Perceptually Based Learning of Shape Descriptions for Sketch Understanding

by

Olga Veselova

B.S. Computer Science
University of Washington, 2001

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Science in
Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

May 2003

Signature of Author: ______________________________
Department of Electrical Engineering and Computer Science
May 29, 2003

Certified by: ______________________________
Randall Davis
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: ______________________________
Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

Perceptually Based Learning of Shape Descriptions for Sketch Understanding

by

Olga Veselova

Submitted to the Department of Electrical Engineering
and Computer Science on May 29, 2003
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

ABSTRACT

We are interested in enabling a generic sketch recognition system that would allow more natural interaction with design tools in various domains. Instead of writing recognizer code for each new domain, new shapes should be added by describing them in a shape description language. While writing such descriptions is easier than writing code, it is still not a particularly easy or natural mode of interaction. The most natural way to teach new symbols to the system would be simply drawing them. This thesis presents a learning system that takes in a drawn symbol and produces a textual description of it appropriate for using in a recognition engine. The main challenge is to decide which properties of the example are relevant. People cope with this task in part, we believe, through innate perceptual biases. We use studies of human perception of geometry to understand these biases and use them to help select the relevant properties from a single example. The main generalization power of the system is derived from two sources: 1) a qualitative description vocabulary that reflects properties that people pay attention to and 2) mechanisms, derived from the observations about perception, that adjust the relative importance of different properties based on the overall configuration of the geometric primitives in the example. Using this approach the system is able to adequately describe complex symbols by identifying a small number of relevant properties.

Thesis supervisor: Randall Davis
Title: Professor of Electrical Engineering and Computer Science

Acknowledgements

- I would like to thank my research adviser Randy Davis for all the help and inspiration that he provided throughout this work and during my time at the AI Lab in general. Randy has helped me figure out a lot of difficult problems and provided great guidance in this research, preventing me from straying into “bottomless pits” and helping me keep focus. I am very grateful for his patience, support, encouragement, feedback, and all the things he helped me learn.

- I would like to thank the students at the Design Rationale Group. It has been a great experience working with them and learning from them.

- I would like to thank sponsors of Project Oxygen for financial support.

- Finally, I would like to thank all my friends at the MIT AI Lab for the advice and discussion of many of the ideas that went into this research.

Chapter 1 Introduction

1.1 Research context: multi-domain sketch understanding

Informal sketches are often an important part of early stage design in many domains [Ullman, 1990]. Sketching helps people explore new ideas, brainstorm designs and reduces cognitive load of the design process. Many designers still use pen and paper for trying out ideas, since CAD tools available to date do not accept free-hand input. These tools require precise specification of all parameters and well-formed designs. Only when the design matures, can it be entered using a CAD tool for more detailed analysis and documentation. Often the valuable information about the design intent expressed in the paper sketches never gets documented. The designers also lose the benefit that computers could potentially provide even at the early stages of design. Useful analysis, qualitative simulations, or exploration of alternatives can be done even on a rough sketch, if only the computer could recognize the objects sketched.

We feel that interaction with design tools could be made more natural if they not only provided powerful analysis of precise designs, but also recognized sketched input at the early design stages.

The work reported here is part of the effort by the Design Rationale Group (DRG) that has developed sketch understanding systems for several design domains including mechanical engineering and software [Alvarado and Davis, 2001], [Hammond and Davis, 2002]. Those systems used hand-coded recognizers for the domain shapes, which made creating a system for each new domain or adding more shapes very tedious and time-consuming.

The DRG is currently interested in enabling generic sketch recognition [Alvarado and Davis, 2002], and is building a system that would reuse the recognition engine for multiple domains. The intent is that a new domain can be added simply by providing descriptions of the domain symbols using a shape description language. Each symbol is described in terms of geometric primitives (lines, arcs, ovals, etc.) and constraints between them (connects, parallel, above, horizontal, shorter, etc.) [Hammond and Davis, 2003]. Symbolic, easily readable textual descriptions make shape representation explicit and allow any user to define new symbols.

While being able to type new shape descriptions is clearly easier than writing code, describing shapes textually is itself not a particularly natural mode of interaction. This thesis describes a system we have developed that is capable of learning a symbolic description of a shape from the user's drawing. The system provides a way to automatically produce the textual descriptions needed by the generic recognition engine from examples provided by the designer of the domain. These descriptions can be further checked or edited by the designer, if required. Figure 1.1 presents the overall view of the generic sketch understanding system and shows the role of our work.

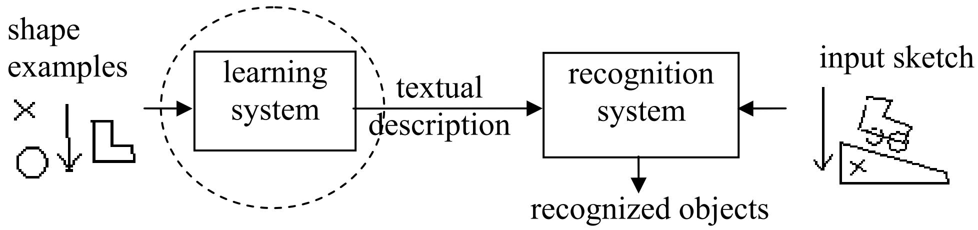


Figure 1.1 Generic sketch understanding system

1.2 The learning problem

Like handwritten characters, symbols in commonly used graphical languages can be drawn with some variation. For instance, all of the drawings in Figure 1.2 are examples of an inverter symbol in electric circuits:

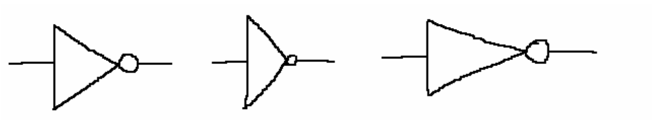


Figure 1.2 Variations of the inverter symbol

Despite the variations, there are important properties that are going to be present in all the examples, such as the lines forming the triangle or the relative size of the circle and the triangle, and unimportant properties that can be varied, such as the relative sizes of the sides of the triangle. We are faced with a classic problem in learning from examples: how can we generalize, i.e., how can we identify which subset of properties is relevant?

One common approach to this is to ask the user to draw the symbol numerous times (e.g., hundreds of times for neural nets), in the belief that the inessential elements will "average out." We find this undesirable for our task of teaching new symbols to the system. The system would be more natural if one could interact with it as if communicating with another person. And typically, seeing one example of each of the symbols in the domain is sufficient for people to learn them. Furthermore, even if people see only one symbol without knowing the other symbols in the domain, they are able to extract enough information to often make a correct decision on whether some new drawing should be recognized as the symbol or not. Aiming at achieving this capability, in our work we have focused on the problem of learning as much as possible from a single example.

1.3 Motivating example

Consider how people learn new symbols such as the one in Figure 1.3.

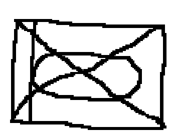


Figure 1.3 Symbol for mechanized infantry used in military planning diagrams

Most people would describe this symbol as a rectangle with diagonals, with an oval in the center and a vertical line adjacent to the oval. A single example is often enough to understand the structure of the symbol. People are likely to recognize it again, even if drawn with some variations (Figure 1.4). The goal of the learning system is to do the same, producing a description of the symbol that is adequate for later recognition.

Figure 1.4 Perceptually similar symbols

Both instances of the mechanized infantry symbol in Figure 1.4 differ from the original example (e.g. in the aspect ratio of the rectangle, the orientations of the slanted lines, and the relative size of the oval). Yet most people would recognize these instances. They do not pay attention to the exact values of the varied properties in the original example from Figure 1.3.

To understand what properties people attend to we have turned to studies of human perception and memory of geometric shapes. We looked at Goldmeier's studies of similarity [Goldmeier, 1972], [Goldmeier, 1982], Arnheim's work on art and visual perception [Arnheim, 1974], and the perceptual grouping principles identified by the gestalt psychologists [Wertheimer, 1923]. Inspired by the phenomena described in these bodies of work and following our own introspection, we have developed a number of heuristics for ranking different geometric properties on perceptual saliency. We show that they are an important step towards matching people's ability to learn from one example.

Our approach clearly depends on the assumption that the drawings in Figure 1.5 are in fact to be interpreted as the same symbol.

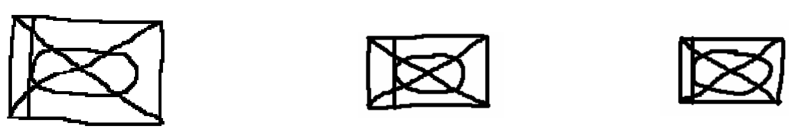


Figure 1.5 Perceptually similar symbols

We feel that it is reasonable to assume that the above figures should be recognized as the same symbol, because similarity and perceptual saliency play an important role in the design of graphical languages. If two symbols that are perceptually similar – i.e. differ on a property that people don't pay attention to – it would be unwise to use them to mean different things. They would be easily confused and the difference would be hard to remember. We thus suggest that a well-designed graphical language is unlikely to contain such ambiguous symbols.

1.4 Domain-specific knowledge

Geometric saliency is not the only source of people's capacity to learn symbols. In some cases we also use domain-specific information. Consider the symbol of an AND-gate in Figure 1.6.

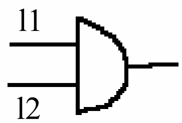


Figure 1.6 AND-gate symbol

When students first learn this symbol in a logic design class they know that the lines labeled l1 and l2 do not have to be the same length because they represent wires. Domain-specific knowledge and graphical conventions sometimes help identify which properties are not important, even if these properties are perceptually salient. Our system currently does not incorporate such knowledge. We feel that relying only on geometric information is still a step in the right direction. There are domains, like military diagrams, where most of the symbols are abstract and do not resemble the objects they represent (like the military symbol in Figure 1.3). Most people would still be able to learn these symbols from one example, using only the geometric clues. In the future, the system could be extended to incorporate domain information or common conventions.

1.5 Measure of success

Ideally, the measure of success for the system is whether the descriptions produced are adequate for recognition. By adequate we mean that the description would cause the recognition engine to admit all and only the instances that the user intended to be recognized when teaching the system an example of the symbol. As the system uses only geometric information, it is bound to make domain-related errors in some cases. For example, it would conclude from Figure 1.6 that lines l1 and l2 have to be the same length. Hence, we prefer to evaluate the system's descriptions by comparing them to the geometric properties a person shown the same symbol would pay attention to, without taking into account the knowledge of the domain or of how the symbol is to be used.

One way to test this is to show people a symbol from an unfamiliar domain and to ask whether different variations of it should be recognized as the original symbol. The variations people accept should match the description produced by the system, and the variations they reject – should not. We have conducted such a study with several military planning symbols. On examples with high agreement between the subjects, the system achieved 83% accuracy (i.e. it agreed with the majority answer 83% of the time). We describe the study in more detail in Chapter 6.

1.6 Example and approach

This section illustrates the system's performance on a simple example. Suppose the user would like to teach the system the symbol in Figure 1.7.

