OBJECT CLASSIFICATION AND IMAGE INTERPRETATION WITH AN OBJECT-ORIENTED SYSTEM

R. Cecchini, A. Del Bimbo, P. Nesi

University of Florence, Italy

INTRODUCTION

Scene analysis is concerned with processing of single or multiple images. One of the final goals of the analysis is to get information about the objects represented in the scene: a synthesis by reasoning process captures the significance of the picture in the first case, and collects the information from the sequence of pictures as a whole in the second case, tracking the moving objects in the frames.

A large amount of work with proposals for algorithms is actually available, Young and Fu (1).

Traditional approach is generally devoted to algorithmic aspects; a lot of work is available in Image Preprocessing, Image Segmentation, and Pattern Recognition both in 2D and in 3D. Object models features are stored into a file system. This file system is often supposed to be reduced in size, leading to very specific Image Processing systems; larger systems typically suffer owing to performance reasons.

Although these small dedicated system have had and still have good applications, the need of large databases for storing image related informations is growing in order to build extensive and multiple application Image Processing systems.

In Artificial Intelligence, first order predicate calculus,(relational programming), which is completely declarative, exhibits good formal properties as consistency and completeness. A suitable reasoning component, the inference engine, can perform deductive analysis. All inferences in classical logic can be seen as set membership. Anyway no information about structural relations is foreseen.

This approach does not seem adequate to treat with some practical applications such as those which require persistent objects or sophisticated Abstract Data Types: performance problems on traditional machines, limitations to database extension limit their usage in real applications.

Owing to the large database requirements, the relational data model has been proposed for capturing spatial information in many systems, especially in geographic data bases, as for example in Berman and Stonebraker (2).

Although this kind of model is easy to implement, it leads to a representation of the spatial information which is "flat", in the sense it does not capture the relationships among the elements or objects in the frame. In particular it has not the necessary flexibility in order to represent hierarchical structures as those pertaining to part-subpart relationships into a frame or to composite objects in the frame, and in general is unable to specify semantic information related to the relationships between the objects.

In fact, if we look inside a typical first generation Image Processing system, we can generally recognize that:

1. Programs in Pattern Recognition work by abstractions. They search for parts and subparts according to visual inclusion criteria, individuate relative positions of parts, detect moving objects in multiple scenes; by reasoning they can individuate the parts names or the class abstractions to which the detected instances belong, guess scene significance, and/or analyze motion.

2. Data models are flat representations (records) of the objects through their features. No semantic information is added.

In general we can say that no correspondence exists between the abstractions and the abstraction hierarchies interpretated inside the application program, and the flat representation made of the features record into the file system or relational database.

We believe that in order to make an Image Processing system more flexible and hopefully faster, the data structure should be Object-Oriented and the knowledge about data structures and relationships should be captured as much as possible.

In the present paper we propose and discuss an innovative integrated architecture of an environment for a multi-user Image Processing System, based on the extensive usage of the Object-Oriented paradigm concepts, applied both in image processing programs and object representation in the database. This will result in a tighter correspondence between the abstraction mechanism present in the program and the permanent storage and organization of abstractions into the database, enhancing both performance and safety.

In addition the adoption of an Object-Oriented DBMS will allow for reusability of data models and model sharing for different applications; concurrence of different tasks which is a key point in large and multi-user Image Processing Systems is allowed.

OBJECT-ORIENTED PARADIGM AND IMAGE PROCESSING

Very recently, Object-Oriented Programming paradigm is receiving a lot of interest in several areas, like software production and language development Cox (4), (5), Meyer (6).

The main advantages of this approach can be individuated in software reusability and extensibility and code sharing.

The Object-Oriented Programming paradigm is the result of the convergence of concepts from many areas of computer science, such as programming languages, Stroustroup (7), Data Bases, Andrews and Harris (8), Operating Systems and Artificial Intelligence, Wegner (3), and is centered around the concepts of abstract data type, Shaw (9), and information hiding and inheritance, Meyer (10).

