

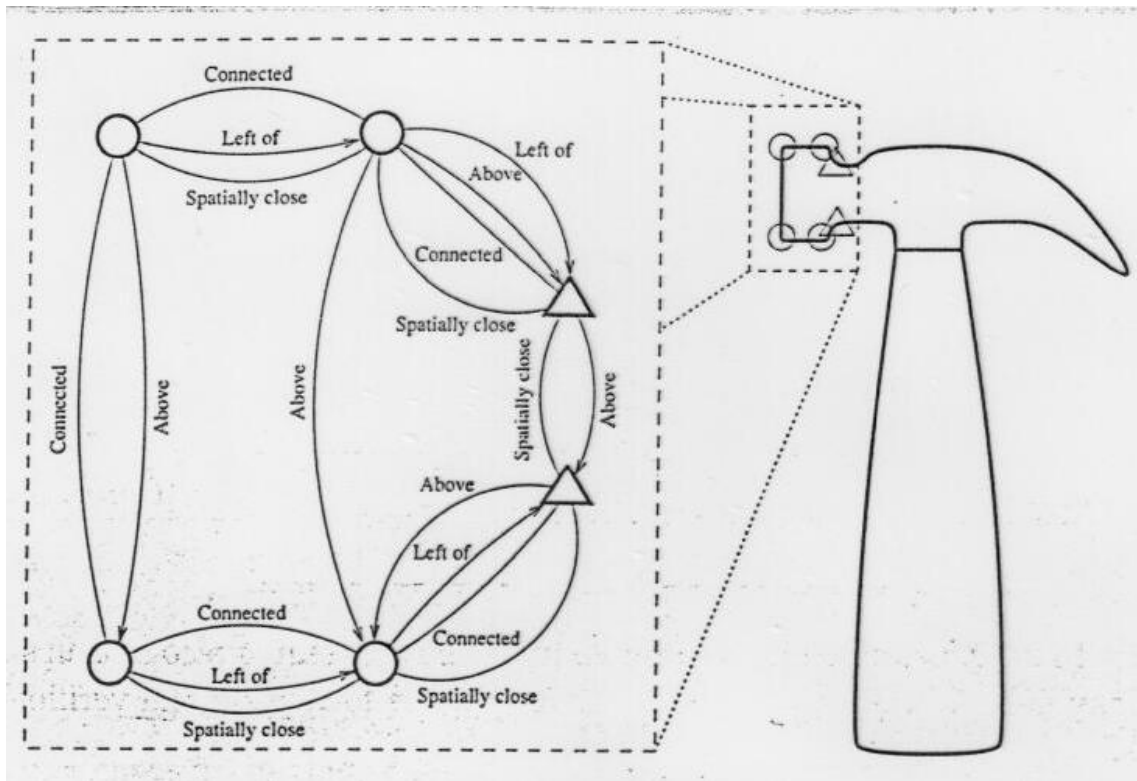
## Lecture notes: Object recognition

Another classic problem in computer vision is to recognize an object from an image of the object. The problem generally has two parts:

- **recognition** - What is the object?
- **pose** - What is the location and orientation of the object?

Template matching is not scalable to a large number of objects and orientations. It is simply not possible to directly compare all possible views of all possible objects.

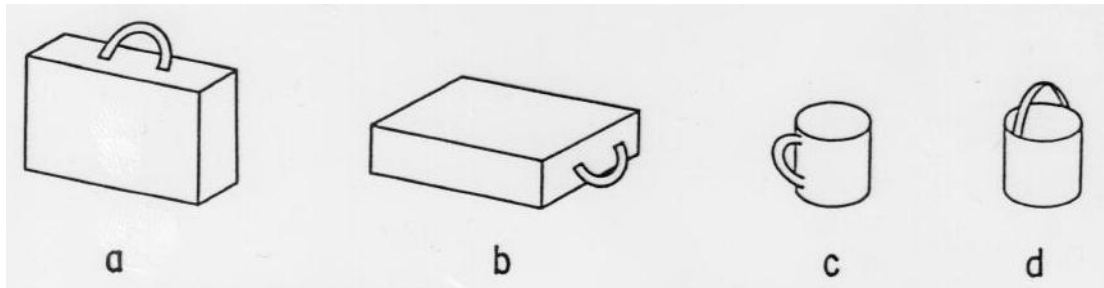
Instead, the general approach involves feature matching, where an object is modeled by a set of features and compared to features seen in an image. For example, a hammer has key components including a handle, blunt striker, and nail remover. The following figure demonstrates an image of a hammer and a possible model. The goal of an object recognition algorithm is to recognize the parts of the model in the image, and then match the segmented parts to the best-matching model.



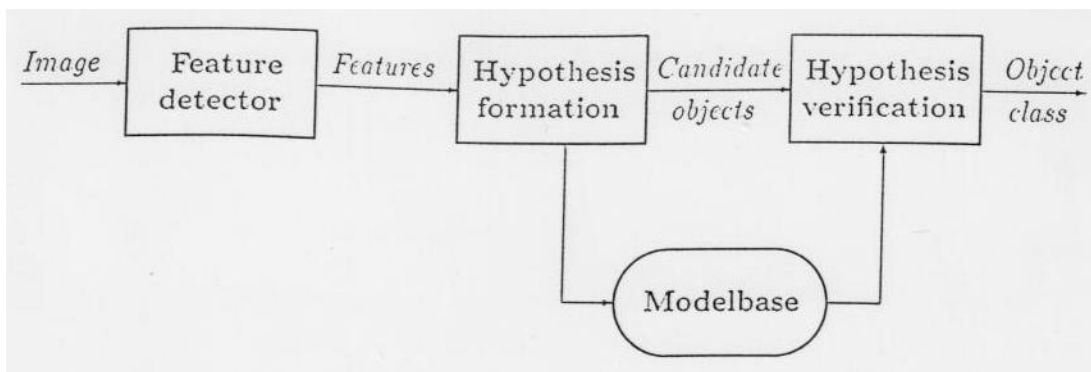
Parts of the problem are:

- **bottom up** - Image properties drive processing.
- **top down** - Goal or expected object models drive processing.

