

ImAge: an extensible agent-based architecture for image retrieval

Hiranmay Ghosh¹, Santanu Chaudhury^{2,*}, Chetan Arora², Paramjeet Nirankari²

¹ Centre for Development of Telematics, 9th floor, Akbar Bhawan, New Delhi 110021, India;

E-mail: ghosh@cdotd.ernet.in

² Department of Electrical Engineering, Indian Institute of Technology, New Delhi 110016, India;

E-mail: santanuc@ee.iitd.ernet.in

Abstract. We present an open and extensible architecture, ImAge, for content-based image retrieval in a distributed environment. The architecture proposes the use of system components with standard public interfaces for implementing retrieval functionality. The standardization of the components and their encapsulation in autonomous software agents result in functional stratification and easy extensibility. Collaboration of the independent retrieval resources in ImAge results in enhanced system capability. Reuse of existing retrieval resources is achieved by encapsulating them in agents with standard interfaces. The addition of independent agents with domain knowledge adds the capability of processing conceptual queries, while reusing the existing system components for feature-based retrieval. A communication protocol allows the declaration of the capabilities of the system components and negotiations for optimal resource selection for solving a retrieval problem. The use of mobile agents alleviates network bottlenecks. This paper describes a prototype implementation that validates the architecture.

Key words: Content-based image retrieval – Digital library – Multi-agent system – Distributed architecture – Conceptual query interpretation

1 Introduction

The availability of digital images for different application domains calls for effective retrieval tools. An image, which is a two-dimensional array of image pixels, encodes an enormous amount of information. Research in

content-based image retrieval investigates new ways to interpret image data (pattern recognition algorithms) and establishing similarities between the images using such an interpretation. The effectiveness of different retrieval algorithms depends on the application. The same set of images needs to be interpreted differently by different retrieval methods to meet diverse user requirements. In a networked world, the image collections, the retrieval tools and the users are expected to be distributed across multiple locations. This paper addresses the problem of designing an image retrieval system that can fulfil the needs of a distributed environment and disparate constraints.

The existing image repositories adopt different retrieval paradigms and implement a few retrieval methods. Some of them use aggregate image features such as the color histogram and texture [5, 10], some use segmentation information, i.e., image regions with relatively homogeneous properties [4], while some others associate semantic meaning to the image segments using some domain knowledge [6, 15, 18]. An image repository adopts some data model for representation of the image data. The images are indexed with one or more entities in the data model and are retrieved using a combination of these indices in the context of a query. The nature of the queries that can be satisfied by a repository is limited by the data model implemented in the collection. For example, a retrieval system like Webseek [5] that supports some semantic categorization of images and indexing based on aggregate image features in its data model cannot support a query requiring segmentation. The different image repositories on the internet exhibit heterogeneity with respect to the data models and hence, with respect to their access mechanisms.

An integrated framework for a multimedia digital library reusing the existing heterogeneous network re-

* Correspondence to: Santanu Chaudhury

sources has been attempted in UMDL [2] and the Stanford University Digital Library [19] projects. These systems use some *mediator* software to coordinate retrieval from the multiple repositories, which may have different organizations and different built-in retrieval methods. The loose coupling between the producers and consumers of information and the mechanism of dynamic resource discovery make the systems amenable to easy extension. The systems direct a query transparently to a set of capable repositories. However, the architecture does not enhance the capabilities of the individual repositories. As a result, retrieval is restricted to the repositories having a built-in capability to process a query. Moreover, there can be heterogeneity in the local interpretations resulting in an inconsistent set of documents being retrieved.

There are some examples of *extensible* image databases, where the data-model of a repository can be enhanced by the action of external agencies. In MOODS [12], the system stores a set of low-level media features, while a user can provide the rules for their interpretation using a script language. Thus, the data model of the system can be extended by adding new user scripts. In Mirror [7], some demons visit the database to extract new media features to augment the capability of the system. In either case, the extension becomes a permanent feature of the system, is done *in anticipation*, and cannot be dynamically tailored to the needs of a specific query.

In this paper, we present ImAge, an open and extensible architecture for a digital image library, where the retrieval functionality is implemented through the interaction of standard reusable components. These components represent various entities required for a retrieval system, for example, the query interface of a repository, the data entities that populate a repository, and the pattern recognition routines, that transform a data object into another. The components may have different internal structures but are encapsulated with standard public interface definitions. The different repositories may support different sets of the data and query objects, thereby having their own individual character.

The approach followed in ImAge has quite a few advantages. The definition of standard component interfaces allow separation of the different functional units of the retrieval system, such as query interpretation, classification and pattern recognition methods. These independent modules can be encapsulated into autonomous *software agents*. The agents collaborate with each other during retrieval using the methods defined in their public interfaces. New agents conforming to the interface specifications can be dynamically incorporated in the system, resulting in its extensibility. The agents can declare their capability set, which is used for negotiation in the context of a retrieval. The architecture includes a mechanism for benchmarking these agents against some common benchmark data to ascertain their relative merits. Components encapsulating semantic knowledge can also be added to the system resulting in the capability to process concep-

tual queries. The standardization of the interfaces result in the possibility of independent research teams to contribute image analysis routines and domain knowledge to the system independently of the underlying repository structures. These routines can be used with any image repository resulting in effective resource sharing. They can also build upon one another using public interfaces. It is also possible to include the existing image retrieval resources (e.g., WebSeek [5], QBIC [10], BlobWorld [4], etc.) in the architecture, by encapsulating them into autonomous agents conforming to the standard interface definitions.

The ImAge architecture, which is motivated by UMDL, proposes a new communication framework which allows the autonomous agents encapsulating the different system components to collaborate during a retrieval. The different retrieval resources can be contributed by independent research groups and may exist anywhere in the network. We encapsulate the pattern recognition routines as mobile agents, so that they can travel across a wide area network to the repository sites and analyze the images at their source.

We have implemented a prototype image retrieval system, ImAge, based on this architecture. The basic system supports query by example using the extracted image features. Though the implementation is generic, we have experimented with the system on a collection of tourism-related images. An extended implementation includes conceptual knowledge in the domain of tourism and supports conceptual query. The system can be easily extended to other applications by incorporating appropriate domain knowledge.

The aim of our research is the development of an architecture that will support content-based image retrieval from a multitude of distributed repositories which support a standard retrieval protocol. We explore the possibility of encapsulating the retrieval resources to realize standard interfaces, so that they can collaborate during retrieval. We do not consider the development of specific retrieval algorithms, such as data models and pattern recognition algorithms, as part of this research. There is currently a strong research interest in multimedia retrieval methods and adequate availability of the retrieval resources has been assumed.

The rest of this paper is organized as follows: Section 2 presents an overview of the multi-agent architecture and describes the various roles played by the agents in the system. Section 3 describes the protocol for capability negotiation and selection of agent teams. Section 4 describes the communication architecture for the agents constituting the system. Sections 5 and 6 describe some global policies for formulating search strategy. Section 7 describes vertical extension of the basic feature based retrieval system for conceptual query processing. Finally, we conclude (Sect. 9) with a summary of our contribution and scope of future work.

2 Architecture

ImAge has been modeled as an open society of autonomous and communicating software agents. Each agent in the society implements an independent unit of retrieval functionality. The collaboration of these agents results in solving a retrieval problem. New agents can dynamically join the society and contribute to its growth. In order that an autonomous agent can contribute in an open society, we define some definite roles in the system. An agent participates in the system in one of these predefined roles. We have followed an object-oriented approach. An agent class has been associated with each of the roles in the system. Every agent is viewed as an object belonging to an agent class. Each agent class is characterized by a public interface definition, which defines its functional behavior. Different agents in an agent class implement the public interface in its own way. Each agent class has a generic implementation that implements its public interface. Every agent belonging to a class extends the generic agent and implements its technology specific methods. For example, the generic Search Agent (SA) defines an abstract method *similarity()*, that returns the similarity value between two images. It is extended by every agent of that class with a feature specific algorithm. ImAge puts no restrictions in the internal design or knowledge representation techniques of an agent.