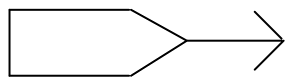
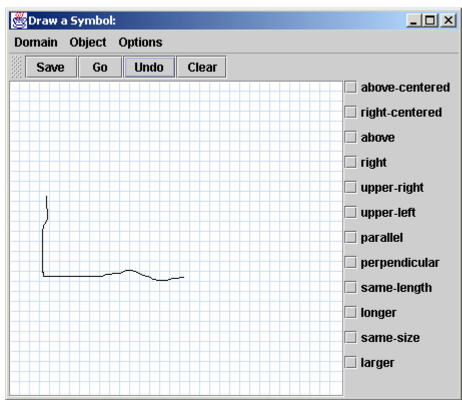


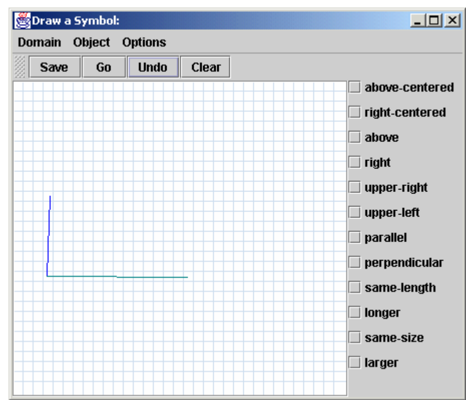
Figure 1.7 Military planning symbol

Our system expects the user to draw carefully – i.e. the lines that the user intends to be straight, perfectly vertical (or horizontal), or well connected, should be drawn that way and only a small amount of noise is allowed. We think this is a reasonable requirement for the teaching phase, since the symbol has to be drawn only once.

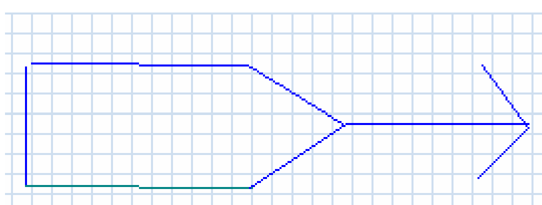
As the user draws the symbol (with a mouse or pen-based input), each individual stroke is segmented into simple geometric primitives – lines and ovals – using pen-speed and stroke curvature data [Sezgin, 2001]. After the drawing is completed the user presses "Go" to generate the description.



a)



b)



c)

Figure 1.8 a) Single stroke. b) Segmentation into geometric primitives. c) Completed symbol

The system straightens out the lines that are almost horizontal or vertical and connects line endpoints if their separation is within a small threshold. The straightened and labeled primitives are shown in Figure 1.9:

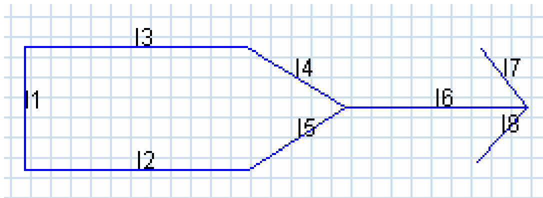


Figure 1.9 Straightened and labeled primitives

Next, the system finds all pairwise constraints that hold in the drawing. The constraints do not have to hold exactly: the system includes small thresholds on distances and angles to account for noise. For example, even if there is a small horizontal offset between the centers of lines l2 and l3, the system will consider these centers to be on the same vertical line.

There were 96 constraints found for the symbol above (see Appendix A). The challenge is to pick only the relevant subset of these constraints for the description. For example, both of the constraints "same-length (l4 l1)" and "same-length (l4 l5)" hold in the drawing. However, people would typically include only the second of those in their description of the symbol.

The system uses several mechanisms to generalize the description (i.e. filter out irrelevant constraints), inspired by the results of psychological studies and our introspective analysis. We give a brief summary of these mechanisms and provide more details in Chapters 3 and 5:

- **Qualitative vocabulary:** Initial generalization is achieved by using qualitative terms to describe constraints and properties. For example, the orientation of a line is described as "horizontal", "vertical", "positive-slope", or "negative-slope."
- **Different default relevance scores:** Different types of constraints have been shown by psychologists to have different perceptual importance. For instance, the structural composition of the primitives in the symbol is more important than their individual properties. In recognition of this, for example, the system assigns higher relevance scores to "connects" constraints than "longer" constraints.
- **Score adjustment based on global properties:** The system increases or decreases the relevance score of each constraint using three heuristics that analyze the global properties of the symbol:
  1. *Obstruction*: This heuristic relies on the assumption that it is harder to pay attention to constraints between two primitives if several other primitives separate them (create obstruction). For example, in Figure 1.9 there are several lines between lines l1 and l7. Hence, the relevance of constraints like "longer (l1 l7)" will be decreased.
  2. *Tension lines*: People pay attention to horizontal and vertical alignments of primitives. We call such alignments tension lines. We increase relevance of constraints breaking the alignment, if violated. For example, in Figure 1.9 line l3 is centered above line l2. Their endpoints are aligned vertically. The

relevance of "above-centered" and "same-length" constraints would be increased even if these primitives were separated by several others.

3. *Grouping*: People tend to group primitives together and see them as a whole. The system currently supports grouping by connectedness and familiarity of shape. When people see several primitives as one whole object, they pay less attention to individual interactions of primitives that form different objects. The system decreases the relevance of a constraint on a pair of primitives if they belong to different groups. In Figure 1.9 lines l6, l7, and l8, are recognized as the previously learned symbol "right arrow", so the relevance of constraints like "longer l6 l4" is decreased.

The system uses these mechanisms to calculate a relevance score between 0 and 1 for each constraint. We picked the middle of the range to be the cut-off threshold. Only constraints that end up with a score above 0.5 are considered relevant enough to remain in the final description of the symbol. Examine the description shown below produced for the symbol in Figure 1.9:

The GROUP HIERARCHY part of the description represents how the symbol was broken down into groups of elements – perceptual grouping. Each group is what the system interprets as a perceptual unity – a separate object within the symbol – either because all of the elements are connected, or because they form a symbol that the system has been taught before (the system has mechanism for recognizing such symbols). In this case, the whole symbol (Group g1) is a connected component which in turn consists of two groups: the "right arrow" (Group g2), which is a previously learned symbol, and a connected group formed by the rest of the elements (Group g3). This grouping influences the relevance score of various constraints according to the grouping heuristic mentioned above.

The CONSTRAINTS part shows all the constraints on the groups and individual elements of the symbol that the system considered relevant. Only half of the original constraints got a relevance score above the filtering threshold and remained in the description. Note that the constraints pertaining to the arrow like "horizontal: l6" are not included, since they are already specified in the description of the previously learned "right arrow" symbol.

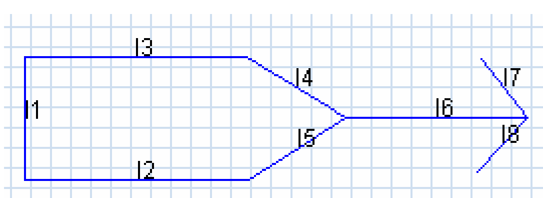


Figure 1.10 Military planning symbol

GROUP HIERARCHY: Group g1 connected-component: l5 l4 l3 l1 l2 l6 l8 l7 Group g2 symbol - right arrow: l8 l7 l6 Group g3 other: l5 l4 l3 l1 l2 CONSTRAINTS: elongated: (g3) connects: (l5.p2 l6.p1) (l4.p2 l5.p2) (l4.p2 l6.p1) (l3.p2 l4.p1) (l2.p2 l5.p1) (l1.p1 l3.p1) (l1.p2 l2.p1) horizontal: (l3) (l2) vertical: (l1)	pos-slope: (l5) neg-slope: (l4) right: (l5 l1) (l4 l1) upper-right: (l5 l2) (l4 l1) (l4 l2) (l3 l1) upper-left: (l3 l4) (l3 l5) (l1 l2) above-centered: (l4 l5) (l3 l2) same-length: (l4 l5) (l2 l3) longer: (l3 l1) (l3 l4) (l2 l1) (l2 l5)

This description reasonably captures the salient properties of the symbol. It would cause the recognition engine to admit all the variations of the symbol in Figure 1.11 and reject the variations in Figure 1.12.

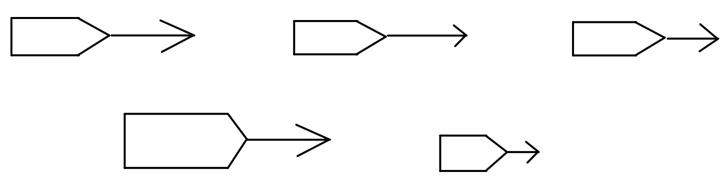


Figure 1.11 Variations that would fit the description

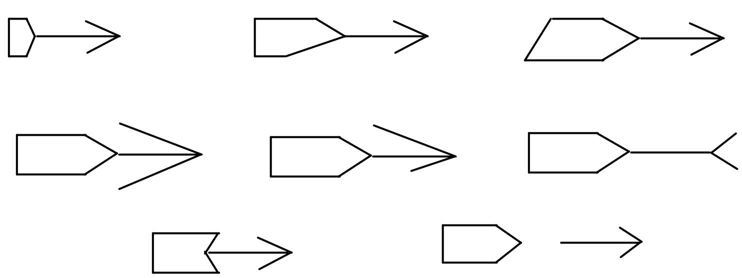


Figure 1.12 Variations that would contradict the description

We have also started exploring ways of displaying the system's conclusions graphically, in order to enable the user to check the result without having to read the textual description. The user selects line l3 and the system shows all constraints related to this line (Figure 1.13). Short double dashes indicate the "same-length (l3 l2)" constraint and the dashed line indicates relative position and center alignment – "above-centered (l3 l2)".

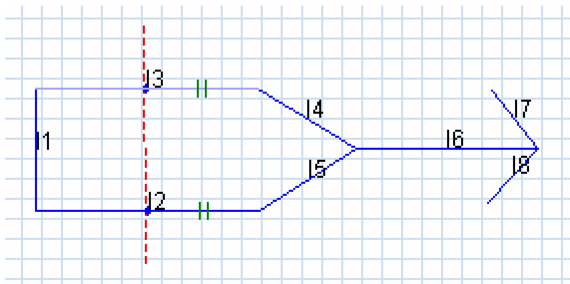


Figure 1.13 Graphical display of the constraints that the system considers relevant.

1.7 Scope and limitations

There are a number of limitations in the current sytem:

- The system currently supports only symbols composed of lines and ovals. It can describe symbols that can be expressed in terms of qualitative constraints, like "same-length" vs. "longer". So for example, it would not be able to learn a constraint like "three times longer." The assumption built into the system is that the exact length ratio is not likely to be the important distinguishing feature between two symbols in a typical graphical language, since a difference between two length ratios would be perceptually hard to notice for people – unless, of course, this difference is too large.

- Our qualitative vocabulary lumps several property values into one term and does not capture that some values may be "too much." The system would describe the relative size of the circles in Figure 1.14a as "larger o1 o2". The drawing in Figure 1.14b would fit the description, even though most people would probably say that the difference between the sizes is too large for Figure 1.14b to be recognized as an instance of the symbol in Figure 1.14a.

Figure 1.14 a) Original symbol. b) Variation that fits the description of the original symbol

A potential solution would be to add a "much larger" constraint. However, it may not be easy to define a good boundary between "larger" and "much larger."

- The system uses only positive constraints, i.e. it specifies only which constraints should hold in the symbol. It does not include "must not" constraints. So it would not be able to describe, for example, a closed shape (say, a four-sided polygon) that should not have self-intersections.

- Currently the system uses only pairwise constraints. So certain constraints like interval equality or alignment of multiple elements are not represented, which makes it impossible to properly describe configurations like the one in Figure 1.15:

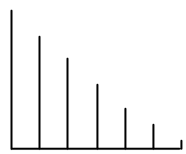


Figure 1.15 A symbol requiring alignment and interval equality constraints

All the limitations mentioned above refer to the system's inability to sufficiently constrain the description of certain symbols. This may create a problem if symbols in the domain are distinguished based only on such properties – for example, if a normal rectangle and a very long thin rectangle are intended to be two different symbols. The system would have the same description for both.