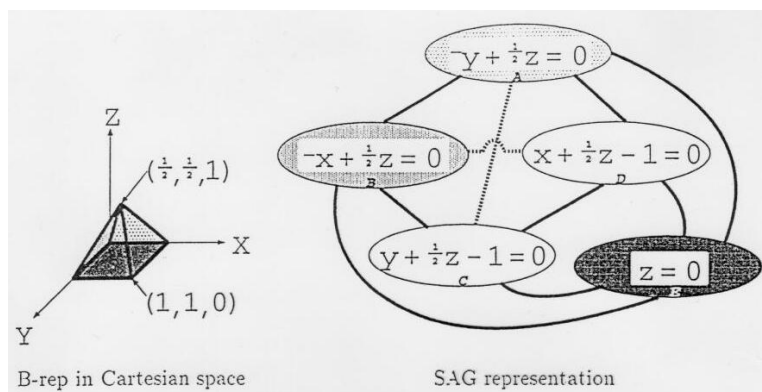
It is important to note that the arrangement of components is just as important as the shapes of the components. The following example demonstrates that changing the orientation or connection of even two components can change the object. In the first example, a suitcase becomes a drawer, and in the second case, a cup becomes a pail.



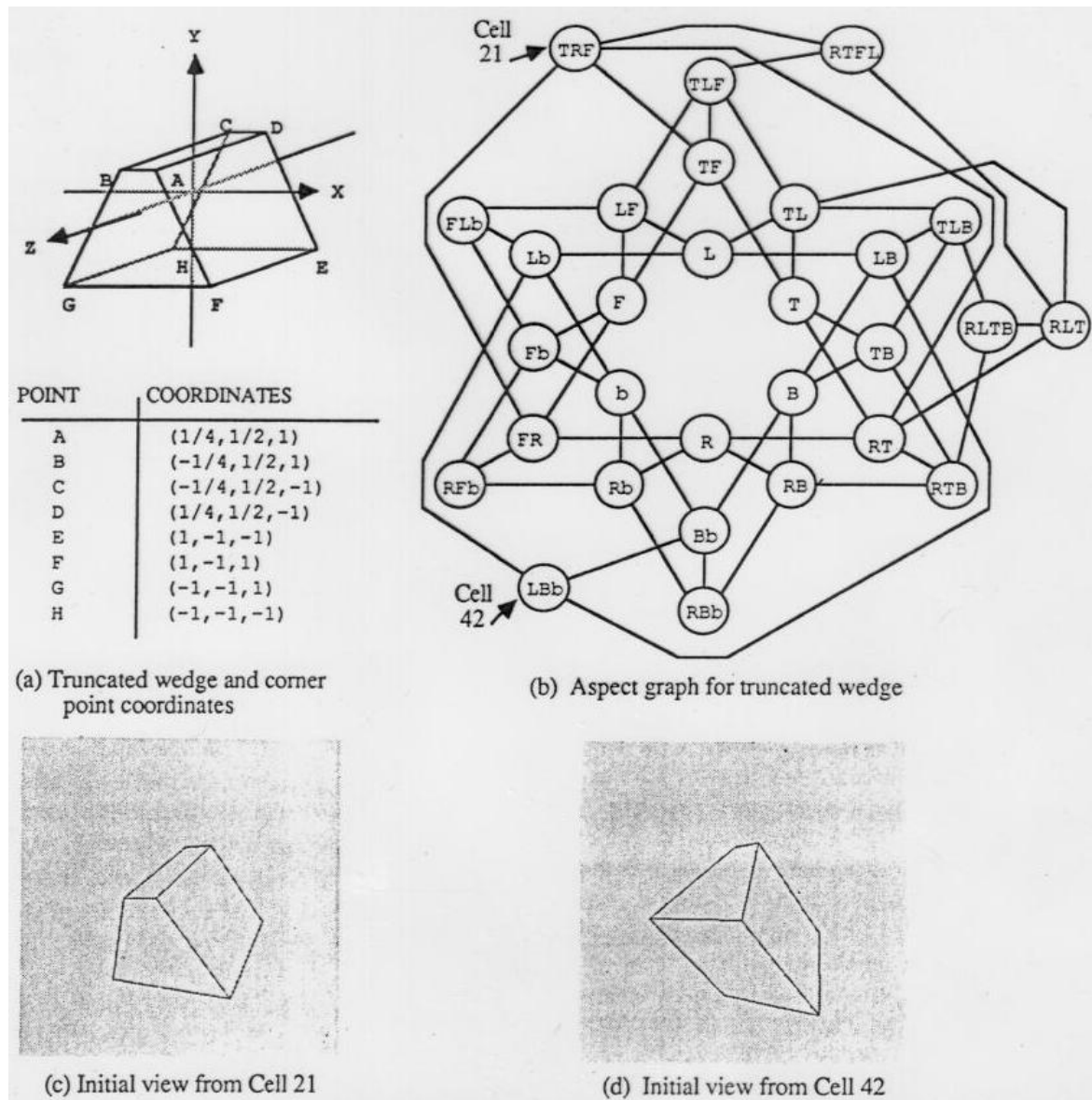
The general approach follows the following graph. First, a feature detector is run on the image to recognize components. Second, component hypotheses are compared against a database of object models. Third, the hypothesized models are oriented according to a given pose and projected into the image to see how well they match. Finally, the best matching hypothesis yields recognition.



