

# GRUFF-3 : Generalizing the Domain of a Function-Based Recognition System<sup>1</sup>

Melanie Sutton<sup>1</sup>, Louise Stark<sup>2</sup> and Kevin Bowyer<sup>1</sup>.

1. Department of Computer Science and Engineering  
University of South Florida  
Tampa, Florida 33620 USA  
sutton or kwb @csee.usf.edu

2. Department of Electrical & Computer Engineering  
University of the Pacific  
Stockton, CA 95211 USA  
stark@napa.eng.uop.edu

## Abstract

Representation systems which support “generic” object recognition offer promising advantages over current model-based vision. Systems applying function-based reasoning are one such approach. In this approach, specific geometric or structural models are disregarded, in favor of analyzing the shape to determine functional requirements for *category membership*. This paper presents an explanation of the ideas behind function based modeling and a description of the extensions made to create the **Generic Representation Using Form and Function-3 (GRUFF-3)** system. This system analyzes the 3-D shape of an object and classifies the object according to its possible function as some (sub)category of the superordinate category *dishes*. The initial **GRUFF** system implementation was restricted to the *furniture* domain and required five knowledge primitives (clearance, relative orientation, proximity, dimensions and stability) to realize the functional requirements of the categories represented. The important contribution of our current work is that a significantly larger domain of objects can now be recognized with the addition of just one new knowledge primitive, *enclosure*. An evaluation of the performance of the system is presented for a database of over 200 3-D shapes.

**Keywords:** computer vision, object recognition, 3-D shape, reasoning about function.

## 1 Introduction

We can distinguish between three levels of increasing sophistication in models which attempt to capture generalized 3-D object shape: (1) *parameterized geometric models*, (2) *structural models* and (3) *parameterized structural models*. A *parameterized geometric model* is created by replacing some

---

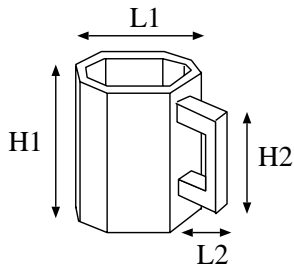
<sup>1</sup>This research was supported by AFOSR grant F49620-92-J-0223, NSF grant IRI-9120895, and a NASA Florida Space Grant Consortium graduate fellowship.

of the constants in a geometric model with parameters that may be constrained to lie within defined ranges. A *structural model* is distinguished from a simple geometric model by specifying an explicit construction of the object as a set of parts (a “part-whole” model). The model may be hierarchical, with several stages in which a “part” is specified by another structural model before the definition is eventually reduced to primitive geometric descriptions. One motivation for using a structural model is to make it convenient to specify sophisticated parameterized geometry. A comparison of these first two approaches is provided in Figure 1-(a) and Figure 1-(b). It also becomes possible to parameterize the structure itself. In a *parameterized structural model*, variations in the essential structure of the model may be modeled by parameterizing the number and location of some of the component parts of the object.

It is clear that any approach which assumes that the recognition system is given an exact model for each object that it is to recognize is inadequate to support “general purpose” vision. The ability to recognize novel objects at the category level is a key step in creating a robust autonomous system. The modeling approaches mentioned above are attempts to capture a category of objects by generalizing the model of some particular prototype object. For any reasonably complex environment, any of these approaches is ultimately limited by either the storage space required for the set of models or the processing speed at which matching can be performed. Even given sufficient computing resources, it is unlikely that all the parameterizations or component descriptions necessary to sufficiently generalize a prototype object model could readily be anticipated.

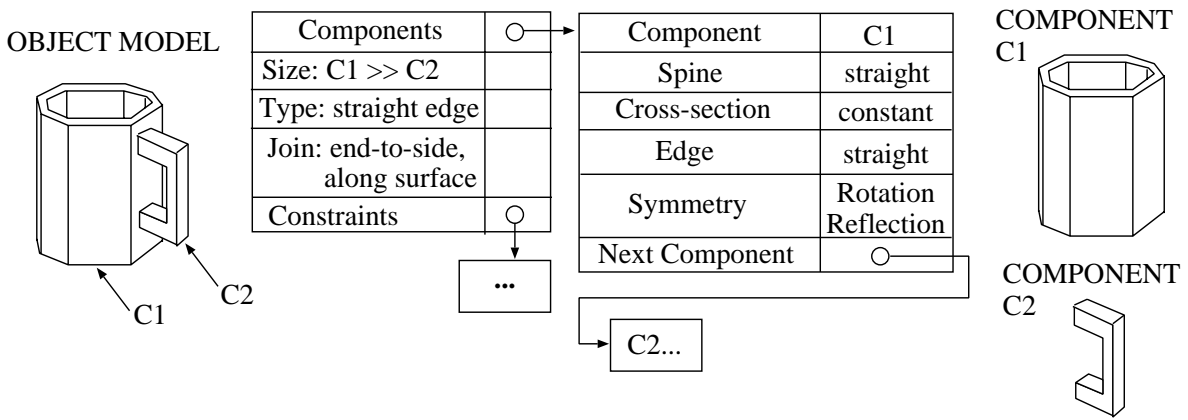
Modeling a category of objects by representing knowledge about how their intended functionality constrains their possible shape is one approach with the promise to avoid these problems. No specific individual object models are stored. Nor are “category” models created by generalizing the model of a specific prototype object. The key advantage over traditional approaches is that a “small” knowledge base suffices to recognize a “very large” domain of objects. As an example, one functional description suffices to recognize all of the objects shown in Figure 1-(c).

Some elements of geometric and structural modeling approaches are still applied here. For example, parameterizations of allowed ranges of shape may be used to test the proximity or the

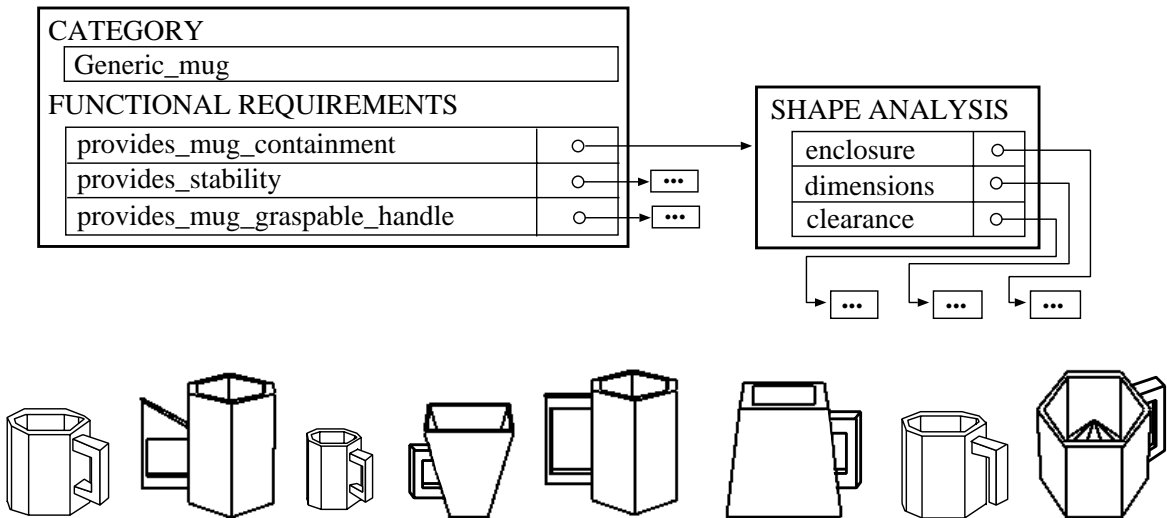


MUG FRAME					
IS_ A	artificial_object				
SUBCLASSES	mug				
CONSISTS_OF	body			handle	
SIZE	LENGTH	min_L1	max_L1	min_L2	max_L2
	WIDTH	min_width		max_width	
	HEIGHT	min_H1	max_H1	min_H2	max_H2

(a) Parameterized Model



(b) Structural Model



(c) Functional Model

Figure 1: Comparison of Geometric and Functional Models

relative orientation of parts of an object. However, function-based reasoning is **not** a model-based approach. For example, in order to recognize the objects shown in Figure 1 using a function-based approach, 3-D shape analysis applying ideas of physics and causation determines if the shape offers *containment*. Alternatively, capturing this idea using a model-based approach requires defining in geometric and structural terms all potential parameterizations or components of a shape which may offer containment.<sup>2</sup> Any geometric/structural model would almost certainly fail to recognize some members of even this simple category of objects. However, functional models would capture a broad variation in allowed shape without reference to any specific geometric or structural plan. For this reason, function-based models seem to provide better support for “purposive” (in the sense of Aloimonos<sup>(1)</sup>) and “task-oriented” (in the sense of Ikeuchi<sup>(2)</sup>) vision.

The function-based approach can also be traced to category concept research in psychology. Humans develop category concepts by grouping objects with similar characteristics and organize categories into a hierarchical taxonomy (Rosch<sup>(3)</sup>). Such a hierarchy has three major levels of specificity. For example, a basic (or entry level) category is the name that we most commonly give to an object, such as “glass.” To represent a refinement or specialization of a basic category we have a subordinate category, like “mug” or “juice glass.” Finally, a generalization of a basic category would be a superordinate category, such as “dishes.” In a function-based approach to defining object categories, a basic category or a subcategory is defined in terms of required functional properties, such as *provides\_containment* or *provides\_graspable\_handle*. As stated previously, the advantage is that a greater variation in possible object shapes can be more easily captured. For example, a specialization within the *basic* category cup/glass is the *subordinate* category mug. This subcategory definition uses a qualitative description of the functionality of a handle. The “to be graspable” property can be satisfied by an infinite variety of shapes. On the other hand, feature-based categorization would depend on a stored set of prototypical handles for matching and verification, as shown in Figure 1.

---

<sup>2</sup>A similar argument can be made about determining the *stability* of an input object.

## 2 Previous Work

Minsky has argued persuasively for a *form and function* approach to representing knowledge about object categories (Minsky<sup>(4)</sup>). Another well-known work in the area of reasoning about object function is that of Winston, Binford and colleagues (Binford<sup>(5)</sup>, Winston<sup>(6)</sup>). The input to this system consists of both the function-based category definition and the segmented, labeled description of the object as a semantic network. Brady, Connell and colleagues discussed the relation between geometric structure and functional significance in their design of the “Mechanic’s Mate” system (Brady<sup>(7)</sup>, Connell<sup>(8)</sup>). For the object category examined, *hammers*, the elements of a structural definition, “handle” and “head,” closely correspond to the elements of a function-based definition, “graspable” and “provides a nail-striking surface.” DiManzo and colleagues proposed a recognition system design that utilizes knowledge about function within an expert system framework (DiManzo<sup>(9)</sup>). Primitives are defined in the form of individual expert systems that evaluate the 3-D information using constraints defined by the user.

Brand described a system for integrating low-level visual processing with causal and functional analysis to interactively build a model of relationships between parts in a scene (Brand<sup>(10)</sup>). The input to the current system includes scenes of simple mechanical devices, and the output provides an explanation of what the device does and how it works. Kise *et al.* described an approach that uses a “simple machine” (e.g., lever) description in modeling functionality-based object categories and reasoning about 3-D shape (Kise<sup>(11)</sup>). Rivlin proposes the use of functional recognition in the context of goal-directed robotics, where functional models can be built and updated from input images (intensity or range) (Rivlin<sup>(12)</sup>).

## 3 Previous Versions of GRUFF

The **GRUFF** (**G**eneric **R**epresentation **U**sing **F**orm and **F**unction) systems differ from many of the approaches described in the previous section because analysis is applied to an *uninterpreted* rigid 3-D shape. (By “uninterpreted” we mean a pure shape description, with no segmentation into

parts and no semantic labeling of any portions of the shape.) In our approach, input shapes are *not* restricted to any particular orientation. Also, **GRUFF** has been extended to recognize a broad range of object categories. The recognition analysis considers shape, physics and causation to realize functional requirements such as “containment” or “stability,” which may fall outside the realm of other approaches. Overall, the **GRUFF** systems provides the most detailed and comprehensive exploration of the “form and function” approach to recognition.

The initial **GRUFF** system implementation was restricted to the *furniture* domain and required five knowledge primitives to realize the functional requirements of the categories represented (Stark<sup>(13,14)</sup>). The **GRUFF-3** expansion to the new domain *dishes* was relatively easy to accomplish because the original system was designed using primitive “chunks” of knowledge intended to be readily applicable to a variety of domains. Specifically, the following changes in system knowledge and design were made for the extension:<sup>3</sup>

1. Adding the concept of the superordinate category *dishes* involved creating functional definitions for the members of this category of objects and incorporating these definitions into the hierarchical recognition control structure (tree).
2. Realizing the new functional descriptions required defining and implementing the additional knowledge primitive *enclosure*.
3. To achieve efficient recognition within the new superordinate category, minor system control changes were made. For example, stable orientations are examined first when confirming recognition for the superordinate category *dishes*, whereas orientations of particular surfaces are examined first for the superordinate category *furniture*.

