MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A. I. Memo 873 November 1985

Diagram Understanding: The Intersection of Computer Vision and Graphics

Fanya S. Montalvo

Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Ma. 02139

Abstract

A problem common to Computer Vision and Computer Graphics is identified. It is the problem of representing, acquiring, and validating symbolic descriptions of visual properties. The intersection of Computer Vision and Computer Graphics provides a basis for **diagrammatic conversations** between users and systems. I call this problem domain **Diagram Understanding** because of its analogy with Natural Language Understanding. The recognition and generation of visual objects from symbolic descriptions are two sides of the same coin. A paradigm for the discovery and validation of higher-level visual properties is introduced. The paradigm involves two aspects. One is the notion of **denotation**: the map between symbolic descriptions and visual properties. The denotation map can be validated by focus on the *conversation* between users and a system. The second aspect involves a method for discovering a natural and rich set of visual primitives. The notion of visual property is expanded, and the paradigm is further illustrated with a traditional business graphics example.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support of the laboratory's artificial intelligence research is provided in part by the Digital Equipment Corporation and in part by the Defense Advanced Research Projects Agency of the Department of Defense under the Office of Naval Research contract N00014-80-C-0505.

Table of Contents

1. Introduction 1
2. Some Underlying Assumptions 4
  2.1. Graphic Input is a Computer Vision Problem 4
  2.2. Abstraction Exists in the 2-D Domain 5
  2.3. Some Form of Validation is Essential 6
3. Some Problems of Reference 7
4. Steps Toward Diagrammatical Conversations 10
5. Acquisition of Visual Properties 12
6. Elements of Bongard Problems 16
  6.1. Concreteness and Precision 16
  6.2. Decomposition and Composition 18
  6.3. Vividness and Conciseness 21
7. Properties Versus Objects as Primary Representations 23
8. What's in a Graph? 25
9. Conclusion 28
10. Acknowledgments 30

List of Figures

Figure 1-1: A sample diagram. 1
Figure 1-2: The denotation map for Vision and Graphics. 2
Figure 3-1: A bar graph of A + B = C. 8
Figure 3-2: The bar graph scaled, a **format** reference. 8
Figure 3-3: The same data, now as a pie chart, a **notational** reference. 9
Figure 3-4: The same bar graph with B changed, a **semantic** reference. 10
Figure 5-1: Bongard problem #3. 12
Figure 5-2: Bongard problem #6. 14
Figure 5-3: Bongard problem #10. 14
Figure 5-4: Bongard problem #97. 15
Figure 6-1: A problem in the style of Bongard. 18
Figure 6-2: Bongard problem #15. 19
Figure 6-3: Bongard problem #5. 19
Figure 6-4: Bongard problem #49. 20
Figure 6-5: Bongard problem #53. 20
Figure 6-6: A difference between a primitive property and a vivid one. 21
Figure 6-7: Test figure. 22
Figure 7-1: A Venn diagram representing each object in a Bongard problem as a set of properties. The thin circles represent the objects on the left and the thick circles represent the objects on the right. 24
Figure 8-1: A typical x-y plot. 25
Figure 8-2: The plot of figure 8-1 with labeling relationships shown. 26
Figure 8-3: The plot of figure 8-1 with denotation relationships and analogies shown. A literal analogy in distance induces the abstract analogy. 27

1. Introduction

When two scientists talk to each other about their ideas, they typically do not restrict themselves to words; they also draw diagrams, label them, and tell each other what these diagrams represent. "Here's a ball rolling down an inclined plane," one scientist may say, for example, and simultaneously sketch a diagram such as in figure 1-1. In order to understand this kind of conversation, the scientists must recognize the circle as such, and associate it with a ball, recognize the arrow and associate it with a vector, and recognize the triangle and associate it with a ramp. After these associations are made, the behavior of the objects in the diagram are then determined. The diagram is a kind of notation with which to solve simple physics problems. Alternatively, the scientists may use a standardized notation whose meaning is known by everyone in the culture.

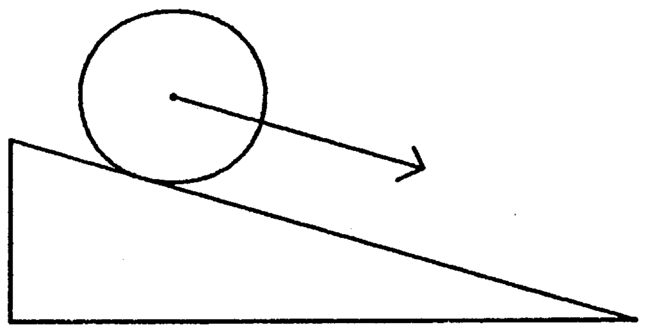

Figure 1-1: A sample diagram.

In order to make sense of this kind of conversation, we need to understand the visual vocabulary being used. In order to embody such a vocabulary in a conversation with a computer system, several conditions must hold. The system must have both *recognizers* and *generators* for the visual vocabulary and methods of associating it with the application domain vocabulary, physics in this case.

The conversation the two scientists are having is a kind of **diagrammatic conversation**. They are generating visual symbols that stand for the things they are talking about, and they are recognizing visual symbols as objects of discourse. The diagrams on paper are just as much a part of their conversation as the words.

When captured by a system, the recognition component of the conversation can be viewed as Computer Vision. The Vision problem is the process of going from a sequence of light intensity arrays digitized by a television camera, to a symbolic description of a situation that can be used by a reasoner to solve a spatial problem, such as navigation or assembly. Aside from the procedural issues of *how* one solves this problem, the declarative part is one half of the denotation relationship: the relationship between the image and its symbolic description. [See figure 1-2 below.]

$$\begin{tikzcd}
\boxed{\textbf{Symbols}} \arrow[d, shift left, "\textbf{Graphics}"] \\
\boxed{\textbf{Objects}} \arrow[u, shift left, "\textbf{Vision}"]
\end{tikzcd}$$

Figure 1-2: The denotation map for Vision and Graphics.

The declarative part of the Graphics problem is the opposite of the declarative part of the Vision problem. It involves going from a user's symbolic description of a picture, a diagram or sequence of these, to the light intensity array typical of modern frame buffers. Currently available graphics systems do only part of the job. For one, such systems are largely procedural in nature. One can tell them *how* to draw in detailed procedural terms, but one cannot describe *what* one wants to draw in higher-level terms. For another, current graphics systems are limited in the number and types of objects to which a user can refer: for example, a user can represent x-,y-coordinates, polygons, and

RGB values, but not properties such as **near**, **larger than**, **inside**, **above**, **convex**, **color by name**, **contrast**, and **squiggliness**. These are properties of diagrams users want to manipulate while interacting with a graphics system. Some graphics experts devote much time and effort to programming these relationships, but in each case the relationship is custom-made to the application, with little room for flexibility by the user. Custom-made graphics are procedural rather than declarative: a user can get the system to do all kinds of interesting things, but cannot *refer* to component parts and properties, and therefore cannot change them. In order to more easily refer to objects in the graphical domain one can be aided by symbolic descriptions and inference machinery applied to the graphics interface part of the system, as well as to the application part. The best way of capturing descriptions and reasoning is with knowledge representation systems. Although knowledge representation systems have been generally applied to specific application domains, they also need to be applied to the visual component of systems. If the visual reasoning is described in the same higher-level language as the application domain reasoning, new ways of displaying complex ideas can be synthesized more flexibly and quickly.

I wish to call the overlap between Computer Vision and Computer Graphics, that which has to do with the correspondence between symbolic descriptions and images, **Diagram Understanding**. Computer Vision and Computer Graphics, at least as originally conceived, were thought of as straight-through, batch processes, ones from image to description and vice versa, with little interaction with a user. But combining methods from the two fields can provide a framework for a system that can have a **diagrammatic conversation** with the user. The diagrammatic conversation paradigm, because of its tight feedback from a human, has a chance of validating the higher-level properties and operations implemented by the system. By "validation" I mean finding a way of testing whether the conversations "sounds" right as in the Natural Language paradigm. Much work has been done in the area of Natural Language Understanding [Winograd 83]. We would like to further this work by including diagrams in natural dialogues. Zdybel et al. [Zdybel et al. 81] and Friedell [Friedell, Barnett, and Kramlich 82; Friedell 84] have shown that including declarative knowledge about display in standard knowledge-based applications can increase their flexibility and power. The present study goes further to address the issues of validation of those descriptions and acquisition of natural, visual primitives and compositional operators for use in such systems.