A key property of good features is their **invariance** to orientation and position. Thus, the length of a line or its specific orientation varies depending on pose, but the angle between two lines rotated in 2D is invariant, and the connection of lines at a corner is invariant to all rotations. Object recognition strives to use invariant features so as to reduce the complexity of the model matching. Previously, we looked at a b-rep and saw that it models surfaces and topology primitives. These are good invariant features.

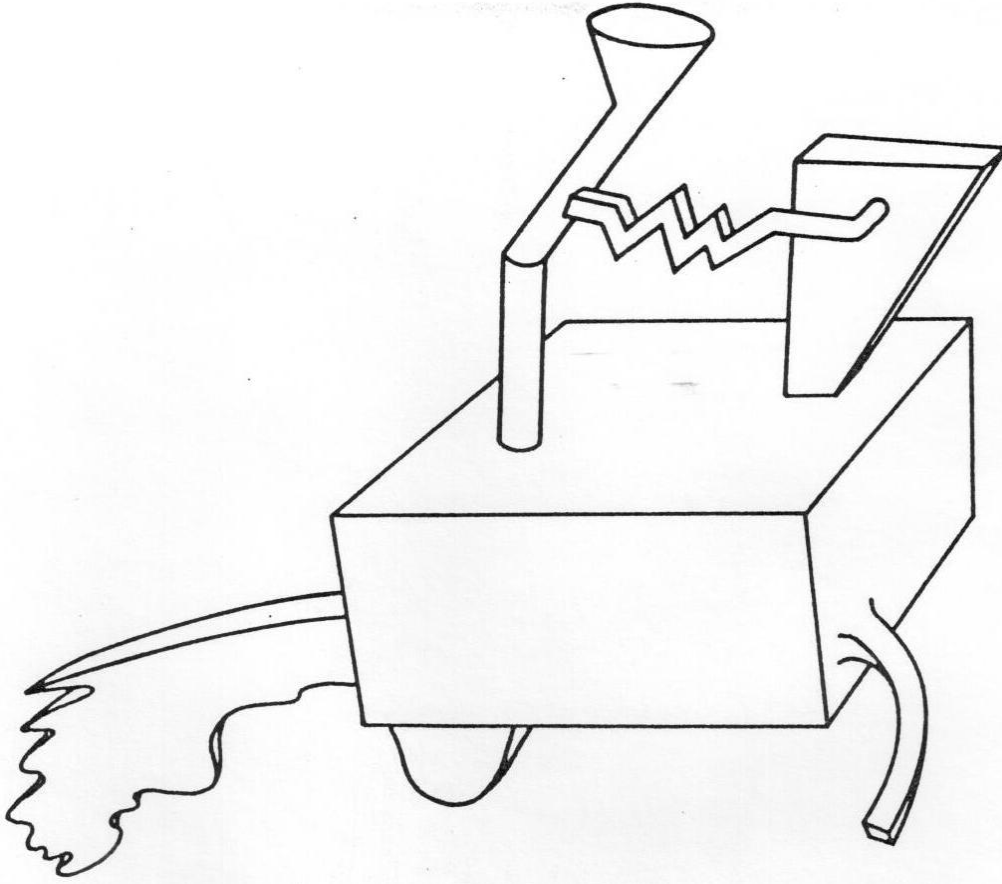


An **aspect graph** is an alternative to an object-centered model like a b-rep. The aspect graph is viewpoint-centered, describing what surfaces or features are visible from different portions of a view-sphere surrounding the object. The following shows an example. An aspect graph partially solves the pose estimation problem at the same time it solves the recognition problem.



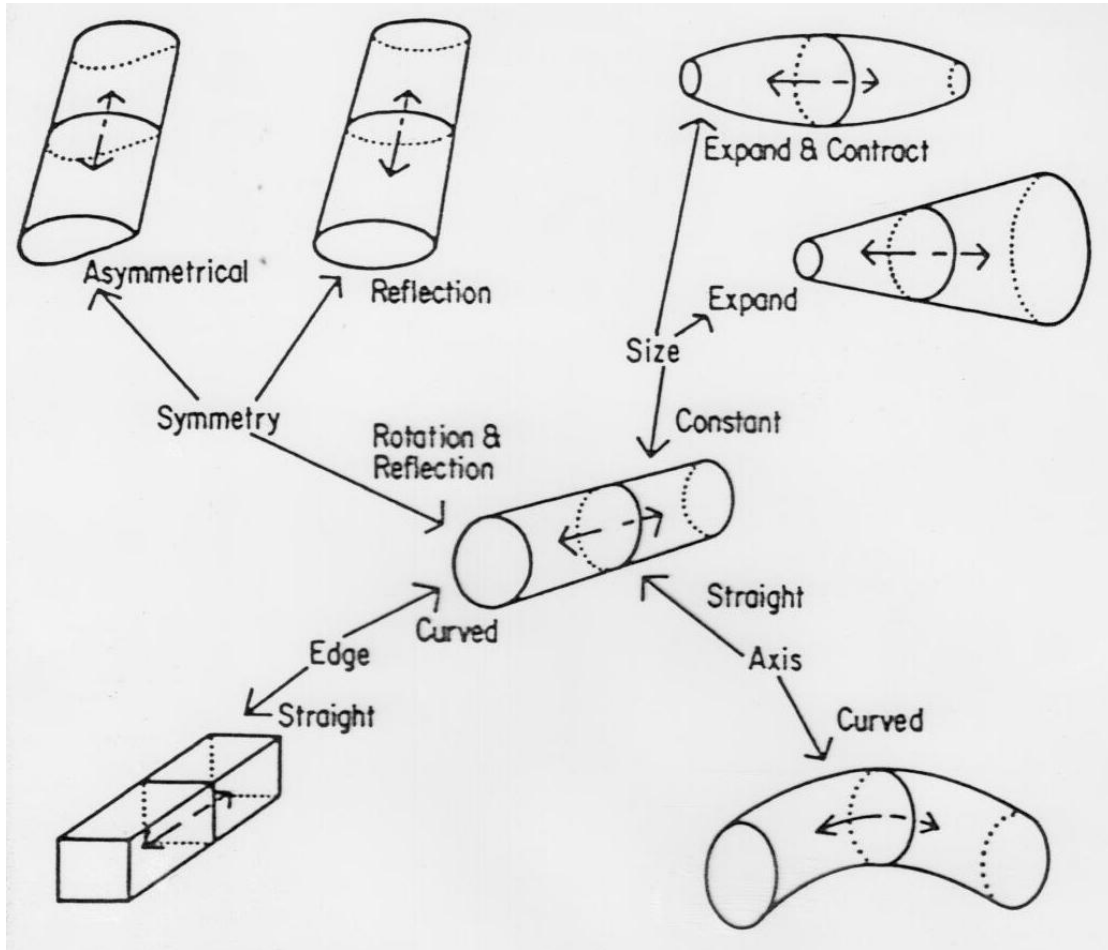
Aspect graphs show the concept of an **accidental viewpoint**. At precise points, features can reduce to singular points or lines. In general these angles are considered accidental and especially challenging for object recognition, just like for when a person views an object from an especially bad angle.