Another set of limitations is related to symbols that the system is bound to overconstrain. Symbols that can have an arbitrary number of certain primitives (Figure 1.16) fall into this category. The system always specifies exactly the number of primitives that a symbol should have. Hence, springs, resistors, inductors, dashed lines, etc. would be impossible to describe properly. We address potential approaches to this problem in Chapter 7.

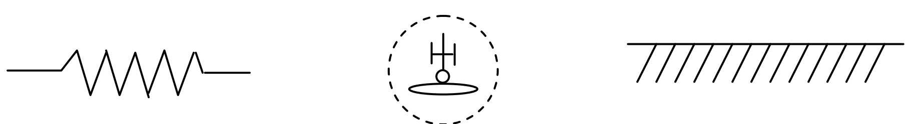


Figure 1.16 Symbols with an arbitrary number of primitives

1.8 Structure of the thesis

Chapter 2 discusses related work on sketching and learning shape descriptions. In Chapter 3 we describe the findings in the perceptual literature that served as inspiration for our approach. Chapter 4 illustrates the performance of the system on several examples and shows how each of the generalization mechanisms is applied. We discuss the details of the implementation in Chapter 5, followed by user study analysis in Chapter 6 and ideas for future work in Chapter 7.

Chapter 2 Related Work

There has been a substantial amount of work on making human-computer interaction more natural by adding interfaces that support free-hand sketching. Work on sketching systems to date falls into two categories: systems that use sketching interfaces without attempting to interpret what the input means and systems that attempt to recognize the sketched objects.

The work in the first category includes systems that transform the user's free-hand input to beautify it [Arvo and Novins, 2000], [Igarashi et al., 1997], systems that support intelligent editing of sketches by allowing perceptually based selection of strokes [LeCun et al, 1995], and systems that allow capturing sketches for documenting designs or knowledge but minimize recognition of shapes, so that the user is free to draw anything [Lin et al, 2002], [Forbus and Usher, 2002].

The systems that are more relevant for our work are the ones that perform recognition of the sketched input. Here we mainly discuss two aspects of these systems:

- **Representation:** We are interested in how the recognized symbols are represented, what features are recorded, and what the descriptive ability of the chosen representation is. For our system we have chosen a symbolic, qualitative representation that corresponds to properties that people typically find perceptually relevant. It is an important source of generalization, because it throws out information on properties that we expect to vary in different instances of the symbol we want to learn. We examine differences and similarities to this approach for the reviewed work.

- **Learning:** We look at how the recognizers for the symbols are acquired, i.e. whether they are specified by hand or can be obtained automatically through training, and if so, how many training examples are required. In our system we learn symbols from a single example, while most of the systems reviewed here need several examples. Yet, some systems are able to learn from much fewer examples than others. So it is important to look at the sources of power for the generalization mechanisms. We believe that in our system, apart from the qualitative vocabulary, one source is the prior knowledge about human perception of geometry. While other systems rely on looking at several examples to "average out" the properties in the symbol that are irrelevant, our system obtains that information from the relevance ranking of the properties based on studies of human perception.

One of the early sketching systems that several other approaches are based on is Rubine's GRANDMA [Rubine, 1991]. Rubine describes a trainable recognizer for single-stroke gestures. Gestures are represented by global features, like length and angle of the bounding box diagonal, the total angle traversed, the sum of the angles at each mouse point, the duration of the gesture, the initial angle of the gesture, etc. The gestures are classified according to a linear function of the features, where the weights are determined during training on multiple examples (typically around 50).

Apart from handling only single strokes, the limitation of this approach is that it uses only aggregate properties of the stroke. The representation does not explicitly capture the detail that may help disambiguate between two gestures with very similar aggregate properties. The representation used in our system makes explicit the properties and constraints on parts of the symbol (like individual lines or ovals).

[Caetano et al., 2002] presents JavaSketchIt – a system that can recognize sketched UI components (buttons, scroll-bars, check-boxes, etc.) and automatically create Java code for them. To recognize UI components, JavaSketchIt uses CALI, a shape recognizer described in [Fonseca et al., 2002]. CALI can recognize simple shapes: squares, rectangles, diamonds, triangles, arrows, crosses, and simple single stroke gestures.

CALI is similar to Rubine's recognizer in that it also uses aggregate properties to represent shapes. Shapes are specified in terms of features of special polygons: enclosing rectangle, convex hull, largest inscribed triangle, and largest inscribed quadrilateral. Using these global features provides certain flexibility. For example, CALI recognizes multi-stoke shapes. Also, shapes can be drawn with overtraced and dashed lines. However, as mentioned above, the recognizer handles only simple shapes. For symbols with more internal detail, like military diagram symbols (see Figure 1.3), these features would be clearly insufficient. Moreover, the number of training examples used to achieve a sufficiently high level of recognition was over 50 for each shape.

Landay and Meyers also describe a sketching tool for designing user interfaces – SILK [Landay and Meyers, 2001]. SILK recognizes sketched interface widgets composed of primitive components – rectangle, squiggly line, straight line, and ellipse. The recognizers for primitive components are based on Rubine's algorithm. A similarity to our system is that SILK uses symbolic spatial relationships (like containment, nearness, and vertical or horizontal sequence) between the primitive components to determine the interface widget that the designer is trying to draw. For example, a scroll-bar is a long skinny rectangle with a box contained in it. However, there is no mechanism for learning these relationships. SILK creators specified the relationships for each UI widget manually. Only the recognizers for the primitive components can be trained (using Rubine's algorithm). Our system, on the other hand, provides mechanisms to learn such relationships from an example. Notice also that the set of spatial relationships in SILK is limited by what is needed for the application at hand. It may be insufficient for describing more complicated symbols in general (for example, the set does not include parallelism or same-length properties).

The Electronic Cocktail Napkin (which is the recognition core of a later system for sketching in conceptual design [Gross and Do, 2000]) uses a two level representation for a symbol similar to that of SILK: low level glyphs and high level combinations of glyphs described by symbolic spatial relationships between them [Gross *et al.,* 1996]. A glyph is a single-stroke or multi-stroke symbol represented by a transition sequence of the pen through the cells of a three-by-three grid. For each glyph the aggregate properties, like allowed number of strokes, number of corners, aspect ratio, and size, are recorded. More complicated symbols can be composed from several glyphs, by specifying spatial relationships between the glyphs. Spatial relationships include adjacency, containment, and overlap of glyphs and intersection, parallelism, and tee conditions for line segments.

This representation allows describing a larger variety of symbols than SILK and CALI. The Electronic Cocktail Napkin also lets the user to specify new glyphs and glyph combinations. Yet, when learning these combinations, the system records all the spatial relationships that it finds for the combination and the user has to manually remove the ones that are unimportant. There is again no automatic generalization mechanism.

Shilman et al. treat symbol recognition as visual language parsing [Shilman et al., 2002]. The visual language consists of the declarative grammar that specifies ranges of allowed values for a set of constraints between elements (distance, angle, width and height ratios, and overlap). Training on many examples is used to turn these ranges into distributions, so that the maximum likelihood parse tree can be calculated during recognition. Again, the visual grammar has to be written by hand, i.e. the designer has to determine relationships that are significant for the statistical model. Only the distributions are obtained through training.

This system, as well as SILK and the Electronic Cocktail Napkin, deal with the potential variability of symbol instances partially through using a small symbolic vocabulary of spatial constraints. However, none of the systems provides capabilities for learning which of these spatial constraints are important – the constraints have to be provided by the user or the designer of the system.

Ferguson and Forbus describe GeoRep – a spatial reasoning engine that generates qualitative spatial descriptions from perfect line drawings [Feguson and Forbus, 1999]. It has been applied for symmetry detection tasks, critiquing simple diagrams of physical phenomena, and spatial reasoning about military course of action diagrams. The paper mentions future applications of GeoRep to sketching, once it is modified to process free-hand input rather than exact line drawings. Apart from using a qualitative vocabulary of spatial constraints, GeoRep also includes generalization capabilities.

The part of GeoRep that is relevant to our work is the Low-Level Relational Describer (LLRD). LLRD produces qualitative spatial descriptions of the input in terms of geometric primitives and relations between them. It handles lines, circular arcs, circles, ellipses, splines, and text strings. It records position constraints like above, beside, etc.; orientation constraints, like horizontal, vertical; connection relations, parallel lines, interval relations, presence of polygons, and boundary description.

Similar to our system, GeoRep attempts to limit the number of recorded constraints between different primitive elements, because not all of them are visually important. The single mechanism it uses for this purpose is proximity. LLRD only looks at constraints between proximal elements. Proximity is calculated as a function of size, shape type, and distance between elements.

Like LLRD, our system prefers local interactions. Locality, however, is defined not through distance but through the obstruction mechanism. The primitives are considered "close" if there are no other primitives between them, regardless of the actual distance. 0 explains how we chose such definition based on observations about human perception.

In addition, our system adjusts the relevance of different properties of the symbol based on alignments (tension lines) and grouping. We show that even if two primitives are far away from each other, the constraints on them may still be relevant for the description of the symbol, and these mechanisms help detect this.

Connell's work on learning shape descriptions for images of physical objects (airplanes, tools, household items, etc.) contains several ideas that are also reflected in our work [Connell, 1985]. The goal of their system is to generalize a description for a class of objects (e.g., "airplane") from images for several objects in the class (e.g.

individual types of airplanes) in order to be able to recognize a new instance of the object. Objects are represented in terms of non-overlapping elongated blobs and their qualitative properties (like straight, curved, tapered, etc.) and constraints on these blobs (like joins, bigger-than, etc.). The description is recorded as a semantic network.

The parallel to our work is in the idea that the representation vocabulary should correspond to perceptually salient properties of objects. The description should make the visually important parts explicit. Connell talks about the importance of reflecting people's notion of visual similarity: "syntactic difference should reflect semantic difference: similar things should give rise to similar descriptions, dissimilar things should yield manifestly different descriptions."

Connell's system generalizes descriptions from a very small set of examples by comparing their semantic networks and removing constraints and properties that are not common between the examples. We believe that the ability to generalize from only a few examples stems mostly from the qualitative description vocabulary that already gets rid of a lot of information about the detailed properties of an object. The system does not have to go through a lot of examples to "average out" the unimportant properties.

Our system differs from Connell's approach in that it defines a ranking of the constraints. It does not have to discover which constraints are unimportant by seeing their absence in additional examples. The ranking already provides this information. This, however, depends on how well the ranking reflects the actual biases in people's perception.

Calhoun et al. presents a system that is most similar to our approach. It is a system that learns and recognizes symbols from relatively few examples. The recognizers are used for interpreting sketches of physical devices [Kurtoglu and Stahovich, 2002].

Like Connell, Calhoun uses semantic networks. The nodes are primitives in the symbol (lines and arcs) and the links are constraints between them. Constraints include: intersections, relative location of intersections, angle between intersecting primitives, and existence of parallel lines. The lines and arcs also have properties: type, length, length relative to the sum of all lengths, slope, and radius. To train the recognizers the system uses several examples of each symbol, including only the relationships and properties that appear with high frequency in the examples. During recognition some degree of matching error is allowed. The important part is that different weights are assigned to different kinds of errors during matching, reflecting different perceptual importance. In other words, if the learned descriptions mandate some unimportant constraints to hold, the system can compensate for that during the recognition stage, because the weights on matching errors for such constraints will be low.