With these relatively minor changes, the system was readily adapted to the new domain. This is what makes the **GRUFF-3** system so powerful as a demonstration of a working generic recognition system. The fact that the number of knowledge primitives grows **very** slowly with the addition of new categories is a significant indication of its viability as an approach to general purpose vision. (Furthermore, since this work, initial implementation of the superordinate category *hand tools* has not revealed a need for any additional knowledge primitives.)

---

<sup>3</sup>Initial results of this work were reported in (Sutton<sup>(15)</sup>). These changes will be described in detail in the following sections.

The original focus of the **GRUFF** systems has been research-oriented. Two major simplifying assumptions are that the input shape description (1) represents a complete 3-D shape and (2) is given as a valid polyhedral boundary representation. Even with these simplifying assumptions, the **GRUFF** approach provides a convincing demonstration of the power of the form and function approach. In addition, we have recently completed initial work on adapting the system to be able to use incomplete shape models of the type that could be extracted from a single view of an object using laser range finder data (Hoover<sup>(16,17)</sup>, Stark<sup>(18)</sup>).

Other limitations are simply those that are inherent to analyzing only a static shape description of an object. For example, the function of a window is most often described as an opening in a wall that allows light to enter. To recognize such a function, confirmation that the panes of the window are made of some transparent material would be required. Obviously, this function cannot be recognized solely through shape analysis. Similarly, distinguishing between a plastic cup and a styrofoam cup goes beyond what can be obtained from shape analysis. Lastly, an object with moving parts, such as a tea kettle with a lid attached, would also be beyond the scope of the current system. However, we have found that for a broad range of categories, *form and function* analysis of rigid 3-D shape can be used quite successfully to determine functionality and ultimately guide further interaction with the object.

The remainder of the paper is organized as follows. Section 4 outlines the overall structure and operation of the **GRUFF-3** (**G**eneric **R**epresentation **U**sing **F**orm and **F**unction-**3**) system. Section 5 gives a trace of the actual evaluation of an input shape, in order to illustrate the use of knowledge primitives and functional properties in the object recognition process. Section 6 describes methods for evaluation of the system's competence in recognition based on a database of 206 shapes. Finally, Section 7 summarizes the current state of the project and suggests directions for continued research.

## 4 Overview of **GRUFF-3**

This section summarizes some of the basic concepts of the **GRUFF-3** system, similar to the original description provided in (Stark<sup>(19)</sup>). However, many of the details are updated and modified here to

reflect the difference in domains.

## 4.1 Function-based Modeling and Representation

The development of a function-based representation system begins with listing the object categories to be known to the system and describing the minimum set of functional requirements for an object to be included in each category. The tree structure created from this information for the superordinate category *dishes* is shown in Figure 2. The structure of this category definition tree guides the processing steps involved in reasoning about an input shape description. Nodes with a shaded border depict (sub)category names. Non-shaded nodes contain the functional requirements associated to each parent (sub)category. This tree shows five basic level categories (*cup/glass*, *pitcher*, *bowl*, *plate* and *pot/pan*) under the superordinate category node for *dishes*. Basic categories are typically given one or more separately named specialized subcategory definitions. For example, in Figure 2, *cup/glass* is a category node which has subcategory node *mug* as a child. The functional plan for *mug* requires all of the functionality specified for its parent category node, and also requires the specialization of *provides\_graspable\_handle*.

To understand how these functional plans are related, each basic and subcategory node can be pulled from the graph and rearranged such that the subcategory node is the root node (see Figure 3). From this figure it is easier to see how the requirements for a *mug* are determined first through the functional plan of the *cup/glass* node shown on the left. The results of evaluation of these requirements is then combined with the additional functional requirement of *provides\_graspable\_handle*.

### 4.1.1 Knowledge Primitives

An input 3-D shape is interpreted by **GRUFF-3** to determine whether it meets the specified functional requirements for any of the categories. The interpretation of whether or not a functional requirement is met is realized through a sequence of “knowledge primitive” (KP) invocations. A “knowledge primitive” can be thought of as a parameterized procedure call which makes some

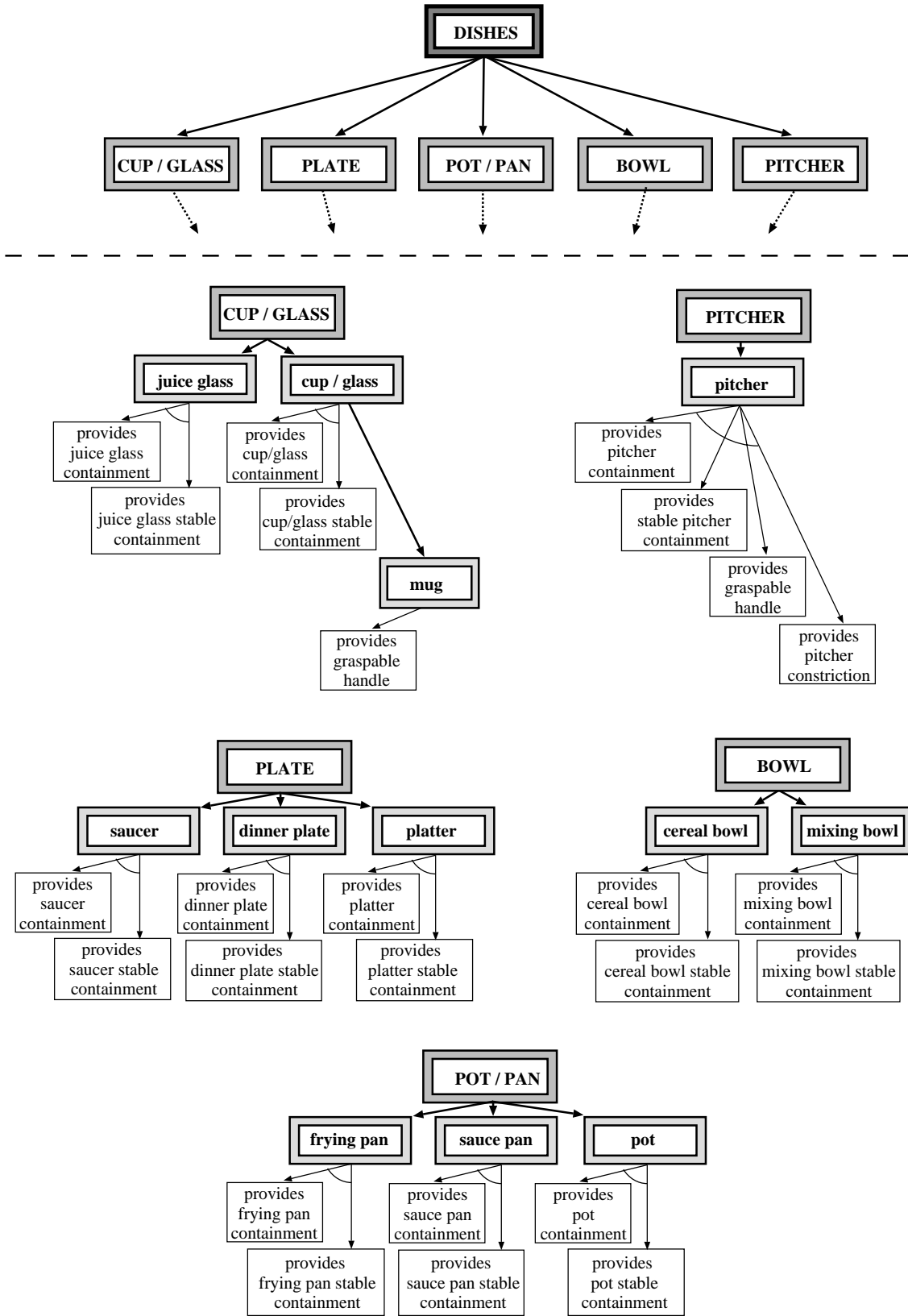


Figure 2: Superordinate Category Dishes

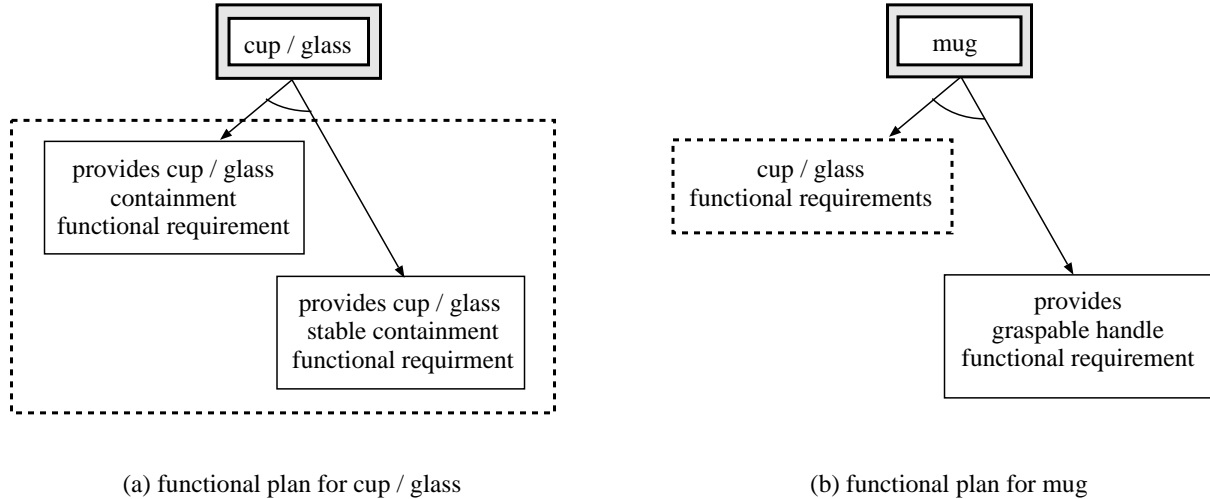


Figure 3: Functional Plans of the Subcategories cup/glass and mug

low-level observation about the object shape. Each knowledge primitive invocation returns an evaluation measure within the interval  $[0,1]$ , reflecting how well the shape meets the ideal desired observation.

The current implementation uses six knowledge primitives to define the functional plans within a superordinate category. These six primitives (*stability*, *clearance*, *dimensions*, *relative orientation*, *proximity*, and *enclosure*) are described below.

- *stability*( *shape*, *orientation*, *applied\_force* ) – Figure 4. This primitive can be used to check that a given shape is stable when placed on a supporting plane in a given orientation, possibly with a force applied. An example of this is to check that a pitcher will rest stably when placed upright on a table. It is assumed that the object has homogeneous density, so that the center of mass may be calculated directly from the shape. This primitive returns an evaluation measure of 1 if the shape is stable in the specified orientation, or a value of 0 if it is not.
- *clearance*( *shape\_element*, *clearance\_volume* ) – Figure 5. This primitive can be used to check that there is a specified volume of unobstructed free space in a particular location relative to a particular part of the shape. The volume is represented by a *clearance polyhedron* which is specified by a set of faces and vertices. The evaluation measure is 1 if the volume specified is unobstructed, or 0 if it is obstructed. An example of this type of check is to check that the area above the opening of a plate or around the outside of a pot handle is accessible to the user.
- *dimensions*( *shape\_element*, *dimension\_type*, *range\_parameters* ) – Figure 6. This primitive can be used to determine, for example, if the width or depth of a surface lies within a specified range. An example of this is to check that the concavity of a bowl or platter should be within a certain height and width/depth range, with an appropriately-sized enclosing face area. The