**Component-based recognition** takes the approach that features define the parts of an object. The most famous component-based recognition scheme was developed by Biederman and is called geometric ions, or **geons** for short. The following shows a classic example object. Although it is nonsense, people strive to make sense of the parts and can easily agree on the segmentation (just not the identification of the object).

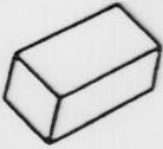













What is this?

Starting with a cylindrical piece, Biederman proposes the following four properties of geon parts: symmetry along axis, size expansion, edge curvature and body curvature:



The following two tables show some of the possible parts:

<u>Geon</u>	<u>Edge</u> Straight S Curved C	<u>Symmetry</u> Rot & Ref ++ Ref + Asymm -	<u>Size</u> Constant ++ Expanded - Exp & Cont --	<u>Axis</u> Straight + Curved -
	S	++	++	+
	C	++	++	+
	S	+	-	+
	S	++	+	-
	C	++	-	+
	S	+	+	+

<u>Geon</u>	<u>Edge</u> Straight S Curved C	<u>Symmetry</u> Rot & Ref ++ Ref + Asymm -	<u>Size</u> Constant ++ Expanded - Exp & Cont --	<u>Axis</u> Straight + Curved -
	S	+	++	-
	C	+	++	-
	S	++	-	-
	C	++	-	-
	S	+	-	-
	C	+	-	-

How many objects can be modeled using this simple set of 24 geons? An analogy can be made to speech:

- 44 phonemes code all English words
- 15 phonemes code all Hawaiian words
- 55 phonemes code all languages world-wide

Can a similar number of parts encode all objects?

Note: the question is meant to be applied to "count" objects, such as a chair or book, rather than "mass" objects, such as sand or water. The latter are more likely recognized by color and texture.

How many objects do people know?

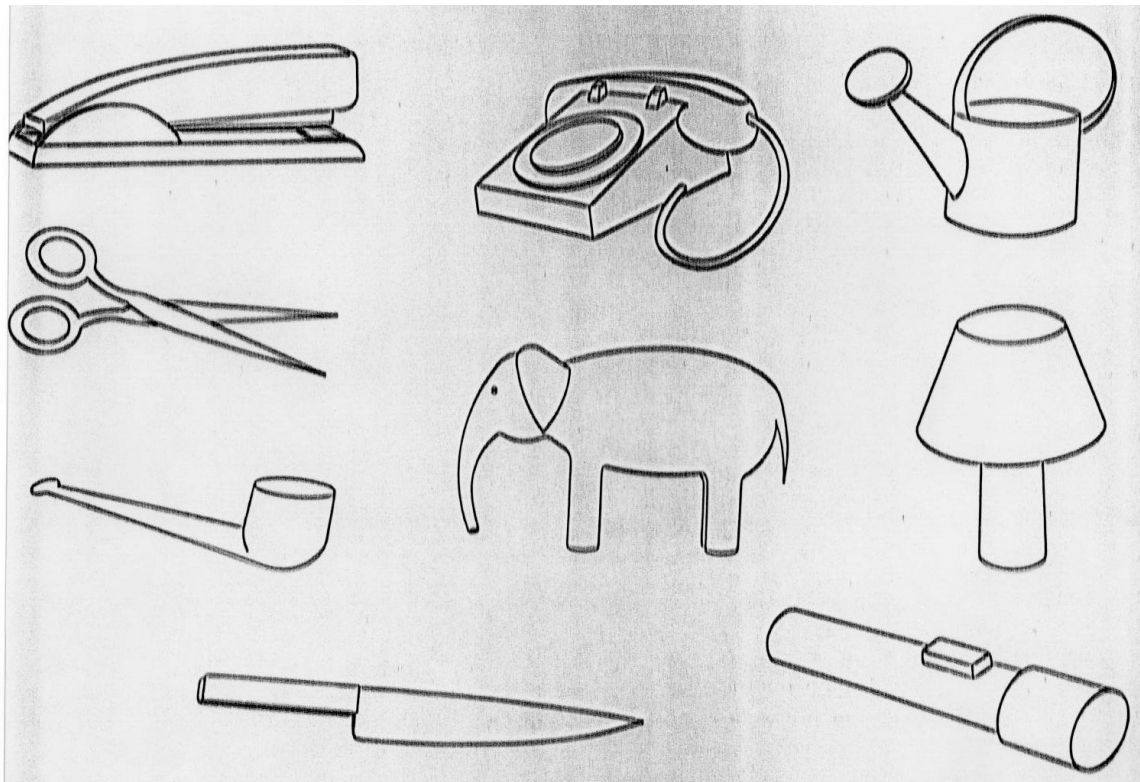
- ~1500 common basic-level count noun categories can be found in a dictionary (e.g. elephant, chair)
- x2 (to be liberal), x10 types/category, yields ~30,000 objects
- ~4.5 new objects learned per day (learning rate seen in children)  
x 18 years = ~16,000 objects