The agent classes in ImAge and their interactions are shown in Fig. 1. A User Interface Agent (UIA) provides the human-machine interface of the system. It encapsulates the knowledge about the users, e.g., a user's preferences, feedback, history, etc. A UIA can implement any type of user interface, e.g. natural language interface, query by example, etc. However, it must communicate the query to the rest of the system in a standard form. A Search Coordination Agent (SCA) encapsulates the knowledge and the heuristic methods for solving a retrieval problem and its optimization. It accepts the user query from a UIA, interprets it and interacts with the

other agents for planning and scheduling the retrieval subproblems. A Collection Agent (CA) forms a layer of abstraction over an image repository. It encapsulates the repository structure and produces a standard view of the different data elements available with the repository. It declares the capabilities of a repository in terms of its query and data services to the external world. A Search Agent (SA) encapsulates a specific image retrieval algorithm. It is developed independent of any repository structure and are made available in the network for public use. These agents can build upon one another to derive a complex data-model. These agents are designed as mobile agents so that they can travel to the collection sites and can analyze the documents at their sources.

Since ImAge allows dynamic growth, the agents in the system cannot be aware of each other's existence. The Registration Agents (RAs) maintain a list of the agents available in the system with their capability descriptions and provide a mechanism for dynamic resource discovery. Since the architecture encourages agents to be freely installed in the system, the system may be populated with a number of agents with similar capabilities but with different performance figures. The Benchmark Agents (BAs) benchmark the agents against a common set of data, which enables optimal choice of agents for solving a retrieval problem. The agent classes are described in more detail in the following subsections.

2.1 User Interface Agent

A UIA provides the human-machine interface of ImAge. It is possible to have different types of user interfaces in ImAge that incorporate different forms of inputs, such as query by example, keywords, natural language input, etc. A user can select an appropriate UIA depending on his/her convenience. Every UIA should, however, translate the query to a standard form which is understood by the rest of the system (see Sect. 4.2). Besides this, a UIA should be able to handle a few other functions, such as the convenient display of results, user registration, maintenance of history, and accepting user feedback.

Since the functionality of a UIA largely depends on the nature of the supported interface, there is no generic implementation for this agent class. A UIA is transparent to the complexity of the actual retrieval mechanism, that involves interaction of many retrieval resources. It views SCA as a complete search engine and submits the user queries to the latter in interactive or non-interactive modes.

2.2 Search Coordinator Agent

A Search Coordinator Agent (SCA) coordinates the retrieval process utilizing the available resources in the system. Collaboration is achieved using a two-phase protocol as in [14]. In the *planning* stage, an SCA identifies an optimal set of image classes in the available repositories and

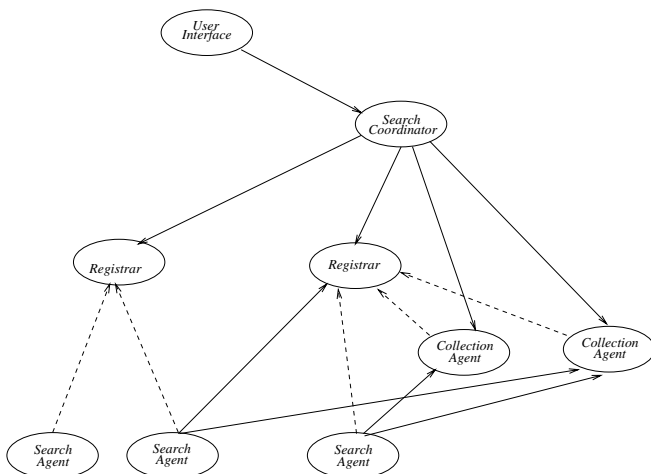


Fig. 1. Agent interaction

an optimal set of search algorithms for each of the classes. The choice of image classes is guided by two factors:

1. The features of the data-model of a repository and the available public PR routines may not match for the repository to take part in a retrieval problem. In that case, the repository is not selected for retrieval.
2. A user is usually not interested in finding out an *exhaustive* set of images that satisfy the query. Thus, it is possible to improve the performance of retrieval by selecting a subset of document classes where the probability of finding the relevant documents is better than that in the others.

The selection of the search agents for a selected document class depends on the query and the data model supported in the repository. In general, while the breadth of search (the selection of the image classes) is determined by the recall requirement, the selection of the search algorithms, where alternatives exist, is determined by the precision requirements and the real-time constraints. These aspects are further elaborated in Sects. 3 and 6.

During the *execution* stage, an SCA contacts the selected CAs, and requests them to schedule the selected SAs on the selected image categories. At the end of the retrieval, an SCA sorts the images by the descending order of their relevance and produces a subset of the images to the user that meets the desired precision requirement.

A generic implementation of the search coordinator implements its overall operational logic, which can be specialized using specific policies and heuristics to select optimal set of resources during the planning stage. It is also possible to implement some specific strategies, e.g. for duplicate removal, while combining the search results from multiple sources.

2.3 Collection Agent

Individual image collections in a distributed environment are characterized by their private data models and retrieval methods. Generally, it is not possible for a software entity to utilize the data, unless it is tightly coupled to the repository. It is in ImAge that we propose an interface that allows the repositories to export a part of their functionality, depending on their implementation. The public view of a repository can be used by software agents that are developed independent of the underlying repository structure.

A CA forms a layer of abstraction over a repository. It declares the public interface of a repository in terms of its public data and query models and exports the data elements as standard objects when required by a retrieval algorithm. Retrieval routines, external to the repository, can complement the native indexing supported in a repository and thereby augment the retrieval capability offered by the collection.

We view different image representations as reusable objects when encapsulated with standard interfaces.

For example, the raw image data can be abstracted to a bitmap format, by supporting public methods like *getHeight()*, *getWidth()* and *getPixel(x,y)* regardless of its physical format. Similarly, a color histogram can be abstracted to 3-D bin values with public methods like *getBinVal(red,blue,green)*. We assume a set of such standard component definitions exist in a development library and the public interface of a repository to be developed using such interfaces. The library can be extended by including new interfaces with innovation of new image data representations.

The existence of standard data objects in a repository does not preclude it from having a private data model, exclusive to the repository. However, the query capability arising out of the private data model must be encapsulated with a standard public interface. For example, the semantic categorization of the images in WebSeek can be implemented using a private data model, while the query interface can be encapsulated as a standard keyword-based query. The images retrieved from one or more semantic categories of WebSeek can be subjected to external image analysis routines to augment its retrieval capability.

The repositories in a heterogeneous environment can implement different data models. However, the data items need to be encapsulated to a standard form for the SAs to build upon them. The static data model of the repository may be built by some manual or automatic process or generated by some of the available SAs. The features extracted by the SAs in the context of a query can also become a part of the data model of a repository, or can be cached for possible future use, depending on the policy of the CA. However, such dynamic capability enhancements increase the complexity of the capability declaration of the CA.

2.4 Search Agent

In ImAge, the capability of a repository is complemented by the action of some *public*¹ pattern recognition (PR) routines in the context of a query. A PR routine accepts raw image data or extracted features through the public interface of a repository and incorporates methods to interpret them in a novel way. These routines are developed independently of any specific repository structure and can be contributed to the system by independent research teams. Moreover, these PR routines can build on one another and can produce a complex view of the image data.