2. Some Underlying Assumptions

Implicit in this view that graphic interaction is a diagrammatic conversation are the following underlying assumptions.

2.1. Graphic Input is a Computer Vision Problem

The first assumption is that graphic input can be viewed as a recognition problem. There are two aspects to recognition: the *how* and the *what*. I will concentrate on the *what* in this paper. The *what* concerns the space of possible objects and properties to which a user can refer. For example, a paint program recognizes pixels in specific colors, bounded regions that can be filled, and frames as in a film. A draw program recognizes connected line segments, circles, rectangles, texture patterns, etc. This is apart from *how* they are recognized. Suppose that in an interactive graphics system, a user wants to select objects and properties of objects by pointing, and input objects by drawing. The system has to have a very good idea of *what* constitutes valid objects in a given domain. For example, if one points to a set of squares, is one selecting the set, one of the squares, one of the line segments in one of the squares, the corresponding line segment in all of the squares, or one vertex of one of the squares? The possibilities are endless. In order to constrain the pointing and drawing process to manageable proportions, it is necessary to know what the common objects of discourse are. This constraint on the domain of discourse is similar to that required in Natural Language. The process must also be constrained relative to a context since denotation can vary over contexts [Labov 73]. For example, Ciccarelli's system [Ciccarelli 84], allows one to draw circles and ellipses around text or other objects. In Ciccarelli's system any sloppily connected curve will be recognized as an elipse in the most general case, or a circle if the aspect ratio is close to one. Elements of Computer Vision are apparently unavoidable in the general graphics input problem: input must be *recognized* as being in a certain semantic category in order to be parsed correctly and disambiguated by context in order to be understood. In any case, graphic interaction restricts the nature of the Vision problem to a more narrow one of finding a correspondence between the user's input gestures and the domain of visual objects in a specific application.

The observation that the input side of full graphical interaction is a Vision problem has not been widely recognized among the Graphics community because of the simple nature of graphic input thus far. However, if one is to proceed to more complex diagrammatic systems, more sophisticated kinds of recognition are desirable. For example, in a text formatter, it would be nice to change the size of spaces between words by just pointing to one and stretching it.

Thinking of the graphic input problem as Computer Vision helps in this regard, because knowledge of human vision can be brought to bear. Conversely, the Vision problem can be simplified if it is, in turn, recognized as partially a Graphics problem in terms of echoing intermediate results of processing to the user as recognition proceeds. In this way, rather than a straight-through, batch process from picture to goal, the process becomes more of a conversation with the user. A user can help the system understand pieces of a picture and thus allow intermediate results in the Vision problem [Glicksman 82]. Using both allows us to concentrate on the structure and content of higher-level descriptions rather than the computation-heavy, low-level parsing and generation.

2.2. Abstraction Exists in the 2-D Domain

A second assumption is that even though lower-level issues of segmentation and contour extraction are bypassed in graphic input, interesting higher-level issues still remain. Even in the

2-dimensional (2-D) domain of diagrams, higher-level issues such as layout and the composition of lower-level properties into higher-level objects remain. I am restricting myself to the kind of diagrams scientists sketch easily: 2-D, without occlusion (2 1/2 D), without motion, and without color. I do not intend to account for the full range of images common to Computer Vision and Computer Graphics: 3-D, shaded, color images with motion; even though it appears that the paradigm will generalize to the 3-D, textured object context [Pentland 84]. Even given this restriction the problems of discovering natural and flexible compositional and layout primitives and relating them in a meaningful way to linguistic symbols is enormous. However, it is a good place to start, because given these restrictions we can concentrate on how higher-level vision connects to the lower levels without diverting energy to lower-level parsing requiring massive amounts of computation, such as segmentation and contour extraction.

It is important to understand that "higher-level" does not equal "3-D". True, many interesting and complex issues exist in reconstructing the 3-D image, but just the issue of which 2-D, symbolic descriptions correspond to which 2-D images is none-the-less abstract. How objects are inferred from groups of primitive properties, and how diagrams represent complex domains are also difficult and interesting problems worth studying. There is some evidence that a 3-D description is not the only "highest-level" output of the human visual system [Sloman 85].

2.3. Some Form of Validation is Essential

Finally, there is the assumption that human perception can be brought into the **conversational loop** in order to validate the correspondence between higher-level descriptions and lower-level image components. By the "conversational loop" I mean the tight feedback between a system and a user which continuously confirms what the user and the system are talking about. An example of this loop is the echoing of a rubber-band line when a user is drawing a line segment. After the first point is selected, the system continuously echoes a line segment from the first point to the current cursor position. This echoing gives the user feedback on what the line segment indicated by the current cursor position looks like. The loop signals a kind of agreement of reference between the user and the system: they both mean "this" line (the one currently being pointed to).

The conversational loop in graphic interaction may provide a bridge between low-level and high-level visual representation by necessitating an intermediate vocabulary. At the low-level some validation for symbolic descriptions can be obtained through correspondence with single-celled feature detectors [Montalvo 76]. At the highest levels, descriptions become more accessible to introspection. One way of discovering intermediate levels is by providing feedback that checks for the correspondence of reference between a user and a system as the user constructs complex objects from primitive properties. Thus, the intermediate levels are spanned by echoing. Echoing is a kind of experiment in which the system continuously asks a user "is this what you mean?" and the user responds by continuing. If the echo is not what the user intended, the conversation stops.

3. Some Problems of Reference

The problems of reference in diagrammatical conversation become very complex as they do in verbal conversations. As alluded to earlier, graphical manipulation is ambiguous. It is not just a problem of *scope*, by which I mean, how much of an object is being pointed to, but one of *focus*: what is the meaning of the object being pointed to? Do we intend to manipulate the appearance of the object or the semantic domain underlying the interface? Typically, systems allow drawing of strictly visual objects, as in paint programs [Smith 78; Shoup 79; Levoy 81], or they allow manipulation of application domain objects, as in CAD systems [Barsky 82], but seldom both. Very few, with some notable exceptions [Ciccarelli 84; Morse 84], allow one to, first, interactively create visual objects, then attach them to application objects, and finally manipulate application objects *through* the visual objects that represent them.

An example with some terminology may make this point a little clearer. Suppose we want to

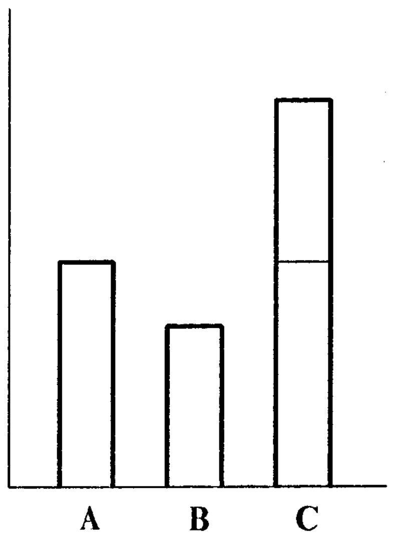

Figure 3-1: A bar graph of A+B=C.

represent the datum A+B=C. We choose a bar graph. Now suppose we want to change the way the graph looks. For example, we may want to scale the graph. Scaling in this context, changes the appearance of the graph but not its meaning. I will call this kind of selection in which only the visual object has been changed, a **format** reference.

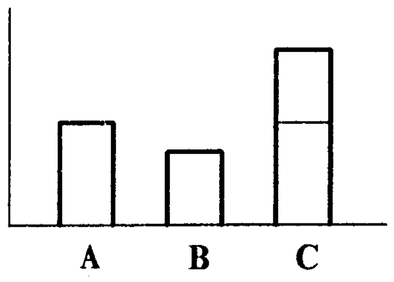

Figure 3-2: The bar graph scaled, a **format** reference.