With size, direction, and type of join:

- 2 geon objects: ~75,000
- 3 geon objects: ~150,000,000

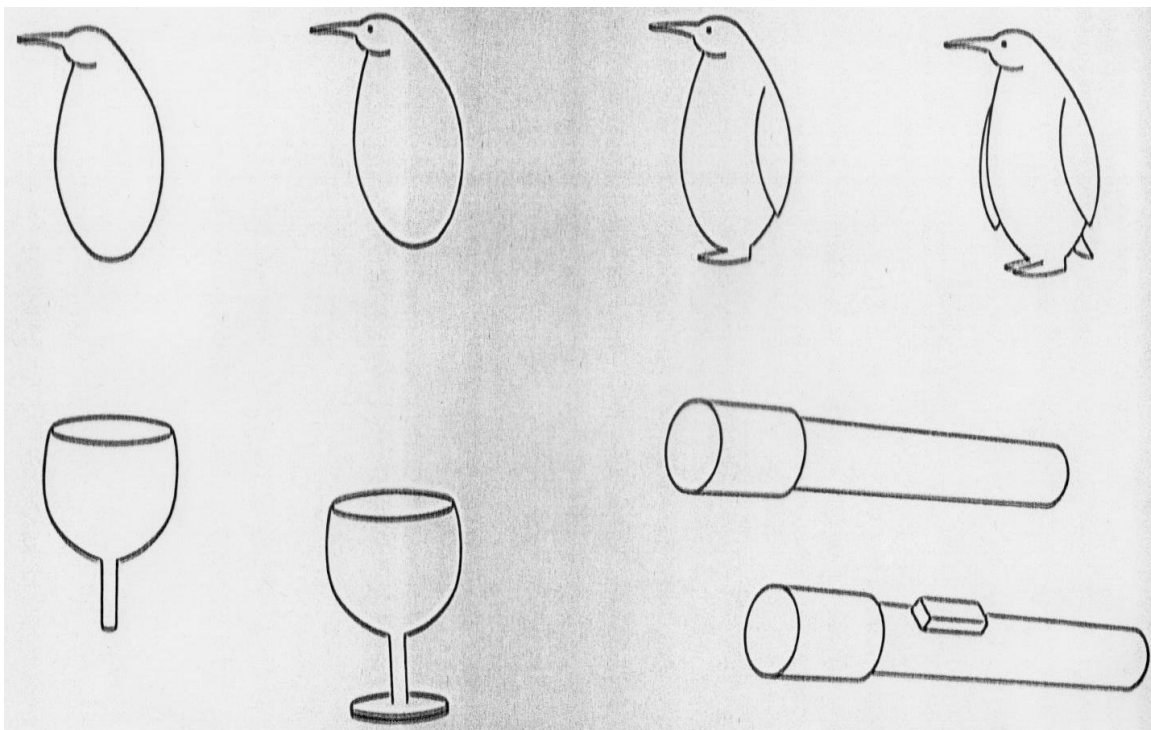
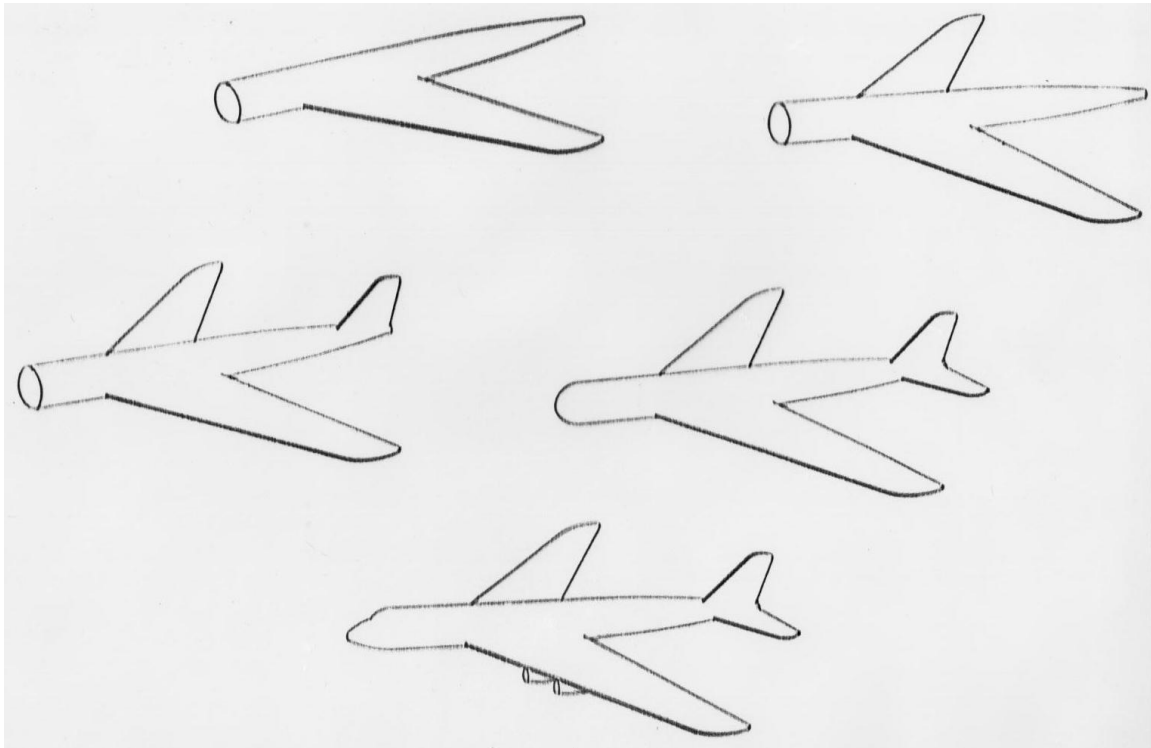
Note: not all geon combinations are assumed to represent real objects.

