Integrating Object Oriented Programming Paradigm Concepts in Designing a Vision and Pattern Recognition System Architecture

V. Cappellini
Dip. di Elettronica

A. Del Bimbo
Dip. Sist. ed Informatica

P. Nesi
Dip. Sist. ed Informatica

Dipartimento Sistemi e Informatica, Facolta' di Ingegneria, Universita' di Firenze
Via S. Marta 3 - 50139 Firenze - Italia

ABSTRACT

The present contribution proposes the usage of the Object-Oriented Programming paradigm for the architectural design of Image Processing and Object Recognition systems. This approach can be important mainly when large Vision systems are involved, which means when we have to deal with large bases of models, even representing complex objects. In these cases issues such as data representation and integration between the object model provided in the programming languages and the one in the database result central to the system design. These concepts have been implemented in an innovative Image Processing and Pattern Recognition System developed at the University of Florence. The system works on a blackboard structure, suitable to be integrated into an Object-Oriented architecture in order to perform the recognition process.

Introduction

Scene analysis is concerned with the 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 hence captures the significance of the picture (or of the picture's sequence) performing recognition and classification [1].
Most current experiences in automatic recognition are small systems dedicated to specific applications. Features of the object models are stored in a file system that is often supposed to be reduced in size; larger systems suffer, typically owing to performance. However, in order to perform recognition, the need of large databases for storing object models, (eventually representing complex objects), is rapidly growing, thus making object recognition applications more 'data-oriented' than in the past. This results in the growing importance of data structuring mechanisms as well as of investigation of suitable language paradigms to model objects to be managed. In fact as systems become larger, integration of several different system components such as knowledge models, recognition process, programming languages and databases, using a single consistent model, become a central issues for the designer in a similar way than in general software systems. This is a different approach from the typical first generation systems in which no correspondence exists between the flat representation of the models in the file system and the abstractions and abstraction hierarchies interpreted in the application program. Much of the work in AI applied to Vision has involved structuring of human knowledge; different kinds of knowledge structuring and representation techniques have been proposed: first order logic, rule based, semantic networks and frames [2]. Frame systems in particular have been very popular in AI because they give the ability both to arrange information in semantic networks and to associate to any object attribute values as well as procedures in order to make the object able to answer appropriate questions. More recently Object-Oriented Programming (OOP) paradigm has gained large acceptance both in software design and in Database communities. The Object-Oriented Paradigm supports a different version of frames called 'Classes'; classes represent object definitions and are distinguished by 'Objects', representing thus Class instances. Both knowledge representation and software engineering issues are present in the Object-Oriented approach even if its main concepts emerged mostly as programming concepts and were therefore driven by implementation considerations rather than by constructs for modeling the problem domain [3]. OOP supports modeling the real world entities at different abstraction levels thus being of interest to most data-oriented applications that are strongly committed with real world modeling and in addition it aids design, implementation and maintenance of complex systems by supporting modularity, reusability of code and extensibility. OOP paradigm has also set the basis for evolution in database systems [4], [5]. Differently from what actually happens with the conventional record oriented systems, Object-Oriented databases make persistent object structures and semantic information. A single object model can thus be used in the language and in the database. OOP paradigm is a consistent framework for the designer in which the previous concepts about integration can be definitely formalized [6]. Central to the OOP paradigm are the concepts of 'Object', 'Class' and 'Inheritance'. It is widely accepted that these three concepts define the subset of programming languages denominated Object-Oriented languages. No consensus exists on the model supporting such basic concepts in programming languages, (for example there is different support of the object concept as passive, autonomous or slot-based..) and this results in different power of modeling available. Even if Vision and Pattern Recognition are naturally object oriented, however Object-Oriented concepts, representations, languages and databases have not here a wide diffusion. However, some papers recently propose hybrid systems merging some Object-Oriented concepts with a rule based approach [7], [8]. In this paper we describe how a system architecture for an Image Processing and Pattern Recognition system can be based on and supported by the OOP paradigm. As a result this allows the coupling of Object-Oriented languages and Database technologies with Image Processing and Pattern Recognition, which is an important issue when the system has to deal with large sets of high structured information (models) or the system has to serve different applications at the same time. In addition this, it allows implementation on loosely coupled multiprocessor architectures, so that fast recognition can be performed.