An SA encapsulates a specific pattern recognition algorithm. Besides a feature extraction algorithm, it must implement a method to compare the feature object with a similar one to determine a similarity score for a pair of

¹ The word public does not have any commercial connotation. The PR routines may be used for a fee or free of cost depending on the policy of its provider.

images.² Like a CA, an SA can allow public access to its data model, so that other SAs can build over it. In such cases, the data elements must be encapsulated as standard objects. The interface specification for such data objects are made publicly available in ImAge.

We consider two types of SAs in ImAge. Generic SAs implement some feature-based retrieval algorithms, like colors [22], textures [23] or simple shapes [11]. These agents can be used for many different applications. Specialized SAs are application specific and incorporate a specific intelligence and training set for recognition of complex image objects, e.g., the face of an important personality. They can build upon low-level image features extracted by Generic SAs.

The SAs are designed independently of any repository structure. They complement the native capabilities of the repositories during a retrieval process. It may be necessary to use more than one SA in succession to satisfy a query. The SAs are implemented as mobile agents, so that they can travel to the collection sites and process the media data or meta-data at their source. This feature alleviates network traffic when a large number of multimedia documents from different repositories are to be processed using a series of SAs. It also alleviates the computational bottlenecks by distributing the processing to multiple repository sites.

2.5 Registration Agent

The agents in the open-ended system do not have a priori knowledge of each others capabilities and communication addresses. The Registration Agents (RAs) aid the agent community in dynamic resource discovery. Every agent, which implements some services to be utilized by others, registers itself with an RA. A capability-based search by a client with the RAs yields the set of prospective agents with the required capabilities. There can be several RAs on the network. An agent can get registered with any of these RAs. The existence of multiple RAs distributes the workload and reduces the network traffic. The RAs themselves register with a *meta-registrar*, which helps in their discovery. The address of the meta-registrar is universally known in the system.

A generic RA defines methods to register and to withdraw an agent. An agent is uniquely identified by its URI in a global network. An RA maintains a list of capabilities of an agent in quantitative terms. It defines a lookup method, which produces a ranked list of agents with respect to the desired capability set.

² There may be different similarity computation algorithms associated with the same media feature, for example, histogram intersection and vector space distance for color histograms. We assume exactly one of the methods to be implemented in an SA. Another similarity measure may be implemented in another agent which is known in the system using a different URI.

2.6 Benchmark Agents

Since ImAge encourages uncontrolled addition of PR algorithms in the form of SAs, it is possible that the system will be populated with many SAs with similar capabilities but different performance figures. It is necessary to compare their performances against a common set of representative media data, so that the most suitable agent can be selected for a retrieval. A Benchmark Agent (BA) benchmarks a set of similar SAs against a common set of benchmark data. It typically contains a reasonably large set of sample images drawn from multiple domains. A BA is designed to support benchmarking of a set of SAs, all of which can work with similar data models. A number of BAs can co-exist in the system, having different data-models, and hence capable of benchmarking different sets of SAs.

An RA looks for suitable BAs that can benchmark an SA at the time of its registration. If no such agent is found (which is more likely in case of the Special SAs), the agent cannot be benchmarked and the performance declaration by the provider of the agent is relied on. If more than one BA is found, the performance data for the agent is computed as a union of all benchmark results. The performance data are stored with the RA for future reference.

The retrieval algorithm in ImAge relies on fusion of information from many SAs. The algorithms employed for similarity computation may produce results in different ranges with different semantic interpretation. It is therefore required to normalize the results to a common scale before combining them. We normalize the values to the range [0,1] using Gaussain normalization as in MARS [17].

3 Capability negotiation for agent team formation

The different image repositories in a heterogeneous environment, in general, implement different data models and have different capabilities. A repository can be encapsulated in a CA to conform to some standard query interface definitions. The SAs can exploit these standard interfaces to retrieve image data (or metadata). The standardization of the interface, however, does not imply uniformity. A repository may support a few of the several different interfaces defined with the system. For example, while WebSeek provides keyword-based semantic retrieval, a system like NetView [26] provides color-based classification. Therefore, the SAs required for solving a retrieval problem depend not only on the the query but also on the public interface of the repository. The problem can be modeled as a search problem where the objective is to reach any of a set of destinations (available access methods of the repository) from a source (query requirement) through the intermediate states that can be generated by the set of available SAs. The situation is

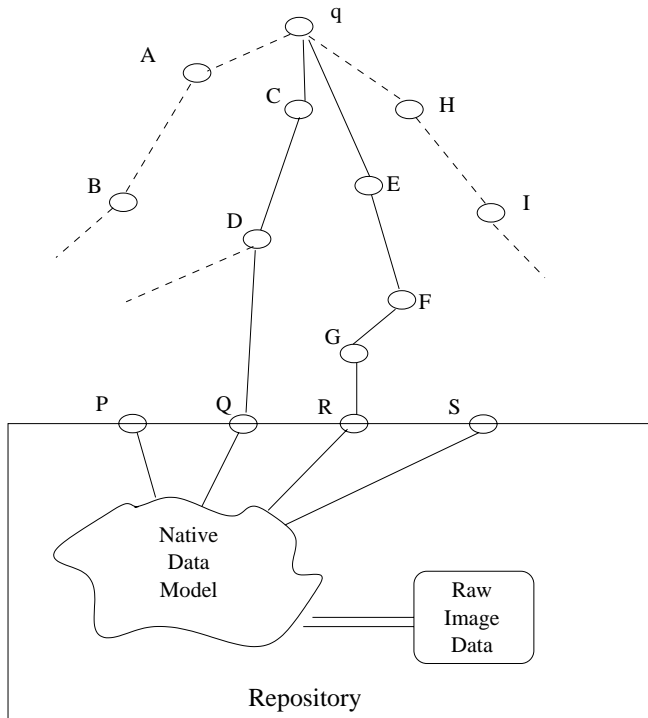


Fig. 2. Agent cooperation

pictorially depicted in Fig. 2. In the figure, q represents a query object, $\{A, B, \dots, I\}$ represent a set of intermediate metadata objects, and $\{P, Q, R, S\}$ represent the public data interfaces in a repository. The edges Aq , BA , etc., represent the action of SAs in transforming one form of metadata to another. The graph is generated dynamically in the context of a query by a planning module, encapsulated in a SCA with the knowledge of the capabilities of the CAs and the SAs in the system. In the example, we find two paths $qCDQ$ and $qEFGGR$ that connects the query to the public data interfaces in the repository. The least-cost path satisfying the performance constraints is chosen for satisfying the query. If no such path is found, the problem is intractable, and the repository cannot take part in the retrieval process. A capability description language (Sect. 4.3) helps the SCA in this reasoning process.

4 Agent communication

The retrieval system proposed in this paper has been modeled as an open-ended multi-agent system, where problem solving is achieved through coordination of a set of autonomous agents, and where new agents can be dynamically added to the system. This necessitates adoption of a standard communication language (ACL) throughout the system. There have been some initiatives on the development of ACL for heterogeneous environ-

ments, namely the KQML³ and the Arcol⁴. FIPA⁵ has released a draft for a standardized version. The motivation behind the development of these languages has been to enable a set of independently designed agents to communicate to each other. The languages focus on the transport and the language levels, i.e., they deal with the mechanism of sending and receiving messages and their interpretation at the performative level. Since these languages are quite generic, their use requires some further understandings between the communicating agents. In order to make a meaningful communication, a sender agent needs to make some assumptions about the receiver agent and vice-versa. Besides this, these languages assume that the agents are implemented using some belief states, resulting in restrictions on the agent design. Finally, these generic ACLs do not provide for any validation of communication policies and communication architecture. As a result, a complex protocol is required to notify the capabilities of the agents to each other and to recover from a situation when an agent receives a message that it cannot interpret.