Here are some examples:

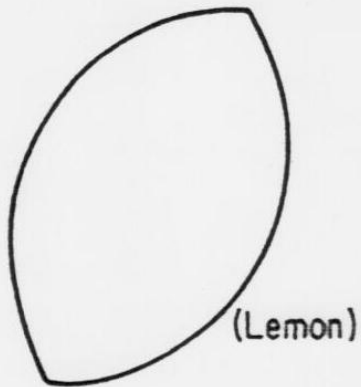




Some objects can be recognized with very few geons, but more simple geons provide increasing levels of detail. This is similar to how cartoons can be drawn.



Some objects can be recognized from a single geon:



(Lemon)

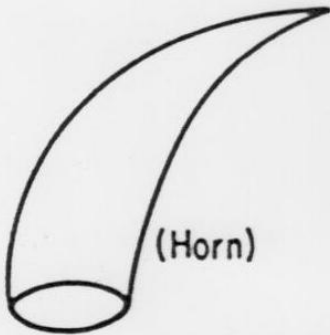
Cross Section:

Edge: Curved (C)

Symmetry: Yes (+)

Size: Expanded & Contracted (--)

Axis: Straight (+)



(Horn)

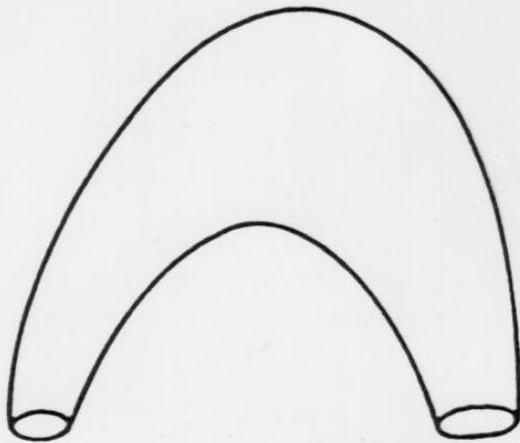
Cross Section:

Edge: Curved (C)

Symmetry: Yes (+)

Size: Expanded (+)

Axis: Curved (-)



(Gourd)

Cross Section:



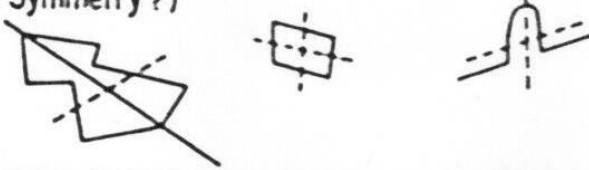

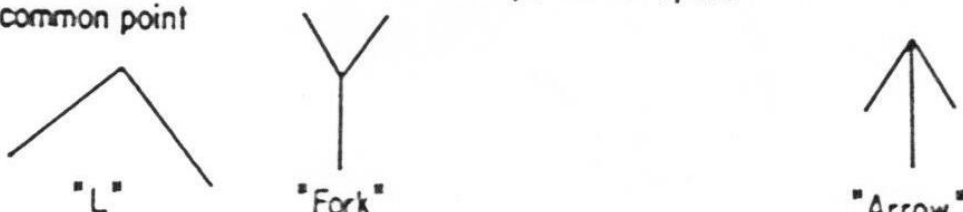
Edge: Curved (C)

Symmetry: Yes (+)

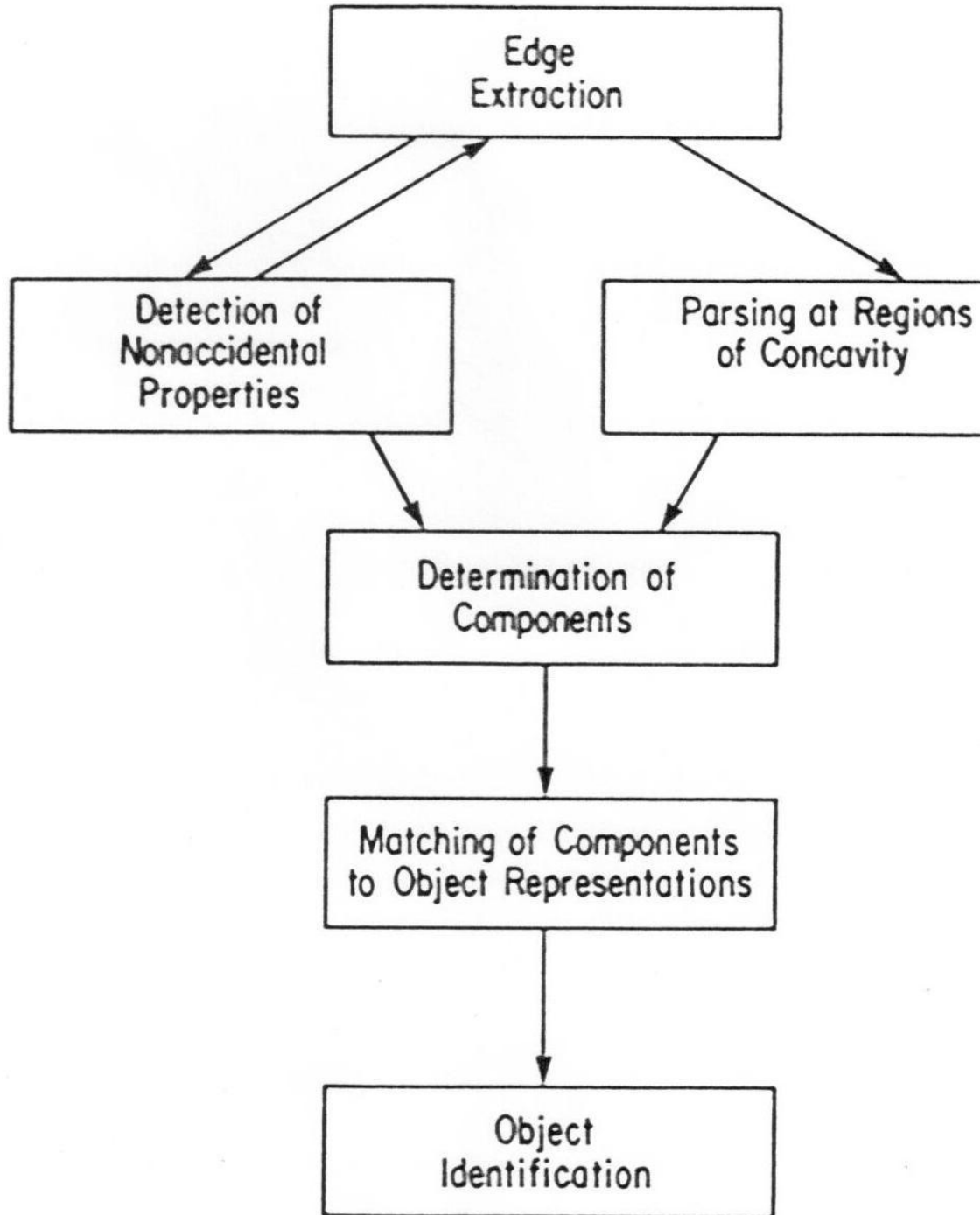
Size: Expanded & Contracted (--)