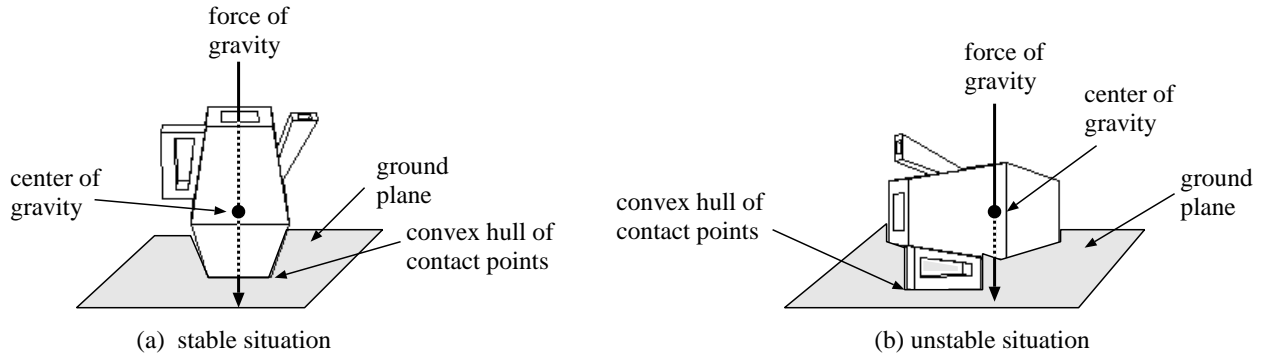


Figure 4: Primitive Stability

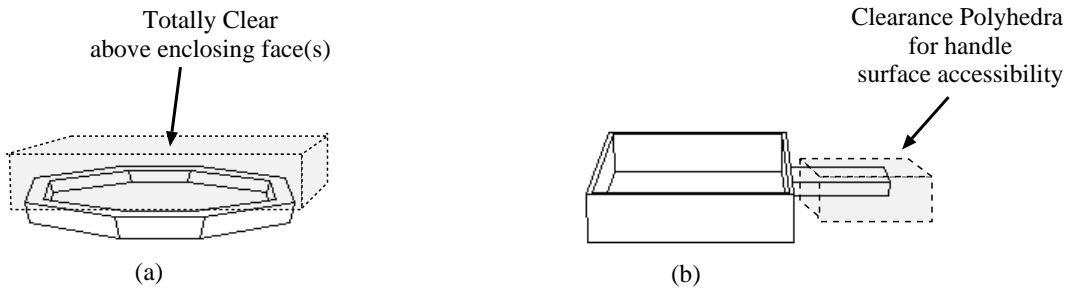


Figure 5: Primitive Clearance

evaluation measure is calculated using four *range parameters*: least, low\_ideal, high\_ideal and greatest (see Figure 7). These parameters are used to calculate a value for the evaluation measure based on where the calculated width or depth falls within the specified range. Any value between low\_ideal and high\_ideal results in a measure of 1. Values outside of this range but between least and greatest fall off linearly to 0.

- *relative orientation*( *normal\_one*, *normal\_two*, *range\_parameters* ) – Figure 8. This primitive determines if the angle between the normals for two surfaces (*normal\_one* and *normal\_two*) falls within a desired range. An example of this is to check that the opening of a pitcher allows a clear area for fluid flow, oriented along the vector from the midpoint of the handle. This primitive can also be used to calculate the transformation which would position the shape in a desired orientation. For example, a common operation is to find the transformation which would orient a specified surface parallel to the support plane. When relative orientation is used to simply reorient the object, a measure of 1 is always returned.
- *proximity*( *shape\_element\_one*, *shape\_element\_two*, *range\_parameters* ) – Figure 9. This primitive can be used to check qualitative relations between shape elements within the object, such as *above*, *below* and *close to*. An example of this is to check that the handle of a pitcher must be situated above the average center of mass of the object and within a given range to make it easy to lift.

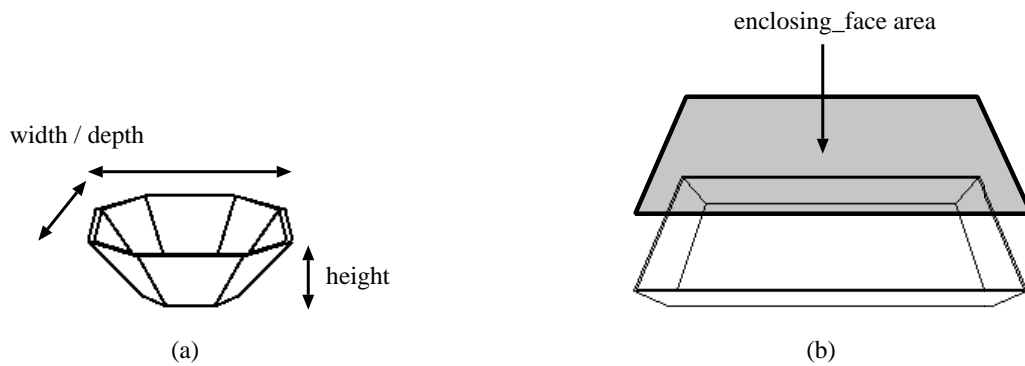


Figure 6: Primitive Dimensions

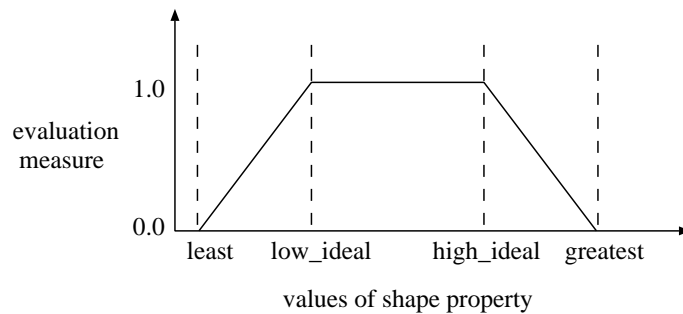


Figure 7: Using Shape Properties to Calculate an Evaluation Measure

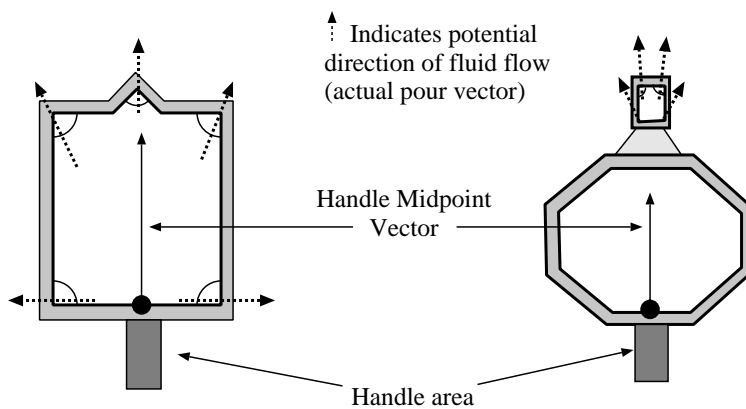


Figure 8: Primitive Relative\_Orientation

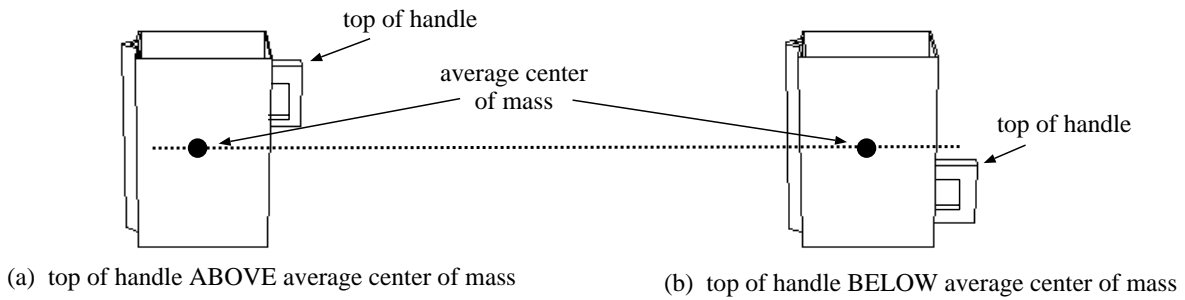


Figure 9: Primitive Proximity

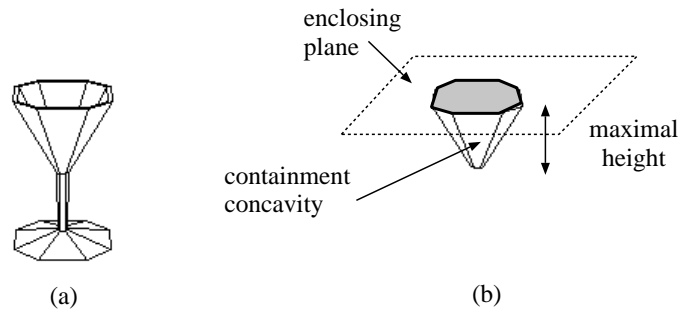


Figure 10: Primitive Enclosure

- $enclosure(orientation, concavity, enclosing\_plane)$  – Figure 10. This primitive is used to determine if there exists a concavity in the shape which can be “closed” by a single plane introduced parallel to the support plane in a given orientation. Because an infinite number of planes could be introduced at different *levels*, each enclosing a different volume, it is desired not only to confirm that a concavity can be enclosed but also to find the maximal volume that can be enclosed. This primitive returns an evaluation measure of 0 if no maximal concavity can be found for the given orientation, or a measure of 1 along with the list of potential containment concavities and enclosing planes.

#### 4.1.2 Hierarchical Control Structure of Dishes

The control structure for the system is based on defining the functional requirements for the categories of objects with a sequence of invocations of the above KPs. Each category of dishes represents a class of objects which provide stable containment of some constrained range of volume and/or shape. Figure 11 depicts a more detailed representation of a portion of the *dishes* category tree shown previously in Figure 2. Each node of this tree is represented by a frame having four fields: *Name*, *Type*, *Realized\_By*, and *Functional\_Plans*. There are four possible frame types: *Superordi-*

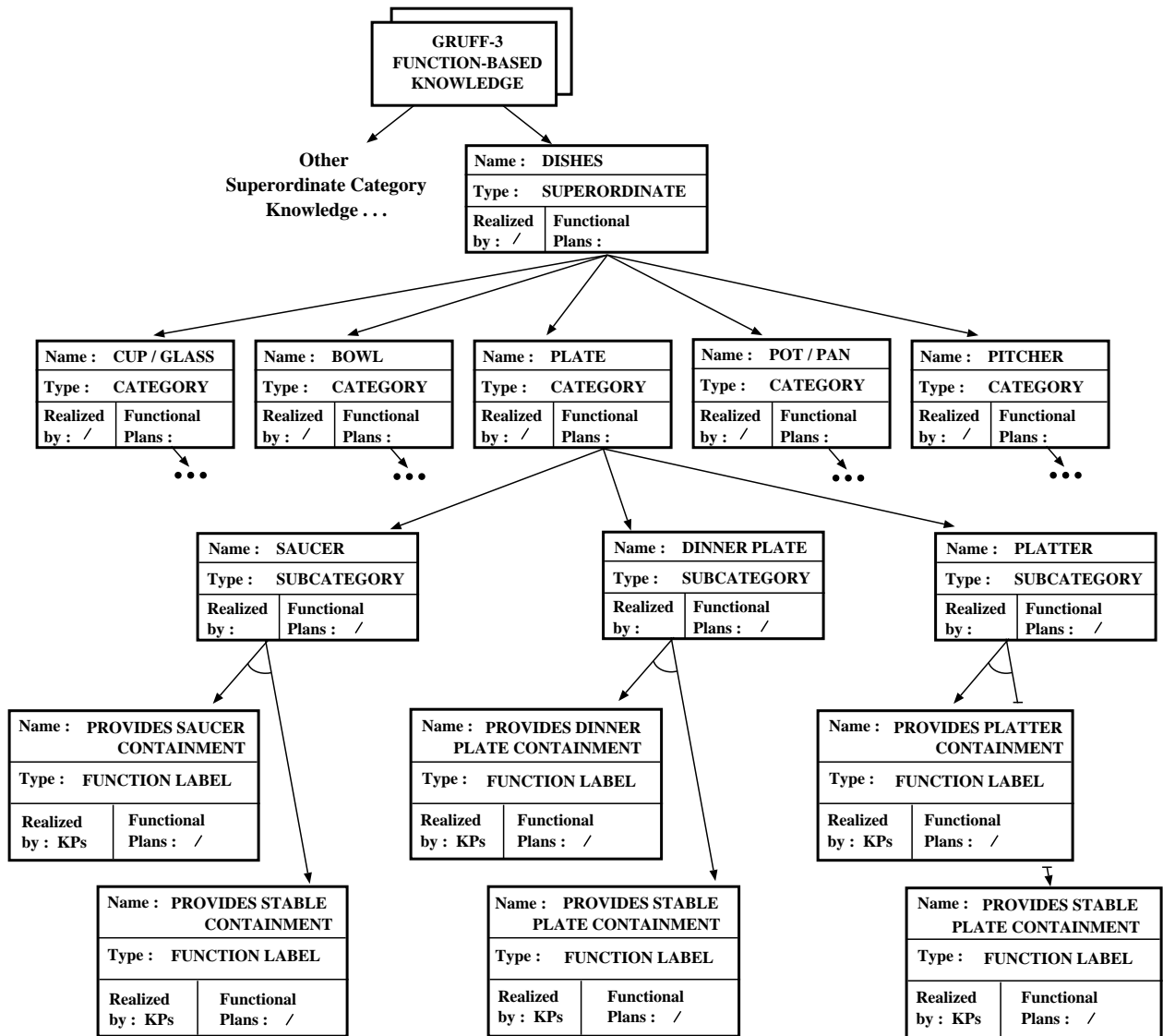


Figure 11: Representation of **GRUFF-3** Function Based Knowledge

*nate* (i.e., DISHES), *Category* (i.e., Cup/Glass, Bowl, Plate...), *Subcategory* (i.e., Saucer, Dinner-Plate, Platter) or *Function\_Label* (i.e., Provides\_Saucer\_Containment). Each *Superordinate* frame has associated with it one or more *Category* frames (in this particular case, five) which have links from their *Functional\_Plans* fields to *Subcategory* frames. At the *Subcategory* level, links in the *Realized\_by* slots point to *Function\_Label* frames for definitions of specific functional requirements, whereas links in the *Functional\_Plans* slots can point to further refinements in *Subcategory* frames.