In object-oriented design, it is possible to define the communication between the objects in terms of the public methods defined with these objects. One object can communicate some information to another by invoking a public method defined with the latter. The semantics of the communication is established by defining the meanings of the objects exchanged, either as a parameter or as the return value, in the method invocation. Distributed computing platforms, such as Java or CORBA, support Remote Method Invocation (RMI), which hides the underlying transport mechanism and enables an object to invoke the method defined in a remote object transparently, as if they were collocated [8]. We have used RMI as the underlying message transport mechanism. Each agent is designed as an object and extends its services to others through a set of public methods, which can be remotely invoked. The use of RMI considerably simplifies the communication architecture. Since the messages that can be interpreted by an agent is defined in its public interface, it is possible to validate most of the messages at compile time, eliminating the need of a runtime protocol. A stronger semantics for the communication is established by the communication objects, since the interpretation of a communication object is largely built into the object itself. Besides, RMI eliminates the need for development of a communication subsystem, such as the KQML router interface library (KRIL).

In a proprietary system, where there are a handful of agents, it is possible to define an interface for every agent individually, and those interfaces to be globally known to every other agent in the system. However, in an open-ended system where agents can be dynamically added,

³ University of Michigan: www.cs.umbc.edu/kqml

⁴ France Telecom: www.arcol.asso.fr

⁵ Foundation for Intelligent Physical Agents: www.fipa.org

there will be a large diversity of messages and the communication will break down, if every agent were allowed to define its own communication language, i.e., its own public interface. To overcome this difficulty, we have developed an ACL based on social commitments [21]. In this model, an agent-based system is viewed as a community of agents with some defined social roles. Every agent in the system participates in a problem-solving exercise in one (or more) of these roles. The communication between any two agents in such system is guided by the roles they perform. With a finite and predefined number of roles in the system, the diversity of communication needs can be contained without either limiting the number of agents or imposing any restrictions on their internal design. The definition of the messages in the system forms the functional specifications for the roles, and any agent can participate in the role by adhering to those specifications. In the proposed architecture, the agents are classified into a few broad functional categories as described in Sect. 2. Each of these agent classes corresponds to a role in the system required for solving a retrieval problem. A public communication interface, that includes a set of methods, is defined for every class of agents. Every agent in an agent class implements the corresponding public interface.

The standardization of the communication interface does not restrict the diversity of operations of the agents belonging to an agent class. The methods defined in the public interfaces of an agent class exhibit polymorphism, since every agent in the class has the flexibility to implement the method in its own way. While some of the communication objects have a fixed interpretation in the system, many are defined in a flexible way with the rules of interpretation encoded in those objects themselves, so that they can be customized to the needs of some specific agents using a content language. The content language is further extended to a query language and to a capability description language, which are used to represent the queries at various stages of refinement and the capabilities of the different agents in the system. These languages are described in the following subsection.

4.1 Content language

In order to offer flexibility in agent communication, some of the message fields are loosely defined as generic *objects* and no type checking is enforced on them at compile time. Examples of such fields include agent capability description, query specification, etc. These objects are polymorphic in nature, i.e., they can be overloaded with different types of information depending on the context. We encode these objects in a descriptive way similar to ASN.1 [1] so that they may be unambiguously interpreted by the recipient agents. The complex data structures are built recursively using fundamental data types, *integer*, *float* and *String*, and construction mechanisms, *fields*, *repetition* and *choice*. We have extended the *repetition*

construct to include logical operations *AND*, *OR* and *NOT*. Every data type is represented by a name. Each data is encapsulated in a class *TypeValue* which is an ordered pair of the type (name) and the value of the data item. A value field can contain an elementary value, fields, repetition, or another *TypeValue* specification.

The vocabulary of the content language is not restricted to a finite set but is left open-ended. Thus, it is possible to extend the language to include new elementary or complex data types which are relevant for some new agents. As a result, it may not be possible for an agent to interpret every message fully. In the distributed architecture, where an agent solves a part of the problem, such a capability is not required. An agent interprets only that part of a message that is relevant for it, i.e., for which it is designed. In fact, the selection of an agent team at any stage of solving a retrieval problem is based on the capability of the agents to interpret the different parts of the (refined) query.

In the following sections, we use the notation $t = v$ to denote a *TypeValue* pair. A more complex type (a recursive buildup of *TypeValue* pairs) is represented as $t_1 = (t_2 = v)$ where any depth of nesting is allowed. The fundamental data types are expressed directly, e.g., by “*string₁*” instead of *string* = “*string₁*”, or *true* for *boolean* = *true*. The absence of a value is represented by \emptyset (null). Repetition is denoted by $[v_1, v_2, \dots]$ and logical operations by $op[v_1, v_2, \dots]$ where *op* denotes a logical operator. For example,

keyword : *OR*[“*tiger*”, “*panther*”, “*leopard*”]

indicates a logical disjunction of the three keywords. Fields are represented by $\{f_1, f_2, \dots, f_n\}$, for example *quality* : $\{cost, performance\}$ indicates that the quality information includes cost and performance fields.⁶ Comments are enclosed between $/ \dots /$.

4.2 Query language

ImAge has been designed as an open-ended system, where different types of retrieval mechanisms may co-exist. This motivates the development of a generic query language which is expressive enough to integrate a variety of query objects. At the same time, the language should not be limited by a static set of vocabulary but should be easily extensible to yet unforeseen image features as well as conceptual query objects.

The query language is defined on top of the generic content language and a query object is defined as a repetition of smaller query objects. Each of these sub-queries, in turn, can be a repetition of even smaller query objects or can be a *choice* of specifications, such as image categories, search specifications, performance specifications, and time constraints. This choice can be extended

⁶ A *TypeValue* pair could also be expressed as fields, but we use a different notation for its special significance.

to include other types of specifications, such as conceptual specifications. Every specification object can be an elementary specification, or a logical combination of elementary specifications. Every elementary specification is associated with a score, indicating its importance in the query.

As an example, consider a query to retrieve *yellow flower* from image category *nature*, with a precision level of 0.8. A “*flower*” is specified as a blob with yellow color and shape similar to that of any of a set of alternate sample blobs, $sample_1, \dots, sample_n$. The query can be expressed as:

```
query = {
  category = (keyword = “nature”),
  feature = (blob = [
    {color = {255, 255, 0}/*yellow*/,
     weight = 0.4},
    {shape = (blobData =
      OR[sample1, ..., samplen]),
     weight = 0.6}])
  performance = [(precision = 0.8)]
}
```

Since the architecture is intended to be open-ended, the vocabulary of the query language is not restricted to a closed set of terminology. Thus, any agent in the system may not be able to interpret an entire query specification. An agent is designed to interpret only the part of the query language required to perform its role successfully in the system. For example, while the field *category* can be interpreted by a CA, a *feature* with *color* attribute is interpreted by an SA incorporating a histogram comparison algorithm.

4.3 Agent Capability Description Language

In ImAge, the SCA forms a team of CAs and SAs for solving a retrieval problem using a capability negotiation process. In order to form an agent team that can effectively solve a retrieval problem, the capabilities of the individual agents need to be known. The Capability Description Language (CDL) allows the CAs and the SAs to declare their capabilities. The language is an extension of the generic content language.