The error weights play the same general role as the default relevance scores in our system. For example, the relative length constraint in Calhoun's system is always allowed a larger error than the relative orientation constraint (the same is true for the default scores in our system – relative length is less relevant than relative orientation). Yet our system also adjusts relevance scores from the default scores, based on the overall properties of the particular symbol, like obstruction, tension lines, and grouping. Using these mechanisms our system approximates the observation that the same type of constraint may have different perceptual importance depending on the global configuration of primitives.

So far we have talked about approaches that use symbolic descriptions (except for Rubine's system). Commonly used statistical machine learning techniques are mostly not applicable for our system because we have chosen to learn from only a single example and these approaches typically require a very large number of examples. For instance, classifiers for handwritten character recognition such as LeCun et al.'s convolutional networks, that achieve performance that is close to human subjects, use 6000 samples of each character [LeCun et al, 1995].

Yet among these techniques we would like to mention one approach from the area of handwritten character recognition that is similar in spirit to our work. Miller et. al. describes a system for learning characters or digits from one example [Miller et al., 2000] The authors create a classifier that is based on only a single training example for each class. They achieve this by including "prior knowledge", which is the shared probability density on common transforms (deformations) of digits or characters. They create an artificial data set by sampling transforms from the distribution and applying them to the single example. Then a classifier, like nearest neighbor, for example, can be trained using this data set.

Our system would not be able to use this approach directly because their current work is limited to affine transformations (translate, rotate, scale, and sheer). We believe that affine transformations are not the only variations that produce an image perceptually similar to the original, so the distribution would not capture all the possible variations. Consider the example in Figure 2.1. The second arrow cannot be obtained by an affine transformation on the first arrow, because it would involve disproportionately scaling different parts of the symbol:

Figure 2.1 The second arrow cannot be obtained by an affine transformation of the first arrow.

Even though Miller's et al. approach is not directly applicable, the idea of including prior knowledge to be able to learn from one example is very similar to our approach. By providing perceptually based constraint ranking we allow the system to extract the important information from a single example of the symbol.

In summary, our approach is strongly determined by the fact that we are learning from one example. Partially the generalization power comes from the qualitative vocabulary of constraints that reflects the relevant properties. Several systems have used this approach to address the variability of the symbol instances. In addition, instead of using several examples as have been done in other systems, the generalization in our system is guided by the relevance ranking based on default scores and global properties of the symbol.

Chapter 3 Knowledge About Human Perception

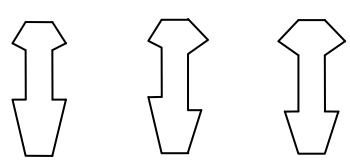
The challenge in learning symbols from a single example is to extract just the right subset of properties from it. We believe that the relevant properties are the ones that people pay attention to when looking at the symbol. A well-designed graphical language would not distinguish symbols by properties that people tend not to notice. If two symbols are very perceptually similar, but are intended to mean different things, they would be often confused, making the language ineffective. Thus, we suggest that it is the perceptually salient properties that constitute the essence of the symbol and should be learned by the system for each example.

We have turned to studies of human perception and memory to understand what people attend to and what they ignore in a geometric configuration. We rely mostly on Goldmeier's work on perceived similarity of geometric shapes and on memory traces [Goldmeier, 1972], [Goldmeier, 1982]. We also draw useful insights from Arnheim's book on art and visual perception [Arnheim, 1974] and studies of the perceptual grouping by the gestalt psychologists [Wertheimer, 1923]. This chapter describes the findings of these studies that inspired the main generalization mechanisms used by the system:

- Qualitative vocabulary
- Default relevance ranking
- Adjusting relevance scores based on global properties of the symbol:
    - Tension lines
    - Obstruction
    - Grouping

3.1 Singularities as the basis for qualitative vocabulary

Goldmeier attempted to discover which properties of a geometric figure people tend to notice when looking at a symbol. He uses people's perception of similarity to explore this: "Some features of a figure are more important for the over-all impression than others, so that changes of these features have a marked effect on similarity" [Goldmeier, 1972]. Figure 3.1 illustrates a typical experiment. Examine the symbol in Figure 3.1a and ask yourself which of 3.1b and 3.1c is more similar to 3.1a?

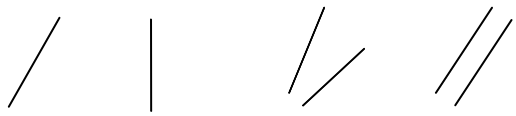


a) b) c)

Figure 3.1 Which of b and c is more similar to a?

The majority of subjects chose c. Note that the left side of b is exactly the same as a, yet even though in c all the lengths and angles are slightly changed, it is considered more similar because of preserved symmetry. It is the symmetry that was perceptually salient in the original figure.

Goldmeier's experiments showed that people frequently attend to properties that he called *singularities*, special cases in the space of geometric configurations (see examples below).



a) b)

Figure 3.2 a) A vertical (or horizontal) line is a special case of possible line orientations. b) Parallel lines are a special case for possible angles between two lines

Features such as verticality, horizontality, parallelism, etc., are singular in the sense that a small variation in them makes a qualitative difference: Rotate a vertical line slightly and it is no longer vertical; do the same to a line described as "slanting upward" (i.e., positive slope) and its qualitative description stays the same.

Goldmeier's work showed that, while singularities significantly affect perception of the symbol, people are relatively insensitive to variations in nonsingular properties. Consider Figure 3.3a:

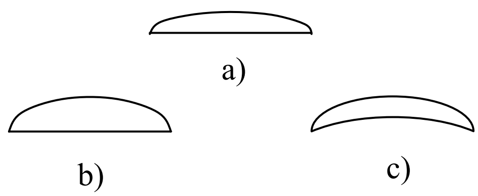


Figure 3.3 Which is of b and c is more similar to a?

Even though the thickness of the figure is preserved in c, the figure does not preserve the straight line, so the majority of subjects chose b. The subjects tolerated a large distortion in thickness and curvature, which are non-singular properties, because the more salient singular property (straightness) was preserved in b.

Goldmeier describes the way people generalize geometric properties that they see in a symbol. For each property "the value is coded either as singular, nearly singular, or nonsingular… This system combines coding accuracy in the narrow singular range with information reduction in the broad nonsingular range" [Goldmeier, 1982]. We use this observation to reduce the description vocabulary to a few qualitative states, lumping the range of nonsingular values into one term. For example, for a slanted line it is not necessary to record the exact angle. It is enough to learn only that it has a positive or a negative slope.

Goldmeier notes that the nearly singular values are perceived as a distortion to the singularity. Taking into account the nature of sketching where it is natural to expect sloppy drawing, this distortion can typically be considered accidental. Our system interprets nearly singular values as intended singularities, so the vocabulary consists only of singular and nonsingular terms.

Goldmeier explicitly mentions some of the singularities, like parallelism, horizontality, verticality, and straightness. We have picked the rest of the terms for the vocabulary based on our own introspection and relying on Goldmeier's description of singularities as the "more regular, better, more unique" [Goldmeier, 1982], p. 44] and as properties a change in which significantly alters the perception of the symbol.

The system records the properties of the symbol in the form of unary and binary constraints on the geometric primitives (lines and ovals) in the symbol. The table below shows the list of supported constraints (constraints that we consider singular are shown in bold):

Touch constraints:	**Connects**, **meets**, **intersects**, **touches**, **tangent**, **overlaps**
Orientation:	**Horizontal**, **vertical**, positive-slope, negative-slope
Aspect ratio:	Elongated, **non-elongated**
Relative position:	**Above-centered**, **right-centered**, **left-centered**, **below-centered**, above, below, right, left, upper-right, upper-left, lower-right, lower-left, **inside-centered**, inside
Relative orientation:	**Parallel**, **perpendicular**
Relative length:	**Same-length**, longer
Relative size:	**Same-size**, larger

3.2 Default ranking: relative importance of different singularities

In addition to showing that singular properties are perceptually more important than nonsingular ones, Goldmeier also compared singular properties with each other [Goldmeier, 1972]. Figure 3.4 and Figure 3.5 illustrate how this is done for different axes of symmetry. The subjects were asked which of b and c is more similar to a.

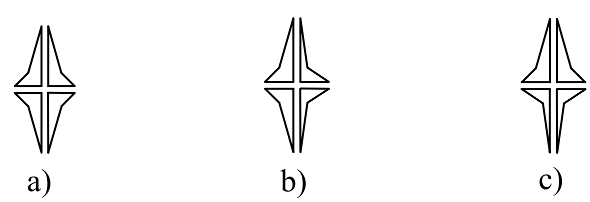


Figure 3.4 Which of b and c is more similar to a?

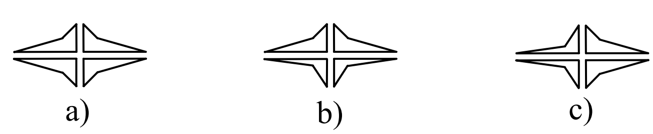


Figure 3.5 Which of b and c is more similar to a?

In Figure 3.4 the majority of the subjects chose c, while in Figure 3.5 the choice was b, even though the shapes in Figure 3.5 are simply rotated versions of Figure 3.4. In both cases the viewers preferred the vertical axis of symmetry.

Although this example is not directly applicable to our system (since it currently does not support symmetry detection), it illustrates the experimental framework in which the importance of different properties can be compared. Goldmeier presents several similar experiments. They are not sufficient, however, to construct a ranking of the different constraints used by our system. We had to use our own introspective analysis to rank the average perceptual importance of different types of symbol properties. We did this by studying common symbols in several domains (electric circuits, military planning, mechanical engineering, etc.) and determining which of their properties allowed most variation without large perceptual alteration to the symbols. The list below shows the order in decreasing importance:

1. The parts that the symbol is composed of.
2. Touch constraints (connects, meets, etc.)
3. Orientation.
4. Relative orientation.
5. Relative position.
6. Relative length and relative size.

Note that this is the default ranking of constraints. Goldmeier argues that the saliency of a given property depends on the particular configuration of the primitives in a shape. The next section describes the observations that helped us develop heuristics for adjusting the relevance of different constraints based on global properties like alignment (tension lines), obstruction, and grouping of primitives.

3.3 Effect of global properties on constraint relevance

3.3.1 Tension Lines

In his book *Art and Visual Perception*, Arnheim argues that people pay attention to regular alignments of geometric primitives in a symbol, particularly horizontal and vertical alignments. In Figure 3.6a the circle is perceived to be "out of balance," while placing it on one of the dashed lines in 3.6b would create a more "stable" configuration [Arnheim, 1974]:

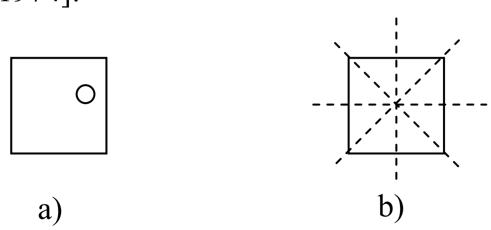


Figure 3.6 Regular alignments

Arnheim talks about "the hidden structure of a square" that can be explored by placing the circle in different places inside the square. The lines shown in Figure 3.6b emerge as axes of stability, especially the horizontal and vertical lines. The alignment of corners and the centers of the sides of the square form a kind of perceptual grid that other elements are "pulled" toward.

