Blackboard-Based Concurrent Object Recognition Using an Object-Oriented Database

Alberto Del Bimbo Paolo Nesi

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

ABSTRACT The increasing interest in large automatic recognition systems requires investigation on database organization and parallelism support. In this paper, a blackboard-based framework is presented for concurrent object recognition in the presence of a large model database. A new object model is defined with a knowledge organization with two-object hierarchies. Concurrency has been introduced in order to provide a separate execution thread for each scheduled model. Results are presented in terms of performance figures.

1. INTRODUCTION

In recent years, an increasing attention has been given by the research community to the opportunity of using database systems, as well as parallelism support in order to perform efficient object recognition and image understanding [1], [2], [3], [4], [5], [6]. This is motivated by the fact that in many applicative fields automatic recognition systems grow in size while still requiring real-time responses.
In large recognition systems, traditional data models and organizations are inadequate [7], [8], [9]. Difficulties arise from the fact that models should be stored in the form of very complex objects, taking into account different levels of abstraction, different image resolutions, and possible different semantic interpretations according to the context of the analysis [10], [11], [12], [13], [14]. Moreover, inadequacy of sequential processing is recognized. Multiple interpretations may exist for each spatial object which is identified in the image data, and an exponential growth of the number of feature matching operations is possible.
Generally speaking, several requirements can be identified for a large recognition system:

- supporting the ability to define models of real world objects taking into account the context in which they are considered. Depending on the context, models should include different object attributes, relationships with other objects and object behavior;
- modeling different representations for the spatial appearance of any real world object. Several descriptors for the spatial object should be defined which can be used in different operating contexts;
- defining a suitable database organization which facilitates the recognition process. An internal indexing scheme and refinement, possibly through different abstraction levels, should be used to narrow down the search space during object recognition and/or image understanding;
- modeling the uncertainty in the object recognition process;
- learning from the user the domain of object recognition;
- supporting real-time responses. Memory and storage management as well as data organization should support efficient object retrieval in the database, while parallel processing facilities should help in performing the right associations.

The capability of the object-oriented data representation for imaging and recognition applications was accepted recently [1], [9], [7].
In particular, object-oriented database systems [15], [16], [17], [18] permit to extend databases in order to support efficient storage retrieval of complex objects.
Objects are created on the basis of object classes which define the attributes for the object type and a collection of type-specific operations. Different abstraction levels at which objects can be regarded may be defined through an inheritance class hierarchy. When the same object model is used in the programming language, a uniform framework both for the short- and long-term memory is given [19].
A multitasking extension for the object-oriented model could be provided with a separate thread of execution for each object method and for each object. Therefore , at least in principle, both intra- and inter-object concurrences can be supported.
Most of available object models (used in current languages and databases) present critical constraints and inefficiencies which limit their use in practical recognition applications. In fact, concurrency support is provided only to a limited extent [20], [21]. Moreover, inheritance is only defined at the class level and therefore it is not possible to define specialization links between objects. On the other hand, in automatic recognition applications with very large databases, both concurrency and object specialization are key features. In fact, object specialization provides facilities to perform database navigation by progressing along specialization links from a more generic to a more specific view of a certain object (subobjects). However, (sub)objects selected at a certain level could, at least in principle, act concurrently in order to determine the best matching object at recognition step.
Present experiences in the use of object orientation for

imaging and recognition applications are limited to augment the programming language with specific new classes and class hierarchies [22], [23], [24], [2]. However, no concurrency support was introduced or discussed in these systems, neither any modification of the object model was proposed.
In this paper, an advanced object recognition system which is based on the above concepts and complies with the major features depicted are presented.
The distinguishing features of this system are:

- an extension of the object-oriented programming language and of the database with specific classes (data types and operations) in order to support image processing and recognition applications;
- a new object model which is a modification of the currently available object-oriented models and supports object specialization and high performance retrieval;
- a specific data organization according to object hierarchies. The database stores a set of object models which are organized according to two specialization hierarchies, that were called *application* and *object object* hierarchies, respectively;
- concurrency of active models. As each object model can answer its confidence to match a specific spatial object which was extracted from the image data, several models can be active at the same time and synchronize their operations.

The object-oriented approach is enforced in the programming language and in the database system, thus providing a uniform framework for the recognition process. Results achieved are presented in terms of performance measures on the model database and performance prediction for parallel processing.
The paper is organized as follows: in Sect. 2 the general architecture of the system is reported, and in Sect. 3 the data model and database organization are addressed. In Sect. 4 support of parallel processing is introduced. In Sect. 5 the schema of the recognition process is described. Performance measures and performance prediction on multiprocessor architecture are expounded in Sect. 6. Conclusions and future developments are discussed in Sect.7.