Secondly, we may want to change the way the datum is represented. Doing so changes the point of view, but not the original datum. For example, suppose we would rather see the equation as a pie chart (figure 3-3). I will call this kind of manipulation in which the relationship between the visual objects and what they represent in the application domain has been altered, a **notational** reference.

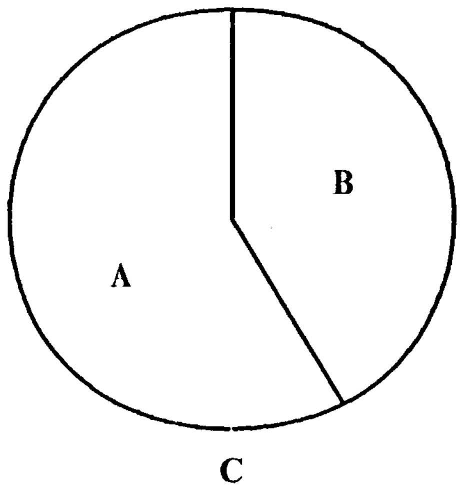

Figure 3-3: The same data, now as a pie chart, a **notational** reference.

Thirdly, we may want to change the datum itself by manipulating the display. We move bar B to a higher position and C automatically moves up. This is a **semantic** reference to the application domain value. Here, the application domain value itself has been changed and the rest of the diagram automatically changes to maintain semantic consistency. Here we can see that a constraint-satisfaction style of graphic interaction, as in THINGLAB [Borning 79], and VISICALC [Ramsdell 80], is a special case of full Diagram Understanding.

In order to deal with these three very different kinds of reference, the graphical knowledge in the system must be explicit and be allowed to interact with the application domain knowledge. To do so requires that the visual knowledge be expressed in the same symbol system as the application knowledge. Given a uniform symbol system, we can explicitly construct the denotation map, associate visual properties with semantic properties, and be able to distinguish them when

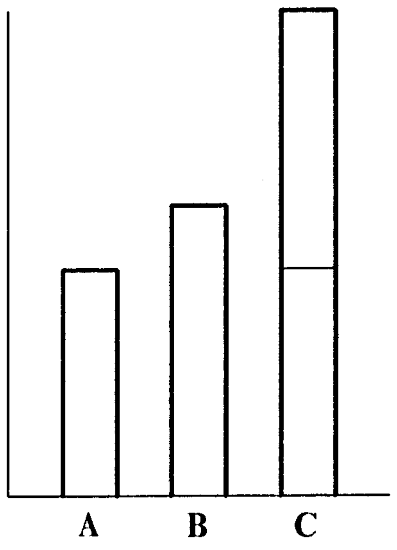

Figure 3-4: The same bar graph with B changed, a **semantic** reference.

necessary. The reasoner must be able to evaluate statements about the visual knowledge, the application domain knowledge, or the relation between them. Weyhrauch's FOL [Weyhrauch 80] is an example of a knowledge representation system that can express all three. A graphical application in FOL can be found in Filman, Lamping, and Montalvo [Filman, Lamping, and Montalvo 83]. The present study addresses the problem of acquiring and expressing visual knowledge given an already existing knowledge representation system and an application.

4. Steps Toward Diagrammatical Conversations

A diagrammatical conversation is not complete if we cannot break out of the application domain we are in and refer to the diagrams themselves in some way. The ability to refer to the diagrams themselves allows flexibility in the interface. Being able to refer to the diagrams and the application domain in the same statement is a meta-operation. It requires a rich vocabulary for the diagrammatic domain as well as the application. Typically, the graphic interface builder and the application expert are different people. Because it takes so much time and effort to build a good graphics interface, the graphic interface builder must be an expert at graphics interfaces and is usually not an expert in the application domain too. The people most expert in visualizing the domain are the domain experts themselves, yet they do not have the time to become graphics experts to build good interfaces. So, users of the end products usually find the graphical interfaces inflexible and constraining. It would be better for domain experts to build and change the interfaces themselves. What domain experts need are graphical tools in which they can freely conceptualize how something should look without having to spend too much time getting it to look that way. They need a graphics tool kit with a rich enough set of visual primitives and operators to attach a graphical interface to a particular knowledge domain.

Before we can have a diagrammatical conversation with a system, we have to have a vocabulary in which to speak, one that is natural to people and understandable to computers. An example of the kind of vocabulary we would like to have is shown by the following example. Suppose we ask for a plot of three variables over time and we don't wish to specify exactly how to draw the x-y axes. Graphics packages exist that do this, but typically the user has no direct control over layout except through coordinate related input parameters. After the plot has been output, the user cannot then say "move the y-axis label a little to the right" or "color the wiggly line red." Interactions are not typically in terms of high-level descriptors such as "a little", "to the right", "red", or "wiggly". Pointing doesn't help very much by itself, because it is ambiguous.

So we see that there are two conditions for a smooth diagrammatical conversation: 1) that the set of visual primitives be natural to humans and computable by machines, and 2) that the system have knowledge about visual properties as well as application properties so that manipulations respect both visual constraints and application domain constraints. A summary of these two conditions could be stated as: finding the right vocabulary and finding the right rules.

5. Acquisition of Visual Properties

In order to draw complex diagrams, or select visual properties from an existing set, the system must have an extensive and flexible set with which to start. The requirements of a smooth diagrammatic conversation will help hone a system to a more natural set of diagrammatic concepts, but they will not help find such concepts in the first place. The starting point should be to capture some of the visual categories that humans have about the world. However, because most of the human visual system is not introspectable, the standard knowledge engineering paradigm of asking an expert will not do. Rather than being a drawback, I believe this lack of introspective ability to be a feature: It certainly prevents us from falling into the introspection fallacy. We do not have access to such knowledge without an experimental paradigm that will make the knowledge explicit.

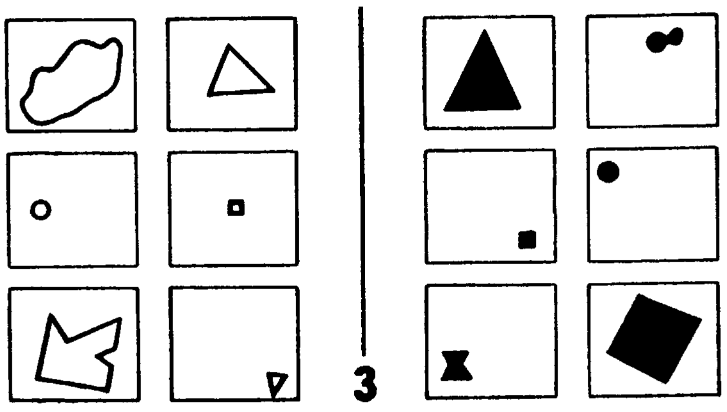

Figure 5-1: Bongard problem #3.

The paradigm that I wish to sketch out is just such an experimental paradigm. It takes advantage of the large set of visual puzzles composed by Bongard as problems for Pattern Recognition [Bongard 70]. See figure 5-1 for an example. The problem posed by the figure is to find the minimal description that distinguishes the six figures on the left from the six on the right. In this case the distinguishing property is that of being **filled** as opposed to **hollow**. The idea is to make people's visual knowledge explicit by presenting Bongard problems to them. Unlike standard intelligence tests that involve visual analogy, Bongard problems are simpler. They embody a wide variability in a set of objects all having the same property on either side of a dividing middle line. The solution makes explicit only the one property that distinguishes all the elements on the right from those on the left. If there are alternative descriptions it is easy to tell which is the most concise description given the whole set of Bongard problems (there are on 100 Bongard problems in all). It is also easy to construct a variation of the property embodied by one of the distinguished sets that still fits the distinguishing description. Therefore, each explicit solution yields a visual property in symbolic form (a person has to state the solution in words), and a set of visible objects that embody the property and span a space of parameters within which the property can vary and still be the same identifiable property.

Thus, the first step in the paradigm is to gather sets of symbolic descriptions along with their corresponding images by presenting Bongard problems to people. The next step is to build recognizers and generators for those properties. The third step is to build a system in which all the properties co-exist, can be referenced, composed and manipulated in objects embodying those properties. Given these three steps, users of the resulting system can then verify whether their input matches the symbolic description given by the system, and whether a sample object produced by the system when given a symbolic property by the user, does in fact embody that property. In other words, the recognizers and generators can be checked by supplying human input at either end and checking the results of the mapping at the other end. In this way a diagrammatic conversation can be used to verify the correspondence between symbolic descriptions and visual properties.