Each subgraph having a *Category* frame as its root (i.e., Plate) denotes a separate basic level category. Different basic level categories represent alternate interpretations of the functionality of an object. At lower levels, subcategories represent specializations of the functional plan instantiated by the parent basic level category.

The tree is traversed in essentially a depth-first manner. Each of the functional requirements must be met by some portion of the structure in order for processing to proceed within a subgraph. Evidence is gathered in the form of evaluation measures which are accrued to produce an association measure for each (sub)category. If a “best used as” result is desired, then recognition is determined by choosing the (sub)category node with the maximum association measure above some defined *functionality threshold* value. If an “all plausible uses” result is desired, then all nodes containing an association measure above this threshold are reported.

## 4.2 System Evaluation Process

The key steps in the system’s shape interpretation process are depicted in Figure 12. The following sections discuss each of these blocks in more detail.

### 4.2.1 Evaluate Shape & Enumerate Key Functional Elements

The first step involves pre-processing of the object shape to accumulate information that is potentially of use at multiple points in the later recognition processing. Basic information extracted includes the following: the volume of the shape, the center of mass of the shape, the convex hull of the shape, and the volume and center of mass of the convex hull. “Potential” functional elements of

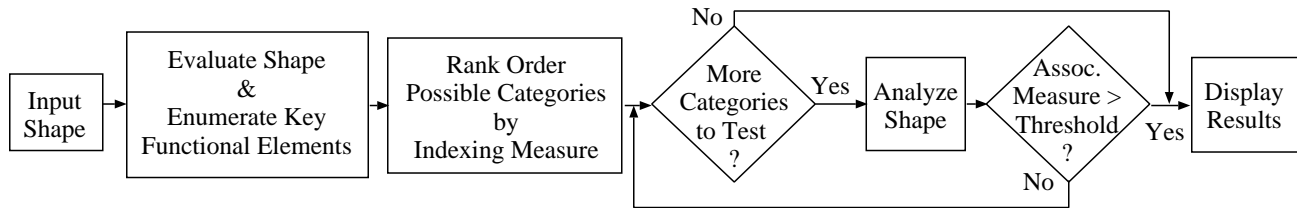


Figure 12: Flow of Control in Interpreting an Object Shape

the shape are also enumerated by building lists of portions of the object shape that could *possibly* fulfill some functional requirement. The top portion of Figure 13 provides short descriptions of example functional elements which can be enumerated during this *pre-processing* stage. For example, a list is formed of sets of potential concavity defining faces. Initially these lists contain all the faces of the shape which are not part of its convex hull. The lists are then partitioned into disjoint sets of faces such that each set defines a separate concavity within the object. In the bottom portion of Figure 13, we can see where later (in subsequent *recognition* stages) the concavity defining faces will be important for association to functional requirements of *dishes* objects, whereas the sets of approximately coplanar faces will be important for association to functional requirements of *furniture* objects.

#### 4.2.2 Rank Order Possible Categories by Indexing Measure


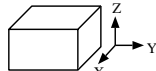
In the next stage, the system hypothesizes an ordered list of categories to use in evaluating the shape.<sup>4</sup> This processing begins conceptually at the root of the category definition tree and flows toward the leaves. Each category node has associated with it an indexing range based on the volume of the convex hull of the shape. The indexing range corresponds to the maximum potential volume that is consistent with the functionality of the containment (hence the use of the volume of the convex hull, rather than the overall volume of the shape). Within the superordinate category *dishes*, each known object category has some specific volumetric requirements whose ranges only

---

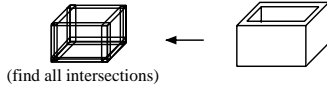
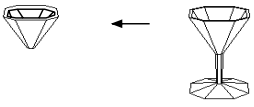
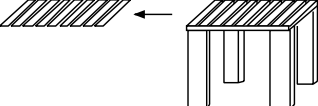
<sup>4</sup>Here, we discuss the indexing heuristics used for superordinate category *dishes*. For more details on indexing heuristics for different superordinate categories, see (Sutton<sup>(20)</sup>).

**During Pre-processing stage:**

**Calculations:**

1. volume - of object and of object's convex hull		
2. center of mass - of object and of object's convex hull	0.001 m <sup>3</sup> (0.0, 0.0, 0.04)	0.004 m <sup>3</sup> (0.0, 0.0, 0.08)

**Formation of Lists of "Potential" Functional Elements**

- list of convex subvolumes which object is composed of
 
- lists of individual surfaces (related to faces of the object)
  - concavity defining faces
 
- lists of virtual surfaces (combinations of faces of the object)
  - collections of faces which are essentially coplanar
 

**During Recognition stage:**

**Formation of Lists of "Known" Functional Elements Found to Fulfill Specific Functional Requirements**

**Three types:**


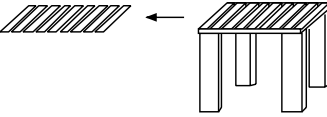
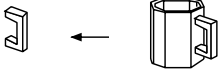
- a single face
  - the face which is determined to function as the enclosing surface of a concavity
 
- a group of object faces
  - the faces which are determined to function as the "sittable surface"
 
- a 3-D portion of the object structure, or the entire structure (e.g., stability is achieved by the entire structure of the object)
  - the faces which are determined to function as a "handle"
 

Figure 13: Enumeration of Key Functional Elements

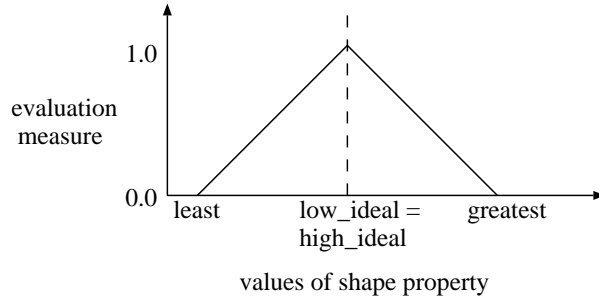


Figure 14: Calculation of Indexing Measures

partly overlap, so this yields a simple yet reasonably effective index to the set of possible object categories.

An *indexing measure* is computed for each category, depending on how well the shape’s calculated volume (determined in the pre-processing stage) falls within the specified range.<sup>5</sup> The indexing measure is one if the volume falls directly on the midpoint of the upper and lower limits of this range, and falls off linearly to zero otherwise, as shown in Figure 14. The index ranges were chosen from empirical testing (using prototypical dish descriptions) to be very broad, due to the varied shapes possible within each category, but help to eliminate “impossible” shapes. For example, these heuristics help eliminate evaluation of shapes the size of a satellite dish for possibly fulfilling the functional requirements of a bowl or platter.<sup>6</sup>

This indexing strategy is incorporated at every level in the control structure. Thus at each subcategory node, there is a  $(min, max)$  range for the key property for the subcategory. At each basic level category node, a similar  $(Min, Max)$  range for the key value is computed by taking the minimum of the minimums for each subcategory and the maximum of the maximums for each subcategory. At the superordinate category node, another  $(MIN, MAX)$  range is computed by taking the minimum of the minimums and the maximum of the maximums for the basic level categories.

---

<sup>5</sup>The *indexing measure* is distinct from the individual knowledge primitive *evaluation measures* and also distinct from the *association measure(s)* indicating the object’s category membership value(s).

<sup>6</sup>For clarity, using Figure 14 for the volume measure of category *cup/glass*,  $least = 0.00006 \text{ m}^3$  and  $greatest = 0.001 \text{ m}^3$ . A satellite dish, with a volume of say,  $1.125 \text{ m}^3$ , will receive an indexing measure of 0.0, and so will be eliminated from consideration as a potential member of the category *cup/glass*.

After the shape's volume has been checked, the categories are ranked according to the greatest indexing measure value generated. A similar indexing process can then be applied to determine the order in which to consider the subordinate categories. In the next processing stage, the input shape can be considered against the possible categories in the order of their ranking, until the shape is found to fulfill the functional requirements of some category with a final association measure of 0.4 or better (the *functionality threshold*). This leaves open the possibility that the system would recognize the shape as one category when in fact there is a lower-ranked (by the indexing strategy) category which would result in a higher association measure.

The ideal indexing strategy would require a “small” amount of work which was independent of the number of categories known to the system, and would always select as the first category to be considered that category for which the shape would have the highest final association measure. The amount of processing required in the pre-evaluation depends on the complexity of the shape but is independent of the number and type of categories known to the system. More details on the indexing scheme can be found in (Stark<sup>(14)</sup>).

### 4.2.3 Analyze Shape

Once a category is selected for further analysis (by passing the indexing range test described above), that subtree of the category definition tree is used as a control structure in the analysis of the shape. For each hypothesized subcategory, there is a subtree that defines the *functional plan* of that subcategory. The traversal of each subtree proceeds in a depth-first manner. As interpretation continues, the system begins to associate *function labels* of a shape (e.g. *provides\_cup/glass\_containment*) to possible *functional elements* of the shape. For example, analysis of the shape depicted in Figure 10 begins with confirming that an orientation exists which could satisfy *provides\_cup/glass\_containment*. The system then analyzes whether the shape could *provide\_stable\_containment* with this candidate concavity.

#### 4.2.4 Accrual of Association Measure

As the traversal proceeds, the system attempts to categorize an input shape as belonging to some subcategory by confirming that all functional requirements can be met. In addition, the system maintains an association measure to reflect how well these requirements are satisfied. As noted earlier, evidence is gathered from each KP invocation in the form of an *evaluation measure*. As each new requirement within a given (sub)category is encountered, this evidence is combined with the current association measure at that level. The final association measure for each (sub)category node therefore represents a “goodness of fit” of the object into the specified (sub)category.

The association measure is accumulated in a manner such that if a required knowledge primitive invocation returns an evaluation measure lower than the current association measure, it lowers the overall evidence for that subcategory. Representative instances of three families of aggregation calculi (Bonissone<sup>(21)</sup>) were evaluated and compared for performance on a set of example shapes for the object category “chair” in the original **GRUFF** system (Stark<sup>(22)</sup>). The operation which provided the best performance at this category level is the T-norm or conjunctive operation:

$$T(a, b) = a * b$$

However, when the subcategory being investigated is a refinement of its parent subcategory (in the way that a mug is a refinement of a cup/glass), the evidence gathered at the parent subcategory node is used in a way which raises the association measure calculated at the present subcategory node by some factor associated to the evidence gathered at the parent node, using a T-conorm operation:

$$S(a, b) = a + b - a * b$$

For example, the functional requirements of a *cup/glass* are defined as *provides\_cup/glass\_containment* and *provides\_cup/glass\_stable\_containment*. A *mug* is a subcategory of this level which requires the additional functional requirement *provides\_graspable\_handle*. The use of the T-norm and T-conorm operations for specific returned evaluation measures is demonstrated in Figure 15.

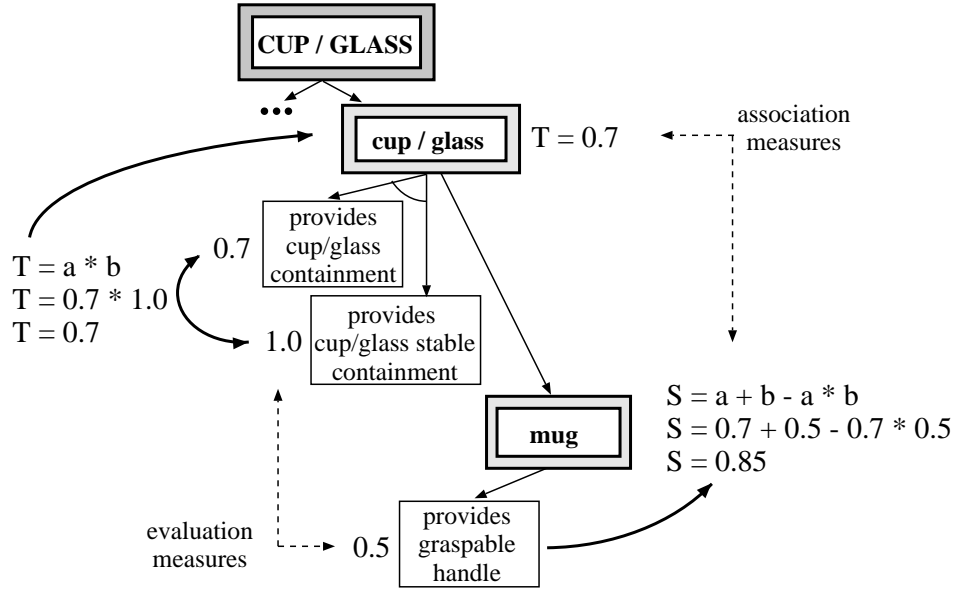


Figure 15: Use of Aggregation Calculi

Here, we find that overall the shape is able to function as a *cup/glass* with a final association of 0.70. However, the same shape functions as a *mug* with a final association measure of 0.85.