Using the Object-Oriented Paradigm in Database Design and Implementation

The proposed system architecture is based on different information bases bearing different kinds of information. The overall database is Object-Oriented and mainly collects instances of images and models that are crucial to the vision task. Object-Oriented database support results in several new facilities that the system can offer. In particular:

a) Capturing semantics allows definition and organization of suitable classes in such a way to model the recognition process and the environment in which the system operates. This will result in a system as modular and flexible as possible, and an architecture easily adaptable to multiple environments: a further step towards that 'general purpose' image recognition system which has still to be designed.

b) Storing data and functions will result in the ability to easily extend the system operation and also to model object motion. In fact, on one hand, adding different operations such as noise reduction or segmentation algorithms to the Raw_Image class (describing the raw image acquired by the TV camera), results in augmenting the power and flexibility of the system without introducing architectural changes; also adding different matching functions to the object model class will result in the ability of the system to adapt to different situations. Some rules or criteria can help in choosing the appropriate functions for the actual environment.

On the other hand when defining object models in addition to features describing form, functions describing the object behavior can be stored. In order to capture the motion model of the object under observation, a suitable adaptive function, synchronous with the frame grabbing process, can be defined into a generic object. A set of possible appropriate coefficients, of a second or upper order equation, describing the object motion law can be associated to the object as a property of the object being.

Recognition process modeling

One of the more formidable problems of an image processing system is the difficulty to process in reasonable amounts of time the enormous quantity of data produced by the external sensors. The observation of human behavior in the same kind of task can be of great help to find a solution: the resulting approach has been called 'smart sensing', [9]. In brief its main characteristics can be summarized as:

a) control of resolution: it is always used at the lowest resolution compatible with the task;

b) windowing: high resolution analysis is limited to selected zones of interest;

c) model-driven processing: the analysis is driven by previous results;

d) coarse-to-fine analysis: first analysis at low resolution is performed, then, if the results are promising, is refined at higher resolution.

It is just this kind of approach which we tried to simulate in our system. It is well known that a reasoning process - e.g. image understanding, text translation, etc. - cannot prescind from general knowledge about the problem domain besides, of course, the one specific for the task. In our system we separated these two different kinds of information, by defining two databases: the first one, closely related to the visual recognition process, and the second which contains environment domain information. Of course, there is a strict correlation between the two: each model in the former corresponds to an object in the latter, which, can thus be used as a kind of 'index' to the first one, which, by its nature has a flat structure with respect to different applicative environments (Fig.1).

We have subdivided the system classes into four categories:

a) Image classes. The low level classes related to the image acquisition and segmentation.

b) Model classes. The classes which model the understanding process. They are collected in the first of the two databases;

c) Environmental classes. The classes which model the external environment, stored in the second database: the one with