Axis: Curved (-)

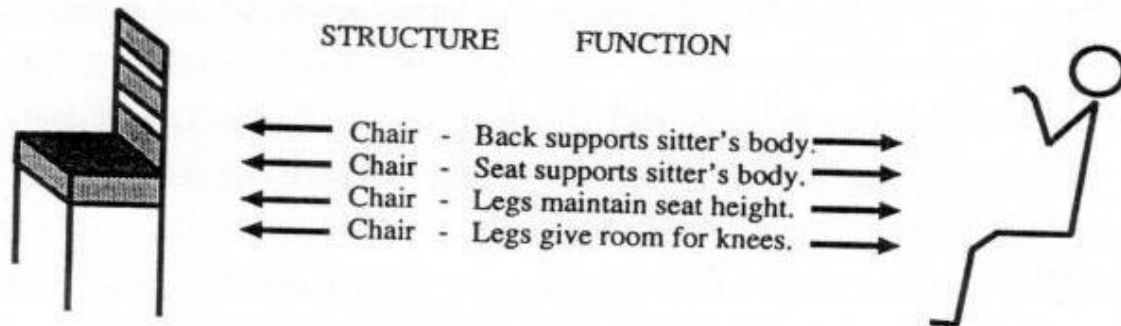
Like aspect graphs, geon-based recognition assumes non-accidental features. For example:

<u>2-D Relation</u>	<u>3-D Inference</u>	<u>Examples</u>
1. Collinearity of points or lines	Collinearity in 3-Space	
2. Curvilinearity of points of arcs	Curvilinearity in 3-Space	
3. Symmetry (Skew Symmetry?)	Symmetry in 3-Space	
4. Parallel Curves (Over Small Visual Angles)	Curves are parallel in 3-Space	
5. Vertices — two or more terminations at a common point	Curves terminate at a common point in 3-Space	

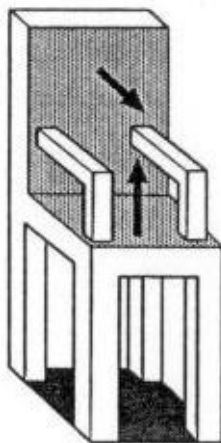
The general flow of Beiderman's geon recognition works as follows:



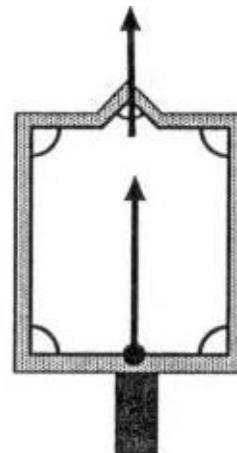
**Function-based recognition** is another possible paradigm. It assumes that the parts of an object serve a purpose, and that the object can be recognized through the collective set of purposes of its parts. For example, consider a chair:



The primitives are determined via geometric reasoning on a 3D model that is recovered from images. For example, relative orientation is important between the seat and back of a chair, and between the handle and pouring spout of a pitcher:



back support approximately  
orthogonal to seat support



pouring direction  
opposite handle

Figure 6: Examples of the knowledge primitive “relative\_orientation.”

Other knowledge primitives include dimensions (size), proximity (closeness of parts), stability (assuming a direction for gravity), clearance and enclosure.

The following shows some more examples of primitives and how they can be scored in the recognition process:

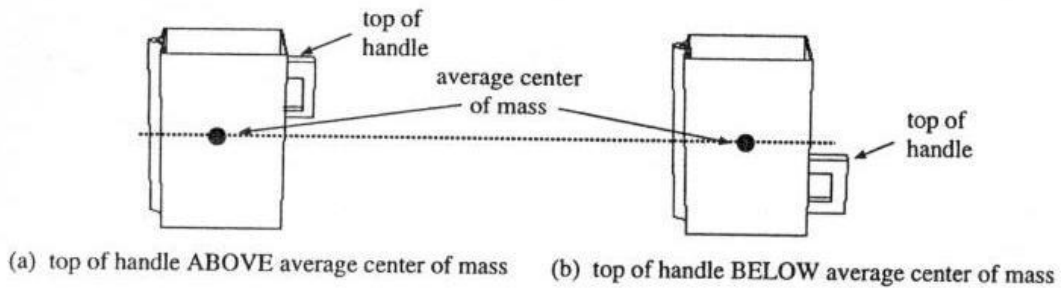


Figure 9: An example of the knowledge primitive "proximity."