Finally, recognition is accomplished by traversing the control tree, visiting each subcategory node, and identifying all subcategories with an association measure above a pre-defined *functionality threshold* value. All of these categories can be considered as potential results. The node with the maximum association measure can be chosen as the “best fit.” Given the example in Figure 15, we would output category *mug* as the best fit for the input shape.

### 4.3 Complexity of Interpretation

The input shape description to **GRUFF-3** is a valid 3-D polyhedral boundary representation. Each shape, with  $N$  faces, is  $\theta(N)$  in the number of faces, edges, and vertices. The two stages of processing are the pre-evaluation analysis and the function-based interpretation.

Pre-evaluation analysis involves three important operations. First, the volume of the shape is calculated using an  $\theta(N^5)$  algorithm due to Edelsbrunner (Edelsbrunner<sup>(23)</sup>). Next, assuming uniform density for the shape, its center of mass is calculated, using an  $\theta(N^3)$  algorithm. Finally,

the system performs some initial interpretation of the shape by creating lists of all potential functional elements (as described in Figure 13). This includes potential functional surfaces (“essentially coplanar” collections of faces which form “virtual faces”), enclosable concavities, and all the convex cells which make up the shape. This step is  $\theta(N^2)$ .

The function-based interpretation stage begins with the system forming a list of hypothesized categories. It then begins to interpret the shape according to each of the functional plan nodes determined to be visited in the category definition tree. Each functional plan node is defined with a series of required functional properties. Each functional property is realized by a sequence of invocations of the knowledge primitives (KPs). The complexity of each KP is as follows:

- stability -  $\theta(N \log N)$
- clearance -  $\theta(N)$
- dimensions -  $\theta(N^2)$
- relative orientation -  $\theta(N^2)$
- proximity -  $\theta(NM)$ , N surfaces with at most M vertices
- enclosure -  $\theta(N^3)$

The number of nodes in the category tree, and so the maximum possible number of primitive invocations, is fixed and is independent of any particular input shape description. The order of complexity of the shape interpretation is therefore just the maximum complexity for any individual step. The maximum complexity step is that of computing the volume of the shape during the pre-processing stage, and so the overall complexity is  $\theta(N^5)$ .

## 5 Evaluation of Example Input Shape

The sequence of operations performed in interpreting a given shape is determined by the particular path followed through the category definition tree. Considering a specific example in detail probably provides the best means to gain a better understanding of the interpretation process. Consider the

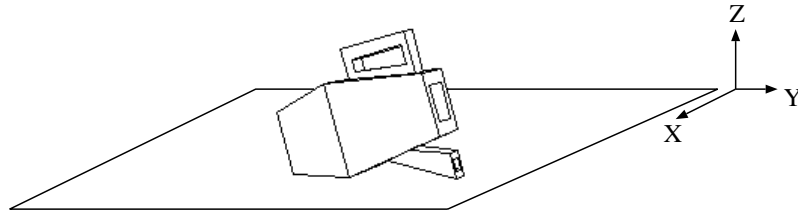


Figure 16: Sample Input Object

input shape shown in Figure 16. The “ground plane” is considered to be parallel to the  $XY$  plane, and gravity is assumed to act in the  $-Z$  direction.

The system begins with the pre-processing stage described previously. As a result of this step, the information available to later steps includes the original boundary representation, a list of essentially coplanar groups of faces that could act as virtual faces, the list of unique convex 3-D sub-volumes, the volume and center of mass of the object, and all sets of concavity-defining faces. When evaluating these concavity-defining faces, the system specifically determines the maximal volume concavity that can be enclosed when introducing an enclosing plane. Because a continuous range of levels exists at which to introduce this plane, a systematic approach is applied at each orientation. From the set of faces which define the object, ordered sets of vertices are available, which represent potential maximal points of a concavity.

At this point in the evaluation, in order to consolidate a potentially large amount of computation required for each category, it is necessary to confirm the object is self-stable in a given orientation.<sup>7</sup> A self-stable orientation is one in which the object has at least 3 non-collinear points of contact with the ground plane, and the normal vector from the center of mass of the object to the ground plane touches the ground plane within the 2-D convex hull of the contact points. These orientations correspond to those which align a face of the convex hull of the object parallel to the ground plane. These orientations are collected as a list of potential results.

In such orientations, the enclosing plane is initially introduced at those vertices furthest from the ground plane. Concavity verification is performed, and if unsuccessful, the plane is introduced

---

<sup>7</sup>This is an example of one of the minor system control changes made to achieve efficient recognition within the *dishes* superordinate category.

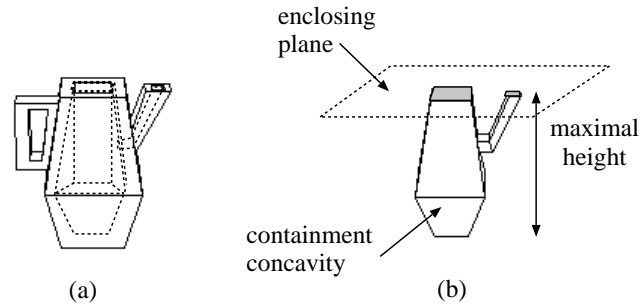


Figure 17: Introduction of Enclosing Plane

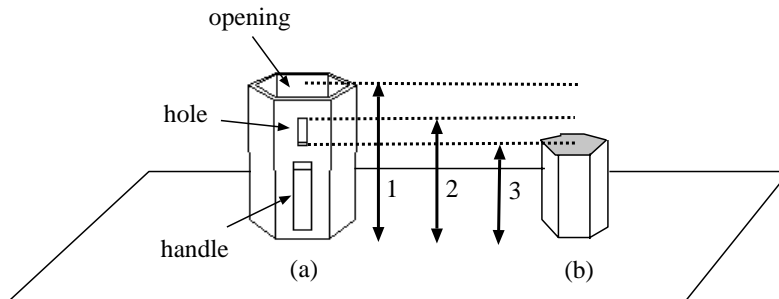


Figure 18: Levels of Enclosing Planes

at successive lower distances, as necessary. Verification is successful when the introduced plane intersects the concavity defining faces in such a way as to form a closed volume. Figure 17 depicts the situation where the enclosing plane introduced at the maximal distance encloses the concavity being tested. In this case, it is the entire inner concavity that forms the containment volume.

Tests at different levels would be applicable if the input shape had instead been the one shown in Figure 18. In Figure 18-(a), a mug is depicted in its normal upright position. Figure 18-(b) depicts the containment concavity of this mug. At the maximal level (level 1), the enclosing plane which is introduced parallel to the ground plane does not enclose a valid concavity due to the hole in the side of the mug. The next vertex of the object concavity is at level 2, so an enclosing plane is introduced at that level. The second plane (level 2) does not enclose the concavity either, because the lower portion of the hole is still present. The plane introduced at level 3 does enclose the maximal volume that can be formed when introducing a plane parallel to the ground plane. The primary goal of the procedure is therefore to enclose the maximum volume possible for each stable orientation. This volume can be used later in dimensional tests.

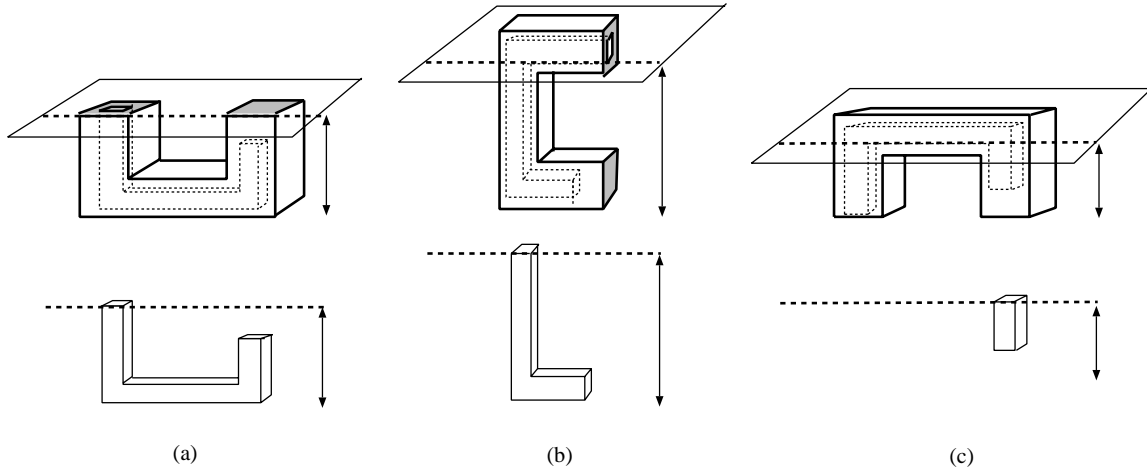


Figure 19: Valid Orientations for Enclosing Planes

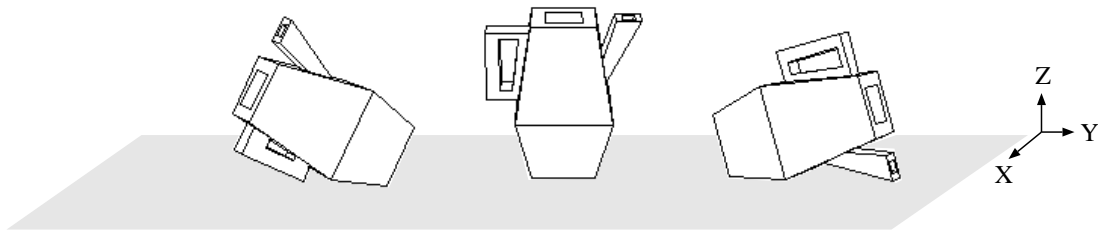


Figure 20: Hypothesized Stable Orientations

It is interesting to note that the system is able to find unusual orientations in which a valid concavity exists, such as those shown in Figure 19. The object depicted is a *U-shaped block* with a single opening hole running through approximately two thirds of the structure. The system identifies a maximal concavity for each of three different orientations of the *U-shaped block*. Figure 19-(a) shows the typical expected orientation and concavity description, whereas Figure 19-(b) and Figure 19-(c) show other valid concavity descriptions.

With the previous sample input object (Figure 16), a total of nine orientations pass verification with concavity descriptions. Three of these are pictured in Figure 20. Here, the middle orientation is the expected one. However, the object resting on either the handle or the spout also provides a valid concavity description. The six other orientations are derived from various ways in which the object can rest on the bottom base and/or combinations of corners on the handle or spout.

Once concavity verification and other pre-processing are completed, the system begins function-based interpretation by investigating the superordinate category *dishes*, according to the category definition tree of Figure 2. As we can see from the example objects in Figure 21, **all** of the potential results (all concavities, in all orientations) obtained up to this point will be considered. At any point, if a particular orientation and concavity passes for a given category, this has no effect on the orientation or concavities which will be checked for subsequent categories. All will be sequentially verified again.

For the given input shape, we ultimately wish to evaluate the category *Pitcher*, with the following definition:

$$\begin{aligned} \textit{Pitcher} ::= & \textit{Provides\_Pitcher\_Containment} \ \& \ \textit{Provides\_Stable\_Pitcher\_Containment} \ \& \\ & \textit{Provides\_Graspable\_Handle} \ \& \ \textit{Provides\_Pitcher\_Constriction} \end{aligned}$$

indicating that it is defined as a shape placed in an orientation that satisfies the conjunction (indicated by  $\&$ ) of these four properties.<sup>8</sup> Let us now trace through the category definition tree to see how and when the *Pitcher* node is encountered, noting that each basic level category will be evaluated until the entire functional plan has been traversed or until a function label cannot be associated to the object structure.