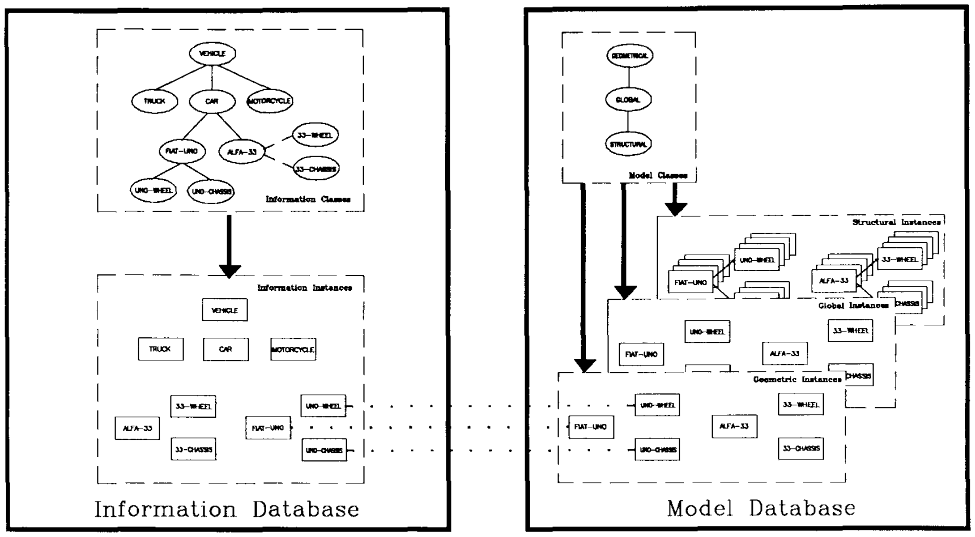

Fig.1-System databases classes and class instances example.

general information, which plays a support role in the recognition process.
d) Control classes. The classes which monitor and control the flow of the process.
The structure of the database of the model classes closely follows the smart sensing approach we described, with its incremental steps of processing and presently is designed to perform recognition through shape analysis. In particular when comparing the acquired object, as resulted from the segmentation step, with object models, some suitable organization of these models can help in shortening the matching process. Using OOP paradigm suitable classes can be defined and arranged in order to recall the usual human process of recognition. We found convenient the subdivision of the process into three levels, subdivision which is mirrored by the class tree structure of the database (see Fig.2). We distinguish between a first level, to be done quickly at low resolution, in which the enclosing geometric figure - e.g. rectangle, ellipse, etc. - is found, and a second, at which global features - like, contours, etc. are analyzed - and a last one, at different resolutions where decomposition into subparts is taken into account. We want here to stress that the consequences of this kind of approach are that our model database contains multiple instances of the same object: one for each analysis resolution, this in particular for the third level. This 'database explosion' is the price we have to pay for the reduction of the processing times: we however believe it is a fair price. The general information database contains an Object-Oriented model of the problem domain. As just pointed out, its information is of help in pruning of the tree of the possible models, (as can be seen in Fig.1).

Motion analysis support

In some situations objects under monitoring do not exhibit a specific or clear shape, useful for recognition, (that is for example the case of night environment processing of different light sources, or also undefined shapes during the day like smoke and clouds etc). In this case the only way of performing recognition is the use of behavior modeling and matching of behavior. Using an Object-Oriented model these objects can thus be modeled in a consistent way as those with some definite form. This will allow extended recognition based on behavior in addition to the usual recognition, based on form. OOP paradigm does not directly support the notion of time. However, encapsulation of attributes and behavior in the class structure offers the means to support time analysis. The undefined dynamic object grabbed by a TV camera can be associated to a 'dynamic_object' class instance bearing a time variable, a clock, initialized at proper time when monitoring starts, and some generic adaptive function describing motion, (Fig.3). A second order model can approximate object behavior in short observation time intervals under the hypotheses that the system is time invariant. Going on to execution, motion law parameters can be estimated and hence compared with those of the reference models. Rule based descriptions can be used for more complex behavior modeling.