Object-Oriented Programming mainly differs from relational programming in that relations are captured through predicates in the latter while relations among types are captured and in type hierarchies in the former.

In the Object-Oriented Programming paradigm, the "object" entity encapsulates both status and behaviour, through respectively, attribute variables and procedures or methods. The object is the only proprietor of its attributes and the only one authorized to manipulate them through its methods. Message passing is usually the way in which object procedures can be activated by external objects.

The Object-Oriented paradigm provides a framework in which conceptual modeling of external entities can be made inside a computer system and a close correspondence between real world and the computer can be built, Felsinger (11). Objects are seen as instances of abstract entities, the classes, which capture the data description and behaviour modeling.

Classes are related each other through standard abstraction techniques: specialization (class-subclass relation) and aggregation (class-class_component relation).

Some programming languages are actually available implementing Object Oriented Programming concepts as Smalltalk, C++, Eiffel, etc..

About database systems, there are actually only few examples, e. g. Andrews (8). On the other hand, actually available database system models, like the relational model, lack the power and expressibility of Object-Oriented Languages and have no structuring facility. The information about combining different relations can only be extracted through processing by the user.

As compared with the relational approach the Object-Oriented model provides a very powerful paradigm for knowledge representation.

Pattern Recognition and Object Classification can greatly advance by using Object-Oriented concepts both in programming languages and database. Any spatial entity as we can extract from some frame can be represented as an individual object belonging to some specific class.

The entire image frame can also be stored taking into account both the spatial relationships among its entities and the conceptual relationships.

Communalities of properties can be included through the IS-A (class-subclass) relation and spatial inclusion is instantiated by IS-PART-OF (class-class_component) relation.

An Image Processing system has to work on the main significant classes identified: they will have to be defined in order to take consideration of the main entities involved into the entire process of Image Processing and in particular to allow modeling of pictorial information at different levels of abstraction.

Model objects, which are instances of model classes, will be stored into the system database. They can be build having both attributes, defined in terms of constraints to which the identified object has to cope with in order to match, and methods which resume the standard model behaviour. This will allow object simulation and behaviour matching in addition to shape matching.

SYSTEM MAIN CLASSES

Image Classes

As can be seen from Figure 1, the main classes of the system are related to the main entities which we can recognize in the Image Processing and Pattern Recognition job. They are:
- classes related to the image: "Raw_Image" and "Processed_Image".
- classes related to the Data Base: "Model_Entity" and "Real_Entity".

Raw Image

To this class belong images or sequences of images taken by the recording device (TV camera). Segmentation algorithms, which are fundamental for the Pattern Recognition process are stored as Raw_Image methods.

These and other preprocessing functions are activated through message sending, during the recognition process by the model classes of the Data Base.

Processed Image

The objects belonging to this class are produced by the application of some preprocessing function to some Raw Image. They are mainly subparts of the original image.

These objects can be related to each other through aggregation relations according to different level of resolutions at which the image is analyzed (see Figure 2).

The methods are mainly algorithms for the extraction of structural and behavioral features.

Also these methods are selected during the recognition process by the model classes of the database.

Database Classes

The Object-Oriented database allows a direct representation of information by adding semantic information like generalization-specialization and part-subpart relationships. It provides the best framework for pictorial information storage, which is most naturally organized at different levels of abstraction. In fact though if the image data can be stored as a set of independent objects, more information is gained through storing these objects with their semantic links and in an object based representation.

As can be seen from Figure 1, the objects of our Database are instances of some subclasses of two basic classes: "Model_Entity" and "Real_Entity". The former contain all the

knowledge (at various abstraction levels) needed for pattern recognition and behavior simulation, the latter correspond to already identified or partially identified real world objects.

The hierarchical structures of the two parts in which the Database is conceptually subdivided are not necessarily the same, as there could be the necessity - for pattern recognition purposes - to define model objects without corresponding real world ones. For example, in the "model" part of a data base devoted to vehicle recognition there could be classes like "Two_Wheels" and "Four_Wheels",that could be unnecessary in the "real" part, where all the objects belong to more specific classes, like "Scooter", "Car", etc. In other words, usually the hierarchical structure of the "real" part of the Database is more simple than the one of the "model" part.