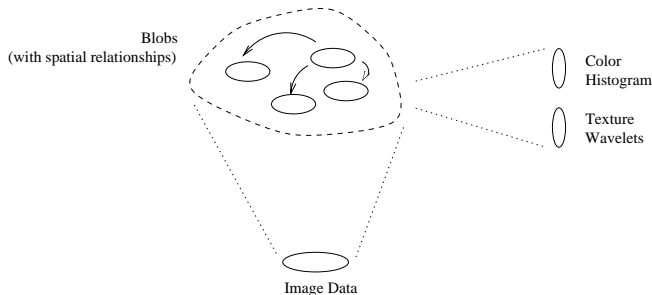


Fig. 3. Data model for a Blobworld-like repository

The capability specification for an SA includes the different types of input objects it can accept, the cost for processing each of the inputs (to extract the metadata of interest) and its performance. For the generic agents, the performance is computed in terms of the axes of the perceptual space through a benchmarking process described in Sect. 5. For special agents, their performance towards their specific functionality is declared by the designer of the algorithm. The capability description for a SA, that implements color matching through histograms, is as follows:

```
capability = {
  input = [
    {data = bitmapImage, cost = 2.0},
    {data = colorHistogram, cost = 0}],
  performance = [
    {attribute = color, score = 0.95}
    {attribute = texture, score = 0.80}]
}
```

In the above example, the SA can accept either bitmap image data or color histogram data with costs 2 and 0 (zero) units, respectively. Since the SA uses histogram for image similarity computation, the feature extraction cost is zero when histograms are directly available. The agent has the performance scores 0.95 towards color and 0.80 towards texture, as judged by a benchmark process.

The CAs need to specify its query services and its data services, i.e., the image features used for indexing the images, and the views to its data model that is available to the external world. The CDL used in ImAge is similar to the Resource Description Framework (RDF) [20] in semantics, but is based on our Generic Content Language rather than XML. To illustrate the capability description of the CAs, let us consider a repository like BlobWorld [4], where every image is segmented into blobs of relatively uniform color and texture. Let us assume that the repository indexes the images based on the color, texture and the relative position of the blobs, i.e., it is possible to support a query like “Find the images comprising (at least) two blobs, $blob_1$ and $blob_2$, where $blob_1$ and $blob_2$ are similar in color and texture to two given samples, and that $blob_1$ is left to $blob_2$ in the image”. The data-model required to support such a query is pictorially shown in Fig. 3. Let us also assume that the repository provides public access to the raw image data and to the blob representation of the images for the SAs to build upon, but the histogram and texture data are private to the repository. The capability description of the CA is as follows:

```
capability = [
  {data = “bitmapImage”, parent = ∅,
   access = “public”, key = false}
  {data = “blobImage”, parent = bitmapImage,
   access = “public”, key = false},
  {data = “colorHistogram”, parent = blobImage,
```

```

    access = "private", key = true}
{data = "gaborTexture", parent = blobImage,
  access = "private", key = true}
]

```

In the above example, the *parent* field indicates the relationship between the different elements of the data model. The *access* field indicates whether the data item is available for public access or is private to the repository. The *key* field indicates whether the images are indexed by this data. It is assumed that instances of *bitmapImage*, *blobImage*, *colorHistogram* and *gaborTexture* (texture as determined using Gabor's functions [23]) are encapsulated in objects with standard interfaces, and that these strings serve as the Uniform Resource Identifier (URI) for those classes.

5 Search Agent characterization and selection

The planning process in ImAge relies on an optimal selection of the search agents. It is therefore necessary to evaluate the SAs against a common set of benchmark data. An agent may be benchmarked for many of its attributes. We have implemented a simple benchmarking scheme for Generic SAs that operate on raw media data.

The content of an image is, in general, characterized by several independent features [17] and the performance of retrieval is determined by the quality of the SAs in comparing the images with respect to these features. We use the subjective judgment of a sample of users as the basis for benchmarking the agents for their performance. We consider an orthogonal perceptual space comprising a set of features, each of which has a definite meaning to the human beings. It is possible for a person to judge the degree of similarity between two images in terms of these features. We have chosen four orthogonal features to define a perceptual space:

1. *Color*. It is judged by the color contents (RGB components and their combinations) of an image.
2. *Texture*. It is judged by the dominant textural pattern of the images.

3. *Shape*. Shape is judged by the contours of the objects contained in the images.
4. *Distinguishing feature*. It is judged by other image characteristics, which cannot be expressed as a combination of color, texture and shape. For example, two images may be considered similar if they contain a common emblem though they are different in all other aspects. Similarly, the signature of an artist on his paintings may be a distinguishing feature to identify the artist.

We assume that the similarity value between two images i_1 and i_2 in a perceptual dimension j can be represented by a number $sim_j(i_1, i_2)$, normalized in the range $[0,1]$, where 1 represents maximum similarity and 0 represents minimum similarity.

The similarity measure between two images in the perceptual space is expressed as a *similarity vector*, each element of which represents a similarity measure in a perceptual dimension. The benchmarking data comprises a set of sample images and the similarity vectors between every pair of images. The sample images are selected at random from a large collection on various themes, believed to exhibit a spectrum of values in the perceptual dimensions. The similarity values of every pair of images in each of the perceptual dimensions are determined as the average of the subjective judgments from a sample population of users in a normalized scale $[0,1]$. The data is represented as a matrix, where every element corresponds to a pair of images, and comprises a *similarity vector* in the perceptual space. This matrix is referred to as the *reference similarity matrix* (Fig. 4).

When an SA m is registered with a registrar, the latter runs it to evaluate the similarity vectors between the image pairs in the benchmarking database. The similarity score awarded by the search agent, which is also in the range $[0,1]$, is then compared with the corresponding score in every perceptual dimension, wherever available, and an error function $\epsilon_{ji}^m = s_{ji}^m - s_{ji}^r$ is computed. Here, s_{ji}^m represents the similarity score awarded by the search agent m to the i -th image pair, and s_{ji}^r represents the similarity score in the reference matrix for the same

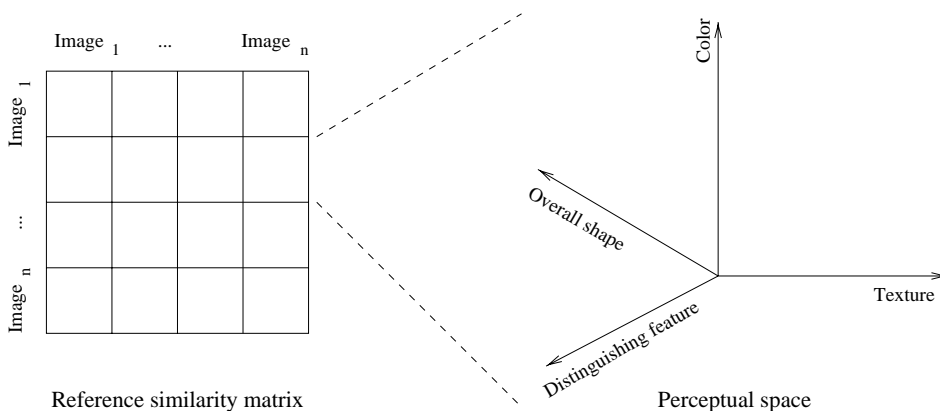


Fig. 4. Perceptual space and similarity matrix

pair of images in the perceptual dimension j . The score of the agent m in a perceptual dimension is then given by

$$\sigma_j^m = 1 - \left(\frac{1}{\kappa_j} \sum_i \epsilon_{ji}^m \right)^{\frac{1}{2}},$$

where the similarity measure for the j -th dimension is available for κ_j pairs of images in the reference similarity matrix. The benchmark data b_m for a search agent m comprises its scores in the dimensions of the perceptual space and its average time of execution for comparing a pair of image, i.e., $b_m = \langle \sigma_1^m, \sigma_2^m, \dots, \sigma_n^m, \tau_m \rangle$, where there are n dimensions in the perceptual space and τ_m represents the average time taken by an agent to compare an image pair. This processing time is measured under some ideal conditions and is weighted by a penalty factor depending on the target execution environment.

In order to select the optimal search agent(s) in context of a query, we need a ranked list of SAs with respect to a requested search criterion. The different search criteria are interpreted as different points in the perceptual space with different projections in the feature dimensions. For example, the criterion *appearance* maps to 0.5 in each of the color and shape dimensions.