Since there are no function labels associated to the root node, processing passes to the first Subcategory node, named *Juice\_Glass*. This node does have associated functional requirements. The first function label encountered when traversing the tree is *Provides\_Juice\_Glass\_Containment*, which is realized by a sequence of KP invocations similar to those shown in Figure 22-(a). (Details on the primitive calls and parameters used will be described shortly.) The list of potential orientations derived during pre-processing and determined to pass self-stability and enclosure are now tested one at a time as valid containment volumes for the present category. The *dimensions* primitive is then invoked a series of times to test volume, area of enclosing planes, width and height of the containment concavity and the object overall. This functional plan will fail for the test shapes in

---

<sup>8</sup>The functional properties of each of the known categories in superordinate category *dishes* can be derived similarly from Figure 2.

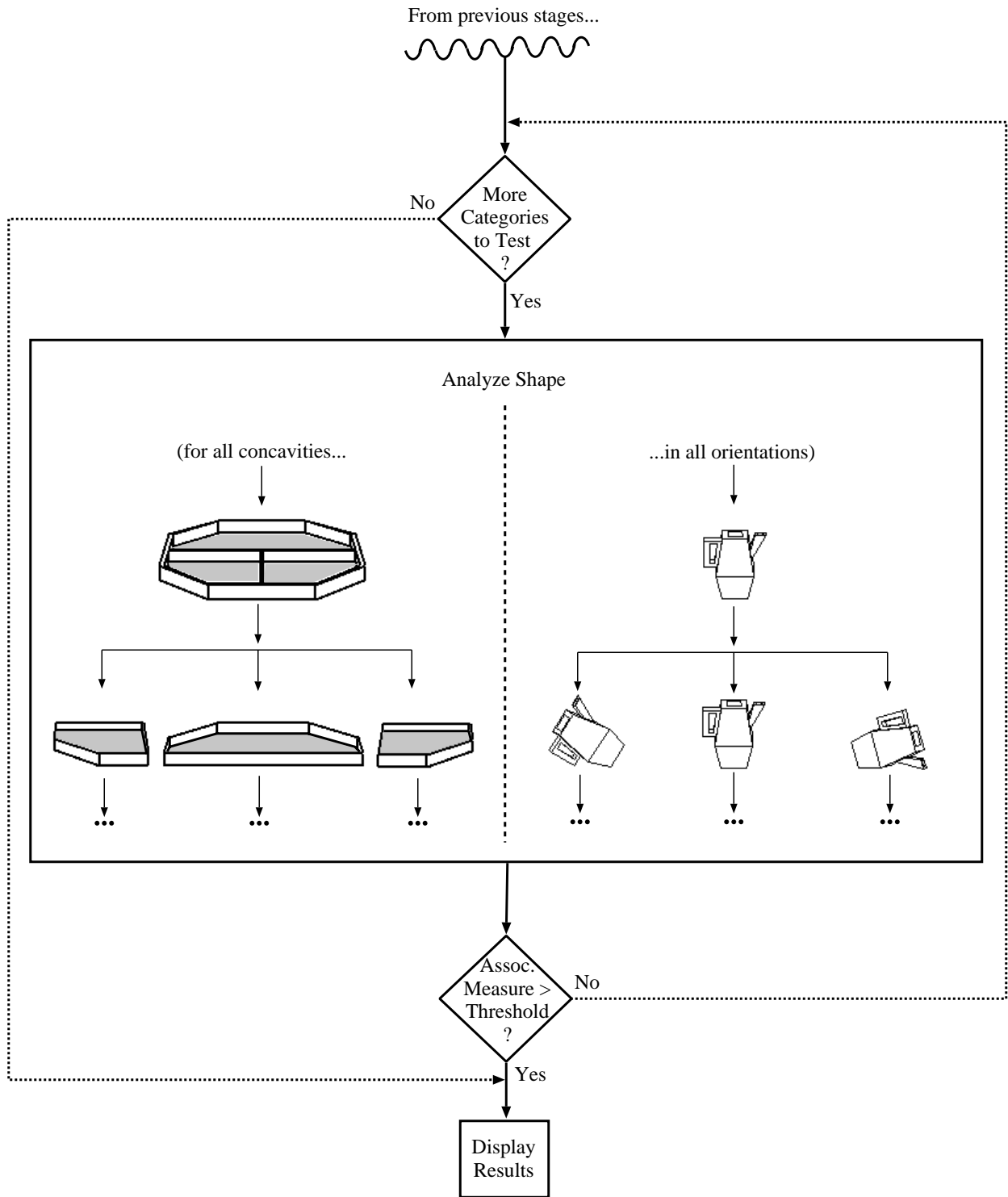























Figure 21: Flow of Control in Analyzing Shape

knowledge primitives for Provides_Pitcher_Containment	resulting evaluation measure at each step			
stability ( object_shape, given_orientation, self_stable ) /* checks for self stability of object with no external forces except gravity */		primitive=1.0 aggregate=1.0		primitive=1.0 aggregate=1.0
enclosure ( given_orientation, concavity_shape, enclosing_plane ) /* checks for a concavity in this orientation */		primitive=1.0 aggregate=1.0		primitive=1.0 aggregate=1.0
dimensions ( concavity_shape, 3d_volume, range_parameters ) /* checks for the appropriate volume of concavity */		primitive=0.95 aggregate=0.95		primitive=0.95 aggregate=0.95
dimensions ( enclosing_surface, area, range_parameters ) /* checks for the appropriate area of enclosing surfaces */		primitive=1.0 aggregate =0.95		primitive=1.0 aggregate=0.95
dimensions ( concavity_shape, width/depth, range_parameters ) /* check for the appropriate width/depth of the concavity */		primitive=1.0 aggregate=0.95		primitive=1.0 aggregate=0.95
dimensions ( concavity_shape, height, range_parameters ) /* checks for the concavity being within the appropriate height range */		primitive=0.95 aggregate=0.90		primitive=0.77 aggregate=0.73
dimensions ( object_shape, height, range_parameters ) /* checks for the surface being within the appropriate height range */		primitive=0.95 aggregate=0.86		primitive=0.95 aggregate=0.70
clearance (object_shape, 3d_volume_above_enclosing_area ) /* checks for appropriate clearance above the enclosing area */		primitive=1.0 aggregate=0.86		primitive=0.0 aggregate=0.0



(a)

knowledge primitives for Provides_Stable_Pitcher_Containment	resulting evaluation measure at each step		
stability ( concavity_shape, given_orientation, external_forces_at_concavity_cm ) /* checks for function stability of object w/ external forces applied to center of mass */		primitive=1.0 aggregate=0.86	

(b)

knowledge primitives for Provides_Graspable_Handle	resulting evaluation measure at each step		
dimensions ( handle_shape, height, range_parameters ) /* checks for handle being within the appropriate height range */		primitive=0.95 aggregate=0.82	
dimensions ( handle_shape, width/depth, range_parameters ) /* checks for the appropriate width/depth of the handle */		primitive=0.95 aggregate =0.78	
clearance ( handle_shape, 3d_volumes_around_handle ) /* checks for appropriate accessibility of handle surfaces */		primitive=1.0 aggregate =0.78	
proximity(handle_top, center_of_mass, range_parameters ) /* checks for stability of handle, relative to the center of mass */		primitive=1.0 aggregate =0.78	

(c)

knowledge primitives for Provides_Pitcher_Constriction	resulting evaluation measure at each step		
relative_orient( pour_vector, handle_vector, range_parameters ) /* checks for appropriate orientation of pour and handle_midpoint vectors */		primitive=0.95 aggregate=0.74	
clearance ( pour_area, 3d_volume_around_pour_area ) /* checks for appropriate accessibility of pouring area */		primitive=1.0 aggregate =0.74	

(d)

Figure 22: Evaluation of Sample Input Object

Figure 20 due to the *dimensions* test for volume. A pitcher is expected to hold a larger volume of liquid than a juice glass. Once a function label fails, evaluation is discontinued for the current subcategory branch. Therefore the function label *Provides\_Juice\_Glass\_Stable\_Containment* under the *Juice\_Glass* node will not be evaluated. Processing continues at the next Subcategory node, *Cup/Glass*.

Much the same as the *Juice\_Glass* node, the first function label encountered is *Provides\_Cup/Glass\_Containment*. This function label differs from the *Provides\_Juice\_Glass\_Containment* function label in the parameter values passed to the *dimensions* primitive. Although the dimensions of this node are somewhat larger than the previous one, the object will again fail dimension tests, and processing will again be discontinued along the current branch.

Next, the category node *Pitcher* is encountered. Figure 22 provides a detailed representation of the knowledge primitive invocations which must be confirmed for each functional requirement of this category. Each of the calls is represented in C-like pseudo-code. The parameter lists have been simplified for easier comprehension, with terms such as “object\_shape” or “handle\_shape” used to indicate that individual face lists or specific lists of surfaces (portions of the shape) are being passed for analysis. For example, the parameter “handle\_shape” represents those surfaces of the object found to function as a handle.

Note that the same primitive (*dimensions*, for example), may have differently structured, though similarly typed, parameters in different cases. For example, the *dimensions* primitive always receives a face list and the type of dimensions test to be performed. In the case of the *dimensions* invocation for *3-D volume*, the entire object’s concavity face list description is passed for evaluation. In the case of the *dimensions* invocation for *area*, a face list with a single face is passed.

The parameter *range\_parameters* is an abbreviation for the values described previously in Figure 7. For example, for the first *dimensions* call, the actual values passed for *range\_parameters* will be (0.0007, 0.001, 0.003, 0.005) for the volume of a pitcher (in cubic meters). While the endpoint values may seem extreme, recall that the evaluation measure returned will be low for these. A pitcher which has a smaller than average volume is also inclined to receive low evaluation measure

values for the remaining dimensional tests and is likely to be discarded for further processing. The actual range parameters are constant for the various categories, however, each input object will receive an evaluation measure according to how well its 3-D shape meets these ranges for each primitive invocation involving them. The columns on the right of Figure 22 show for each functional requirement both the individual primitive evaluation and the current aggregate value of these measures. The *functionality threshold* is applied to the final aggregate (association) measure of the figure.

Continuing with the example input object, for category *Pitcher*, the first function label encountered is *Provides\_Pitcher\_Containment*. This time, after testing self-stable orientations with concavities for the present category, the *dimensions* primitive calls will also pass. Processing then continues with a call to *clearance* to confirm the accessibility of the concavity for filling. For the given example, the only remaining result is the proper containment concavity and orientation (see Figure 22-(a)). The rightmost orientation fails because there is no clearance above the enclosing plane for filling the pitcher.

The stability primitive is invoked once again to confirm *Provides\_Stable\_Pitcher\_Containment*, as shown in Figure 22-(b). This invocation tests to confirm that the object is *function-stable* in the given orientation. Function-stability entails applying external forces at points associated with the functionality of the object. The pressure of the liquid can be approximated by a point force exerted in the direction of gravity from the center of mass of the containment concavity. This assumes that the containment concavity is expected to be able to hold liquid to its maximum volume. Objects are flagged as potentially being unstable, but are not eliminated from processing, if found not to pass function-stability. This is based on the idea that at this stage, the exact content or level of the concavity may be unknown. The objects are still considered to be potential dishes, although with the ability to contain possibly smaller quantities of food or liquids to maintain function-stability.

The next function label to be evaluated is *Provides\_Graspable\_Handle*, which must confirm that there is a portion of the object structure that can be used as a handle. The list of primitives used to realize this function label is shown in Figure 22-(c). Using the edges of the enclosing face(s),

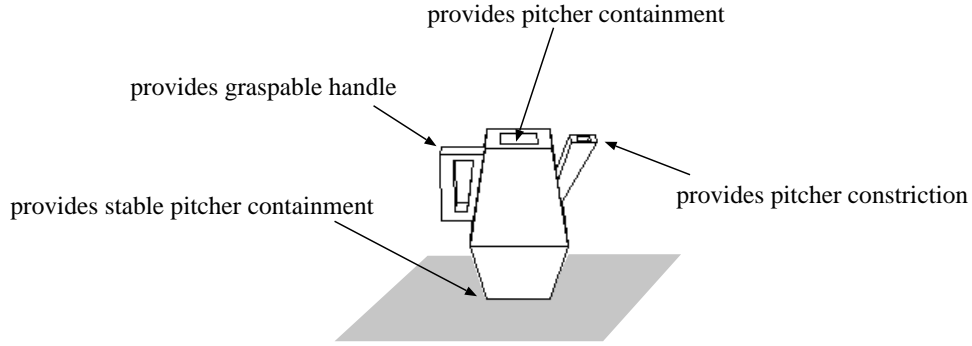


Figure 23: Recognition: Pitcher Subcategory  $\rightarrow$  Measure 0.74

surfaces are located outside the containment area and tested with *dimensions* to ensure they are within the proper size range to be grasped as a handle. Next, *clearance* is used to ensure the handle is accessible to be grasped.