Integration of Blackboard Support for Recognition

Control in a system is mainly an implementation issue; different control mechanisms can be used in an Object-Oriented system; they address the way in which an ordered interaction among the entities involved can be obtained. The native way of interaction between objects supported by the OOP paradigm is 'message passing' that acts as a remote procedure call mechanism. An object executes an operation defined in its scope on behalf of someone other that has requested it through

```
class Model_Geo: public Visual_Model
{  // geometric inclusion form
      cl_nam whoami;
      mod_lev mylev;
      char* myname;
      void* specialization_link;
      void* generalization_link;
      void* point_of_view;
      float Centroid_Position_X; // x-pos of centroid (ratio)
      float Centroid_Position_Y; // y-pos of centroid (ratio)
      float Centroid_Position_Offset;
      MER my_mer;                              // mer
      MEE my_mee;                              // mer
   public:
      Model_Geo (char*, mod_lev=GEOMETRIC_CHA);
      virtual char* get_name ();
      void start ();
      void matcher ();
      env_info get_environment_info ();
      mod_hypo* examine_blackboard (info_rqt* request);
      void ask_p_i_info (p_i_rqt* request);
      void formulate_hypos (hypo* hypothesis);
};

class Model_Gbl : public Model_Geo
{   // global characteristics model
      float Area_Perimeter_Ratio;
      float A_P_R_Offset;
      DFT_Contour Fourier_Contour_Transform;
      float F_C_Transform_Offset;
      Hu_Invariants Hu_Cont_Invt;
      float H_I_Offset;
      Histogram Gray_Level_Hist;       //normalized
      float G_L_H_Offset;
      int Histogram_Expansion_Factor;
   public:
      Model_Gbl (char* nam, mod_lev mlev=GLOBAL_CHA) :
                              (nam, mlev) {}
      void matcher ();
};

class Model_Str: public Model_Gbl
{   // structural characteristics model
      Set_of_Part part_ptr;  // list of ptrs to components
      Set_of_Posn posn_ptr;   // list ptrs to pos. descriptors
      Set_of_Resl resl_values;  // list minimum corr. resolu.
   public:
      Model_Str (char* nam, mod_lev mlev=STRUCT_CHA) :
                                       (nam, mlev) {}
      int check_subpart_existence (part_rqt* request);
      void ask_model_activation (sch_rqt* name);
      void matcher ();
};
```

Fig.2-Implementation of the main database model classes.

the defined interface. Objects can however also be implemented as autonomous objects that come into execution through a self triggering mechanism. A Blackboard architecture [10] is well suited to be used as the control part of an Object-Oriented system and can easily be integrated with the native Object-Oriented message passing paradigm. The Blackboard control model relies on a set of independent knowledge sources, (the object model instances in our Vision system), and communication and interactions between the knowledge sources take place only through a common memory (the blackboard itself). The problem solution is obtained

```
class dynamic_object: public object{
private:
   char *id;
   time instant;
   float space[MLEV];  //known position

  // finite difference equations coefficients
   float alfa,beta;
   float heading; //known heading
   float speed;   //known speed
public:
   void initialize();
  //assign time value to instant
   void set_time(time);
  // evaluate the current predicted position
   float predict_pos(float);
  // evaluate the coefficients: alfa, beta
   void model_eval(float);
}
```

Fig.3-Motion analysis support through class definition.