The score of a search agent m with respect to a criterion c is computed as $\sigma_c^m = \sum_j p_j^c \sigma_j^m$, where p_j^c is the projection of the criterion c on feature j and σ_j^m is the score of the agent m with respect to the feature j as a result of benchmarking. The agent with the highest value of σ_c^m is normally selected for retrieval.

While many pattern recognition algorithms are designed to deliver good performance for one or more features in this conceptual space, there could be some agents that encapsulate algorithms for specialized applications, for example, identifying the face of an important personality. Benchmarking on the perceptual space is insufficient to capture these capabilities. To account for these agents, we have defined a distinct class of SAs, namely the *special search agents*. Rather than benchmarking, the RA records the feature(s) and score(s) of the agent as declared by the supplier of the agent using the capability description language as described in Sect. 4.3. These scores are processed in the same way as the benchmark scores in the context of a retrieval.

6 Collection expertise

In principle, it is possible to analyze every image in every possible repository online using a collaboration of CAs and SAs in response to a query. Such an endeavor will be prohibitively costly on the internet, where millions of image documents exist. Besides, searching a general collection of images with a limited set of media features is likely to result in poor precision⁷.

A CA encapsulates an image repository and provides a standard query interface if supported by an underlying data model of the repository. The query interface can be used to identify some document classes (subsets of documents in a repository produced as a result of a priori classification), where the query is more likely to be satisfied. In the absence of any classification information, all images in the repository may be considered to belong to a single image class. ImAge utilizes available classification information to prune the search space. For example, consider a query like *yellow flower*, which would be very difficult to satisfy on a general collection of images. However, a collection like WebSeek organizes its collection in several semantic categories, including “*nature/flowers*”. A search for a predominance of yellow color in this image class is likely not only to reduce computational overheads⁸ but also to yield more satisfactory results.

In general, the different repositories may classify the documents with different perspectives independently of the other system components. Therefore, the image category requested in a query may not directly map to one or more categories of a repository. The different categories in the repository will rather have different degrees of similarity with the query specifications. The degree of similarity between a query and the available categories can be measured in many ways. We have implemented a vector-space based method. Every image class is associated with a set of keywords. Some relevant keywords are also associated with a query, either by the user or by some domain knowledge (see Sect. 7). The independently supplied keywords are mapped to some controlled vocabulary using a thesaurus, so that a document class and a query represent two points in a defined vector-space. A similarity value in the range [0,1] is computed as a function of the angles of the two vectors in the vector space. An image category having some non-zero similarity value (where at least one of the keywords matches) with respect to a query qualifies for participating in the retrieval process. Ideally, we need to select a subset of the qualifying image categories which satisfies the required recall value with minimum computational overheads for optimal retrieval.

We use the following heuristic method to find a subset of qualifying image categories that satisfies the requested recall at a low computational cost. Let n_j denote the total number of images in a qualifying image category σ_j , and s_j the similarity value of the category with respect to a query q . An estimate of the number of relevant images in σ_j is given by $\hat{n}_j = s_j \times n_j$. Let $\{m_1, \dots, m_{\kappa_j}\}$ be the set of SAs selected for query q . Let τ_k represent the standard cost of execution for an agent m_k and p_{jk} be the penalty function for that agent for image category σ_j . The total retrieval cost for σ_j in the context of the query is given by $c_j = n_j \times \sum_{k=1.. \kappa_j} (\tau_k \times p_{jk})$.

⁷ Try any feature-based image retrieval engine on the web, e.g., QBIC, Webseek or Blobworld.

⁸ The nature/flowers category of WebSeek contains 853 images against a total of 65 000 image in the whole collection.

We define a figure of merit for an image category σ_j as $f_j = \hat{n}_j / c_j$, and prepare a ranked list of categories using this figure of merit. From the top of the list, we select the least number of image categories so that the requested recall is satisfied. The estimated number of relevant documents in a set of image categories $S_\kappa = \{\sigma_1, \dots, \sigma_\kappa\}$ is given by

$$\hat{N}_\kappa = \frac{\sum_{j=1.. \kappa} s_j \times n_j}{\sum_{j=1.. \kappa} n_j} \times N_\kappa,$$

where N_κ is the total number of image documents in $\bigcup_{j=1.. \kappa} \sigma_j$. The estimated recall when S_κ is selected for retrieval is given by $r_\kappa = \hat{N}_\kappa / \hat{N}$, where the denominator denotes the estimated number of relevant documents in the set of all qualifying image categories.

7 Vertical extension: semantic interpretation of query

A user is usually interested in a semantic category of images, for example, images depicting medieval monuments or snow peaks, rather than low-level image features like color and texture. A concept is an abstract entity and cannot be directly “observed” in an image. However, it leads to some definite patterns in the image forms, recognition of which leads to the belief in the presence of the object, with some underlying assumptions. For example, a medieval monument can be identified in an image by recognizing a combination of media objects representing its domes, minaret and the facade, assuming that the image depicts a place of tourist interest (other assumptions are also possible). A combination of media objects that can be used to identify a concept is called its *observation model*. The recognition of the media objects often requires specialized pattern recognition algorithms, called the *recognition functions*. Experience gained through many observations of the concept allows the association of a number of alternative observation models to a concept in a specific domain. A feature-based retrieval system requires the user to specify a few image features, which is an oversimplification of an observation model and hence, rarely produces good results. Semantic classification of an image requires the combination of evidences from a number of media features and some domain specific assumptions.

Knowledge-assisted interpretation of image documents has been attempted in specific domains [6, 25]. However, the knowledge representations in these systems are tightly coupled with the underlying data model of the repositories. In an “open” architecture, it is possible to interpret a query by an independent agent to a set of alternative observation models, comprising some standard, possibly qualified, image patterns. For example, the media object “dome” can be represented as a set of al-

ternative blobs, each characterized by a specific sample shape. Since the query refinement is undertaken by an independent agent, it is not guaranteed that all the specified image properties will be supported at any particular repository. Once the observation models are available, the SCA negotiates with the other agents (see Sect. 3) to identify teams of CAs and SAs, which can participate in the retrieval.

In ImAge, a concept is interpreted by an autonomous agent, called a *Thematic Agent (TA)* having the requisite domain knowledge. A TA is designed to operate in a closed domain of knowledge. It associates a finite number of concepts in the domain with some property values that can be observed in image documents. It captures the specialization and containment relationships between the concepts that imply property inheritance. A TA can generate the observation model for a concept using its encoded knowledge.

The process of query refinement by a TA results in the selection of a set of observation models from a specified conceptual query. A query intends to express one or more concepts using some descriptors (e.g., keywords). The descriptors are viewed as special observation models which are observed in a query. The concepts and the observation models have a many-to-many relationship. Therefore, it is not possible to uniquely identify a concept using a set of descriptors. We have modeled the relationship as a cause-effect relationship, where (the intention of) a concept can cause a descriptor in the query with some non-zero probability. Similarly, a concept can cause an observation model to materialize in an image. The probability values are associated to the cause-effect relationship with the experience of numerous actual observations by a domain expert. The TA constructs a belief network [16] with the states of the root node representing the set of concepts in its knowledge domain, and the descriptors and the observation models as the leaf nodes. Without any assumption about the user behavior, the states of the root node (concepts) are initialized with equal a priori probabilities. The observation of the descriptors in a query provides virtual evidences for the nodes. These evidences are propagated in the belief network, and the set of observation models that provides the most probable explanation to the observation set are selected for retrieval.

The open architecture has several advantages for conceptual retrieval. The existing SAs can be reused in the context of conceptual retrieval. Thus, the researcher of a semantic recognition algorithm can focus on the knowledge domain and does not have to worry about the feature recognition algorithms. The separation of the semantic knowledge and the media knowledge allows mixed mode queries, where a concept can be qualified by one or more media features. For example, in a query like “white medieval monument”, the semantic entity “medieval monument” requires a knowledge-based approach. The media feature “white” can be detected by a histogram evaluation method. Thus, such a query can

be evaluated in a repository like the Blobworld⁹ with the support of suitable TAs and SAs.

