Recognition and Learning of Unknown Objects in a Hierarchical Knowledge-Base

Lipika Dey* P.P. Das† Santanu Chaudhury ‡

Abstract

We present here a knowledge updation mechanism based on a semantic-net like knowledge organization scheme. By considering learning as an extension to recognition of unknown objects, pre-updation analysis ensures consistency in the resulting knowledge base.

1 Introduction

Object recognition is a complex perceptual task. It involves synthesis of multiple object attributes describing various aspects of an object, like its shape description or functional qualities etc. Also the ability of the vision system should extend beyond identification of only the known objects. We argue that an instance of an unknown object should be, if possible, at least categorized to its generic class and its similarity with the known objects should be established and if it is not non-sensical with respect to its domain of discourse, the new object should be incorporated into the knowledge-base. Hence, it is essential that the capability of learning is intertwined with the process of recognition.

In this work we present the process of unknown object recognition and learning as an integrated phenomenon. Our work is motivated by the idea of conceptual clustering and we have considered this particularly for an object recognition system where attributes of objects are of varied kinds - geometric, functional, physical property linked and spatial. The approach is in the context of a new semantic-net based knowledge representation scheme in which objects and classes are encoded as concepts built up of simpler concepts (HAS_PARTS link) and possessing sets of features and relations of any kind as mentioned above. We have been able to integrate the various types of links successfully into a unified mechanism of reasoning for recognition.

The learning process is aided by background knowledge pertaining to domain and control knowledge designed for the KR scheme which is domain independent. The domain knowledge is used in the recognition of the input object i.e. to find out its generic class and the closest match. The control knowledge is first used to decide whether the object is known or unknown. If unknown, then it aids in the process of incorporation of the new object into the knowledge base. The KB is updated ensuring its consistency. The updated knowledge base is optimized to achieve recognition efficiency with minimum storage requirement. Finally, saliency of attributes are extracted to help in future recognition processes.

Unlike previous generic object recognizers like [7] and [5], we have considered learning as an integral part of recognition. The system also differs from other learning mechanisms like [6], [1] and [4] since the position of the new object is preceded by a structural analysis and not a statistical analysis of the values of the attributes (see [2] and [3]).

2 Knowledge-base Organization

Objects are hierarchically organized in the knowledge-base(KB). The generic properties of a group of similar objects are associated to their common parent. Individual specialities are attached to the specific concepts. Part decomposition of objects is also encoded into the structure. The description of an object comprises of local or global features which may be geometric, physical or conceptual in nature. This also includes spatial relations constraining parts or subparts of an object thereby providing a shape description.

The Knowledge-base is a network like structure represented by $\mathcal{G} = \langle \mathcal{K}, \mathcal{L} \rangle$, where $\mathcal{K}$ is a set of nodes and $\mathcal{L}$ is a set of attributed links. Elements of $\mathcal{K}$ consist of

Geometric primitives $\mathcal{G}$- these are the elemental entities involved in any shape recognition scheme. They cannot be further subdivided and combine with each other to form complex concepts.

Part nodes $\mathcal{P}$- these nodes essentially denote typical parts or subparts of objects in a domain. Part nodes may be linked among themselves in an hierarchical fashion and also combine with each other to form composite objects.

Object nodes $\mathcal{O}$- these represent the individual object models and object categories.

*Dept. of Computer Science & Engg., I.I.T., New Delhi - 110016, India. email :- lipika@cse.iitd.ernet.in

†Dept. of Computer Science & Engg., I.I.T., Kharagpur, India - 721302. email :- ppdas@cse.iitkgp.ernet.in

‡Dept. of Electrical Engg. I.I.T., New Delhi - 110016, India. email :- santanuc@ee.iitd.ernet.in

Feature nodes $\mathcal{F}$ and Relation nodes $\mathcal{R}$ - these nodes represent the attributes of the entities.

The first three kinds of nodes are grouped as *Active nodes* since these represent entities finally recognised by the system. The attributes of these nodes are *a) type* and *b) label* indicating its type and name respectively. The attribute nodes are termed as *Passive nodes* since they enable the recognition process by providing factors based on which it proceeds but are not recognised themselves. The attributes of these nodes are discussed below.

Feature nodes have two attributes FEATURE_TYPE denoting the type of the feature represented at that node and OF_TYPE denoting the category of node for which the description is applicable. For example, FEATURE_TYPE *function*, OF_TYPE *handle* is *gripping*, in association to hammer describes the function of handle in hammer.