In addition to primitive properties, Bongard problems can also suggest operators for combining primitive properties. Typically, solutions are compositions of several properties. For example, problem #6 involves the ability to recognize a polygon, and also count the number of sides of the

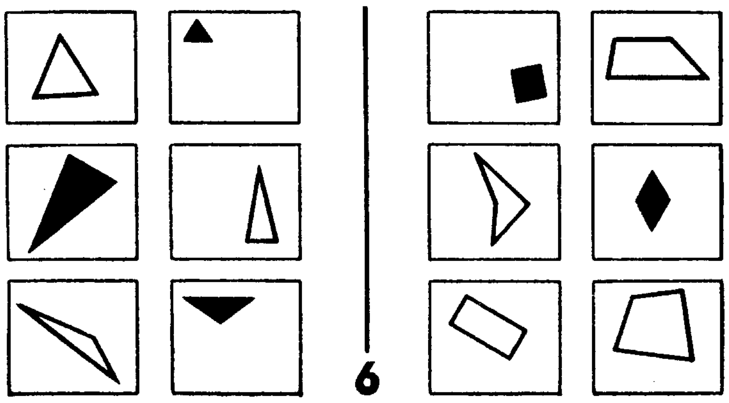

Figure 5-2: Bongard problem #6.

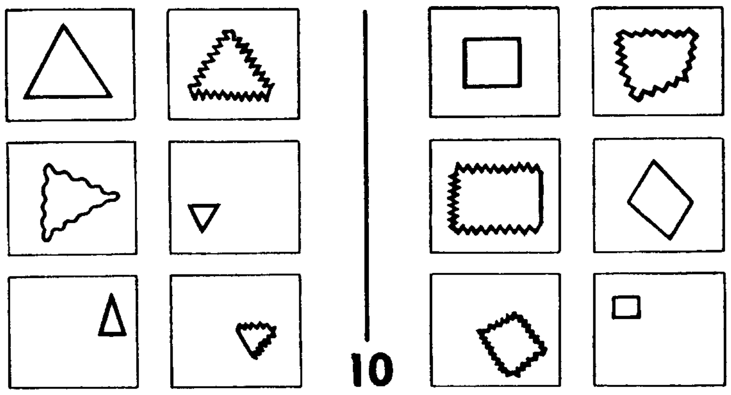

Figure 5-3: Bongard problem #10.

polygon. In problem #10, the situation is a bit more complicated: edges must first be filtered before the sidedness count can be taken. Finally, in problem #97, the pixel array must be smoothed and thresholded in order to extract predominant contours. In many cases, it is not just the primitive properties that matter, but how they are composed. It is with this explicit identification of the *relationships* between properties that we can see how higher-level descriptions are built up from lower-level primitives. Solutions to Bongard Problems provide explicit descriptions of visual relationships and abstractions, as well as primitive properties.

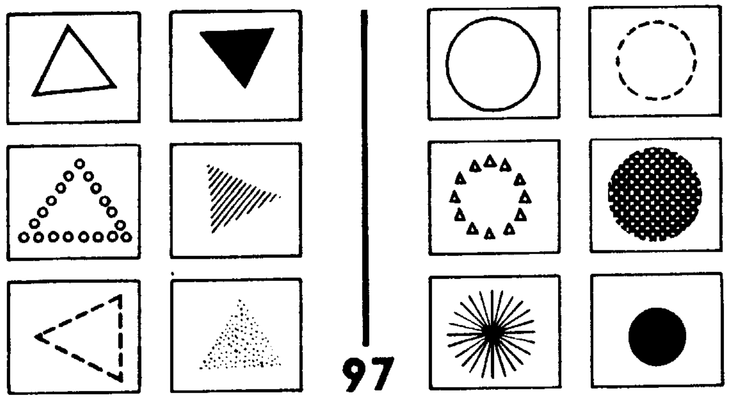

Figure 5-4: Bongard problem #97.

To summarize, the experimental paradigm is this. First, a visual property is made verbally explicit by presenting a Bongard problem to human subjects. The system itself does not solve the Bongard problem. Secondly, the verbal descriptions are translated to symbolic descriptions in the system's terms. If there are alternative descriptions, the most concise and implementable is preferred. Thirdly, the denotation map consisting of recognizer and generator is implemented for each property. Subjects can then verify the map by checking the correspondence between graphic input and the resulting symbolic description; and, similarly, by checking the correspondence between symbolic input and a sample object produced by the system. Finally, by incorporating the set of visual properties gathered in this way into a graphics tool kit having mechanisms that allow attachment to an application domain, we can verify how the properties fare as part of a full diagrammatical conversation in some application domain. By testing the set against a wide set of applications we can get a measure of the degree of completeness and generality in the kit as well as of its utility.

Sometimes the applicability of a certain property is not known in advance. The Bongard paradigm affords a certain amount of generality independent of particular application domains. I was at first concerned that a particular property, squiggliness of a closed curve, in problem #20, was not obviously applicable in some real graphics context. On showing it to Gene Ciccarelli, however, he immediately thought of computer networking examples in which the central node is represented by a cloud in order to deliberately keep the process vague. It's not clear in advance how application experts visualize their domain. It's better to provide as wide a range of properties as possible and allow them to select and construct their own idiosyncratic presentation. Bongard problems provide a very rich domain from which to choose.

In addition, new Bongard problems can be created as needed to verify new properties. We can throw out unused properties and create and test new ones. Doug Hofstadter has invented 46 new Bongard problems that are much more abstract than Bongard's original 100. The point is that once a tool kit is available, the overhead of building new presentations from scratch need not be paid for every time a user wants to change the style of interaction.

6. Elements of Bongard Problems

Let us examine Bongard problems more closely in order to understand the elements that make them a good tool for extracting knowledge about visual properties from human subjects. There are three general characteristics: 1) the solutions to the problems establish the correspondence between symbolic descriptions and image properties, 2) the problems provide a test for the decomposition of and examples of the composition of visual properties, and 3) vividness on the visual side and conciseness on the symbolic side are related.

6.1. Concreteness and Precision

The first characteristic makes symbolic descriptions more concrete. Descriptions alone are not very well grounded if not attached to visual examples that make the exact nature of the target property explicit and compelling. The set of figures in a Bongard problem do this in several ways. First, they show six positive examples --- the figures on the left --- of the target property embodied in objects having several variations. For example, in figure 5-1 the property **hollow** on the left is varied in several different ways: **number-of-sides, size, curvilinear/rectilinear,** and **concave/convex.** Second, a problem presents six different examples of near misses --- the six figures on the right --- all of which similarly share properties of **three-** and **four-sidedness, large** and **small, curved** and **rectilinear,** and **convex** and **concave,** but do not exhibit the target property of being **hollow.** Symmetrically, positive and negative examples are also present for the property embodied by the six figures on the right. In this way the strictly **visual** properties of **hollow-ness** and **filled-ness** are unambiguously associated with the symbolic terms we've been using, that is, by the solution to the problem.

There are several aspect to this concrete association between descriptions and visual properties. Visual examples constrain the property being defined to a small concise example. (See section 7.) The set of six figures is actually an example of *a single* property, *not* just one object. In addition, the association of the property with a symbolic term, or set of terms, makes the target property explicit on both sides of the problem: the left and right fields. The two target properties are points on a dimension. In this case, the dimension is just binary, **filled/hollow**; but in other cases it may be a continuous dimension, such as **size**, or **orientation**; or it may be a many-valued discrete dimension, such as **sidedness.** So an experimental paradigm having only two sample objects to distinguish is not as precise because it does not narrow the target property sufficiently. A paradigm having only one set of objects as examples of the target property is also not precise enough, because it provides no negative examples, leaving the universe of potential properties unbounded. A paradigm, like this one, for distinguishing two sets of objects having variations over other common property dimensions, does have the requisite elements of concreteness and precision.