Contemporary semantic databases are restricted to a specific knowledge domain. The open architecture of ImAge allows several semantic classification engines, each specializing in one (possibly overlapping) knowledge domain, to co-exist in the system and to allow retrieval from a common set of repositories. New agents, encoding new descriptions of knowledge domains, can be dynamically added to the system. The knowledge of an individual TA may be limited to a tiny domain and be hand-coded by some domain experts. Many such agents collectively provide a scalable and non-trivial knowledge base. This feature is particularly useful when the same image can be interpreted from multiple semantic perspectives, for example a photograph of a festival has sociological, cultural, as well as tourism-related implications.

The extension to the ImAge architecture for processing a conceptual query is depicted in Fig. 5. The UIA interacts with a TA for query refinement before submit-

⁹ Assuming that it has been designed with an open architecture and can export its data model.

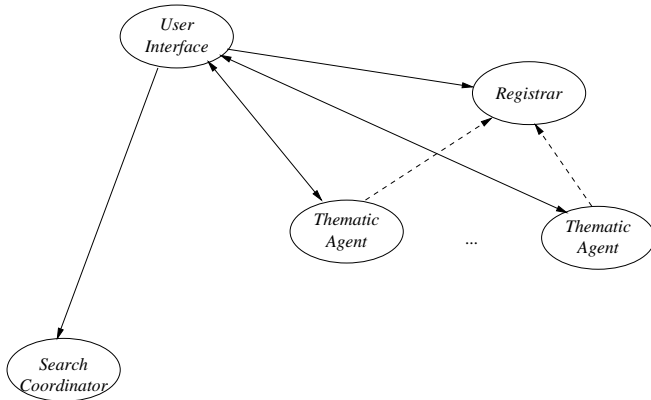


Fig. 5. Extension of agent architecture for conceptual query

ting the feature-based query to the SCA. The several TAs that may co-exist in the system register themselves with the RAs specifying their domain of expertise. The UIA selects one or more of the TAs for query refinement depending on the domain of the user's interest.

The decomposition of a conceptual entity into alternative observation models has another advantage. Even if any particular observation model can be realized at a few repositories only, the retrieval can be supported on a larger set of repositories, since several alternatives exist.

We have extended the query language to incorporate specification of the conceptual entities. The simplest representation of a concept in our system is through one or more concept descriptors. We have used simple keywords, e.g., *monument*, as the concept descriptors. Visual descriptors are more expressive and can be incorporated at the cost of added computational complexity. More complex concepts can be represented by associating concept modifiers with a concept descriptor. We envisage different types of concept modifiers.

1. A concept descriptor may be modified by another to represent a specialization of the concept, for example, *monument.medieval* represents a special type of monument.
2. A concept may also be specialized by associating some property attributes with it. For example, *(monument, [color = white])*. This construct represents a *mixed mode* query, where conceptual specifications are complemented with feature specifications.
3. A third type of concept modifier associates two concepts with a connective to indicate a set of new concepts, for example, *of(hills, India)* indicates a set of mountain ranges, e.g., *{Himalayas, Bindhyas, Nilgiri, ...}*.

The concept specification can have a combination of modifiers, for example, *(monument.medieval, [color = white])*. The concept specification in the query is complemented with a specification for the knowledge domain. It



Fig. 6. A snow peak and the Lotus Temple: violation of closed domain assumption

is necessary, because the same descriptor can have different semantic connotations in different domains. For example, the keyword *monument* has different connotations in the domains of *architecture* and *tourism*. The selection of the TA is guided by this domain specification.

Content-based retrieval for conceptual queries are based on some assumptions about the domain. For example, a white object of triangular shape can be interpreted as a snow-peak assuming that the images being analyzed pertain to natural scenery. The existence of an image like that of the *Lotus-Temple* violates its assumptions and will produce unsatisfactory results (see Fig. 6). The TA associates some descriptors (e.g., keywords) with a concept domain, which are used to select a subset of image classes (see Sect. 6) for retrieval. We implicitly assume that the underlying assumptions for the content-based assumption are satisfied in the selected image classes.

8 Implementation

We have implemented a prototype system to validate the architecture. Our emphasis has been on the development of a set of standard interfaces to achieve separation of the retrieval functionality and encapsulation of available retrieval resources.

A shell for the CAs declares and exports the supported retrieval capability and the data model of a repository. A small private collection in the tourism domain has been encapsulated in this shell. The collection contains about a 100 image documents. We have also encapsulated an available retrieval resource on the internet, namely WebSeek, to demonstrate reuse capability. In the private collection, the images are referenced by some HTML documents. The image file names and text in the HTML documents serve as the annotations to the images. We have manually categorized the image collections in a few semantic categories based on these annotations from the perspective of tourist interest and have associated a set of keywords as descriptors with each of the image categories. Some of the categories are as follows¹⁰:

architecture	/fort	(fort, quila ...)
	/temple	(god, mandir ...)
	/tomb	(chhatri, maqbara ...)
nature	/beach	(beach, shore ...)
	/flora	(flower, green ...)
	/mountain	(hill, snow ...)
	/waterfall	(fall, jhora ...)
	/wildlife	(fauna, tiger ...)

The CA caches the image feature extracted by an SA in the context of a query and thus dynamically augments its data model (though such increments are not declared to

the external world). The cached features are utilized in any future query requiring the same feature description.

In order to encapsulate WebSeek, we have downloaded its semantic classification scheme and encoded it with associated thesaurus terms. In response to the keywords in a query, one or more semantic categories are selected using the method described in Sect. 6. The CGI commands of WebSeek are emulated and the resultant HTML files are parsed to find the URL of the images in a semantic category.

We have so far encapsulated five general purpose image analysis algorithms as SAs. These algorithms include color matching (using the RGB histogram intersection method) [22], chromaticity (using the illumination invariant YV histogram) [9], Funt and Finalyson's method [11], color distribution (correlogram) [13], and texture matching (using Gabor filter) [23]. We have also integrated two special agents, employing Eigen-space based techniques [3, 24], one for recognition of faces and the other for recognition of emblems and miniature flags. These two agents use the same code but different training sets.

A Benchmark Agent (BA) that can benchmark Generic SAs has been implemented. These SAs operate on raw image data. The BA maintains a set of images and some perceptual similarity values (see Sect. 5) for every pair of image in its collection. It schedules an SA on its dataset and computes the normalization and performance parameters for the SA.

A UIA implements Query By Example (QBE). A sample image sample can be selected from a set maintained by the UIA for that purpose. The user can also supply an image sample from a collection external to the system by providing the URL. The other query parameters include: (a) a combination of search criteria (e.g., color, texture, appearance, etc.); (b) a few semantic keywords, which are used to select the image classes with the CA; and (c) performance criteria (desired recall and precision values).

We have implemented a prototype TA in the tourism domain for demonstrating the capability of vertical extension. The TA encodes a handful of concepts of tourist interest, e.g., historic relics, places with scenic beauties, etc. The concepts are hand-coded in this TA and are associated to standard image properties and descriptors. The prototype can be extended using proper domain expertise.

To support the distributed system design on heterogeneous computing platforms, we considered the use of Java and CORBA, so that we can focus more on system development rather than sorting out the networking and communication issues. Implementations of Java and CORBA are available on most of the standard machines. While the Java distributed environment is language specific, CORBA allows multi-language development. Use of multiple programming language and integration with relational databases is, however, possible in Java architecture JNI and JDBC. The mobile code for the SAs re-

¹⁰ The keywords are enclosed in brackets. Some of the keywords are synonyms in Indian languages.

quired platform independent code which is possible with Java. The Java IDL for CORBA was yet to be made available when we started the project and public domain implementation of CORBA was rather slow compared to Java RMI. Considering all these factors, we chose Java as our implementation language and RMI as the distributed computing infrastructure.