Model Entity Classes. The hierarchical structure of the "model" part of the database reflects the knowledge required for a successful pattern recognition process.

In the model Database is contained all the available information about the problem domain, hierarchically organized and distributed between the various levels of abstraction. The higher classes (e.g. "Vehicle") embody a more general knowledge than the lower ones (e.g. "Car").

The model tree is structured according to from most general to most specific classes rule. The model tree has to take into account in addition to the generalization links (IS-A) also the aggregation links (IS-PART-OF).

The attributes of each class are the model features specific to its abstraction level and which allow to differentiate it from the other ones at the same level.

The methods belong to four fundamental categories:

1. Preprocessing Selectors: they select between the preprocessing methods of the Raw_Image class the more suitable one for the subsequent elaborations. For some models they could be absent, as the preprocessing done at an earlier level could be sufficient.

2. Extractor Selectors: they select between the methods of the Processed_Image class the more convenient one for the extraction of the features to be matched with that of the model.

3. Matchers: they compare the features extracted from the Processed_Image with the ones of the model to determine if the real world object is an instance of the class of the model (or to one of its subclasses).

4. Behavior Simulators: these methods allow to simulate the behavior of the real world object, and can be used as a help in the identification process or to predict the future behaviour of the object.

Real Entity Classes. The structure of the classes of this part of the Database is of course strongly dependent on the scope to which the system is dedicated and the type of queries that must be answered.

A Real_Entity instance is created by the system at the end or at some intermediate step of the patter recognition process.

PRINCIPLES OF OPERATION OF THE SYSTEM

Processing and recognition can be performed by traveling over the Database. At each node the underlying classes features determine the branching decision and eventually the additional processing needed in order to perform branching. Matching will ultimately be performed by comparison of features and eventually behaviour of the detected object with Database leaves.(See figure 3).

Process can start at each level of the model tree according to the actual knowledge about the reality under examination. In absence of information it starts with the class at the top of the tree.

As the process reach a node of the tree, it compares the knowledge it has at disposal at that step with the constraints in the node itself and in the immediately underlying nodes: in order to select the right branch to the next node it schedules the opportune methods in the Raw_Image and/or Processed_Image classes.

The structure of the model tree must have a general vision of the structure of the Image Processing and Pattern Recognition system and has to be congruent with the facilities available in the sense that it has to know which methods to use from Raw_Image and Processed_Image classes.

The system works according to the Object-Oriented Paradigm by sending messages to external classes in order to schedule methods.

The process, progressing in the recognition task , sends messages to the Raw_Image and Processed_Image to execute their specific methods in order to have an image clean and segmented into its subparts.

Following the first segmentation step, the process tries to find the right branch in the model tree order to match some model node.

This tree structure is suitable to be implemented together a "pyramid" structure for processing refinement, Fairhurst (12), Nevatia (13). Its lowest level is the raw image (the highest resolution), the next higher levels correspond to lower resolution images got by averaging neighbor pixels, with a two or three reduction factor for each step.

The process can select the proper level of pyramid. It will work with the pyramid highest level (low resolution) at the tree root and may require further processing on the lower levels of the pyramid in order to check for the right branch. This approach permits the reduction of computation times by processing the image hierarchically.

A nice property of the Object-Oriented approach in structuring the System is the possibility of implementation on parallel architectures. There is opportunity for parallelism when more than one path is candidate for branching. In that case multiple processes can be created to follow the different branches.

Classified objects are created as instances

of the recognized class in the real element subtree.

In case of model matching, simulation can be started by scheduling some method which resume the standard model behaviour. Simulation can be executed in parallel with recognition, for example in processing a sequence of frames recognition, in order to allow also behaviour matching.

CONCLUSIONS

In this paper we have presented a proposal for an innovative and advanced Image Processing and Pattern Recognition system which fully implements the Object-Oriented paradigm.