6.2. Decomposition and Composition

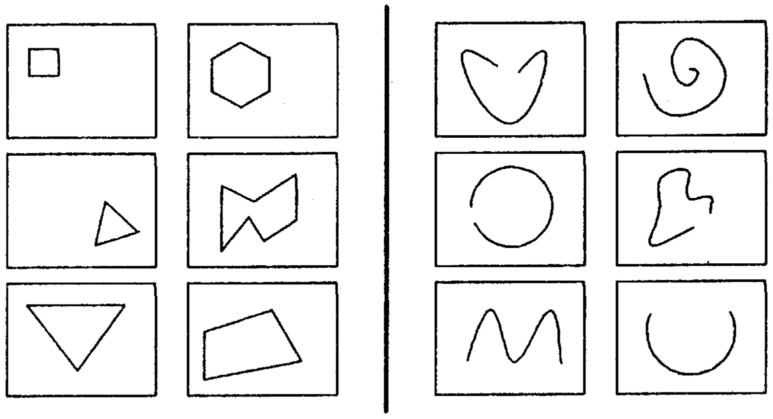

Figure 6-1: A problem in the style of Bongard.

Another characteristic of Bongard problems is that they provide a test of decomposition of properties. There is no prescription for decomposition, but if a decomposition exists, two Bongard problems that distinguish the two properties can be constructed and tested. Let's take the example in figure 6-1. The figures on the left are all **closed** and the ones on the right are all **curved.** Thus, there are at least two solutions to the problem: **closed** versus **open**, and **rectilinear** versus **curved.** If two solutions exist, and can be found experimentally, the problem can be decomposed into two: one distinguishing the property **closed** from **open**, and the other distinguishing **rectilinear** from **curved.** Figures 6-2 and 6-3 are tests of this decomposition.

Treisman [Treisman and Gelade 80; Treisman and Schmidt 82] also has designed a paradigm which results in the decomposition of objects into more primitive properties, but the target properties in each case are represented by only one object, or replicated exactly by several identical objects with no variations. In effect, there are not as many positive and negative examples of the target property to sufficiently narrow its visual definition. The output of the experiments is not a symbolic description in all cases. The paradigm supports findings of the decomposition of some properties, but the emphasis is on finding "illusory conjuncts" rather than on validating symbolic descriptions and finding a complete set of primitives and compositional operators.

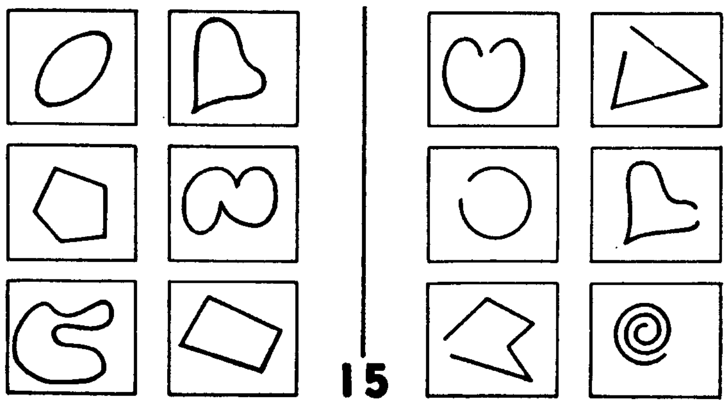

Figure 6-2: Bongard problem #15.

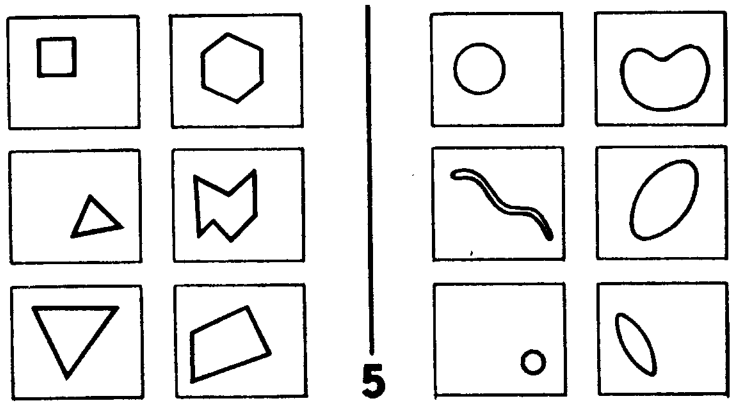

Figure 6-3: Bongard problem #5.

The set of Bongard problems goes further to capture compositional relationships of complex objects as well as primitive properties. One finds properties applied to relationships between objects and properties of properties of objects. Figures 6-4 and 6-5 show examples of properties of properties of sets of objects. Figure 6-4 illustrates the property of **inside** versus **outside** applied to the relationship of **distance** among the small circles. Figure 6-5 again illustrates the **inside/outside** property this time applied to the **sidedness** of polygons. Thus, properties gathered from Bongard problems can be viewed as compositional operators that can be applied recursively to primitive objects to form more complex objects and relationships. Properties isolated in this way illuminate the substructure of higher-level visual properties.

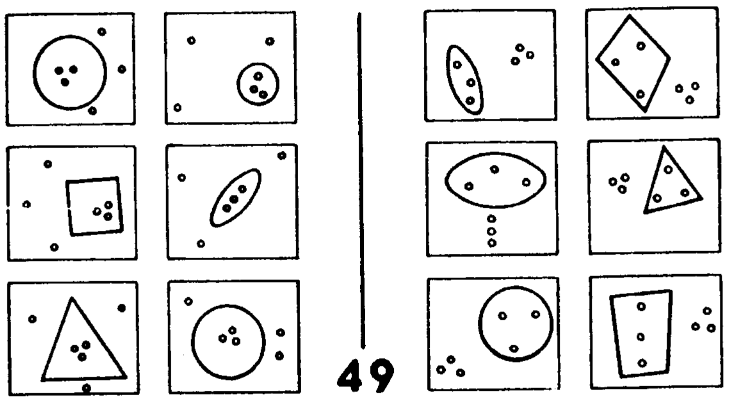

Figure 6-4: Bongard problem #49.

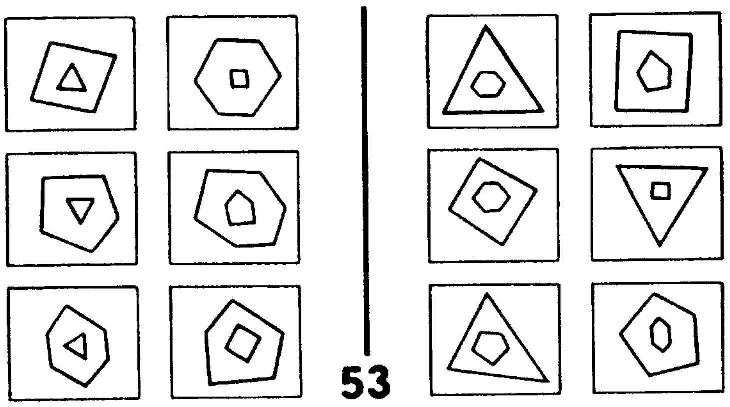

Figure 6-5: Bongard problem #53.

6.3. Vividness and Conciseness

Just as solutions to Bongard problems associate symbols with images, the solutions are influenced by **vividness** in the visual domain and **conciseness** in the symbolic domain. Vividness determines which of the potential properties will be seen first and how fast it will be seen. Conciseness determines which description will be more easily expressed. It appears that the two criteria interact, although the exact nature of their interaction is not clear. Following are two examples of interaction.

In the first example, the only difference between the objects in figure 6-6 [Pomerantz and Garner 73] is the line orientation of the diagonal line. However, the fact that one is **closed** and the other **open** stands out more than the difference in line orientation. The **closed/open** feature is also the more concise description of the two, because even though line orientation is the more primitive and concise feature of the diagonal line by itself, the description of the line in the context of the right-angle vertex is longer when one includes the relationship of the diagonal lines' end points to the other two lines. It is in the whole context that the **closed** versus **open** description becomes the more concise one, because it includes the relationships between the three lines, not just a property of one of the lines. So here the more vivid feature in the image domain is also the more concise feature in the description domain. Vividness appears to influence conciseness.