2. SYSTEM ARCHITECTURE

The system structure is schematized in Fig. 1. The *Image Processing* block performs low-level vision tasks on a single image or on a sequence.
Feedback from the recognition task is achieved through the *Expert System* module which provides help in choosing the more adequate algorithms to improve the recognition task. A simplified prototypal version of this module is actually implemented in the system.
The *Model Database* is a permanent support for storage of models. These are organized according to specialization hierarchies as further discussed in Sect. 3. As in other image understanding and recognition systems a blackboard ar-

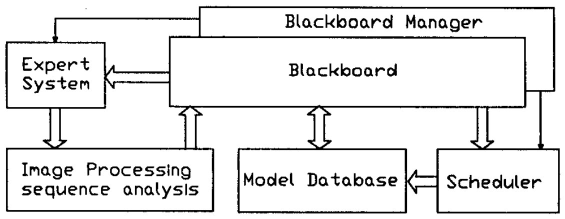

Fig. 1-General system architecture.

chitecture is adopted [10], [25], [26], [27], [6].
The *Blackboard* is a temporary support, which can be accessed by every model, and contains all the features which result from the image processing and the confidences declared by the models when matching the observed data.
The *Blackboard Manager* provides decision support by manipulating the information contained in the Blackboard in order to schedule the appropriate models from the database, through the *Scheduler.*
The *Scheduler* receives the commands from the Blackboard Manager and schedules the models in the Model Database based on the data in the Blackboard.
The system is able to exploit parallelism through concurrent scheduling of models from the database. Scheduled models perform concurrently feature matching with temporary data, in the Blackboard, obtained from image processing.
Blackboard Manager, Scheduler and Expert System are implemented as concurrent instances. Model Database, Blackboard Manager and Scheduler are discussed in detail in the following sections. The programming language used to build the system is an object- oriented extension of an already existing language (C++) [28], [29]. The extensions are added in order to provide the concurrence and the object specialization [30], [21].

3. MODEL DATABASE

In this section, model database organization and structure will be discussed with reference to the class hierarchies which were defined for knowledge representation, as well as to the specialization relationships between objects that were introduced and their implementation.

3.1. Class hierarchies

Classes are used in the system for providing a clear description of object structures and supporting code modularization and reuse.
Two class hierarchies were defined knowledge structuring. A first class hierarchy, *Object Class Hierarchy* (OCH) models the knowledge of each object (see Fig. 2). The knowledge is broken down into specialized chunks regarding both the object visual appearance and the information stemming out of sources other than the sensor, like the object behavior and the possible relationships with other

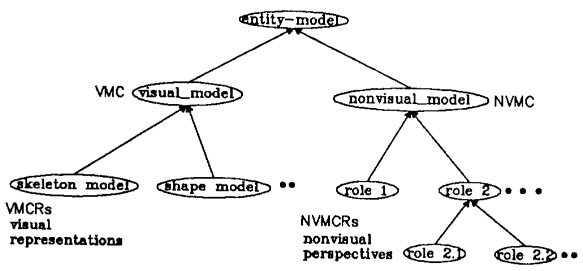

Fig.2-Object Class Hierarchy (OCH).

entities. These are modeled through visual model (VMC) and non-visual model (NVMC) subclasses, respectively. Several subclasses of VMC (VMCRs) may be defined corresponding to the different representation schemes for the visual appearance of the object. Examples are skeleton, shape, surface, volume-based representations, etc...
Analogously, several subclasses (NVMCRs) of NVMC may be also defined which correspond to the different roles that each object can play in the different contexts in which it is regarded. Generally speaking, each role corresponds to specific relationships with other entities.
Class methods implement fuzzy matching operations for the class attributes which are used in the recognition process. When activated, the method inputs the value of the corresponding feature of the spatial object to be recognized and evaluates the matching confidence of the model for the attribute. In the case of composite objects, the composite matching confidences are evaluated.
A second class hierarchy referred to as *Application Class Hierarchy* (ACH), represents the observer's view of this domain. Each class in this hierarchy defines an object category, limited to a certain level of generality, as it can be viewed by the user. In Fig. 3, this is shown referring to a specific application domain of vehicle classification. Leaf classes are descriptions of objects that have a one-to-one correspondence with real world objects. Each class in the ACH is structured according to the OCH discussed above.

Generally speaking, only leaf classes provide descriptions in terms of both visual and non-visual representations, as they actually define models of concrete objects. Intermediate classes can also have visual descriptors, even if more generic. This corresponds to situations in which we know that a category of objects is characterized by some features with a specific visual appearance (e.g., a car has four wheels). Classes which define more abstract object categories, with no counterpart in the real world, have no visual descriptors.

3.2. Object specialization: object model and object hierarchies

Objects are instances of the classes/subclasses which were defined for the knowledge representation. Specialization

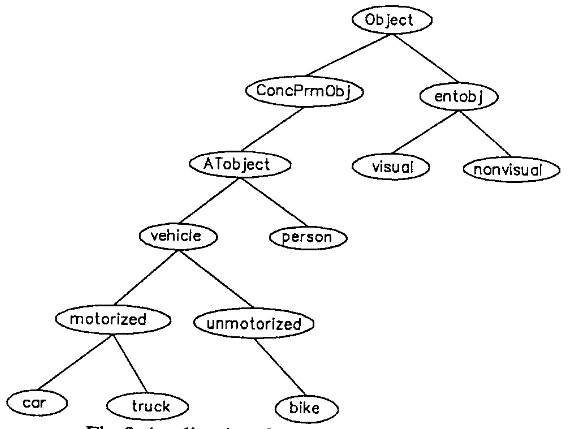

Fig.3-Application Class Hierachy (ACH).