In our system we call these alignments *tension lines*, which we define in terms of alignments of line endpoints and midpoints. The system identifies a tension line wherever at least two such line points align horizontally or vertically (currently, we do not support diagonal alignments). Although this definition of tension lines may not capture the full complexity of the perceptual mechanisms that create the hidden structure, we believe that it can serve as a useful approximation.

Since the hidden structure grid represented by the tension lines is a salient element of the symbol, we increase the relevance of relative length, position, and orientation constraints that contribute to the creation of these lines.

3.3.2 Obstruction

We looked a variety of symbols to try to understand the perceptual importance of different constraints. In the process we have noticed that in the symbols that contain a lot of primitives, our attention seems to be limited to the local interactions between them. Consider the example in Figure 3.7a below:

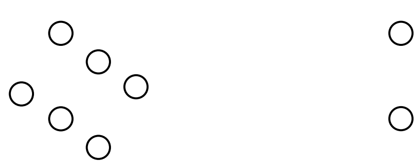
a) b) c)

Figure 3.7 a) Pattern of lines. b) Two parallel lines that are part of the pattern. c) Other pairs of parallel lines that are part of the pattern

The lines in Figure 3.7b are part of the pattern in a. In b it is noticeable that the lines are parallel, while in a, that is not something we would normally notice. Part of the reason for this might be that we perceive the pattern as a whole – a slanted elongated blob of lines. Nevertheless, notice that the parallelism of the pairs of lines in c is more noticeable in the original pattern than the parallelism of pair b. It is easier to pay attention to the local interaction of these lines because there are no other lines separating them. We try to approximate this effect by the notion of *obstruction*, which is measured by the number of geometric primitives between a given pair. The relevance of constraints is decreased for higher obstruction values.

3.3.3 Grouping

Finally, we also use observations of perceptual bias from the Gestalt psychologists, who noted that people tend to combine individual primitives into a greater whole, grouping them by proximity, similarity, etc [Wertheimer, 1923]. For example, Figure 3.8a is perceived as two rows of circles, rather than six individual circles. Properties of a row as a whole are also perceptually more important than properties of its components. We don’t tend to notice the vertical alignment of the circles in two columns the way we do in Figure 3.8b:



a) b)

Figure 3.8 Perceptual grouping. It is generally not noticeable that parts of a are the same as b

Grouping allows describing symbols more concisely. In the figure below the group consisting of the circle and the arrow is centered inside the rectangle. Conveying the same relationship using individual constraints on each of the primitives would be much harder.

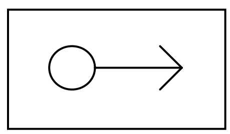


Figure 3.9 Military planning symbol for mortar

Our system currently supports two grouping principles: connectedness and familiarity of shape, i.e. previously learned shapes are recognized as separate groups within a new symbol. We decrease the relevance of constraints between pairs of primitives that belong to different groups.

3.4 Challenges

Studies show that people's view of geometric properties, such as sizes, angles, orientation, and curvature, do not easily map onto exact measurements from the drawing. Goldmeier notes: "Experiments demonstrate that similarity does not vary parallel with simple and obvious geometric parameters." Consider the example below, taken from [Goldmeier, 1972]:

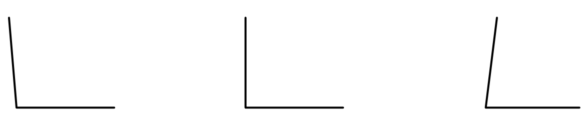


Figure 3.10 Which angle is 90º?

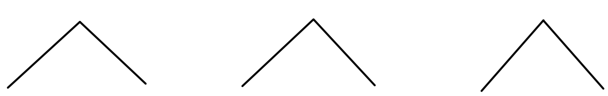


Figure 3.11 Which angle is 90º?

It is much harder to tell which angle is 90º, even though the angles in Figure 3.10 are simply rotated versions of the ones in Figure 3.11.

Another example of a common misjudgment is the famous perceptual illusion given below:

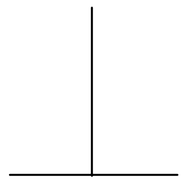


Figure 3.12 Do the lines seem the same length?

Even though the mechanisms causing such misjudgments are not understood well enough to exactly replicate such biases computationally, we include heuristics for the most common cases, decreasing the relevance of perpendicularity, for example, if it is found in other than a horizontal/vertical configuration.

Chapter 4 Examples of Performance

Even with the use of qualitative vocabulary, the number of the constraints initially identified by the system for a given symbol can be fairly large. Most people would find only a small subset of these constraints relevant for describing the symbol. The system uses observations about people's perceptual biases described in the previous chapter to calculate relevance scores and filter out the large number of low scoring constraints. For symbols with many primitives (more than 20) the reduction from the original number of constraints can be more than tenfold.

The calculation is based on the default relevance scores for each constraint type and the three mechanisms for adjusting these scores based on the global properties of the symbol – obstruction, tension lines, and grouping. This chapter illustrates in detail the effect of each of the mechanisms on a number of examples.

Consider the military symbol in Figure 4.1 drawn for the system. For this example, we assume that the system has not previously been taught the rectangle or the triangle symbols, so it is not able to identify them in the symbol.

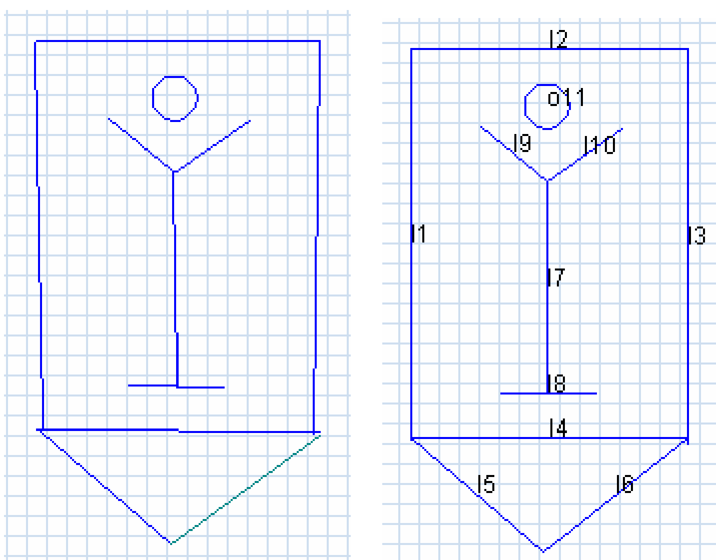


Figure 4.1 Military symbol: a) Strokes segmented into primitives. b) Straightened and labeled primitives

The system initially identifies 166 pair-wise constraints in the symbol (see full list in Appendix A), examples of which include:

parallel: (l10 l6) (l9 l5)
longer: (l2 l5)
lower-right: (l6 o11)
same-length: (l8 l10)

Scoring on perceptual relevance and removing constraints with low scores leaves only 80 constraints in the final description:

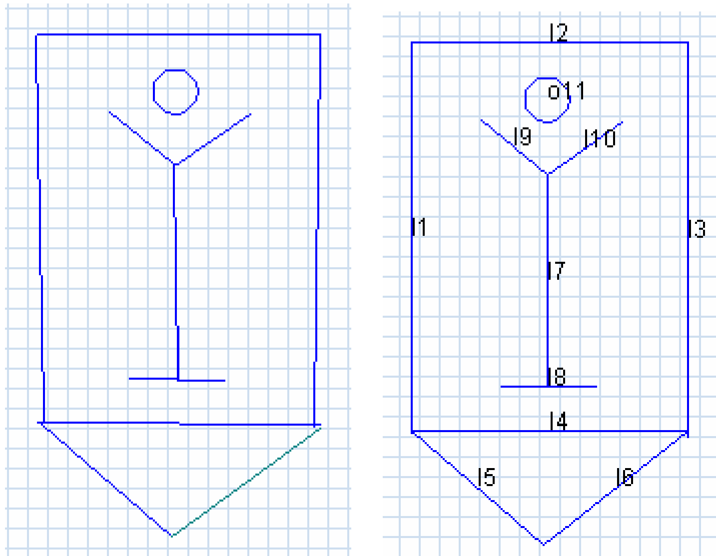


Figure 4.2 Military symbol: a) Strokes segmented into primitives. b) Straightened and labeled primitives

CONSTRAINTS:
connects: (l5.p2 l6.p1) (l4.p2 l6.p2) (l4.p1 l5.p1) (l3.p2 l6.p2) (l3.p2 l4.p2) (l2.p2 l3.p1) (l1.p2 l5.p1) (l1.p2 l4.p1) (l1.p1 l2.p1) (l9.p2 l10.p1) (l7.p1 l10.p1) (l7.p1 l9.p2)
meets: (l7.p2 l8.c)
horizontal: (l4) (l2) (l8)
vertical: (l3) (l1) (l7)
pos-slope: (l10) (l6)
neg-slope: (l5) (l9)
above: (l10 l8) (l4 l5) (l4 l6) (o11 l10) (l9 l8)
right: (o11 l1)
below: (l5 l4) (l8 l10) (l6 l4)
left: (o11 l3)
upper-right: (l10 l7) (o11 l9) (l3 l4) (l3 l6) (l2 l1)
upper-left: (l2 l3) (l1 l4) (l1 l5) (l9 l7)
lower-right: (l5 l1) (l10 o11) (l4 l1) (l3 l2) (l3 o11) (l8 l9) (l7 l9)
lower-left: (l4 l3) (l1 l2) (l1 o11) (l9 o11)
above-centered: (o11 l4) (o11 l7) (o11 l8) (l2 l4) (l2 l7) (l2 l8) (l2 o11) (l8 l4) (l7 l4) (l7l8)
right-centered: (l10 l9) (l3 l1) (l6 l5)
same-length: (l5 l6) (l2 l4) (l1 l3) (l9 l10)
longer: (l4 l5) (l4 l6) (l3 l2) (l3 l4) (l3 l6) (l1 l2) (l1 l4) (l1 l5) (l7 l9) (l7 l10)

The next three sections illustrate the adjustments of the default relevance scores the system makes to arrive at this description. The mechanisms are applied in the order discussed.

4.1 Obstruction

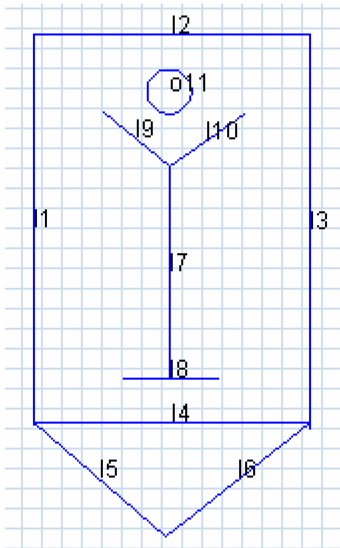


Figure 4.3 Military symbol

For each pair of primitives, the system measures the obstruction value which is approximately the number of other primitives between the pair. For example, there are 5 primitives between lines l2 and l4. The precise definition of the obstruction values is given in Chapter 5.