Relation node also has two attributes RELATION TYPE and OF_TYPE which is a set indicating the category of all the parts between which the relation is defined.

The links in $\mathcal{G}$ are of the following types:-

ISA - A link between a specialized concept and its generic category.

HAS_PARTS- An attributed link between an object and its parts or a part and its subparts. It has attributes **status** indicating the nature of association- whether the part or subpart is *inherited unchanged* or *modified* or *new* and **instance_name** which is a unique identifier for an instance of the part or subpart.

FEATURE - A link between a node and an associated feature node. It has the attributes *a) status* which means same as above, *b) value* indicating the value of the feature in association to the current concept and *instance_name* indicating the instance of part or subpart of the object to which the feature description is applicable. SELF in the last field indicates global association.

RELATION - A link between a node and an associated relation node. Its attributes are same as that of the feature link.

Exceptions are handled with a link having status MOD and value NIL - to indicate that the attribute is disinherited.

A *descriptor* is the description of an association of a part, feature or relation node to a concept. A description of a concept C is defined as

$Desc(C) = \langle label, type, ancestor, part\_descriptors, feat\_descriptors, rel\_descriptors \rangle$, where a set of descriptors consist of the entire set of attributes of C, acquired from its parent, new or modified parts and those which are associated to it with status NEW or MOD.

3 Recognition of a single unknown object

The recognition problem can be informally described as - *given an object description in terms of features and relations known to the system, recognize it.* The given object may be a new but generically similar to a known object. Similarity may be functional or structural. We provide a mechanism to recognize such similar objects and enrich the KB suitably. For this, we consider two phases of recognition -

Generic Classification - i.e. categorization of the input object into some known conceptual class. The most specific class in a path of the existing KB graph, which is subsumed in the input description is taken as the *generic class* of the input object. Hence, the given object has to possess all the properties of the generic class. We note that, generic classification is dependent on the state of knowledge at the time of recognition and later on we show how this knowledge is refined with gradual updation.

Identification - in this phase , the apriori object/s with which the current object maximally matches are found out.

The recognition process is based on the theory of activation. The steps of the recognition process are outlined here.

(i) The input object is presented to the system as a collection of primitive instances, features associated to them and relations between these instances.

(ii) A **binding doublet** consisting of a KB primitive and the indexed input primitive is constructed and associated to the primitive node, if both are of similar type and have identical features associated to them. There may be more than one binding with each node and also one input primitive may be present in bindings associated to different primitive nodes.

(iii) The binding doublets are then propagated to the nodes which have these primitives as parts. All possible combinations of these binding doublets are formed and each binding is checked for feature and relation matches. The exact strategy of matching depends on the type of the feature or relation. This is part of the domain knowledge. The matching routine is encapsulated in the definition of the global feature or relation which is unique for the feature type and the node type to which it is applicable. For example, a match for the feature *feature_type material, of_type head, value metal* occurs only when all the input primitives constituting the subparts of head in the considered binding have the *feature_type material value metal* attached to them.

During the **classification process**, only those bindings in which *no input primitive instance is repeated and for which all the features and relations associated to a node are satisfied*, are accepted as **valid bindings**. If a node has at least one such valid binding associated to it, it is assumed to be activated and an activation value of 1.0 is associated to that binding. During the **identification process**, bindings are propagated from the primitives upward in a similar manner as described above. A **valid binding** is now described as one for which *no primitive instance is repeated.* The activation value of a binding is now calculated as a function of the number of mutual matches and mismatches of feature and relation values between

the input object and the node and their corresponding uniqueness values.

(iv) The bindings retained at an activated node are propagated to the nodes related to them through HAS_PARTS and ISA links.

(v) In the classfication stage, the most specific node in a path which is activated is the generic class descriptor of the input object. In the identification stage, the node(s) with the highest activation value is(are) the identified classes.

4 Learning from Recognition

The recognition process returns two sets of active nodes as its result - one set representing the possible generic classes of the input object and another set denoting the objects with which it matches maximally. The two sets are now analyzed to establish the proper identity of the object and decide its position in the hierarchy such that the resulting KB is consistent.