We have developed a mobile agent framework over the Java platform that caters to the specific needs of the retrieval system. At the minimum, the mobile agents require an environment that supports portability by hiding the heterogeneity of the physical nodes and a facility for agent migration. The Java Virtual Machine Environment (JVM) provides for platform independent executables and thereby achieves portability. The agent migration is implemented as a layer over the JVM using a dynamic class loader and remote method interface (RMI).

We have implemented the UIA with thin client architecture, where a back-end (server) module runs continuously in the system and provides the various user and query management functionality. It has a persistent memory to remember the history of usage of the system. The human interface is built through a small client module. Multiple users can use the same server together. The server is implemented as a Java application, while the client is implemented as a Java applet. The applet is downloaded on a browser on the user's machine when a user wants to interact with the system.

9 Conclusions

We have developed an open and extensible architecture for retrieval where the retrieval logic and the data model are decomposed into well-defined functional components. Each of the components implements a standard interface, and thus, can interact with each other. They are encapsulated into autonomous communicating agents. A collaboration of these agents realize the retrieval capability of the system. The architecture proposes a communication protocol based on social role models of the agents. The various communication needs of the agents are encoded using a generic content language that rides over the protocol. An existing retrieval resource can be encapsulated in an autonomous agent conforming to the communication protocol and be reused in the system. The component oriented architecture leads to easy extensibility of the system. The use of mobile agents leads to economy in use of the network infrastructure.

A limitation of the architecture is that the repositories that do not have a public interface compliant to the defined protocol cannot participate in the retrieval. Even with a public interface, the participation of a repository depends on the query, the capability of the repository and the available public image analysis expertise. ImAge makes the best effort to make the independent system components interwork without a guaranteed success.

Whenever possible, retrieval takes place with optimal resource utilization.

Currently, there are not enough standards for the development of reusable system components to support a full-scale digital library. Some standardization efforts for multimedia data representation have been initiated in the MPEG-7 forum. We propose that some interface definitions be included in a library that serves as the development kit for the system builders and that can be extended with new inclusions. De facto standards will emerge from such activity.

Though a prototype system has been developed for image retrieval, the architecture is quite general and can be easily extended to other media forms and in general to multimedia documents by the addition of appropriate knowledge-base and pattern-recognition agents. Multimedia retrieval has some specific advantage for conceptual query. The domain knowledge can produce observation models in alternative media forms, which can be exploited for retrieval. For example, a railway steam engine may be identified either by its body shape and smoke cloud or by its characteristic whistle and huff-and-puff. We plan to incorporate textual and audio SAs into the system and build thematic agents that can produce observation models in all these media forms. Another area of research is the global planning and optimization policies with multimedia search methods which will result in the overall performance improvement of the system.

References

1. Recommendation X.680 (07/94): Information technology – abstract syntax notation one (asn.1): Specification of basic notation. www.itu.int/itu-t/rec/x/x500up/x680_27252.html, July 1994
2. Atkins, D.E., Birmingham, W.P., Durfee, E.H., Glover, E.J., Mullen, T., Rundensteiner, E.A., Soloway, E., Vidal, J.M., Wallace, R., Wellman, M.P.: Towards inquiry-based education through interacting software agents. *IEEE Comput.* 29(5):69–76, 1996
3. Carey, S., Diamond, R.: From piecemeal to configurational representation of faces. *Sci.* 195:312–313, 1977
4. Carson, C., Belongie, S., Greenspan, H., Malik, J.: Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Trans. Pattern Anal. Mach. Intell.*, submitted
5. Chang, S.-F., Smith, J.R., Beigi, M., Benitez, A.: Visual information retrieval from large distributed on-line repositories. *Comm. ACM*, 40(12), 1997. <ftp://ftp.ee.columbia.edu/pub/CTR-Research/advent/public/papers97/CACMdec97.pdf>
6. Chu, W.W., Hsu, C.-C., Cardenas, A.F., Taira, R.K.: Knowledge based image retrieval with spatial and temporal constructs. *IEEE Trans. Knowl. Data Eng.* 10(6):872–888, 1998
7. Vries, A.P., de Blanken, H.M.: Database technology and the management of multimedia data in Mirror. In: *Proc. SPIE*, November 1998, pp. 443–455
8. Downing, T.B.: *Java RMI: Remote Method Invocation*. IDG Books Worldwide, 1998
9. Drew, M., Wei, J., Li, Z.N.: Illumination invariant color object recognition via compressed chromaticity histograms of color channel normalized images. In: *Proc. 6th Int. Conf. on Computer Vision*, 1998, pp. 533–540

10. Flicker, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., Yanker, P.: Query by image and video content: The QBIC system. *IEEE Comp.* 28(9):23–32, 1995
11. Funt, B., Finalyson, C.: Color constant color indexing. *IEEE Trans. Pattern Anal. Mach. Intell.* 17(5):122–129, 1995
12. Griffioen, J., Yavatkar, R., Adams, R.: A framework for developing content based retrieval systems. In: Maybury, M.T. (ed.). *Intelligent Multimedia Information Retrieval*, AAAI Press, 1997, pp. 295–311
13. Huang, J., Kumar, S.R., Mitra, M., Zhu, W.J., Zabih, R.: Image indexing using color correlogram. In: *Proc. Computer Vision and Pattern Recognition*, 1997, pp. 762–768
14. Jennings, N.: Controlling co-operative problem solving in industrial multi-agent systems using joint intentions. *Artif. Intell.* 75:195–240, 1995
15. Manmatha, R., Croft, W.B.: Word spotting: indexing handwritten manuscripts. In: Maybury, M.T. (ed.). *Intelligent Multimedia Information Retrieval*, AAAI Press, 1997, pp. 43–64
16. Neapolitan, R.E.: *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. Wiley, 1990
17. Ortega, M., Rul, Y., Chakrabarti, K., Porkaew, K., Mehrotra, S., Huang, T.S.: Supporting ranked Boolean similarity queries in MARS. *IEEE Trans. Knowl. Data Eng.* 10(6):905–925, November–December 1998
18. Ozkarahan, E.: Multimedia document retrieval. *Inf. Process. Manage.* 31:113–131, 1995
19. Paepcke, A., Cousins, S.B., Garcia-Molina, H., Hassan, S.W., Ketchpel, S.P., Roscheisen, M., Winograd, T.: Using distributed objects for digital library interoperability. *IEEE Comput.* 29(5):61–68, 1996
20. Resource description framework (rdf) model and syntax specification (rec-rdf-syntax-19990222): www.w3.org/TR/REC-rdf-syntax/, February 1999
21. Singh, M.P.: Agent communication languages: Rethinking the principles. *IEEE Comput.* 31(12):40–47, 1998
22. Swain, M., Ballard, D.: Color indexing. *Int. J. Comput. Vision* 7(1):11–32, 1991
23. Swapp, D.: Estimation of Visual Textual Gradient using Gabor Function. PhD thesis, University of Aberdeen, Aberdeen, UK. www.csd.abdn.ac.uk/publications/theses/swapp.html, 1996
24. Turk, M., Pentland, A.: Eigenfaces for recognition. *J. Cogni. Neurosci.* 3(1):71–86, 1991
25. Yoshitaka, A., Kishida, S., Hirakawa, M., Ichikawa, T.: Knowledge-assisted content-based retrieval for multimedia databases. *IEEE Multimedia* 1(4):12–21, 1994
26. Zhang, A., Chang, W., Sheikholeslami, G.: Netview: Integrating large-scale distributed visual databases. *IEEE Multimedia* 7(3):47–59, 1998