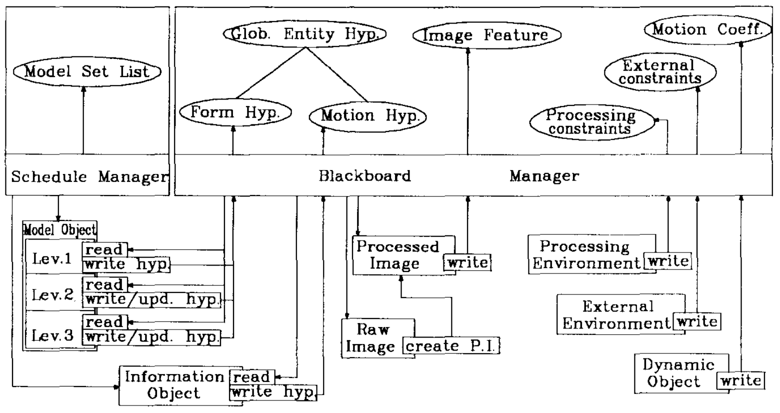

Fig.4-Schematization of data flow in the recognition process

through a cooperation of all the model instances that behave like inference engines, as each one has access to the results of all the others. The activity of a classical Blackboard has in our system been decoupled into a set of different classes, the Information Management classes, which cooperate for the system information management. These classes pilot the recognition process, from the generation of the Processed_Image (the result of the segmentation process on the grabbed raw image), to the final conclusions; they schedule the activation of the model class instances, mediate their requests for the information from the images and store their hypotheses in a common database (the blackboard).
They are:
a) Schedule_Manager. Its task is to control the model class instance activations, trying to optimize hardware and software resources. It calculates, by heuristic methods, the priorities of the entries of its scheduling list, and dynamically updates it at each significant event in the system.
b) Blackboard_Manager. This is the common repository of all the formulated hypotheses, available to each entity in the system. Each model class instance publishes its results by writing them on the Blackboard. Solution is built incrementally; as each object model is represented with several instances at different abstraction levels, higher level instances update hypotheses of lower level instances for the same object model. The Blackboard Manager finally analyzes the hypotheses from the object models about the object form (and behavior, if motion analysis is involved) and selects those with the maximum confidence level. The use of such a common database allows to 'decouple' the knowledge sources, their internal structures can be completely different from each other and optimized for their specific task: only the results must be in a common format. Additional information in the Blackboard are general information relevant to the recognition process. These are environment information, allowed resolutions, maximum allowed processing time, thresholding of the confidence level, etc.. In addition the Blackboard Manager collects all the requests from the model class instances and, if necessary, transmits them to the appropriate objects (as, for example, when some derived attribute has to be extracted from a Processed_Image), (Fig.4).

Conclusions

In this paper we have presented a fully Object-Oriented approach to a general purpose object recognition system. This, together with a blackboard architecture, allows to build a flexible and easily maintainable system. The knowledge information is distributed into two Object-Oriented databases: one, hierarchically structured into three layers according to visual abstractions, contains the object models (the knowledge source of the recognition process); the other, which reflects in the most natural way the problem domain knowledge, helps in the pruning of the decision trees and allows to analyze the behavior of the objects.
The novelty of the system is that it is fully Object-Oriented; this allows in particular to be able to use in Image Processing and Pattern Recognition area the advances of research both in Object-Oriented languages and in Object-Oriented databases bringing these research areas nearer to the Vision world. The choice of a Blackboard model makes the system particularly well suited for an implementation on a loosely coupled parallel architecture, and will hopefully make the system faster. The current version of the system has been implemented in C++ OOP language.

References

[1] Young R., King-Sun Fu, in "Handbook of Pattern Recognition and Image Processing," Young R. and King-Sun Fu, eds., Academic Press, Orlando, USA 1988.
[2] A. R. Rao, R. Jain, "Knowledge Representation and Control in Computer Vision Systems," IEEE Expert, spring 88, pp.64-79, 1988.
[3] B. Cox, "Object Oriented Programming, an Evolutionary Approach," Addison-Wesley, Reading 1987.
[4] T. Andrews, C. Harris, "Combining Language and Database Advances in an Object-Oriented Development Environment," in Proceedings OOPSLA '87, Springer-Verlag, Berlin 1987.
[5] A. Purdy, B. Schuchardt, "Integrating an Object Server with Other Worlds," ACM Trans. on OIS, Vol.5, p.27, 1987.
[6] V. Cappellini, A. Del Bimbo, A. Mecocci, "An Object Oriented Approach for Object Recognition and Classification," in Proceedings ICASSP.89 IEEE Int.Conf.on Acoustic Speech and Signal Processing, Glasgow 1989.
[7] L. Mohan, R. L. Kashyap, "An Object-Oriented Knowledge Representation for Spatial Information," IEEE Transaction Softw. Eng., Vol.14, no.5, pp.675-688, 1988.
[8] D. I. Moldovan, C.-I Wu, "A Hierarchical Knowledge Based System for Airplane Classification," IEEE Trans. Softw. Eng., Vol.14, pp.1829, 1988.
[9] P. J. Burt, "Smart Sensing in Machine Vision," in Machine Vision, H. Freeman, eds., Academic Press, S. Diego, 1988.
[10] R. Engelmore, T. Morgan eds., "Blackboard Systems," Addison-Wesley, Reading 1988.