4.1 Analysis of the Recognition results

At this point we may note that, we now have the description of the input object as $Desc(O) = \langle ?, ?, ?, \textit{part\_descriptors}, \textit{feat\_descriptors}, \textit{rel\_descriptors} \rangle$, where the label, type and ancestor have to be determined. The feature descriptors and relation descriptors are currently associated to primitives only and once the identity of the object is established, proper associations are made with the global descriptors. We now outline the various cases that may occur and their interpretations.

In the following discussion we assume $Desc(n)$ is constituted of only the feature, relation and part descriptors, O represents input object, G represents the node in KB determined as the generic class of O and I is the node in KB which is determined as the identified class of O. O is assumed to be a connected, unoccluded object.

Case - I G and I are the same node. There are two possibilities in this case.

(i) $Desc(G) = Desc(O)$.

This implies that the node G is an exact match for the input object and the input object is recognized with absolute certainty.

(ii) $Desc(G) \subset Desc(O)$.

This implies that node G is not an exact match of O and O has additional features and /or relations and /or parts. Also there is no subclass of G with which there is a greater match for O and therefore the system concludes that O *is a* new *subclass of* G.

Case II I IS_A G.

In this case I and O belong to the same generic class G and O has similarity with I. Hence the system decides in favour of *constructing a new subclass* C such that C IS_A G, I IS_A C and O IS_A C.

Case III I is not a successor of G.

(i) I has no ancestor - This indicates that $\mathcal{I}$ and $\mathcal{G}$ may have a common ancestor $\mathcal{C}$ and one of them is an exception to the generic concept. A *new concept* C is created with G and I as its subclasses and O is put as O IS_A G. This ensures that future instances of O would not be identified with I.

(ii) If I has an ancestor - then as experience shows, the commonality is attributed to some unique features and relations which were so far only in I and now shared by O and therefore I is ignored for the establishment of actual identity of O. O is introduced as a new subclass of G. However, this may not be the best possible position for O and the possibility of incorporation of new classes as parent for ancestors of I and G is being examined in this context.

We note that all of the above cases are applicable to both PART and OBJECT type nodes of $\mathcal{K}$.

Case IV There may be no single generic class but *a set of PART nodes in* G.

(i) I may also be a set of part nodes with IS_A links to members of G or identical to members of G.

In this case, first the identity of the various parts are established by using the above mentioned decision making procedures. And then O is considered as *a new generic class* having HAS_PART links to the recognized part nodes. The type of node O is determined from the existing elements of the KB that have HAS_PART links to the members of G and I.

(ii) I is an object node or a part with HAS_PART links to the members of G.

(a) I has no ancestor.

Then the system decides to create a new generic concept C such that O IS_A C and I IS_A C.

(b) $\exists\ P \in \mathcal{K}$ *s.t.* I *IS_A* P and P has no ancestor.

In this case the system *creates a new concept* C such that P IS_A C and O IS_A C. C has no parent. This case accommodates exceptions.

(c) Otherwise O is recognized as a new concept not related to any existing concept through IS_A link.

Case V G is NULL.

This means even the primitive nodes are not activated. Thus the object is considered as nonsensical with respect to the current KB. The system now has to decide on the incorporation of new primitives into the KB which would mean analysis of their construction from geometric primitives. We have kept it currently outside the scope of our work though we notice that the current system can be easily extended to do that.

4.2 Updation of the Knowledge Base

We have mentioned above the conditions as well as positions of creation of new concept nodes in the KB. The Updation module now adjusts the link attributes of the enhanced knowledge base to ensure consistency. The attributes of the feature, relation and HAS_PARTS links of the entire knowledge base is updated using the following rules.

Case I - A new parent concept C is created:-

(a) C has an ancestor

(i) C initially inherits all the attributes from its ancestor with status set to LIVE and values same as those of its ancestor.

(ii) All the attributes common to all the chil-

dren of C are now associated to C in the following way. If all the children have the same value for an attribute, the common value is accepted as the value for that attribute of C also, else it is accepted as PHI. If the attribute already exists in C it is checked whether the value is same as desired or not. If they are not equal, the attribute link is changed to MOD and the value is set to the desired value. If the attribute does not exist in C a link with status NEW and the desired value is set up between C and the attribute type node.
(iii) If C has any NEW part, C acquires the attributes of the part with status LIVE and values unchanged.
(iv) The input object O and the set of nodes I are incorporated as subclasses of C.
(v) Steps i to iv are repeated for each subclass of C.