links were maintained also between objects.
As previously introduced, this provides the ability to perform database navigation by directly using specialization links between objects, thus progressing from generic to specific views of the objects in the database.
In order to support object specialization [21] a new object model was introduced into our system. In according to this object model, if two objects o_1 and o_2 are instances of *classes* c_1 and c_2 with c_1 being the superclass of c_2, then o_1 is made the superobject of o_2. Therefore, in the system, two distinct object hierarchies exist for the object structure and the object user's view, respectively. For symmetry purposes, the structure of each object will be referred in the database as *Object Object Hierarchy* (OOH). Every object is regarded as a composite object made up of several specialized subobjects. A one-to-one relationship exist between the subclasses in the OCH and the subobjects in the OOH. For each object model several visual/non-visual submodels exist (OVMRs/ONVMRs): each of them acts as a separate conceptual focus in the recognition process at the specific abstraction level. In particular, non-visual submodels allow to focus on the applicative context and implement specific perspectives of the object, according to the fact that each object can play only selected roles into a specific environment. As an example, looking at a *car* object, different motion laws and proximity relationships with other objects can be assumed in the following roles: *at_paytoll_station, on-highway, on-road....*
Separate visual submodels exist depending on the resolution which has been selected, as well as on the selected point of view. The storage of different resolutions and visual perspectives could overwhelm the system, in terms of large storage requirements and hard maintenance. Following the object-oriented paradigm the user of the system has two options: explicit definition, and implicit definition through some appropriate procedure. A trade-off exists between storage occupancy and expected performance.
Object models in the database are hierarchically organized

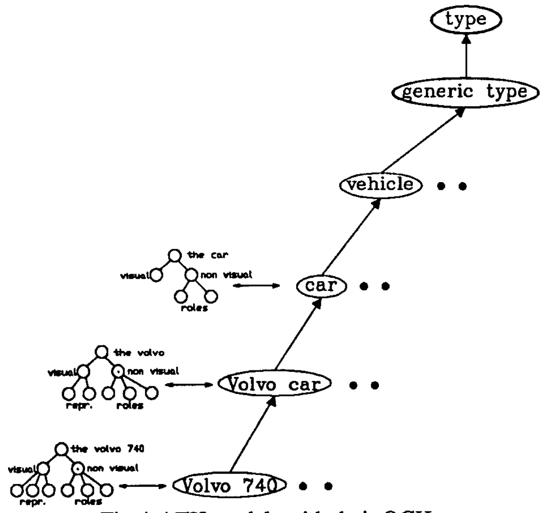

Fig.4-ATH models with their OCHs.

along a hierarchy which describes the different abstraction levels at which the object can be regarded, and it is referred as *Application Object Hierarchy* (AOH). Models in the AOH may have different structures (OOHs), depending on the level of the AOH at which the object is located. An example of AOH and OOH is reported in Fig. 4 with reference to a vehicle recognition application.
The introduction of object hierarchies provides a performance increase in the retrieval of complex composite objects into the database. The performance comparison between the implementation of object searching with respect to the usual approach of object-oriented retrieval mechanism (search through *obj_id*) is shown in Fig. 5 where the mechanism used is referred as T-Ref). In this case AOH is a four-level hierarchy as in Fig. 3. Please note that a first access to the hierarchy has to be always performed by key.

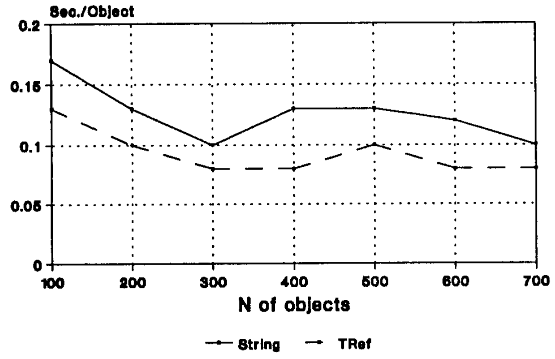

Fig. 5 - Comparison between performance on retrieving the ACH-OCH using string and T-Ref.

```
Class ConcTmpObj
{
private:
Protected:
 Int state;
 TaskID tid;    // this is the thread identifier which is set
                // at runtime. 0 value means inactive task.
 Semaphore mysem;//this is the semaphore associated
                       //to the object task
 Int kill;       //when 1 the task is killed
 void TaskEx(); //when called this concludes the task.
              //It is called only by the TTask()procedure.
Public:
 ConcTmpObj(); // This is the Constructor
 ~ConcTmpObj(); // This is the Destructor
 int RUN(void _FAR*); //procedure to start methods
 void FAR TTask();  // prototype of concurrent method
 void _FAR ICTTASK();//proto. of command interpreter
 void Suspend();        //to suspend task execution
 void Resume();         //to restart a suspended task
 void SetPriority(PCLS,PVAL); //to set task priority
                    //PCLS refers to class priority (0:4)
                    //PVAL refers to priority inside
                    //the class (-31:+31)
 void GetPriority(PCLS*,PVAL*);//to read task priority
 void SemSetWait(); //to set the semaphore and wait
 void SemWaitSet();//wait for semaphore free and set it
 void SemClear();    //to reset the semaphore
 void SemWait();     //to wait for semaphore free
 TaskID getTaskID(); //to recover the task number
}
```

Fig.6-ConcTmpObj class definition.