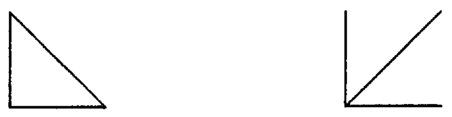

Figure 6-6: A difference between a primitive property and a vivid one.

The second example involves a brief experiment. Glance at the figure on the next page for about a second. Then cover it and try to describe it before reading on. About half of the people to whom

I've shown this figure (including myself), when constrained by a short glance, described the figure as two overlapping pentagons (or regular polygons), each with different width outline. As you can see on closer inspection, there are indeed two overlapping pentagons, and two different weight outlines, but their weight does not correspond one-to-one with pentagons. The pentagons each have thick and thin segments, and the polygons of uniform weight are irregular and six-sided.

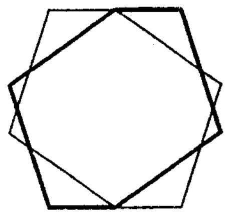

Figure 6-7: Test figure.

My interpretation of this result is that subjects only have time to parse properties: **weight** and **five-sidedness**, and that the process that associates properties with single objects for the purposes of description happens after viewing and without the chance to verify the objects for discrepancies. A description in turn, must be concise in order to be remembered. Building a description of these irregular polygons is difficult and presumably requires more than one second of scrutiny. A pentagon is much more concise. One is forced to construct a plausible object to account for the simple properties already parsed, because properties do not exist by themselves. So an object with five sides and thick borders is inferred even thought one does not exist in the figure. The lack of conciseness of the description of the actual figure appears to influence the way we see, i.e. it's vividness. Again vividness and conciseness seem to go hand in hand, but in this case conciseness seems to be influencing vividness.

7. Properties Versus Objects as Primary Representations

The irregular polygon example of the last section shows how early recognition of properties can influence later object description. Recognition of objects appears not to be as straightforward as introspection suggests. It seems to be a complex inference consisting of a judgment that a set of perceptual properties come together at a specific location and time. The judgment may be wrong, and yet the correct properties may survive identification. Treisman's work [Treisman and Gelade 80; Treisman and Schmidt 82] also supports the claim that properties are more primitive and less divisible than objects.

I think this suggests that the properties of objects are more primitive than objects, or that their recognition occurs prior to object recognition. Objects are composite entities. In order to construct more complex objects from primitives, we need to begin with properties not objects, because objects are already composite things.

An object is an instance of a set of objects or object class. A set of objects can be *described* by a property: all the *red* objects, for example. The way properties are used is relative to the task at hand. In a recognition task an object becomes instantiated only at the end of the process. In a reasoning task, in which a prototypical object must be imagined, an object plays a very different role. An object is a theory about the properties relevant to a reasoning task. The theory, and thus the object, must first be instantiated in order to reason about it. The theory will probably change as a function of the reasoning process. In both cases, visual objects are highly structured, complex, and task relative. So the old example of Clyde, the gray elephant, generates a contradiction because the problem is first posed as a prototypical reasoning problem, but is implicitly meant to generalize to a recognition problem [Fahlman 77; Brachman 85], e.g. all elephants, even pink ones. That our representations of concrete objects are immutable and identical in both recognition and reasoning tasks is an illusion. Theories can change and identification does not require stable objects.

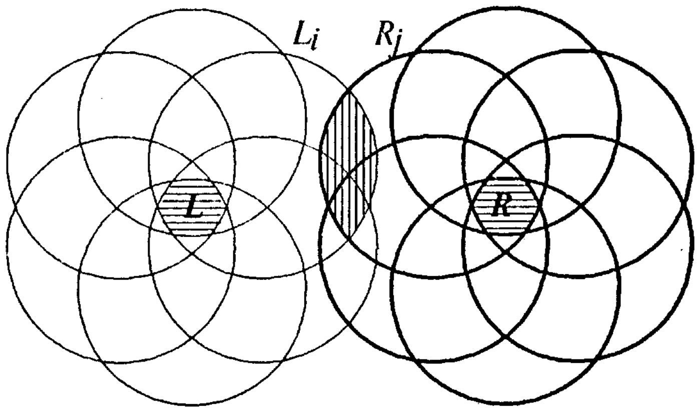

Figure 7-1: A Venn diagram representing each object in a Bongard problem as a set of properties. The thin circles represent the objects on the left and the thick circles represent the objects on the right.

A set of properties *describes* a set of objects. As more properties, not implied by the initial set, are added, a more specific set of objects is described. We can think of the objects in a Bongard problem in this formalism as a set of overlapping sets of properties in property space, one set for each object. (See figure 7-1.) Some sets for the objects on the left of the dividing line, L_i, may overlap some sets for the objects on the right, R_j. Thus, $L_i \cap R_j$ is typically non-empty for some *i,j*, as shown (the vertically cross-hatched area). An example of $L_i \cap R_j$ in problem #3 (figure 5-1) would be the property of **triangularity**. Let $\boldsymbol{L}$ be the intersection of all the objects on the left, $\boldsymbol{L} = \bigcap_{i=1}^{6} L_i$, and $\boldsymbol{R}$ be the intersection of the ones on the right, $\boldsymbol{R} = \bigcap_{i=1}^{6} R_i$. In order to solve the problem we must find some property, $p \in \boldsymbol{P}$, that is in $\boldsymbol{L}$ or $\boldsymbol{R}$ but not in their intersection, $\boldsymbol{P} = (\boldsymbol{L} \cup \boldsymbol{R} - \boldsymbol{L} \cap \boldsymbol{R})$ (the horizontally cross-hatched area in figure 7-1). In figure 7-1, $\boldsymbol{L} \cap \boldsymbol{R} = \varphi$, but in reality there are many properties shared by both sets: they're all black and white, they're on the same page, they're in the same Bongard problem, they're 2-D, etc. The smaller the sets $\boldsymbol{L}$ and $\boldsymbol{R}$, or the greater their overlap, the smaller the search within $\boldsymbol{P}$ for a distinguishing property. In the light of this formalism, Bongard problems are chosen so as to specify ***P***, the space of target properties, as narrowly as possible. If there is an ambiguity as to which property in ***P*** best describes the set ***L*** or ***R***, the available choices are usually not very numerous. So the search for the most concise description of the distinguishing property is not very large once ***P*** is found. However, the confounding factors turn out to be properties in $L \cap R$, the properties common to all figures, or near misses, $L_i \cap L_j \cap R_k$, for some $i \neq j,k$. So if $L \cap R$ is large, the search can be long.

8. What's in a Graph?

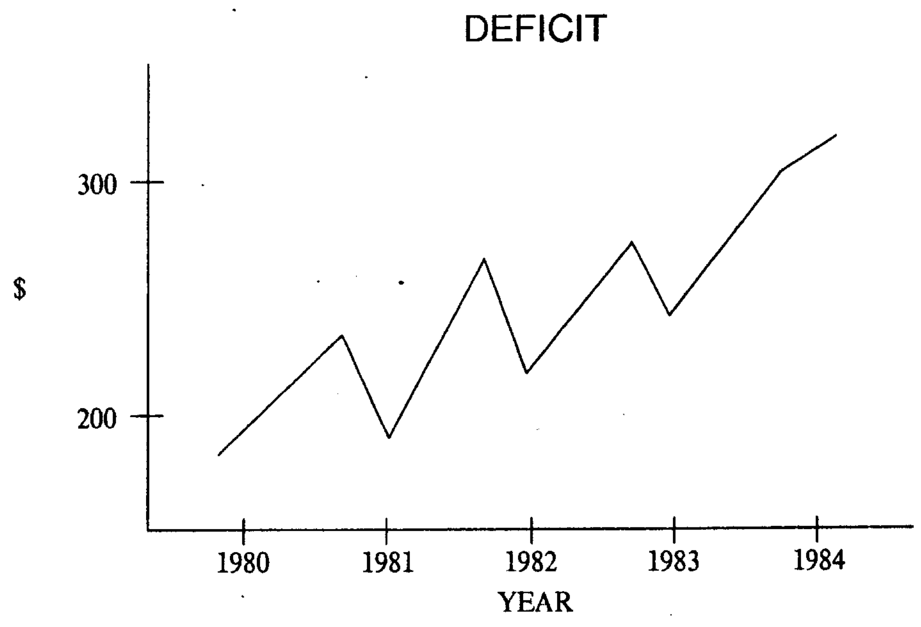

Figure 8-1: A typical x-y plot.