(b) C does not have any ancestor.
(i) An empty node is created and named C.
(ii) Part links are created between C and its possible parts where the set of possible parts are decided from the nodes in G and I as described in the previous section.
(iii) Attributes acquired from the parts are associated to C.
(iv) Steps (iii), (iv) and (v) of Case I are carried on.

Case - II O is identified as a subclass of an existing class.
(i) O is created and step (i) is same as that of Case - Ia. *(ii)* Till now O does not represent the input object exactly. Correspondence between existing attributes of O and the input attributes are established by considering the valid binding lists associated to nodes in I.
(iii) Values of existing attributes are changed if the input values are different from those inherited by O and the status of the corresponding attribute link is changed to MOD.
(iv) For all new attributes present in the input, attribute links with status NEW and the input values are created.

The system now checks for attributes which occur frequently in the subclasses of a class but are not associated to the parent. These attributes have low intra-class saliency values and hence are dragged down to the parent and the links are again updated. This optimizes space.

4.3 Determination of Saliency of Features and Relation

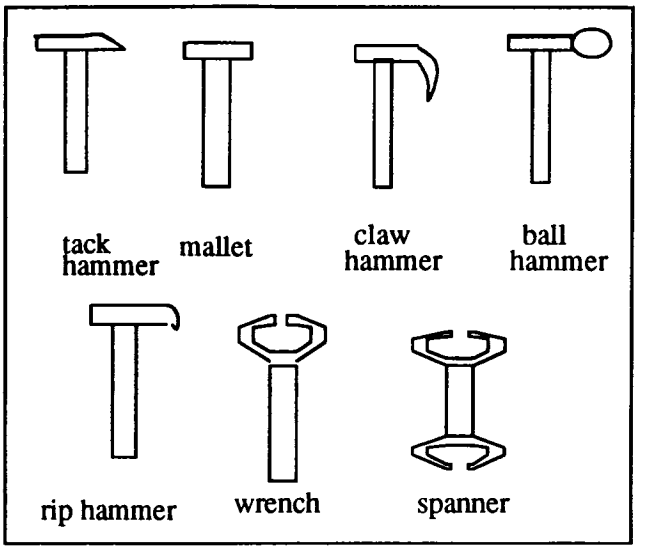
Saliency and uniqueness values are used for optimization and recognition. Saliency of an attribute described by a passive node is defined as the inverse of the frequency of its occurrence in association to an active node. Whenever an active node is created in the knowledge base, there may be attribute links with status NEW or MOD starting from it and incident on an attribute node. With the introduction of each such link, the count indicating the number of occurrence of that attribute is incremented by 1. If the attribute was not there in the KB before, an attribute node is created with the FEATURE_TYPE and OF_TYPE fields derived from the input description and the count of occurrence of the newly created attribute node is set to 1. At the end of an updation operation, the saliency value of the attribute is calculated as the inverse of its count of occurrence. The *uniqueness of an attribute value* is calculated as the ratio of the occurrence of that value to the total number of occurrences of that attribute with any value. Each NEW or MOD link to the attribute has a value associated to it. With each attribute node, we associate a set of values. Each value has a counter associated with it to indicate the number of times it occurs in association to a NEW or MOD link to the attribute node. If the value, already exists in the set, its count is incremented by 1 else the value is appended to the set and its count is set to 1. Finally, the uniqueness of a value is aclculated at the end of each updation operation and the saliency value and the uniqueness value of the attribute and the attribute value is associated to each link.

As already mentioned in section 3, the saliency values and the uniqueness values are used in the identification process to determine the closest match for the input object. The use of these two values handles the cases of adding instances to the system and adding classes to the system excellently. We fed our system with polygonal objects of the same type like squares of varied sizes. Though, the individual line lengths were highly unique, the low saliency value of the attribute *length of side* (since it kept on being MODIFIED with each instance, its occurrence increased) obscured the uniqueness effect and all were clubbed under square. But between a square and a triangle, the high saliency of the attribute *number of sides* which occur only at the generic levels of quadrilateral or pentagon or triangle as well as the unique values (4 or 5 or 3) forced the system to create a new basic generic class triangle to accommodate the various instances of triangle under it.