Finally, the function label *Provides\_Pitcher\_Constriction* is encountered. At this point, using the primitive calls in Figure 22-(d), the system attempts to determine the areas of the containment volume from which liquid can flow. A call to the KP *clearance* ensures that the area is clear of obstructions, and the fluid will flow freely from the shape.

All potential results are consolidated at this point in the evaluation. An association measure is accrued which reflects how well each of the functional requirements has been met. The only surviving potential result is depicted in Figure 23. The categories *Bowl*, *Plate* and *Pot/Pan* will also be traversed, but will fail due to the fact that the input shape cannot fulfill the functional requirements defined for these categories. Final recognition is performed by choosing the (sub)category node with the highest association measure above the set *functionality threshold*. The cumulative association measure returned by the system reflects the system's analysis of how the shape of the object best conforms to the functional requirements of a *Pitcher*.

## 6 Evaluation of System Competence

Several factors must be considered when evaluating the competence of a function-based recognition system. Since the goal is categorization based on what a shape can “function as” rather than what

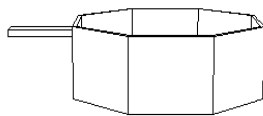


Figure 24: Interpreting Object Functionality

it is “supposed to be,” it is entirely reasonable for a shape to belong to more than one category. Consider the shape in Figure 24. In this case, given the option of cup/glass, bowl, plate, pot/pan, or pitcher, a person would immediately classify the object as “supposed to be” in the pot/pan category. However, based on functionality alone the shape could be classified as belonging to the bowl category as well. The handle would not be considered a significant deterrent for the object to “function as” a large-sized mixing bowl.

Accordingly, one measure of the competence of the system can be determined from the number of times the system and the person agree on the possible functionalities of an input shape. The number of times these categorizations match can be termed “is / is not” competence. For example, an object considered a pot/pan by a person and categorized as both a pot/pan and a bowl by the system would be considered a match. A non-match would occur if the system had instead categorized the object as just a bowl.

A second measure of competence is based on looking at how often the highest “ranking” categorization agrees with human preference. It should be noted that some human bias is inherent in this comparison, due to the fact that human interpretation is based on a larger domain of competence, often using contextual cues for what a shape is meant to be. However, by comparison of the highest ranking categorizations for a given input shape, we can obtain a better idea of the true breadth of a pure function-based system.

## 6.1 Is/Is Not Competence

To begin the evaluation of system competence, a database of 206 sample input shapes was created. A representative subset of these shapes has been shown throughout the text, however more

Table 1: Evaluation of Is/Is Not Competence in Function-Based Recognition.

	HUMAN INTERPRETATION	SYSTEM AGREEMENT WITH HUMAN	PERCENT AGREEMENT
CUP/GLASS (is/is not)	35 / 171	35 / 171	100 / 100
BOWL (is/is not)	44 / 162	39 / 148	89 / 91
PLATE (is/is not)	13 / 193	13 / 191	100 / 99
POT/PAN (is/is not)	26 / 180	25 / 180	96 / 100
PITCHER (is/is not)	10 / 196	10 / 191	100 / 97
NOT KNOWN	78 / 128	75 / 124	96 / 97

comprehensive sets are provided in Figure 25 and Figure 26.

Each shape was categorized by its author as belonging to one or more of the five categories known to the system. If more than one category was applicable, then the “most appropriate” or “first ranked” category was also identified. All of the shapes were then analyzed by the system. During this first evaluation, no indexing strategy was used, meaning that each shape was evaluated to determine if it could meet the functional requirements of each of the five categories. Final association measures were gathered, and those above the *functionality threshold* of 0.4 were considered sufficient for category membership. When a shape was categorized into more than one category, the category with the highest association measure was selected as the one which the system determined the shape could best function as.

Statistics were then gathered on how often the system and the person agreed that the functional requirements of a particular category were met. The results of this *is / is not competence* analysis are summarized in Table 1. The numbers in the **Human Interpretation** column show the number of objects the person thought *could possibly/could not possibly* function as the category associated to the row. The pair of numbers sum to the total number of objects. The numbers in the **System Agreement with Human** column depict the number of those objects the system agreed with. Therefore, each pair of numbers in this column is less than (when a disagreement occurred) or equal (when no disagreement occurred) to the numbers in the **Human Interpretation** column.

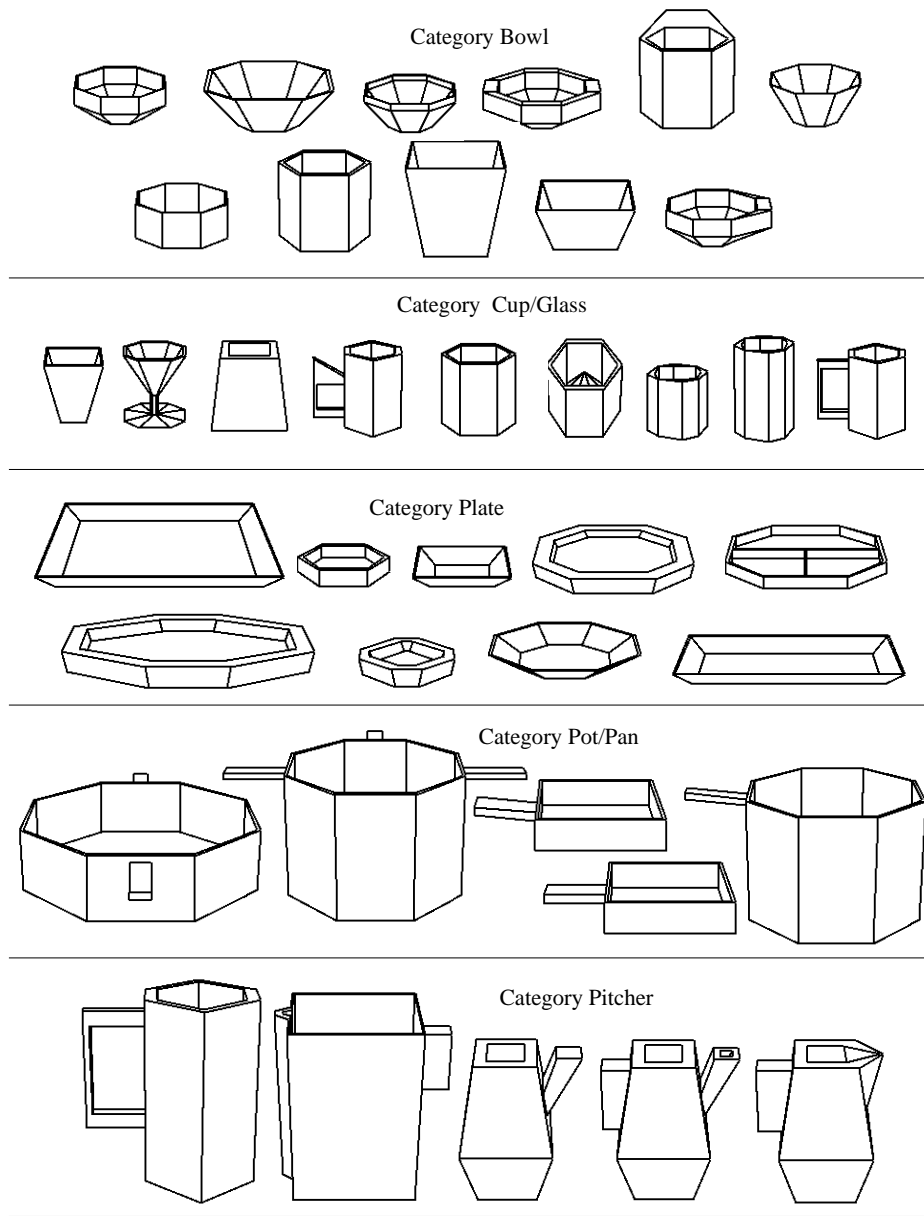


Figure 25: Representative Set of Valid Objects

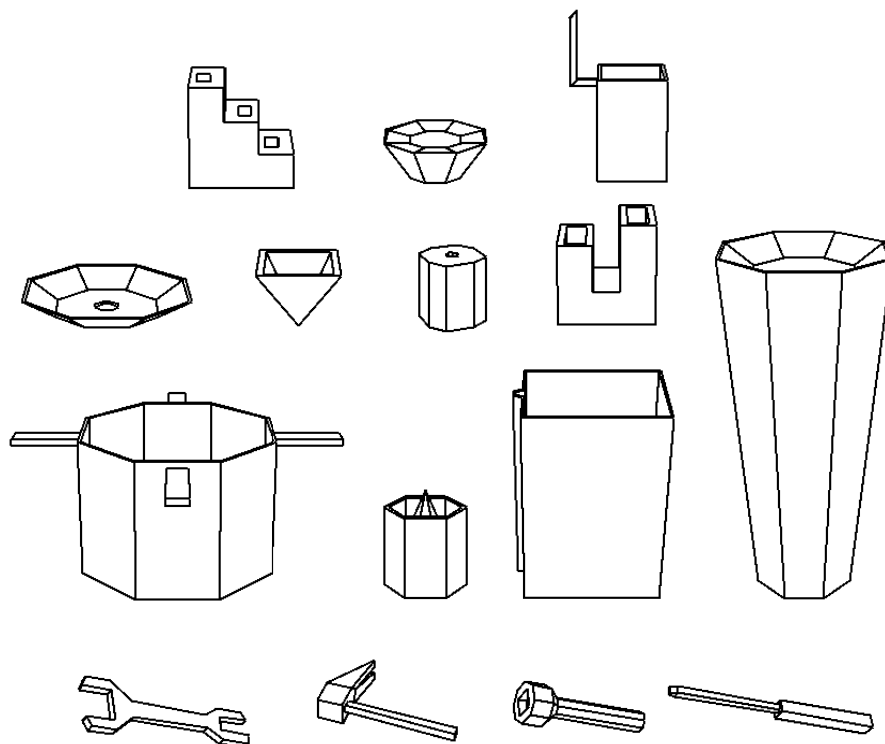


Figure 26: Representative Set of “Not Known” Objects

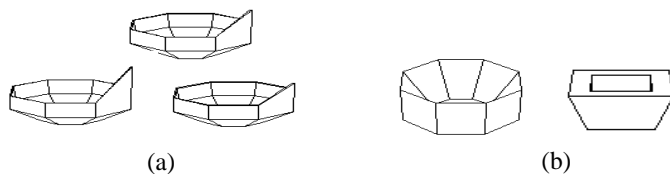


Figure 27: Examples of Is/Is Not Discrepancies

The largest discrepancy was the 89% agreement in category bowl. The discrepancies for this entry are depicted in Figure 27.

The objects in 27-(a) were labeled by their designer as bowls. However, the system found them to be “not known” based on the obstruction of the raised side panel. Similarly, the system found that the objects in 27-(b) were not adequate to function as bowls due to the conflict between ideal height and volume dimensions.<sup>9</sup>

---

<sup>9</sup>Unless otherwise stated, objects depicted within a figure are shown at the same relative scale within that figure.

Table 2: Evaluation of Ranking Competence in Function-Based Recognition

System Interpretation	Human Interpretation					
	cup/glass	bowl	plate	pot/pan	pitcher	not known
cup/glass	35					
bowl		38		4		1
plate		1	13			1
pot/pan				22		
pitcher					10	1
not known		5				75

## 6.2 Ranking Competence

The next level of analysis is based on the first-ranked interpretation of each shape. This is called the *ranking competence* of the system, and demonstrates how well the system makes appropriate distinctions between categories. The results of this comparison are summarized in Table 2. The rows represent the system interpretation that resulted in the highest association measure. Similarly, the columns represent a person’s first-ranked interpretation. If the system’s relative ranking of categories was totally independent of the human ranking of the categories, then the numbers in a given row should be evenly distributed across the columns. If the person and system interpretation were perfectly correlated, then the only non-zero entries would be on the diagonal. Over the entire database of shapes, the system’s primary interpretation of the shapes agreed with the person’s primary interpretation 94% of the time. Figures 25 and 26 show a representative set of those shapes on which agreement occurred over the first ranked category.