4. PARALLEL SUPPORT

Concurrency is one of the major distinguishing features which were introduced in the system in order to provide higher performance in the recognition task. Concurrency was introduced both for short- and long-term objects (models which are in the database and are scheduled on demand) [31], [32], [33].
Temporary objects concurrency . The programming language supports multi-threads with the addition of the class "ConcTmpObj" (see Fig. 6). This class directly interacts with operating system facilities and comprises the definition of a status variable, a task identifier, control variables and a RAM semaphore to provide process synchronization. It includes all the operations which are usually available in a multitasking environment such as task suspension, resume, priority change, semaphore set, free and wait....
Classes are defined as subclasses of the "ConcTmpObj" class (see Fig. 7). The instantiation of a concurrent object is therefore similar to that of static objects.
Permanent objects concurrency. Concurrency has been also introduced for the permanent classes of the system. The "ConcPrmObj" class is similar to the "ConcTmpObj" and was defined owing to the limitations of the running database version which does not support multiple inheritance (see Fig.3). According to this, the ability is provided to concurrently schedule distinct objects, multiple methods for dif-

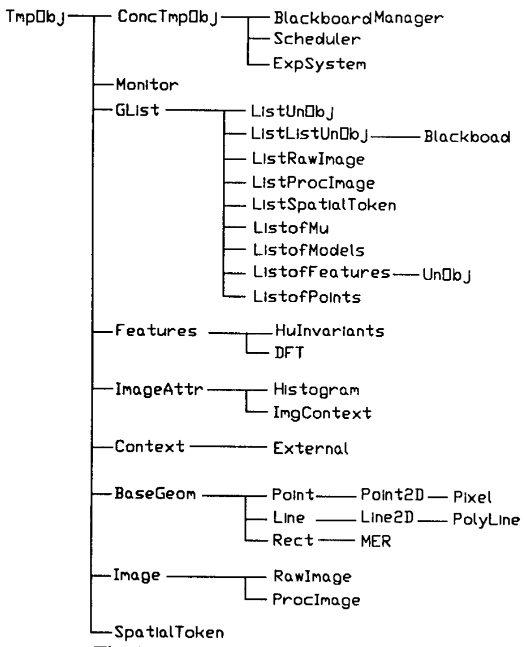

Fig.7-Temporary class tree (a part).

ferent instance of the same class and multiple instances of the same method for the same object. In the first case, every class needs a message interpreter which is specific of the class. In the second case, a semaphore is needed on the instance variables as methods can operate on the object variables concurrently.

In Fig. 8, an example of definition of the command manager for a concurrent object is reported. In addition, whatever method can be scheduled as an independent concurrent task by using the method RUN() of concurrent superclass. Any object models can act concurrently for the recognition process. As each model embeds its distinct matching procedure, several copies of this can be concurrently activated for matching with different unidentified object on the Blackboard.

5. RECOGNITION PROCESS

As the recognition process is affected by uncertainty, the following binary fuzzy relationships were introduced [34], [35], where q stands for the target object (any *SpatialToken* in the image frame), o is the generic object model and d a generic descriptor for the model:

$$F = \{ \mu_i(o,d)/(o,d) \mid o \in O ; d \in D \},$$

with a membership function $\mu_i : O \times D \rightarrow [0,1]$. This describes the degree to which the descriptor value represents the object. A fuzzy transitive relationship is also defined between couples of descriptors giving the measure of the strength of the semantic link l between them:

```
void <class name>::_FAR ICTTask();
{
while(1)
 {
 state=0;
 SemSetWait(); /* wait for new command */
 if (kill==1) TaskEx(0); /* verify the kill condition */
 switch (command)
  {case --:
     //methods like commands

     break;
   case --:
     . . .
     . . .
   default:
     break;
   }
 }
TaskEx(0); /* close task command */
}
```

Fig. 8 - Prototype of concurrent method for a command manager.

$$S_l = \{ \mu_l(d_i,d_j)/(d_i,d_j) \mid d_i,d_j \in D ; l \in L \}.$$

In our system object recognition is regarded as a binary fuzzy retrieval relation. Given a certain $q \in Q$, we get the set R_q which is a fuzzy subset in O :

$$R_q = \{ \mu_{R_q}(o)/o \mid o \in O;\ \mu_{R_q}(o) = \mu_R(q,o) \}.$$

As a consequence, the recognition task can be regarded as a reordering of the collection with respect to the values of μ_{R_q}. Recognition is performed in several steps. Based on the visual context the Expert System starts the image processing task. Data resulting from the segmentation step, (i.e., spatial tokens of the whole image) are referenced in the Blackboard with their identifier *obj_id*. Processing over the Spatial Tokens is carried out in order to extract selected numeric/symbolic features.

As soon as features are evaluated, the Blackboard Manager asks the Scheduler to activate models in order to compare properties extracted for each spatial token (the unidentified object) with those of the models in the database. The Scheduler, uses non-visual knowledge to prune in advance the number of models to be taken into consideration. In particular, indexing facilities are provided which allow the selection of submodels (OVMRs, ONVMRs) on the basis of the application context. For example, if the applicative environment is related to highway monitoring, bike objects will be regarded as absolutely improbable; moreover, specific representations will be selected depending on the daytime. The features of every unidentified object are compared with those of the selected models in the database. In our system, every model can autonomously give an answer, having a separate thread of execution. Therefore, multiple concurrent object models exist in the system during the recognition step. Complex objects activate component objects, and wait for their answers before outputting the global confidence. The confidences for composite object C, with A global attributes a and S subparts s (in total $A+S$ attributes), has been defined according to a generic *fuzzy weighted averaging function:*