Given the kind of graphics tool kit introduced in the previous sections, can we identify the primitives that make up a simple graph, for example, one with axes and a plot of a budget deficit, as in figure 8-1? A traditional graph constitutes a very complicated set of relationships. First, there are the primitive visual objects that the graph is composed of, such as line segments, numerals, and text. Numerals and text introduce the **labeling** relationship into the figure. They don't stand by themselves as visual objects. They are associated with specific graphic objects in the figure. Figure 8-2 shows some of the labeling relationships in 8-1: one between numerals and a tick-marked line, another between an axis label and the axis, and a third between the plot label (**DEFICIT**) and the whole graph. The *labeling* relationship is analogous to the *denotation* relationship, in that graphic objects represent application domain objects, except that the application domain objects in this case are represented textually in the graph as labels. The labeling relationship is an *explicit* representation of the denotation map.

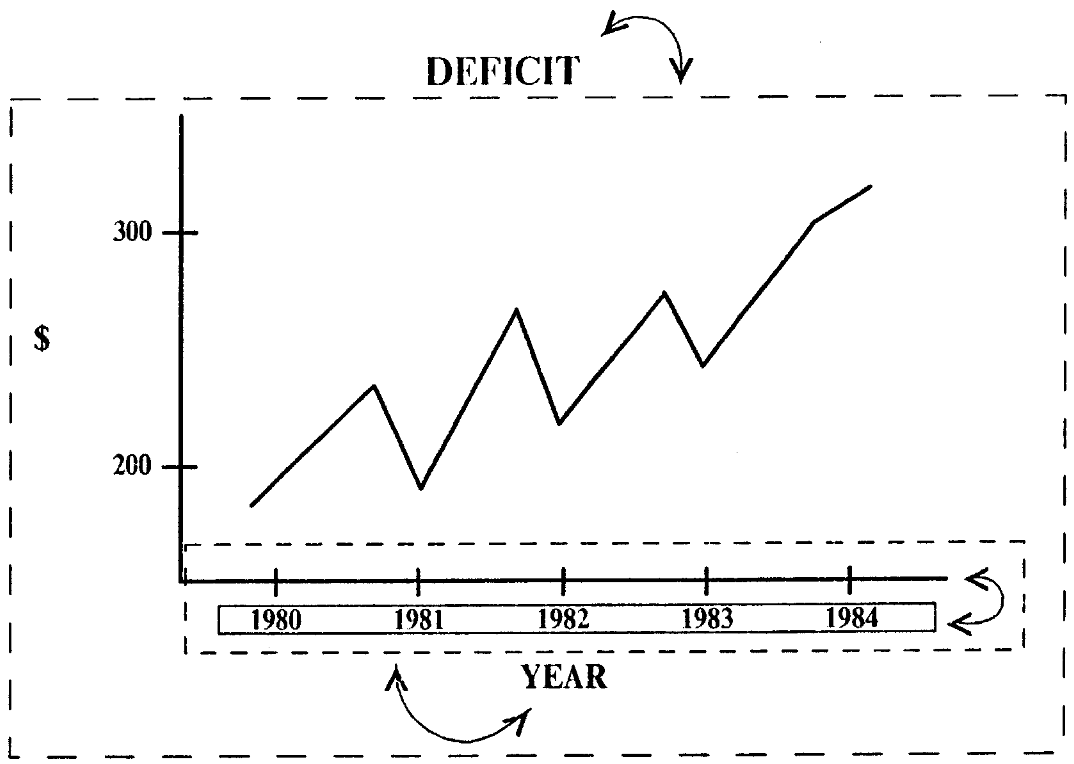

Figure 8-2: The plot of figure 8-1 with labeling relationships shown.

Secondly, the graph exhibits denotational relationships between distances on the plot and real world quantities: horizontal distance representing time and vertical distance representing money. There are similar relationships between the x- and y-axes and abstract numerical scales. It is very easy to get confused here. The axes do not *a priori* represent time and money; such representation is a consequence of the labels associated with them. The axes themselves represent only abstract numerical scales. (See figure 8-3.)

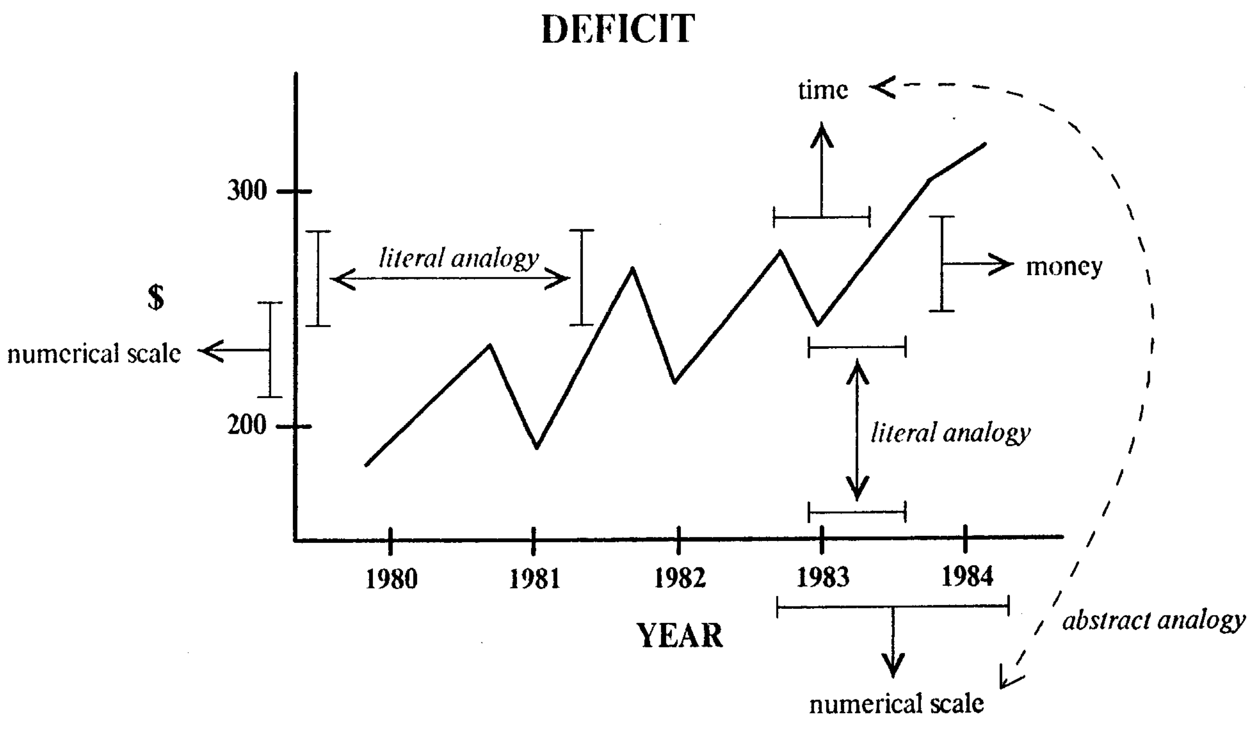

Figure 8-3: The plot of figure 8-1 with denotation relationships and analogies shown. A literal analogy in distance induces the abstract analogy.

Finally, there are analogies in both x and y, between actual real world objects, such as, time and money, and abstract numerical scales. Horizontal and vertical distance on the plot is aligned with x and y distance on the axes. Distance constitutes the *literal analogy* that drives the more abstract analogies between quantity of money and the y-axis in one dimension, and between years and the x-axis in the other (figure 8-3). Because a graph is such a stereotypical way of representing quantities, these analogies may seem so obvious as not to appear to be analogies at all. However, if we try to dissect the primitive structure of graphs into a minimal set of concepts, their essence as analogies is revealed.

There appears to be a complicated web of denotational relationships in a typical graph, as well as a literal analogy between just the visual properties of distance. This analogy between distances induces the association of semantic properties. The result is that we read the height of the curve directly as 200 or 300 billion dollars without noticing the many layout relationships that underlie these subtle analogies. These analogies, in turn, allow us to make the inference that height equals dollars.