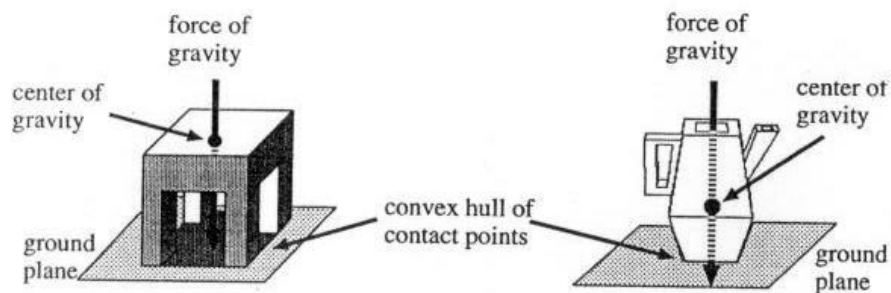


Figure 10: Examples of the knowledge primitive "stability."

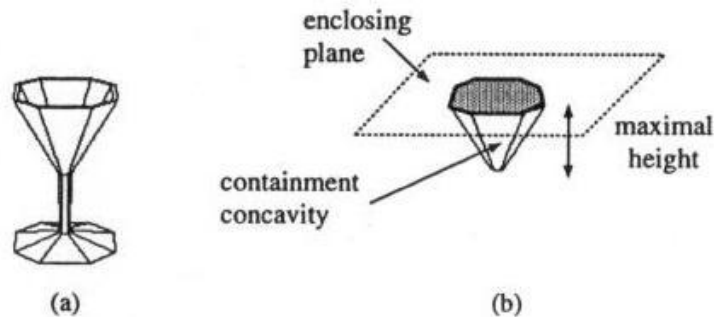


Figure 12: An example of the knowledge primitive "enclosure."

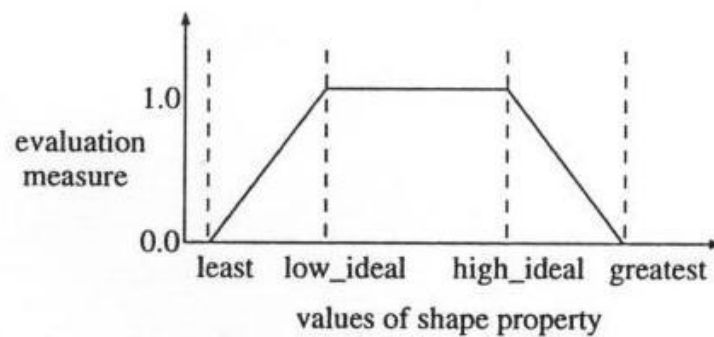


Figure 13: Shape properties are used to calculate evaluation measures.

A test of this method on 450 shapes yielded a 94% agreement with human interpretations of the shapes. An interesting failure was that the system reasoned about a trashcan, turning it upside down and recognizing it as a potential sittable object (stool/chair). Here are some of the objects tested:

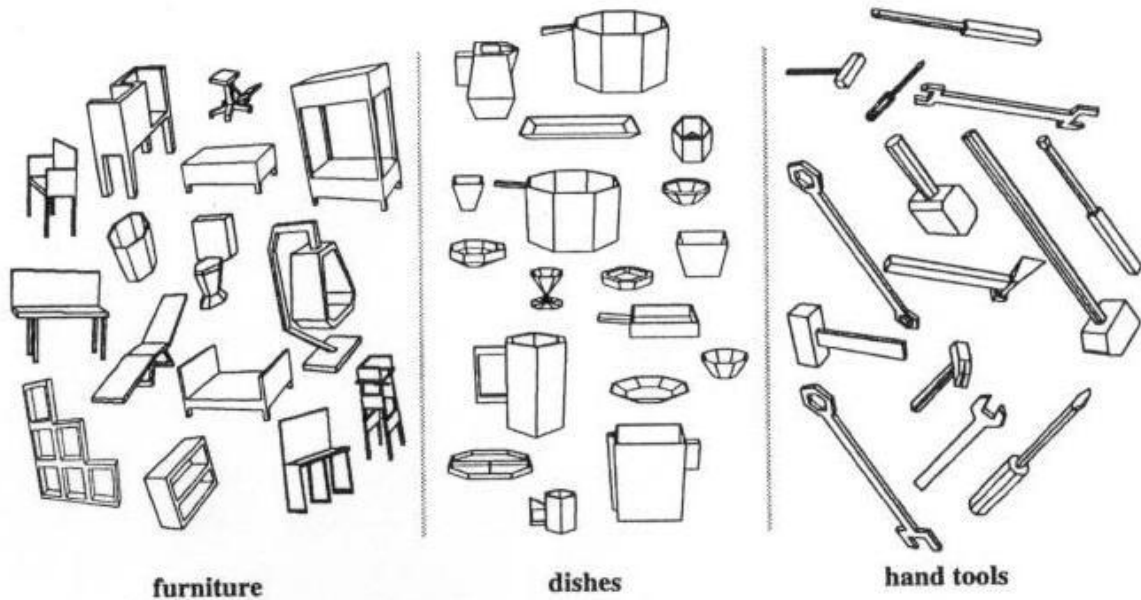


Figure 36: Example object shapes tested by the GRUFF system.

Object recognition is an unsolved problem. The paradigms discussed in this lecture have all made progress towards the goal of general object recognition, but much work remains to be done to see if any of these techniques can scale to the level of a human.