$$(1)\quad \mu_O(C) = \left(\frac{1}{1+\sum_{s=1}^{S} w_s}\right)\left(\frac{\sum_{a=1}^{A} w_a\, T(fr_a, fg_a)}{\sum_{a=1}^{A} w_a} + \sum_{s=1}^{S} w_s\, \mu_O(s)\right)$$

$$\text{with:}\quad T(fra, fga) = \begin{cases} 1 & \text{if } |fr_a - fg_a| \le offset \\ \dfrac{offset}{|fr_a - fg_a|} & \text{if } |fr_a - fg_a| > offset \end{cases}$$

and: fr_a = reference attribute values for the a-th attribute; fg_a = attribute values measured from the grabbed image for the a-th attribute; w_a = weighting factors for the a-th attribute; w_j = weighting factors for the s-th subpart, where *offset*, w_a, w_j depend on the object model construction.

This function is used to evaluate the confidence for the object defined in the OOH as comprised of OVMRs and ONVNRs parts. The number of models to be selected and of the features to be compared may be very large. Object-oriented knowledge organization and heuristics are used to shorten the recognition process. Models at the highest levels in the object hierarchy are activated first and requested to answer their confidences of matching the set of features which have been initially selected. At the same time, an expansion activity takes place, identifying the most plausible features which could be further compared. This is made by using the fuzzy relationships between the model descriptors. If a selected descriptor d_i has a confidence $\mu_{l(d_i,d_j)}$ about the link with the descriptor d_j, then it is possible to assign an a-priori confidence to the descriptor d_j which is obtained as $\mu_{d_i}\, \mu_{l(d_i,d_j)}$. According to this, the features with the highest a-priori confidence values are marked as possible features to be further used for comparison, when those selected will not allow to discriminate between models.

In the Blackboard, a set of tuples [*status, model name*, μ, *p_features, b_features*] is instantiated for each target object (*obj_id*), one tuple per each model which was scheduled for matching.

The field *status* describes the status of the model in the matching process, and the possible distinct values for this field are: execution, wait, stopped. The μ holds the matching confidence that the model has with respect to the target. *b_features* is a list of features which are used to evaluate the matching confidence μ. *p_features* is the list of features which can be possibly used for matching for the model indicate by *model name*.

The Blackboard Manager is in charge of narrowing down the search in the AOH, depending on the computed confidence values. This is made by selecting the models with a confidence value higher than a certain threshold or the highest available. For each selected model *m* the Blackboard Manager asks the Scheduler to retrieve its submodels *n*. These are in fact the new models that will be scheduled as concurrent matching models at the next step. In addition, each of these submodels has to schedule its subparts as concurrent child processes in order to provide a complete fuzzy confidence evaluation according to the eq. (1). If a more specialized model outputs a confidence value which is lower than that of the more general model, this is rescheduled and possibly a new comparison is performed using the additional selected features. Principles of operations of the Blackboard Manager are reported in the pseudocode shown in Fig.9.

```
Blackboard_Manager: DO
 VAR b_features, p_features, req_features :List_of_Features;
 VAR Num_obj,i :Integer;
 VAR selected_models :List_of_Models;
 VAR obj_id :ref_to_unidentified_object;
Init_Blackboard_Manager();
 LOOP
  Num_obj := How_many_obj();
  DO i=1 TO Num_obj
   obj_id:= Get_obj_id(i);
   DOCASE State_of(obj_id) /*verify the status of all tuples
                           for this obj_id */
    ENABLED: DO /*μ is comp. for every tuple for obj_id*/
     /*create the list of selected models*/
     selected_models := Select_Model(obj_id);
     /*request features for matching*/
     req_features := Req_Required_Features(obj_id);
     /*activate Scheduler on selected models for matching
       with the requested features*/
     Scheduler.OkFor(obj_id,selected_models,req_features);
    END DO;
    NOALLOWED: DO /*condition for models selection
                   cannot be sotisfied*/
     p_features := Req_Possible_Features(obj_id);
     b_features := Req_Based_Features(obj_id);
     /*send a request for additional features extraction
       to the expert system*/
     Expert_System.ReqElb(obj_id,b_features,p_features);
    END DO;
   END CASE;
  END DO;
 END LOOP;
END Blackboard_Manager;
```

Fig.9-Pseudocode of Blackboard Manager algorithm.

6. PERFORMANCE ANALYSIS OF THE RECOGNITION SYSTEM

The number of objects (models, M), which can be concurrently scheduled at a certain level of the models object hierarchy depends on the specifics of the application, on the number of unidentified objects in the Blackboard and on the kind of algorithm to select the models from the matching confidences. In general, this number can be estimated by:

$$(2)\, M = \sum_{u=1}^{U} \sum_{m=1}^{J(\varepsilon,u)} \left[N(m) + \sum_{n=1}^{N(m)} S(n)\,(l-1) \right]$$

where:

- U := number of unidentified objects in the Blackboard;

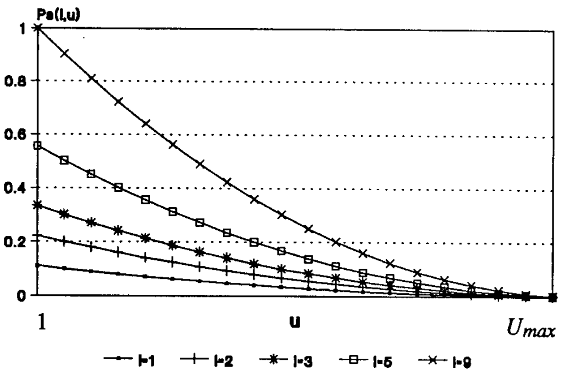

Fig. 10 - Function $P_s(l,u)$ of the hypothesis ii).