The Object-Oriented Database is the central element of the System: it collects Model_Entities and instances of Real_Entities; Model_Entities constraints drive the recognition process by scheduling the appropriate methods in the Raw_Image and Processed_Image instances in order to perform matching.

This architecture is expected to exhibit flexibility and versatility ; it can be implemented on parallel architectures. Shape and behaviour matching can be performed.

A prototype in Smalltalk has been implemented.

BIBLIOGRAPHY

1. Young R. and King-Sun Fu ,1988, "Handbook of Pattern Recognition and Image Processing", Young and Fu Ed., Academic Press, Orlando, USA.

2. Berman R. and Stonebraker M., 1977, "GeoQuel: a System for the Manipulation and Display of Geographic Data", Proc. Conf. on Very Large Databases, 186

3. Wegner P., 1987, "The Object-Oriented Classification Paradigm", in "Research Directions in Object-Oriented Programming", Shriver and Wegner Ed., MIT Press, Cambridge, USA, 479-560

4. Cox B., 1984, IEEE Software, 1, 1

5. Cox B., 1986, "Object Oriented Programming: an Evolutionary Approach", Addison Wesley, Reading, USA

6. Meyer B., 1988, "Object Oriented Software Construction", Prentice Hall, New York, USA

7. Stroustroup B., 1986, "The C++ Programming Language", Addison Wesley, Reading, USA

8. Andrews T. and Harris C., 1987, "Combining Language and Database Advances in an Object-Oriented Development Environment", Proceedings OOPSLA 87

9. Shaw M., 1984, IEEE Software, 1, 10

10. Mayer B., 1987, IEEE Software, 50

11. Felsinger R., 1988, "Object Oriented Design", SAL, London, England

12. Fairhust M., 1988, "Computer Vision for Robotic Systems", Prentice Hall, New York, USA

13. Nevatia R., 1988, "Handbook of Pattern Recognition and Image Processing" Young and Fu Ed., Academic Press, Orlando, USA.

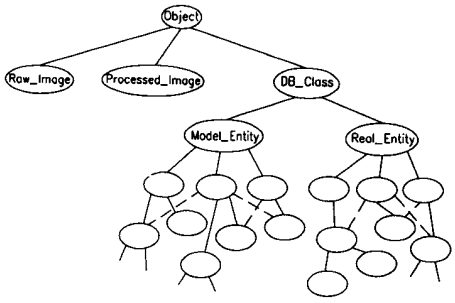

Figure 1. Class Tree for the main system classes.
Dotted lines explicit IS_PART_OF links, otherwise IS_A links.

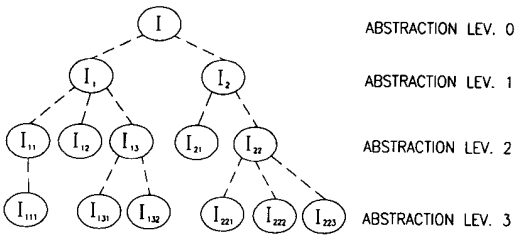

Figure 2. Instances of the Processed_Image class can be hierarchically related in order to specify different resolution levels of processing.

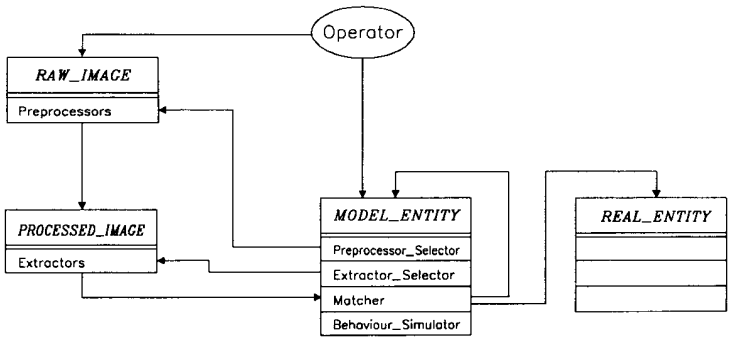

Figure 3. Flow Diagram of the System.