The presence of other obstructing primitives makes the relationship between a pair of primitives less noticeable. To reflect this, the system decreases the relevance of relative orientation, position, length, and size constraints for the pair depending on the obstruction value – larger obstruction values result in greater decrease. Examples of constraints with a significant decrease include:

parallel: (l10 l6) (l9 l5)
longer: (l3 l5)
same-length: (l10 l8)
above: (l10 l6)
upper-right: (o11 l5)

4.2 Tension Lines

Tension lines are horizontal and vertical alignments of two or more line endpoints or center points. Figure 4.4 shows the tension lines (in red) identified by the system in the example:

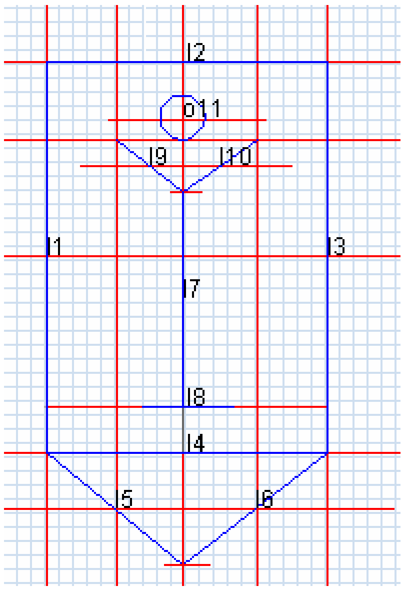


Figure 4.4 Tension lines formed by the primitives in the symbol

The aligned position of the primitives in the user's drawing leads to relative length and relative position constraints. Violating these constraints would break this perceptually salient alignment, so the relevance of these constraints is increased.

For example, line l3 is centered to the right of line l1 and they have the same length. The alignment of their end and center points is considered a salient property of the symbol. The system increases the relevance of the "right-centered" and "same-length" constraints necessary to maintain this alignment. These constraints become important even though their relevance may have been previously downgraded by the obstruction heuristic.

The system adjusts the relevance of 15 relative position and length constraints. Examples of such constraints are given below:

same-length: (l4 l2) (l1 l3) (l6 l5)
right-centered: (l3 l1) (l10 l9)
above-centered: (l2 l4)

4.3 Grouping

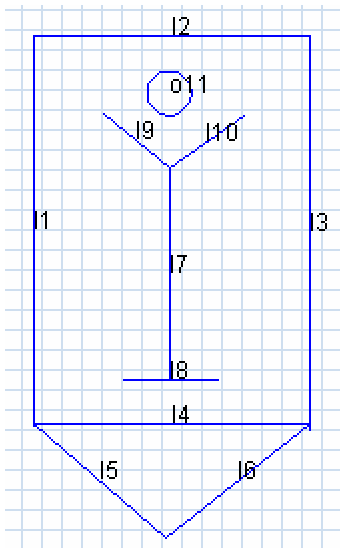


Figure 4.5 Military symbol

People tend to group together subsets of primitives in a symbol, especially when it contains a lot of primitives. The group is perceived as one whole and relationships between individual primitives belonging to different groups become less perceptually important. Our system currently supports two perceptual grouping principles: connectedness and familiarity of shape. In this case only the connectedness principle applies since the system has not previously been taught the rectangle or triangle symbols (which would have been recognized as familiar shapes). In Figure 4.5, the system identifies three connected components:

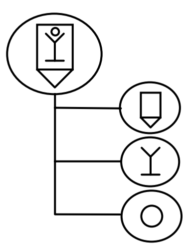


Figure 4.6 Grouping of the primitives in Figure 4.5 based on connectedness

For this symbol, the system decreases the relevance of 35 constraints between primitives in different connected components (like "longer (l4 l8)", or "longer (l3 l7)").

4.4 Another Use of Tension Lines

In the example in section 4.2 the system increases the relevance of a constraint between a pair of primitives if the endpoints of these primitives contribute to the formation of two tension lines. The tension line heuristic is also applied when one tension line is formed by centers of several primitives (more than two are required), even if their endpoints are not aligned. Alignment of several centers creates a "stronger" tension line. The system increases the relevance of each "above-centered" or "right-centered" that has to hold in order not to break the alignment.

Consider the symbol for DC voltage in Figure 4.7:

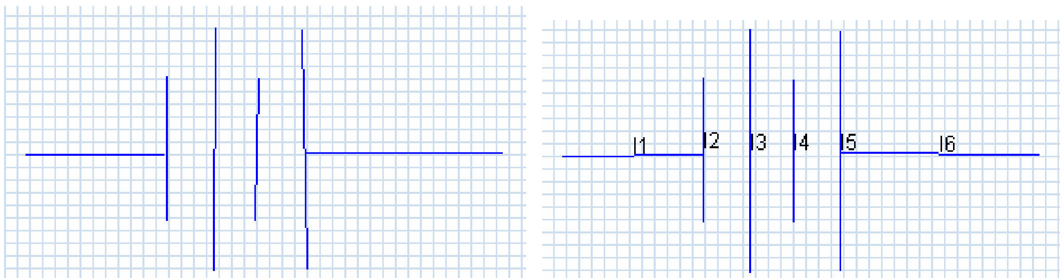


Figure 4.7 Symbol for DC voltage: a) Drawn strokes segmented into geometric primitives. b) Strokes straightened out and labeled

The description for this symbol is shown below. The system initially identified 45 constraints in the symbol. After calculating relevance scores and filtering out low scoring constraints, 28 constraints were left in the description.

CONSTRAINTS:
meets: (l1.p2 l2) (l6.p1 l5)
horizontal: (l1) (l6)
vertical: (l5) (l4) (l3) (l2)
right-centered: (l5 l1) (l5 l2) (l5 l3) (l5 l4) (l4 l1) (l4 l2) (l4 l3) (l3 l1) (l3 l2) (l2 l1) (l6 l1) (l6 l2) (l6 l3) (l6 l4) (l6 l5)
same-length: (l3 l5) (l2 l4)
longer: (l5 l4) (l3 l2) (l3 l4)

All the primitives form a strong horizontal tension line. Consider the constraint "right-centered l6 l1." The relevance of this constraint is first significantly brought down by the obstruction mechanism since there are several lines between lines l6 and l1. However, the presence of the tension line causes the system to increase of the relevance of this constraint and it is not filtered out from the description.

Examples of the constraints that did not make the relevance bar after the scoring are "longer (l5 l2) (l5 l1) (l6 l1)" etc.

4.5 Familiar Shapes

This section demonstrates the effect of the second grouping factor supported by our system – the familiarity of shape. It groups together primitives within the symbol that form a familiar shape – a symbol that has previously been learned by the system.

In many domains (e.g. military planning) symbols are composed of common shapes like rectangles, triangles, diamonds, circles, etc. or a combination of other simpler symbols. A much more concise description of the symbol is often possible in terms of constraints on those shapes as a whole. Those constraints become more perceptually important while the constraints between individual primitives that belong to different shapes are less noticeable.

The system checks whether any of the previously learned symbols are contained in the new symbol and identifies such subparts as separate groups. Consider the symbol in Figure 4.8. Before learning it, the system has been shown a rectangle, a square, a triangle, and a cross and produced descriptions for those symbols. It searches for them in the new symbol.

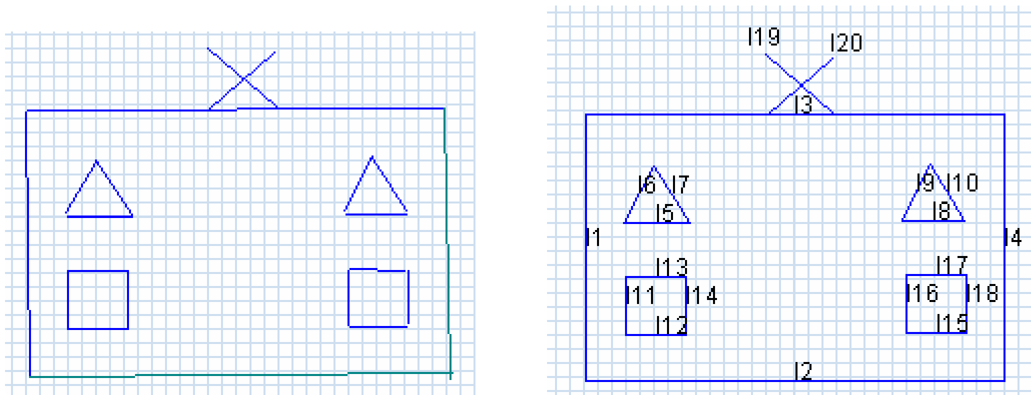


Figure 4.8 Military planning symbol composed of familiar shapes

Below is the system's description for the symbol:

GROUP HIERARCHY:
Group g1: l1 l2 l3 l4 l5 l6 l7 l8 l9 l10 l11 l12 l13 l14 l15 l16 l17 l18 l19 l20
    Group g2 subobject - regular triangle: l5 l7 l6
    Group g3 subobject - regular triangle: l8 l10 l9
    Group g4 subobject - square: l14 l13 l12 l11
    Group g5 subobject - square: l18 l17 l15 l16
    Group g6 connected-component: l19 l4 l3 l1 l2 l20
        Group g7 subobject - cross: l20 l19
        Group g8 subobject - rectangle: l4 l3 l2 l1

CONSTRAINTS:
upper-right:(g3 g4)
upper-left:(g2 g5)
above-centered:(g3 g5) (g2 g4)
right-centered:(g5 g4) (g3 g2)
inside:(g5 g8) (g4 g8) (g3 g8) (g2 g8)
meets:(l19.p2 l3) (l20.p1 l3)
above-centered:(l20 l3) (l19 l3)

The system initially found 500 constraints in the symbol. 340 of these had their scores reduced due to the grouping factor. For example, the relative length and position constraints between lines l14 and l16 are not affected by obstruction and would be considered important using the tension line heuristic. Yet they receive a low relevance score because the lines belong not only to different connected components but also to different previously learned symbols, which in our system makes the relevance decrease even greater.

The use of known shapes allows the system to describe the symbol in terms of more general constraints on the shapes as a whole. There is also no need to include the constraints that are already listed in the descriptions for previously learned symbols (there were 93 such constraints in this example). As a result the description is compact even though the symbol has a lot of primitives and the initial number of constraints is very large. There are only 14 constraints in the final description.

Chapter 5 Implementation

This chapter describes the processing steps the system goes through from the time the user starts drawing the symbol to generating the final description. We provide the definition of all supported constraints and their default relevance scores. We show how the system adjusts these scores to filter out irrelevant constraints, based on three factors: obstruction, tension lines, and grouping. We also present a graphical interface for displaying the resulting constraints.