Some of the disagreements shown in Table 2 are primarily due to a person’s difficulty in considering a shape purely on function-based grounds. Representative examples of these types of discrepancies are displayed in Figure 28. The objects in 28-(a) were equally recognized as bowls and pots. This is because dimensional constraints of these two categories are so close, and the existence of the handle does not limit the ability to function as a bowl. Those shapes shown in 28-(b) had a higher measure as bowls because the existence of additional obstruction on the outside of a pot once one handle is found lowers its overall measure. This reflects the limited ability to

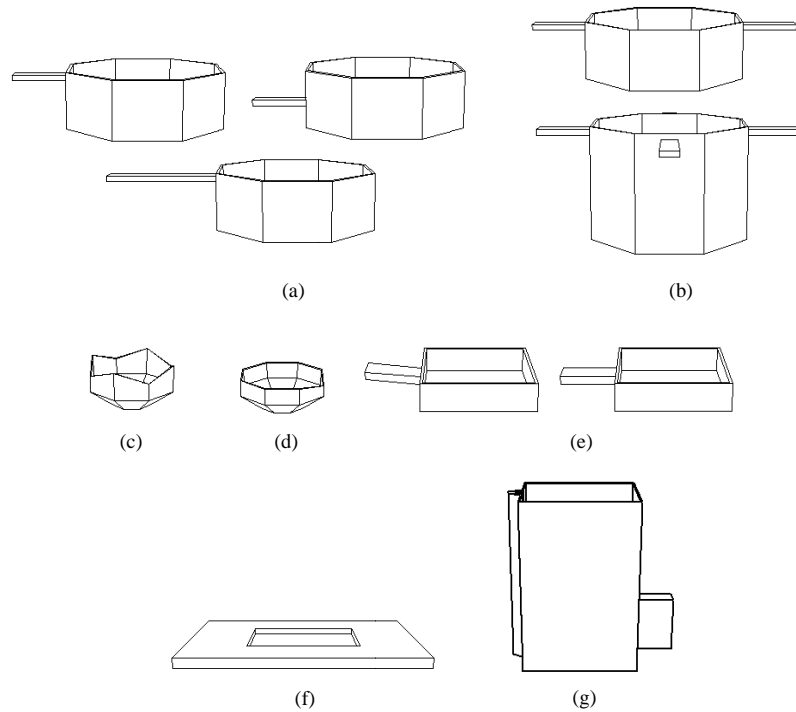


Figure 28: Disagreement between System and Human Interpretation of Objects

grasp the pot by the handle and pour from another side. On the other hand, the object in 28-(c) was labeled as “not known,” but the system was able to classify it as a bowl, because the side panel height differences were not determined to significantly impair the ability to function as a bowl. The object in 28-(d) was labeled a bowl by its designer. However, the system found that the dipping edge constrained the height and volume dimensions of the object such that it was more suitably used as a deep plate. The objects in 28-(e) were labeled as pans by a person, but were classified as bowls by the system, because their height and volume dimensions better matched ideal values for a bowl. Similarly, the object in (f) tended to fit the dimensional constraints of a plate, although its designer labeled it “not known.” Finally, the object in 28-(g) was classified (with a low measure) by the system as a pitcher, although its designer believed the defined handle was inadequate for this category.

Figure 29 provides an clearer picture of the way in which a set of similar shapes can be recognized by the system across a range of categories, due entirely to dimensional constraint analysis. These

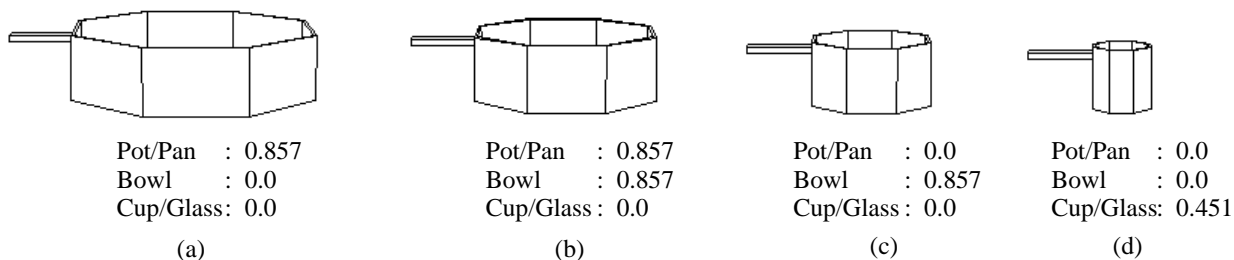


Figure 29: Comparison of Final Association Measures

objects show the origin of the types of discrepancies in Figure 28-(a) and (b). In this example, the shape shown in 29-(a) is most ideally suited to the dimensional constraints of a pot. As the height, width and volume values are adjusted as in 29-(b), a tie develops and the shape is determined to be able to function equally as a pot or large, handled mixing bowl. In 29-(c), the dimensions are outside the range of a pot/pan, but still well within the ideal values of a mixing bowl, so the shape is categorized accordingly. Finally, in 29-(d), the dimensions do not fall within the pot/pan or bowl category, but do match the ideal values of a cup/glass. The lower measure of 0.451 for this category reflects the presence of the handle. The dimensions of the handle are inadequate to categorize the shape as a functional mug, so the overall measure has been lowered to indicate that the shape is adequate to function as a cup/glass, with obstruction on one side which limits accessibility for grasping and drinking in this area.

The results of the *ranking competence* analysis show that no glaring errors in object categorization occurred. Discrepancies tended to be the result of the system applying pure functionality criteria for recognition, without considering possible external cues sometimes used by people. The analysis even suggests that human preference, due to its reliance on such cues, has a more limited ability to recognize based purely on function.

### 6.3 Evaluation of an Indexing Strategy

In addition to evaluating **GRUFF-3**'s recognition *competence* (how frequently it correctly recognized objects), we also consider the *efficiency* of such a recognition system (how much processing is

Table 3: Evaluation of the Effectiveness of the Indexing Strategy

	Number Of # Objects	# Of Possible Evaluations	Evaluations Actually Made	Evaluations Saved	Indexing Efficiency
No Threshold	206	1030	869	161	16%
0.40 Threshold	206	1030	603	427	41%

required). In the above competence analysis, each shape was processed against the functional requirements of each known category. However, for greater efficiency (and therefore less processing), a different mode of operation is actually used in practical applications, by incorporating the *indexing strategy* discussed in Section 4.2.2.

The number of category evaluations saved by the indexing strategy, as compared to not having an indexing strategy, are summarized in Table 3. With 206 shapes tested and five basic level categories known to the system, a total of 1030 category evaluations would be done if the system used no indexing strategy.

With the simple indexing strategy described previously, a total of 161 potential category evaluations are immediately eliminated because the key value list for the shape has no matches to the key value ranges for the superordinate category as a whole and also for some basic level categories. With recognition processing ending as soon as the shape is found to fulfill the functional requirements of some category with a final association measure of 0.4 or better (the *functionality threshold*), an additional 266 category evaluations are avoided. Thus 427 of the 1030 total possible category evaluations are eliminated by the indexing strategy.

There exists the possibility that, with the indexing and cutoff at the first plausible association measure, the system could recognize a shape as one category when in fact there is some other category which would result in a higher association measure. This in fact occurred in 7 cases. These errors are representative of the recognized limitations of this efficiency strategy.

## 6.4 Implementation Details

The **GRUFF-3** system is written in C. The size of the executable program is approximately 1 MB. The shape descriptions analyzed here have approximately 20-40 faces, 50-70 vertices and 80-100 edges, leading to average space requirements of about 10 KB per object. Total processing time on a SPARC station IPX ranges from under a minute to several minutes, depending on the complexity of the object. The longest processing times occur for those objects with the greatest number of testable orientations or concavities. All of the shape models discussed are available to interested researchers through anonymous ftp to figment.csee.usf.edu under the directory pub/errors\_stuff/Objects.

## 7 Discussion

The results of our research show that function-based modeling can be used to create a powerful object recognition system. The recognition is generic in the sense that entire categories of objects are captured, rather than specific exemplars. We have completed an evaluation of a system based on these concepts. The system takes an uninterpreted 3-D shape as its input and reasons to determine if it belongs to the superordinate category *dishes*, and if so, into which (sub)category it falls. The system has analyzed 206 input shapes and the results largely agree with human interpretation of the shapes. Many of the discrepancies between human and system categorization of shapes are due to the system finding novel uses for objects that the human did not recognize. This is a reflection of the fact that the system uses a purely function-based definition of the category.

For the current expansion of **GRUFF**, in order to realize the functional representation of *dishes*, the one knowledge primitive added to the system was *enclosure*. The diversity of object shapes which can now be recognized with these six knowledge primitives supports the thesis that reasoning about functionality provides an effective means to achieve generic recognition capabilities.

Several directions of further research are planned. One is to continue expanding and testing the system's domain of competence to include different superordinate categories, such as *hand tools*. This expansion includes the extension of the function-based approach to objects with articulated

parts. Furthermore, the extension of the function-based approach to analysis of more than just static shape data would also be very important. For example, the combination of visual sensing for 3-D shape and tactile sensing for properties such as surface rigidity and roughness should prove very powerful. Continuing research is also planned using laser range finder data as input for partial 3-D shape descriptions (Hoover<sup>(16,17)</sup>).

## References

- [1] J. Aloimonos. Purposive and qualitative action vision. In *DARPA Image Understanding Workshop*, pages 816–828, 1990.
- [2] K. Ikeuchi and M. Hebert. Task-oriented vision. In *DARPA Image Understanding Workshop*, pages 497–507, 1990.
- [3] E. Rosch, C.B. Mervis, W. Gray, D. Johnson, and P. Boyes-Braem. Basic objects in natural categories. *Cognitive Psychology*, 8:382–439, 1976.
- [4] M. Minsky. *The Society of Mind*. Simon and Shuster, New York, 1985.
- [5] T.O. Binford. Survey of model-based image analysis systems. *International Journal of Robotics Research*, 1:18–64, 1982.
- [6] P. Winston, T. Binford, B. Katz, and M. Lowry. Learning physical descriptions from functional definitions, examples, and precedents. *AAAI*, pages 433–439, 1983.
- [7] M. Brady, P.E. Agre, D.J. Braunegg, and J.H. Connell. The mechanics mate. In T. O’Shea, editor, *Advances in Artificial Intelligence: European Conference of Artificial Intelligence*, pages 79–94. Elsevier, 1985.
- [8] J.H. Connell and M. Brady. Generating and generalizing models of visual objects. *Artificial Intelligence*, 31:159–183, 1987.
- [9] M. DiManzo, E. Trucco, F. Giunchiglia, and F. Ricci. FUR: Understanding FUnctional Reasoning. *International Journal of Intelligent Systems*, 4:431–457, 1989.
- [10] M. Brand. Vision systems that see in terms of function. In *Working Notes on Reasoning About Function*, pages 17–22, 1993.
- [11] K. Kise, H. Hattori, T. Kitahashi, and K. Fukunaga. Representing and recognizing simple hand-tools based on their functions. In *Asian Conference on Computer Vision*, pages 656–659, November 1993.
- [12] E. Rivlin, A. Rosenfeld, and D. Perlis. Recognition of object functionality in goal-directed robotics. In *Working Notes on Reasoning About Function*, pages 126–130, 1993.

- [13] L. Stark and K.W. Bowyer. Generic recognition through qualitative reasoning about 3-D shape and object function. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 251–256, 1991.
- [14] L. Stark and K.W. Bowyer. Function-based recognition for multiple object categories. *Image Understanding*, 59(1):1–21, January 1994.
- [15] M. Sutton, L. Stark, and K.W. Bowyer. ‘We do dishes, but we don’t do windows,’ Function-based modeling and recognition of rigid objects. In *Proceedings of SPIE Workshop on Intelligent Robots and Computer Vision XI: Algorithms, Techniques and Active Vision*, pages 132–142, 1992.
- [16] A. Hoover, D.B. Goldgof, and K.W. Bowyer. Extracting known and inferred shape information from a single view. In *Proceedings of SPIE Workshop on Sensor Fusion V*, pages 2–13, 1992.
- [17] A. Hoover, D.B. Goldgof, and K.W. Bowyer. Building a b-rep from a segmented range image. In *Proceedings of IEEE Second CAD-Based Vision Workshop*, pages 74–81, February 1994.
- [18] L. Stark, A. Hoover, D.B. Goldgof, and K.W. Bowyer. Function-based recognition from incomplete knowledge of shape. In *Proceedings of IEEE Workshop on Qualitative Vision*, pages 11–22, 1993.
- [19] L. Stark and K.W. Bowyer. Achieving generalized object recognition through reasoning about association of function to structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1097–1104, 1991.
- [20] M. Sutton, L. Stark, and K.W. Bowyer. Function-based generic recognition for multiple object categories. In A. Jain and P. Flynn, editors, *3-D Object Recognition Systems*, pages 447–470. Elsevier Science Publishers, 1992.
- [21] P.P. Bonissone and K.S. Decker. Selecting uncertainty calculi and granularity: An experiment in trading-off precision and complexity. In L. Kanal and J. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 217–247. North-Holland Publishing Company, 1985.
- [22] L. Stark, L.O. Hall, and K.W. Bowyer. An investigation of methods of combining functional evidence for 3-D object recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(3):573–594, 1993.
- [23] H. Edelsbrunner, J. O’Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Computing*, 15:341–363, 1986.