- $J(\varepsilon,u)$:= number of models with a confidence value higher than ε in matching the features of the unidentified object u;
- $N(m)$:= number of submodels derived from the m model;
- $S(n)$:= number of subparts of the n-th submodel.

Considering a uniform distribution of models, submodels and submodel subparts it is possible to transform the previous expression in:

(3) $M(u,m,n,s) = u\,m\,(\,n + n\,s\,(u{-}1)\,)$,

where u, m, n, s are local variables.

Please note that, according to the hypotheses above, several instances of the same (sub)model could be concurrently active for matching different target objects in the Blackboard. However, it should be noted that every model could be activated for matching or directly by the Scheduler or as a subpart of another model. In order to take into account this fact, several assumptions can be made.

i) Rescheduling of the same model as a subpart of another model is more probable when the number of the target objects in the Blackboard is larger. According to this, we assume the following expression for the probability P_s of scheduling a subpart of a generic model as a function of the number of unidentified objects u :

(4) $$P_s(u) = \frac{(\,u - U_{\max} - 1)^2}{U_{\max}^2},$$

where $U_{\max}$ is the maximum number of unidentified objects in the Blackboard.

ii) The probability of rescheduling a subpart model is greater when lower levels of the OAH are considered. In fact, in this case, levels usually have many models with a greater number of submodels and subparts. This is generally true as lower levels in the OAH correspond to real objects. This implies that expression (4) can be modified as:

(5) $$P_s(l,u) = \frac{l\,(\,u - U_{\max} - 1)^2}{U_{\max}^2\left(L_{\max} - 1\right)},$$

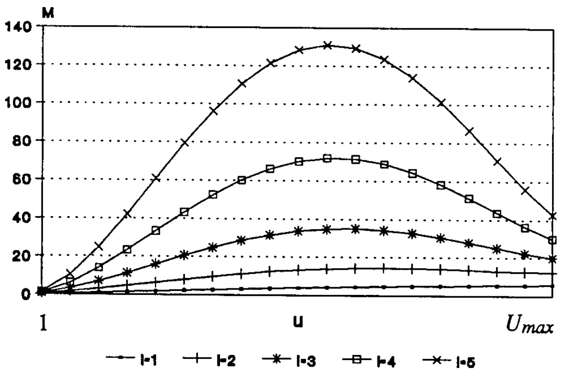

Fig. 11 - Number of models with respect to the number of unidentified objects on the Blackboard u for some levels l in the hierarchy.

where: l is the generic level number and $L_{\max}$ is the maximum number of levels in the OAH. In Fig. 10 the typical behaviors of $P_s(l,u)$ are shown.

In addition, following expressions hold for the submodels distribution and the subparts distribution, respectively:

(6) $$R_s(\,l\,) = l\,\frac{S_{\max}}{L_{\max}},$$

(7) $$R_n(\,l\,) = l\,\frac{N_{\max}}{L_{\max}},$$

being $N_{\max}$ and $S_{\max}$ the maximum number of submodels and subparts in the lowest level of the hierarchy, respectively.

Substituting in (3), the expression for the number of models which are scheduled at a certain level l of the OAH is:

(8) $$M(\,u,m,l) = u\,m\left(R_n(l) + R_n(l)\,R_s(l)\,(u-1)\,P_s(\,l,u)\right).$$

In Fig. 11 figures are presented for this expression in the case in which only the model with the highest confidence level ($m = 1$) is selected at any level of the OAH.

In Fig. 12 an example of a vehicle classification application (see the OAH in Fig. 3) is reported where: $L_{\max} = 6$, $U_{\max} = 5$, $S_{\max} = 4$, $N_{\max} = 2$.

6.1. Performance prediction on parallel architecture

Using the equations of the previous section, the number of concurrent processes (active models and submodels) as a function of the number of unidentified objects in the Blackboard for each level of the hierarchy can be derived. Results provide hints for the evaluation of the speed-up achievable when porting the recognition task on a multiprocessor architecture.

In the following, the allocation of one processor for each tasks: Blackboard Manager, Scheduler, Expert System and of P processors for the M object models selected in the recognition task will be assumed. Using the Amdhal equation [36]. The time of execution of a generic algorithm in a

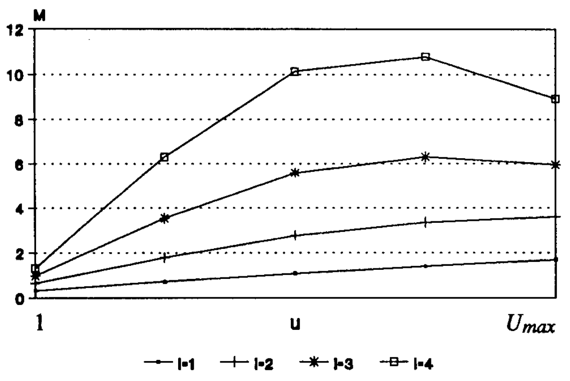

Fig. 12 - Number of models with respect to the number of unidentified objects on the Blackboard u for some levels l in the hierarchy. Example based on the vehicle application hierarchy.