Figure 5.1 below briefly outlines the processing steps to generate the textual description:

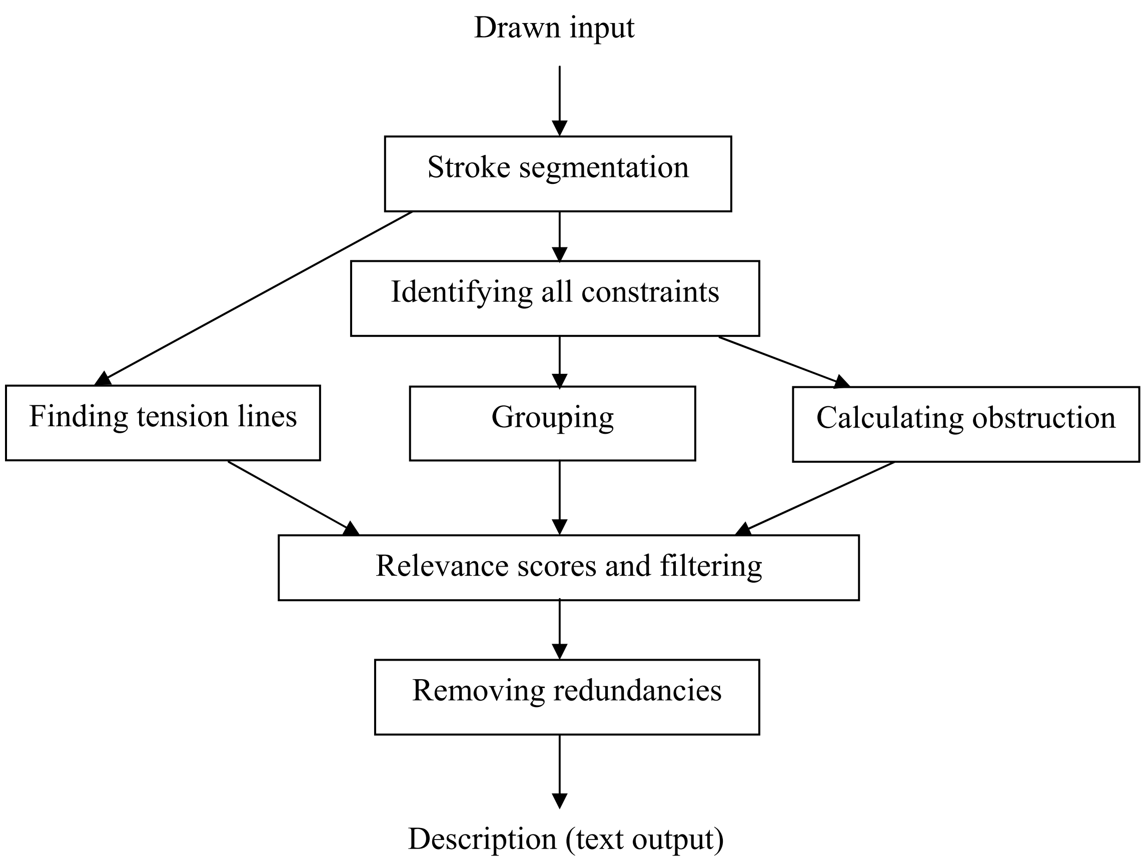


Figure 5.1 Processing steps to produce the description of a symbol

Each section describes one of these steps and illustrates the work of the system on the symbol in Figure 5.2:

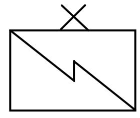


Figure 5.2 Military planning symbol

5.1 Stroke segmentation

The user draws the symbol in the system's drawing window (Figure 5.3), which provides a grid to make it easier for the user to draw carefully. The program accepts any mouse or pen-based input.

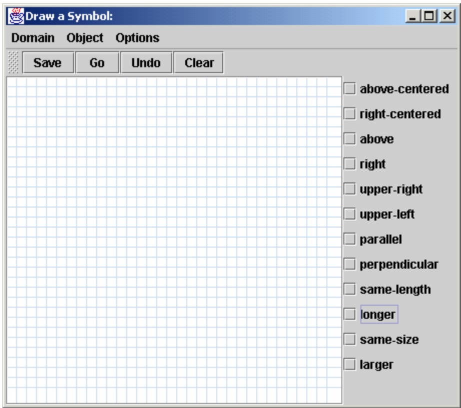
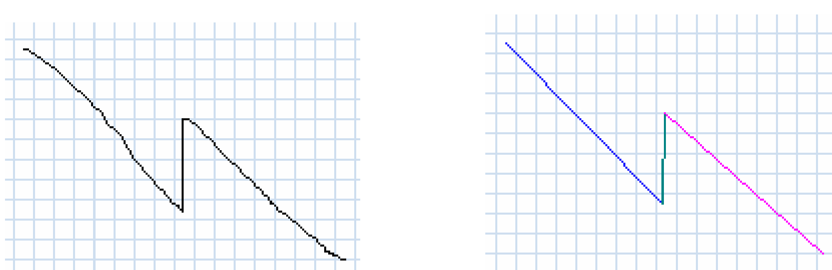


Figure 5.3 Drawing window

We use a toolkit developed by [Sezgin, 2001] to segment the strokes into simple geometric primitives. The toolkit takes into account both stroke curvature and pen speed data to find separate geometric primitives, based on the observation the people often slow down the pen at corners. Our system instructs the toolkit to classify each stroke as either a polyline or an oval.

If the user slows down accidentally (which often happens when using a mouse, rather then pen input) the segmentation may produce spurious corners. We use alternating segment colors for each primitive within a stroke to provide feedback on segmentation (Figure 5.4). The user can press "Undo" and redraw the stroke, if the segmentation is incorrect.



a) b)

Figure 5.4 a) Original single stroke. b) Segmentation of the stroke

Figure 5.5 below shows the symbol from Figure 5.2 in the system's drawing window:

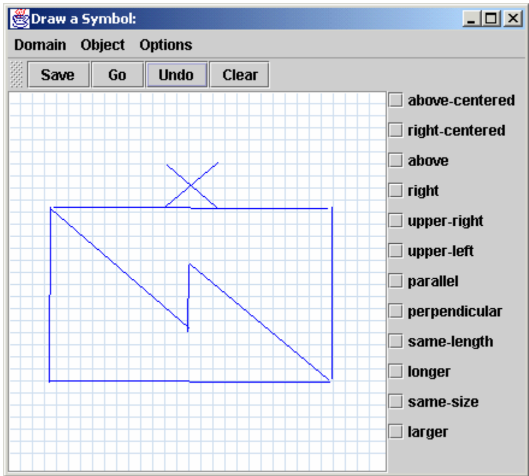


Figure 5.5 Military planning symbol: strokes segmented into geometric primitives

The user presses the "Go" button after completing the drawing to generate the description. All the strokes on the surface are considered to be one symbol.

As we have mentioned, generalization is done on the primitives, not on the stroke data, so the order and the number of strokes does not affect the produced description. The advantage of this approach is that the user is not required to draw the symbol in exactly the same way during sketching as during the teaching phase – the system would recognize it based on what it looked like, rather than how it is drawn. On the other hand, some ways of drawing are more likely to occur than others. For example, one would often draw a rectangle starting from the top-left corner and all in one stroke. The stroke order and number information could give additional clues for the recognition engine for distinguishing between symbols in cases of ambiguity. Our system currently does not explicitly record this information, although the order is implicitly contained in the primitive labeling.

5.2 Identifying all constraints

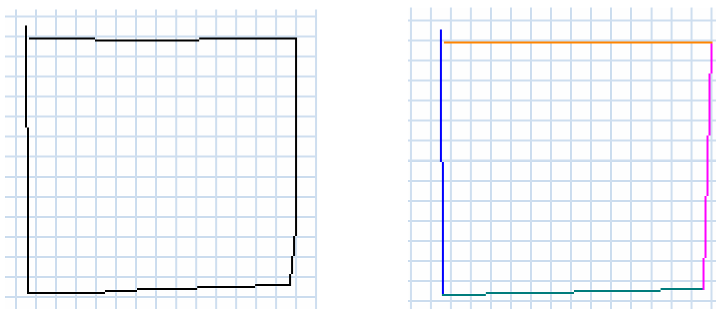
Once the drawing is completed, the system records all the constraints in the drawing. Each constraint type is represented by a graph – one graph for "connects", one graph for "above", etc. Geometric primitives are nodes in the graph and edges signify whether the constraint holds between a pair of nodes. Unary constraints, like "horizontal", reuse the same data structure for uniformity, with all the edges as self-loops. The edges in the graphs are directional, so for each symmetric constraint like "same-length" or "parallel" two edges will be found for a given pair of primitives. The final output description includes only one of each pair of symmetric constraints to minimize redundant information.

To identify constraints the system considers each primitive for unary constraints and each pair of primitives for binary constraints. As mentioned in 0, the vocabulary consists of singular and non-singular constraints. The system first tests whether a singular constraint holds for the primitive (or pair). For example, for line orientation, the system tests whether it can be considered horizontal or vertical. Nearly singular values, like almost horizontal, are treated as accidental noise and recorded as singular. We describe the noise thresholds in the next section.

If the singular constraint is not satisfied, the system then tests the non-singular constraints (like "positive-slope" or "negative-slope" for line orientation). This approach corresponds to Goldmeier's observation that people's perception is sensitive to singularities and codes geometric properties in terms of their relation to the singularity [Goldmeier, 1982], p. 43]. That is, for example, if a line is perceived as horizontal, it cannot be simultaneously seen as positively sloped.

5.2.1 Noise thresholds and constraint definition

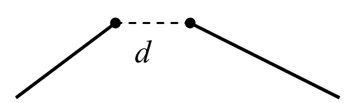
It is hard to draw the symbol perfectly; not all lines intended to be exactly horizontal, connected, or aligned will come out that way (Figure 5.6).

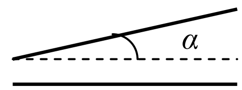
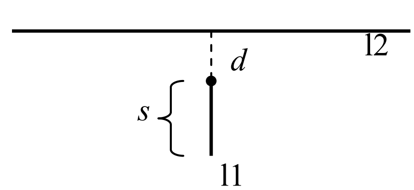


a) b)

Figure 5.6 Noisy drawing of a square: a) Original stroke. b) Stroke segmented into lines

The system allows a certain amount of noise when testing for presence of constraints. Noise tolerances are governed by three constants:

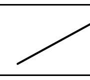
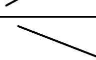
Constant	Value	Example
MAX_OFFSET: This constant is used for testing any constraints where the system needs to determine whether the distance between two points can be considered negligible. The constant specifies that the distance should be less than 7 pixels.	7 pixels	Two lines will not be considered connected if the distance *d* between their endpoints is greater than 7 pixels. 

MAX_ANGULAR_OFFSET This constant specifies the maximum angular difference for which the angle can be considered negligibly small. It is used for constraints like line orientation or relative orientation.	10º	Two lines will not be considered parallel if their angle difference α is more than 10º. 
SIZE_TO_OFFSET_RATIO We do not want to consider the distance *d* between two points negligible if the size of the primitives in question is small (even when the MAX_OFFSET threshold is satisfied). The constant specifies the minimum ratio between the size of the smallest primitive and the distance *d*. The size should be at least 3 times larger than *d* for *d* to be considered negligible	3 times	Line l1 is not considered to meet line l2 because its length *s* is less than 3 times larger than the distance *d* to line l2. 

We use the noise tolerance constants to determine when the system can decide that a constraint holds in the drawing. Below we describe the definitions for each constraint in the vocabulary. In the definition tables singular constraints are shown in bold.

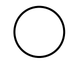
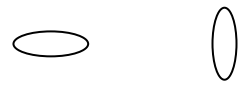
5.2.1.1 Orientation

"Horizontal" and "vertical" constraints hold if the angle difference between the ideal and the actual orientation of the line in the drawing is less than MAX_ ANGULAR_OFFSET and if the change in y (for horizontal) or x (for vertical) coordinates from the center to the endpoints is less than MAX_OFFSET. If the "horizontal" or "vertical" constraint is not satisfied, the orientation is recorded as either "positive slope" or "negative slope," depending on the slope of the line. The orientation of an oval is defined by the orientation of its longer axis and applies only to ovals satisfying the "elongated" constraint (see definition below).

Constraint	Applies to	Example
Horizontal	lines, elongated ovals	
Vertical	lines, elongated ovals	
Positive Slope	lines	
Negative Slope	lines	

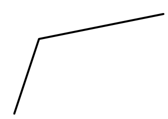
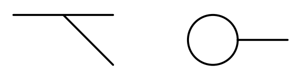
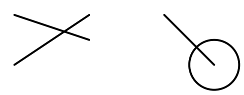
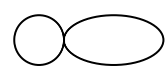
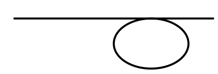
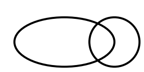
5.2.1.2 Aspect ratio

Ovals are considered “non-elongated” if the ratio of their length to their thickness is less than 1.5. Otherwise the oval is “elongated”.