5 Result

We have presented here the steps in the gradual development of the KB on the domain of handtools. The KB initially consisted only of the description of the parts head and handle and a basic class handtool. The handle was described as a long, thin strip with two flat ends joined to the two ends of a flat, elongated, rectangular region. The head is described as an entity having two ends joined to a region but the exact shapes, sizes or material of the ends etc. are not defined. Handtool is defined as an object class with two parts - a head and a handle. The two parts are defined to be joined but the nature of join is undefined. An associated property of handtool is that the length of its handle is approximately double of that of the head. The system was then fed with the descriptions of various types of hammers, wrenches and spanner as shown on the top of figure 1. Each input description consisted of the physical attributes of an instance of a tool like length of the handle, shape of the ends of head, material of head and handle, values of the joining angles etc. Each tool was catego-

rised to an existing class and then identified either with the generic class itself or a subclass of it. The Knowledge base is then suitably updated as per the rules outlined in the previous sections. Figure 1 shows how recognition is refined with updation. The first instance of tool - a tackhammer was initially classified and identified as a **handtool**. But with updation of the knowledge base, its generic class is refined to STRAIGHT SPINED HAMMER. The ball hammer, which comes later in the sequence is correctly classified and identified as a new type of STRAIGHT SPINED HAMMER straight away. The usage of saliency and uniqueness of attributes are illustrated by the clubbing of rip hammer and claw hammer to form a new class CURVED_SPINE HAMMER since the feature CURVED SPINE for claw hammer is found to be more unique than the feature STRAIGHT SPINE among the subclasses of hammer, as the later is shared by more subclasses of hammer.



6 Conclusion

We have presented a system which can recognise and learn unknown objects from visual inputs. The learning process is preceded by generic classification and followed by a consistency check on the updated KB. The saliency of the attributes are determined by the structure of the KB itself and is later on used for recognition and optimization purposes. We have tested the system on the domain of handtools and polygons.

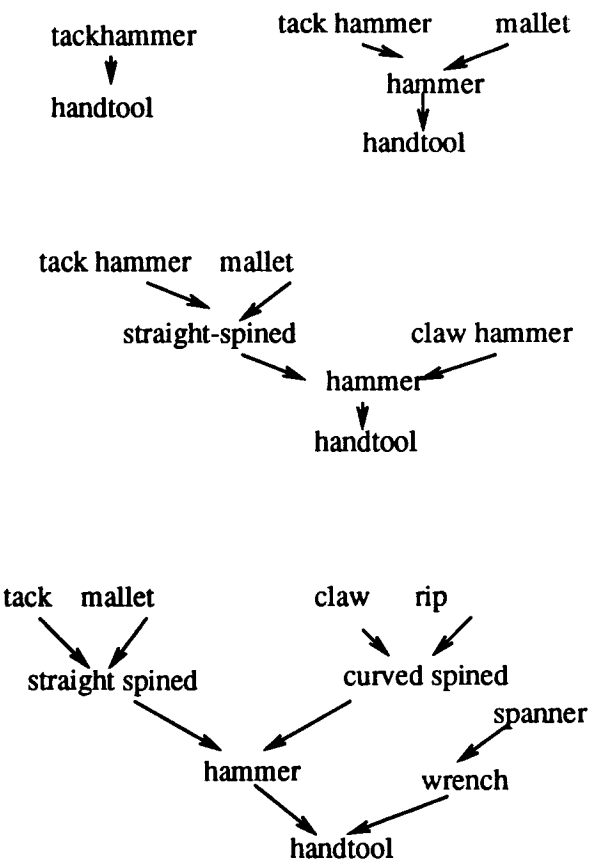


Figure 1: Development of the HANDTOOL hierarchy

References

[1] E.A. Feigenbaum. *The simulation of verbal learning and behaviour*. McGraw Hill, New York, 1963.

[2] D. Fisher. Knowledge acquisition via incremental concept learning. *Machine Learning*, (2):139–172, 1987.

[3] J.H. Gennari, Pat Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, (40):11–61, 1989.

[4] M Lebowitz. Categorizing numeric information for generalization. *Cognitive Science*, (7):1–40, 1983.

[5] Louise Stark and Kevin Bowyer. Achieving generalized object recognition through reasoning about association of function to structure. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(10):1097–1104, October 1991.

[6] R.E. Stepp and R.S. Michalski. Conceptual clustering of structured objects: A goal-oriented approach. *Artificial Intelligence*, (28):43–69, 1986.

[7] A. J. Vayda and A. C. Kak. A robot vision system for recognition of generic shaped objects. *CVGIP: Image Understanding*, 54(1):1–46, 1991.