9. Conclusion

I have introduced the area of Diagram Understanding analogous to Natural Language Understanding. Although previously not named as such, it concerns itself with issues of both the recognition and generation of visual objects by a computer system. Traditionally, these areas have been called Computer Vision and Computer Graphics, respectively. But there is a core issue in this area: the symbolic description of higher-level visual objects and its association with concepts generated by people. In an interactive system, the user can have a diagrammatic conversation when the right sorts of associations are provided. The requirements of a smooth conversation drive the representational issues toward natural, human-based structures. The need for human interaction motivates a test-bed system having both recognition and generation of visual objects, for the purposes of studying higher-level representation. There is no guarantee that the denotational loop will converge unless we have humans somewhere in the loop. There are two ways of doing this. One is to test entire systems for smoothness of interaction, as has been done in the Natural Language Understanding paradigm. Another is to build systems out of small modules that have been tested individually, and whose compositional operators have also been tested against human perception.

In such a system, denotation itself is illusive and ambiguous. It is necessary to have a system that can at least represent these various possibilities of reference in order to have it do the necessary level switching. Various kinds of reference confusion have been identified and illustrated with examples of format , notational , and semantic manipulation.

A method of visual knowledge acquisition has also been presented. It involves an experimental paradigm that makes descriptive properties explicit. Compositional operators can also be explored and isolated with this paradigm. Properties are seen as more basic than objects in a vocabulary for Diagram Understanding.

If we are to dissect visual objects into primitive substructure, we have to at least be able to describe the most commonly used type of graphics display in these primitive terms. An example of the complex nature of a standard business graph has been shown. With it we outline the many, subtle relationships between the parts. Fundamentally, however, a graph can be seen as the association of visual properties with some application domain properties. Proximity and alignment relationships induce analogies between vertical and horizontal distances.

A kit of visual properties and operators is proposed to be used by domain experts to attach graphics to specific domains. It is argued that static graphics interfaces cannot work for everyone at all times. The best solution is for domain experts to design and modify their own interfaces. They are the most proficient in visualizing their own domain, and as need varies, their visualizations can also vary. It is impossible for graphics designers to keep up with changing visualizations of abstract concepts dreamt up by domain experts. Graphics interfaces should be as flexible and versatile as paper and pencil, and as easily used. In order to provide such generality, descriptive primitives specific to human vision should be provided. The implementation of recognizers need not be the same as in human visual systems, but the correspondence between human descriptions and visual objects must be the same.

10. Acknowledgments

Many thanks to Gul Agha, Phillip Apley, Michael Brady, Donald Byrd, Randall Davis, Carl Hewitt, Douglas Hofstadter, Henry Lieberman, Marvin Minsky, Alex Pentland, Thomas Reinhardt, Aaron Sloman, and Patrick Winston for inspiration, guidance, and helpful criticism.

References

[Barsky 82] Barsky, B.A. Computer-aided geometric design. *Computer Graphics 16*, 1 (May 1982).

[Bongard 70] Bongard, M. *Pattern Recognition.* Hyden Book Co. (Spartan Books), New York, 1970.

[Borning 79] Borning, A. THINGLAB - A Constraint-Oriented Simulation Laboratory. Ph.D. Th., Stanford University, 1979.

[Brachman 85] Brachman, R.J. 'I lied about the trees' or, defaults and definitions in knowledge representation. *AI Magazine 6*, 3 (1985).

[Ciccarelli 84] Ciccarelli, E.C. Presentation Based User Interfaces. Ph.D. Th., Massachusetts Institute of Technology, A.I. Lab., 1984.

[Fahlman 77] Fahlman, S.E. A system for representing and using real-world knowledge. Ph.D. Th., Massachusetts Institute of Technology, A.I. Lab., 1977.

[Filman, Lamping, and Montalvo 83] Filman, R.E., Lamping, J., and Montalvo, F.S. Meta-language and meta-reasoning. Proc. of the 8th IJCAI, IJCAI-83, Karlsruhe, West Germany, August, 1983.

[Friedell 84] Friedell, M. Automatic synthesis of graphical object descriptions. *Computer Graphics 18*, 3 (July 1984).

[Friedell, Barnett, and Kramlich 82] Friedell, M., Barnett, J., and Kramlich, D. Context-sensitive, graphic presentation of information. *Computer Graphics 16*, 3 (July 1982).

[Glicksman 82] Glicksman, J. A cooperative scheme for image understanding using multiple sources of information. Tech. Rep. 82-13, Dept. of CS, U. of British Columbia, November, 1982.

[Labov 73] Labov, C.N. The boundaries of words and their meanings. In *New Ways of Analyzing Variation in English*, C.N. Bailey and R.W. Shuy, Eds., Georgetown University Press, Washington D.C., 1973.

[Levoy 81] Levoy, M. Computer Animation Tutorial Notes. SIGGRAPH '81 Conference, Dallas TX, August, 1981.

[Montalvo 76] Montalvo, F.S. Aftereffects, adaptation, and plasticity: a neural model for tunable feature space. Ph.D. Th., University of Massachusetts, 1976.

[Morse 84] Morse, A.C. A system for embedding data displays. Tech. Rep. 84-7-1, Visual Intelligence Corp., July, 1984.

[Pentland 84] Pentland, A.P. Shading into texture. Proc. of the Nat.Conf.on AI, AAAI'84, Austin TX, August, 1984.

[Pomerantz and Garner 73] Pomerantz, J.R., and Garner, W.R. Stimulus configuration in selective attention tasks. *Perception and Psychophysics 14* (1973).

[Ramsdell 80] Ramsdell, R.E. Power of VisiCalc. *Byte 5*, 11 (November 1980).

[Shoup 79] Shoup, R.G. Color table animation. *Computer Graphics 13*, 2 (Aug. 1979).

[Sloman 85] Sloman, A. What are the purposes of vision? Presented at the Alvey Vision Conference at Sussex University, September, 1985.

[Smith 78] Smith, A.R. Paint. Tech. Rep. 7, NYIT, July, 1978.

[Treisman and Gelade 80] Treisman, A.M., and Gelade, G. A feature-integration theory of attention. *Cognitive Psychology 12* (1980).

[Treisman and Schmidt 82] Treisman, A.M., and Schmidt, H. Illusory conjunctions in the perception of objects. *Cognitive Psychology 14* (1982).

[Weyhrauch 80] Weyhrauch, R. Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence 13*, 1,2 (April 1980).

[Winograd 83] Winograd, T. *Language As a Cognitive Process, Vol. I: Syntax.* Addison-Wesley, Reading MA, 1983.

[Zdybel et al. 81] Zdybel, F., Greenfeld, N.R., Yonke, M.D., and Gibbons, J. An information presentation system. Proc. of the 7th IJCAI, IJCAI-81, Vancouver, B.C., Canada, August, 1981.

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 873 / 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Diagram Understanding:The Intersection of Computer Vision and Graphics	5. TYPE OF REPORT & PERIOD COVERED AI-Memo
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Fanya S. Montalvo	8. CONTRACT OR GRANT NUMBER(s) DEC & DARPA/ONR contract N00014-80-C-0505
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209	12. REPORT DATE November, 1985
	13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

vision	diagram understanding
graphics	knowledge-based graphics
representation	knowledge acquisition
shape	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A problem common to Computer Vision and Computer Graphics is identified. It is the problem of representing, acquiring, and validating symbolic descriptions of visual properties. The intersection of Computer Vision and Computer Graphics provides a basis for diagrammatic conversations between users and systems. I call this problem domain Diagram Understanding because of its analogy with Natural Language Understanding. The recognition and generation of visual objects from symbolic descriptions are two sides of the same coin. A paradigm for the discovery and validation of higher-level visual properties is introduced. The para-

ABSTRACT CONTINUED

digm involves two aspects. One is the notion of denotation: the map between symbolic descriptions and visual properties. The denotation map can be validated by focus on the *conversation* between users and a system. The second aspect involves a method for discovering a natural and rich set of visual primitives. The notion of visual property is expanded, and the paradigm is further illustrated with a traditional business graphics example.