parallel architecture can be expressed by:

(9) $$T_P = \alpha\, T_S + \frac{(1-\alpha)T_S}{P},$$

where the first term take into account the portion of the recognition algorithm that cannot be parallelized, the second term about the portion which can be executed concurrently. In our case, the proportionality constant α is considered a function of the number of models according to:

(10) $$\alpha(M) = \frac{(1-\beta)}{M} + \beta M .$$

In fact, the portion of code for models management which must be sequentially executed is linearly dependent on the number of models, while the portion of the algorithm which cannot be executed in parallel inversely depends on the number of models (first and second term of the eq.(10), respectively). According to this, the execution time of a the matching algorithm on the multiprocessor system is given by:

(11) $$T_P(M,P) = \left(\frac{(1-\beta)}{M} + \beta M\right)T_S + \frac{1-\left(\frac{(1-\beta)}{M} + \beta M\right)T_S}{P}.$$

Under the hypothesis that in the presence of a single model there is no part of the code which can be executed in parallel and β is 99.9 per cent of the sequential portion and grows linearly, curves of Fig. 13 can be derived for the speed-up as a function of the number of active models.

7. CONCLUSIONS

In this paper, a system was presented which performs recognition by comparing graphical entities with models that are stored into an object-oriented database. The concurrence and a proper database structure were chosen in order to support fast retrieval and processing. The solution

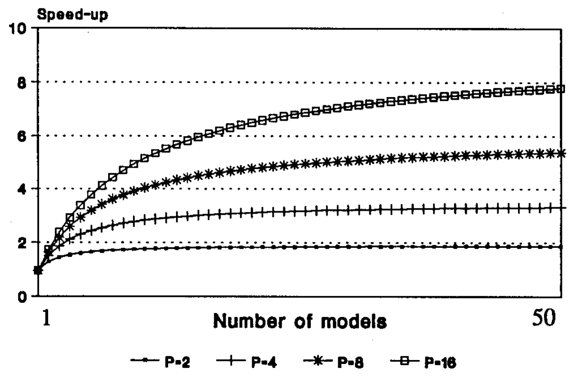

Fig. 13 - Speed-up with respect to the number of the active models for some constant number of processors.

presented herein proposes a new object model which emphasizes object specialization as well as a knowledge organization based on two-object hierarchies with support of changes of conceptual focus. Parallel processing is enforced by introducing concurrence in the object model, so that each model acts as an independent thread of processing and is executed on a separate processor.
Results achieved so far show that a high performance increase can be obtained by using the proposed model with respect to the common object-oriented model. Moreover, efficient indexing and search narrowing is obtained through the hierarchical organization of the model objects. Curves of the performance achievable with a multiprocessor architecture are presented under simplified assumptions.

REFERENCES

[1] A. M. Goodman, R. M. Haralick, L. G. Shapiro, "Knowledge-Based Computer Vision - Integrated Programming Language and Data Management System Design," *IEEE Computer*, Vol. 22, N. 12, pp. 43-54, 1989.

[2] H. V. Jagadish, L. O'Gorman, "An Object Model for Image Recognition," *IEEE Computer*, Vol. 22, N. 12, pp. 33-41, Dec., 1989.

[3] R. M. Bolle, and al. "Visual Recognition using concurrent and layered parameter networks," in: *Proc. of Computer Vision and Pattern Recognition*, pp. 625-631, 1989.

[4] R. M. Bolle, A. Califano, "A Framework for 3D Recognition," in: *Machine Vision for Three-Dimensional Scenes*, (H. Freeman, ed.), Academic Press Inc., pp. 1-24, 1990.

[5] J. K. Tsotsos, "Representation Axis and Temporal Cooperative Processes," in: *Vision Brain and Cooperative Computation*, (M. A. Arbib and A. K. Hanson, ed.), MIT Press, Cambridge, MS, USA, pp. 361-417, 1987.

[6] L. D. Erman, and al. "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty," in: *Blackboard Systems*, (R. Engelmore and

T. Morgan, ed.), Addison Wesley Publishing Company, England, pp. 31-86, 1988.

[7] V. Cappellini, A. Del Bimbo, P. Nesi, "Integrating Object-Oriented Programming Paradigm Concepts in Designing a Vision and Pattern Recognition System Architecture," in: *Proc. IEEE 10th ICPR International Conference on Pattern Recognition, Atlantic City New Jersey*, June 17-21, 1990.

[8] A. R. Rao, R. Jain, "Knowledge Representation and Control in Computer Vision Systems," *IEEE Expert*, pp. 64-79, spring 88, 1988.

[9] L. Mohan, R. L. Kashyap, "An Object-oriented Knowledge Representation for Spatial Information," *IEEE Transactions on Software Engineering*, Vol. 14, N. 5, pp. 675-688, 1988.

[10]A. R. Hanson, E. M. Riseman, "VISIONS A Computer System for Interpreting Scenes," in: *Computer Vision Systems*, (A. R. Hanson and E. M. Riseman, eds.), Academic Press, New York, pp. 303-333, 1978.

