in these tables. The tables are updated by the insertion procedure, which we shall leave beyond the scope of this paper.

3.3 Access Structure

While in the data modeling standpoint the features may be viewed as abstract data types, from the indexing standpoint they are considered as access paths that have been precomputed to grant efficient retrieval. In this section we explore the issues involved in the design of the access structure. We shall present what we have identified as the problems, and outline the characteristics of the solution. The methodology to reach the solution requires extensive research efforts; in the current stage of our preliminary prototype design, we have taken some pragmatic measures to circumvent the problems.

The final aim is to locate a set of suitable image headers along with pointers to appropriate region (or other) in them. For our purpose, all indexing is performed at the level of entities in the IO plane. Queries generated at higher levels, are navigated to their IO level definition and then fast access methods are employed on the attributes. The next step, namely fast location of the region segments from the images, is the problem of spatial data management, and as mentioned is not our major focus in this paper. In terms of the attribute space alone, the access structure problems can be stated in the following stages:

1. The domains of attributes of the IO plane can assume the form of a number of continuous and discrete mathematical structures such as real lines to networks. Not all of them are metric

domains in which a total order is naturally defined. For some structures such as enumerated sets and hash-indexable strings, the transformation to a metric domain is simple. For other structures such as matrices and networks, it is not obvious, and is specific for application domains. Some structures such as networks, *relative* distance measures have been proposed, but they do not provide a total ordering mechanism and usually turn out to be a compute intensive problem. For these methods index updation at insertion time becomes a difficult problem.

2. If we consider only the totally orderable attributes, a mechanism is necessary to create a composite total ordering, without which the multi-attribute information cannot be clustered for efficient access. In our case it makes retrieval difficult on the basis of a composite similarity query, as discussed before.

3. The next level of complexity is introduced if a similarity-based query is specified as a *weighted* range query on multiple attributes, where the weights signify how important an attribute dimension is by the user's estimate. The weights can be viewed as a simple priority ranking on the order of retrieval of attributes. They can also be viewed as a post retrieval processing problem, which at once places a heavy computational burden on the system, both because the retrieval produces redundant information and post processing adds to this computational cost.

4. Our data model implies that relations, unlike normal object oriented databases [17], can also be queried along with object-attribute combinations. The problem here is that the indexing is to be performed on the set {first_object_spec relation_spec} or on {first_object_spec relation_spec second_object_spec}. One can first index on the object and navigate from the object to the relation, but this is an efficient method.

In our current status, we have often sidestepped the issue of metricizing the individual and composite attribute space. The correct approach to the problem involves the rigorous analysis revealing which properties of each type need to be preserved under the metric transformation. We have only begun research into these issues in our example domain. Most of the attributes we have chosen to use are naturally metricized. The weighted attribute problem has been considered as a simple priority ranking of attributes. In our current system, we have not yet addressed the issue of indexing on object relation combinations.

4 Conclusion

This paper describes the current status of our project on visual information management systems. We have introduced a new data model, that directly uses the image features, their inter-relations and representations. The use of multiple representations of image objects, functional relations between the representations, and the overall object-oriented framework are the features of the layered data model. The second contribution is the introduction of a knowledge processing module to refine the user's query. To our knowledge, this is the only system that allows the user to progressively define a similarity-based query, a flexibility that results from the use of a knowledge component. Our work touches base with many research issues that are still quite open.

5 Acknowledgement

The Authors would like to acknowledge the invaluable implementational support provided by Karen Brady, Professor Elliot Soloway, Eric Yager, Deborah Swanberg, Gyan Bhal, Dan Ross, Mike Wynblatt, Eric Krueger, Chris Warock, George Chen and Stuart Freedland. Amarnath Gupta was supported by a UNDP Fellowship during this work.

References

[1] W.I. Grosky and R. Mehrotra, "Index-

Based Object Recognition in Pictorial Data Management", *Comput. Vis. Graph. and Image. Proc.*,Vol. 52 No. 3, pp. 416-436, 1990.

[2] F. Rabitti and P. Stanchev, "GRIM_DBMS: A Graphical Image Database Management System", in *Visual Database Systems*, T.L. Kunii (ed.), Elsivier Sc. Pub., 1989.

[3] S.K. Chang, *Principles of Pictorial Information Systems Design*, Prentice Hall, 1989.

[4] T. Joeseph and A.F. Cardenas, "PICQUERY: A High-level Query Language for Pictorial Database Management", *IEEE Trans. Soft. Eng.*, Vol. 14, No. 5, pp. 630-638, 1988.

[5] N. Rossopoulos, C. Falotsos and T. Sellis, "An effificient Pictorial Database System for PSQL", *IEEE Trans. on Soft. Engin.*, Vol. 14, No. 5, pp. 639-650, 1988.

[6] J.A. Orenstein and F.A. Manola, "PROBE: Spatial Data Modeling and Query Processing in an Image Database App.lication", *IEEE Trans. on Soft. Engin.*, Vol. 14, No. 5, pp. 611-629, May 1988.

[7] M.K. Leong, S. Sam and D. Narsimhalu, "Towards a Visual Language for an ObjectOriented Multi-Media Database System", in *Visual Database Systems*, T.L. Kunii (ed.), Elsivier Sc. Pub., 1989.

[8] W.I. Grosky, "A Logical Data Model for
Integrated Pictorial Databases", *Comput. Vis. Graph. and Image. Proc.*,Vol. 25 No. 3, pp. 371-382, 1984.

[9] W. Kim, "A Model of Queries in Object-Oriented Databases", *Proc. 15th Int. Conf. VLDB*, pp. 423-432, 1989.

[10] C. Lecluse, P. Richard and F. Velz, "O_2, An Object-Oriented Data Model", *Proc. SIGMOD*, 1988.

[11] B. Seeger and H-P. Kriegel, "The Buddy-Tree: An Efficient and Robust Access Method for Spatial Data Base Systems", *16th Int. Conf. VLDB*, pp. 590-601, 1990.

[12] A. Henrich, H-W Six, P. Widmayer, "The LSD Tree: Spatial Access to Multidimensional Point and Non-point Objects", *15th Int. Conf. VLDB*, pp. 45-53, 1989.

[13] R. Elmasri, G.T. Wuu and Y.G. Kim, "The Time Index - An Access Structure for Temporal Data", *16th Int. Conf. VLDB*, pp. 1-12, 1990.

[14] S.Gadia, "A Homogeneous Relational Model and Query Language gor Temporal Databases", *ACM TODS*, Vol. 13, No. 4, pp. 418-448, 1988.

[15] T-L. Wang and D. Shasha, "Query Processing for Distance Metrics", *16th Int. Conf. VLDB*, pp. 602-612, 1990.

[16] B.P. Jenq, D. Woelk, W. Kim and W.L. Lee, "Query Processing in Distributed ORION", *Proc. Int. Conf. on EBDT*, pp. 169-187, 1990.

[17] W. Kim, "Object Oriented databases : Research Directions", *IEEE Trans. Knowl and Data Eng.*, Vol. 2, No. 3, pp. 327-341, 1990.