[11]J. K. Tsotsos, "Knowledge organization and its role in representation and interpretation for time-varying data: the ALVEN system," *ComputationalIntelligence*, Vol. 1, N. 1, pp. 16-32, 1985.

[12]R. A. Brooks, "Symbolic Reasoning about 3-D Model and 2-D Images," *Artificial Intelligence*, Vol. 17, pp. 285-348, 1981.

[13]W. S. Havens, "Representation Knowledge of the Visual World," *IEEE Computer*, Vol. 16, N. 10, pp. 90-98, 1983.

[14]R. Cecchini, A. Del Bimbo, P. Nesi, "IPOOS: an advanced system for automatic classification," in: *Proc. of 3rd International Workshop on Time-Varying Image Processing and Moving Object Recognition Florence Italy*, May 24-31, 1989.

[15]D. H. Fishman, and al. "Iris: An Object-Oriented Database Management System," *ACM Transactions on Office Information Systems*, Vol. 5, N. 1, pp. 48-69, January, 1987.

[16]ONTOS, "Object-Oriented Database Documentation," Rel. 1.42, Operativ System: OS/2, Ontologic, Inc., Ma, USA, 1989.

[17]J. Banerjee, and al. "Data Model Issues for Object Oriented Applications," *ACM Transactions on Office Information System*, Vol. 5, N. 1, pp. 3-26, Jan, 1987.

[18]T. Andrews, C. Harris, "Combining Language and Database Advances in an Object Oriented Development Environment," in: *Proceedings OOPSLA'87, Springer-Verlag, Berlin*, 1987.

[19]J. Eliot, B. Moss, "Object Orientation as Catalyst for Language-Databases," in: *Object-Oriented Concepts Databases and Applications*, (W. Kim and F. Lochowsky, eds.), ACM Press New York, pp. 583-589, 1989.

[20]P. Wegner, "Classification in Object-Oriented Systems," *ACM SIGPLAN Notices*, Vol. 21, N. 10, pp. 173-182, Oct., 1986.

[21]E. Sciore, "Object Specialization," *ACM Transactions on Office Information Systems*, Vol. 7, N. 2, pp. 103-122, April, 1989.

[22]R. Taniguchi, M. Amamiya, E. Kawaguchi, "Knowledge based Image Processing System: IPSSENS-II," in: *Proc. IEE 3rd International Conference on Image Processing and its Applications*, Warwick, UK, pp. 462-466, July, 1989.

[23]M. Flickner, M. Lavin, S. Das, "An Object-oriented Language for Image and Vision Execution (OLIVE)," in: *Proc. of the IEEE 10th Int. Conference on Pattern Recognition Atlantic City NJ*, Vol. II, pp. 561-571, June, 1990.

[24]R. Cecchini, A. Del Bimbo, P. Nesi, "Object classification and Image interpretation with an Object-Oriented system," in: *Proc. 3rd IEE nternational Conference on Image Processing and its Applications,United Kingdom*, July 18-20, 1989.

[25]M. Nagao, T. Matsuyama, "A Structural Analysis of Complex Aerial Photographs," Plenum Press, New York, 1980.

[26]V. Hwang, L. S. Davis and T. Matsuyama, "Hypothesis Integration in Image Understanding Systems," *Computer Vision, Graphics, and Image Processing*, Vol. 36, pp. 321-371, 1986.

[27]B. A. Draper et al., "Issues in the Development of a Blackboard-Based Schema for Image Understanding," in: *Blackboard Systems*, (R. Engelmore and T. Morgan, ed.), Addison-Wesley, 1988.

[28]B. Stroustrup, "The C++ Programming Language," Addison Wesley, Mass. USA, 1986.

[29]M. J. Stefik, D. G. Bobrow, "Object-Oriented Programming: Themes and Variations," *The AI Magazine*, Vol. 6, N. 4, pp. 40-62, 1986.

[30]A. Del Bimbo, P. Nesi, "Riconoscimento concorrente di oggetti su base di dati Object-Oriented," Dipartimento di Sistemi e Informatica, Facolta' di Ingegneria, Universita' di Firenze, RT 6/91, Italy, 1991.

[31]S. E. Hudson, R. King, "Cactis: A Self-Adaptive Concurrent Implementation of an Object-Oriented Database Management System," *ACM Transactions on database Systems*, Vol. 14, N. 3, pp. 291-321, September, 1989.

[32]A. Yonezawa, M. Tokoro, "Object-Oriented Concurrent Programming," The MIT Press, Cambridge Massachusetts, 1987.

[33]C. Tomlinson, M. Scheevel, "Concurrent Object-Oriented Programming Languages," in: *Object-Oriented Concepts Databases and Applications*, (W. Kim and F. Lochovsky, ed.), ACM Press and Addison-Wesley, pp. 79-124, 1989.

[34]G. J. Klir, T. A. Folger, "Fuzzy Sets Uncertainty and Information," Prentice-Hall International Inc., 1988.

[35]A. Del Bimbo, D. Lucarella, "A Fuzzy Object Retrieval System for Image Understanding," in: *Proc. of the IEEE International Phoenix Conference on Computers and Communications (IPCCC'91) Phoenix USA*, 1991.

[36]A. L. Decegama, "The Tecnology of Parallel Processing," Prentice Hall International Editions, 1989.