Constraint	Applies to	Example
Non-elongated	ovals	
Elongated	ovals	

5.2.1.3 “Touch” constraints

We refer to “connects,” “meets,” “tangent,” etc. as the “touch” constraints. The distance *d* between the points that are supposed to be coincident should be less than MAX_OFFSET. The table below shows how we define *d* for each constraint. Also, the ratio of the size of the smallest primitive and *d* should be greater than SIZE_TO_OFFSET_RATIO. The size of a line is its length and the size of an oval is the maximum of its width and height.

Constraint	Applies to	Definition of tested distance *d*	Example
Connects	Lines	The distance between line endpoints.	
Meets	lines, line and oval	The perpendicular distance from the line endpoint to the line segment or oval boundary.	
Intersects	lines and ovals	Not applicable. The system tests for the presence of intersection.	
Touches	Ovals	The smallest perpendicular distance between oval boundaries.	
Tangent	line and oval	The smallest perpendicular distance between the line and the oval.	
Overlaps	Ovals	Not applicable.The system tests for the presence of intersection of oval boundaries.	

It may happen that several of the “touch” constraints are satisfied for a given pair of primitives at the same time. For example, both “meets” and “intersects” constraints would be satisfied in Figure 5.7.

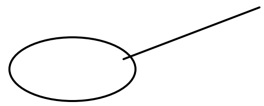


Figure 5.7 The symbol satisfies both "meets" and "intersects" constraints

We want to choose only one interpretation. We define an order in which “touch” constraints are tested and record only the first satisfied constraint:

For lines	For line and oval	For ovals
1. Connects 2. Meets 3. Intersects	1. Meets, tangent 2. Intersects	1. Touches 2. Overlaps

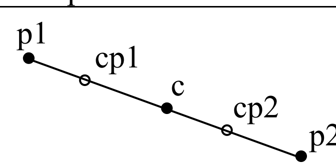
For some “touch” constraints the system also specifies where exactly the primitives touch. For example, Figure 5.8 shows the kinds of cases we would like to distinguish:

a) b)

Figure 5.8 a) Different points where one line may meet the other. b) Different points of intersection of a line with the oval

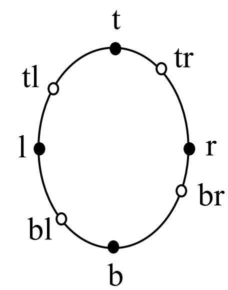
For each of the “connects,” “meets,” “intersects,” “touches,” and “tangent” constraints the system records the points of coincidence on both primitives. For example, “meets (l1.p1 o1.t)” means that point p1 of line l1 meets oval o1 at the top.

As with all constraints, we attempt to reflect perceptual singularities in the specification of coincidence points. The table below shows the definitions of possible coincidence points on a line, with singular points shown in bold. Endpoint labels p1 and p2 are assigned arbitrarily.

Point on a line	Notation	Example
Endpoint 1	p1	
Any point between center and endpoint1	cp1	
Center	c	
Any point center and endpoint2	cp2	
Endpoint 2	p2	

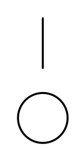
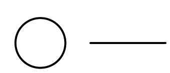
The end and center points are singular, so the system always starts by testing whether a constraint holds for one of these points, that is if the distance from the coincidence point on the other primitive to one of these points is less than MAX_OFFSET and satisfies the SIZE_TO_OFFSET_RATIO threshold. If not, one of the non-singular points is recorded.

Similarly, coincidence points are defined for ovals. Singular points that are tested first are shown in bold.

Point on an oval	Notation	Example
Top	t	
Top right	tr	
Right	r	
Bottom right	br	
Bottom	b	
Bottom left	bl	
Left	l	
Top left	tl	

5.2.1.4 Singular position constraints

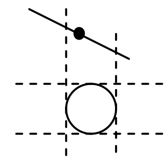
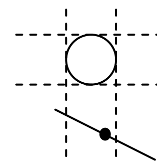
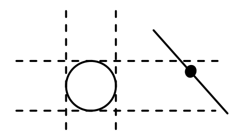
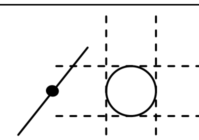
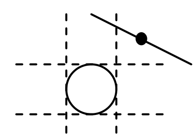
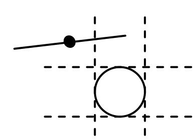
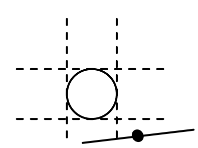
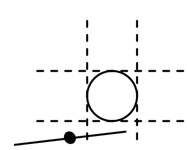
These constraints specify relative position of the primitives and the horizontal or vertical alignment of their geometric centers. The centers are considered horizontally or vertically aligned if the difference in their respective y or x coordinates is less than MAX_OFFSET and SIZE_TO_OFFSET_RATIO is satisfied.

Constraint	Applies to	Example (the position of line relative to oval)
Above-centered	lines and ovals	
Right-centered	lines and ovals	

We do not test for "below-centered" and "left-centered" constraints because their definition is symmetric to "above-centered" and "right-centered" respectively.

5.2.1.5 Non-singular position constraints

These constraints specify relative positions of the primitives and are recorded only in the absence of the corresponding singular position constraints described in the previous section. The recorded constraint depends on the position of the center of the first primitive relative to the bounding box of the second primitive. To test the constraint "above l1 l2," for example, the system would look at the center of line l1 and the bounding box of line l2.

Constraint	Applies to	Example (the position of line relative to oval)
Above	lines and ovals	
Below	lines and ovals	
Right	lines and ovals	
Left	lines and ovals	
Upper-right	lines and ovals	
Upper-left	lines and ovals	
Lower-right	lines and ovals	
Lower-left	lines and ovals	

The limitation of these definitions is that the boundaries between these terms do not correspond to clear qualitative perceptual boundaries. For example, the difference between the two drawings in Figure 5.9 is almost unnoticeable, while the produced descriptions would be different – one would be “above (o1 l2)” and the other “upper-right (o1 l2).” This means that the description produced for the first symbol would prevent the second symbol from being considered an instance of the first one.

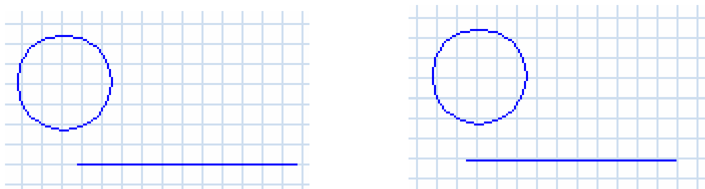


Figure 5.9 Two very similar drawings that produce dissimilar descriptions

5.2.1.6 “Inside” and “inside-centered” position constraints

These constraints apply to primitives inside ovals. The “Inside-centered” constraint holds if the primitive is inside the oval and the coordinate difference between its center and the center of the oval is less than MAX_OFFSET and satisfies the SIZE_TO_OBJECT_RATIO. Otherwise only the “inside” constraint holds.
”Inside” constraints do not have to hold exactly in the actual drawing, as long as they hold if noise were removed from the drawing. For example, in Figure 5.10, the line is considered to be inside the oval because the system decides that it satisfies the “meets” rather than the “intersects” constraint with the oval:

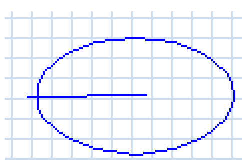
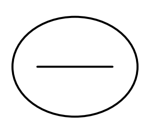
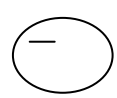


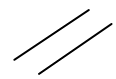
Figure 5.10 The line is considered to be inside the oval

Constraint	Applies to	Example (the position of line relative to oval)
Inside and centered	line and oval, oval and oval	
Inside	line and oval, oval and oval	

5.2.1.7 Relative orientation

“Parallel” and “perpendicular” constraints hold if the actual angle between the lines in the drawing differs from the ideal angle by less than MAX_ANGULAR_OFFSET. The system records these constraints only for lines that it identified as positively or negatively sloped. This is done because, as we show further in Section 5.6, the system never filters out “horizontal” and “vertical” constraints. “Parallel” and “perpendicular” constraints only would only provide redundant information for horizontal and vertical lines, so the system does not record them.

Also, as shown in 0, it is hard for people to accurately tell the angle between the two connected slanted lines (see Figure 3.10 and Figure 3.11), so we do not record the “perpendicular” constraint for such lines.

Constraint	Applies to	Example (the position of line relative to oval)
Parallel	Lines	
Perpendicular	Lines	

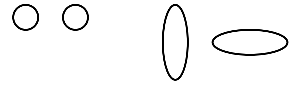
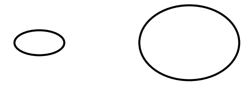
5.2.1.8 Relative length

Two lines are considered to have the same length if the ratio of the length difference over the length sum is less than 0.05. Otherwise a “longer” constraint is recorded.

Same length	Lines	
Longer	Lines	

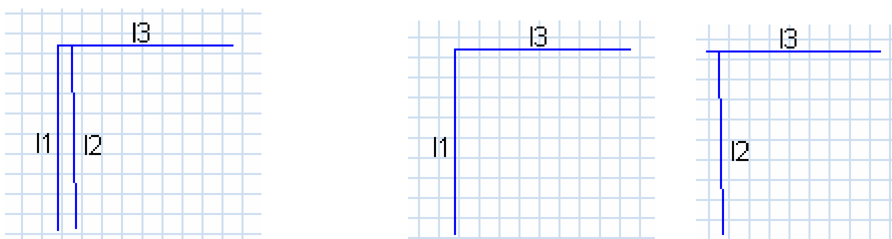
5.2.1.9 Relative size

The size of the oval is defined as the maximum of its width and its height. Two ovals are considered to have the same size if the ratio of the size difference over the size sum is less than 0.08. Otherwise a “larger” constraint is recorded.

Constraint	Applies to	Example (the position of line relative to oval)
Same size	Ovals	
Larger	Ovals	

5.2.2 Possible contradictions

Tolerances for noise make it possible to record contradicting constraints, because the system tests constraints for each pair of primitives separately. For example, in Figure 5.11a, the system will decide that both lines l1 and l2 connect to the endpoint of line l3 if the distance between lines l1 and l2 is smaller than MAX_OFFSET. Remember that a “meets” constraint is never recorded if “connects” is found first. The system examines pairs (l1 l3) and (l2 l3) separately (Figure 5.11b). For both of these pairs, the distance between line endpoints is small enough for the system to identify a “connects” constraint. Yet it also decides that both lines l1 and l2 are vertical, which contradicts the “connects” constraints.



a) b)

Figure 5.11 a) Drawing resulting in potential contradictory constraints. b) Pairs of primitives separately examined by the system

Figure 5.12 shows another example that may cause contradictions. If the noise tolerance is large enough compared to the length of line l3, the system will decide that both lines l2 and l3 are centered to the right of l1. Yet l3 is also above l2.

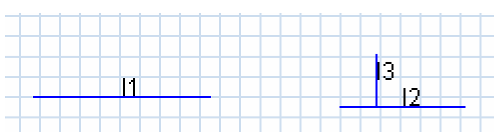
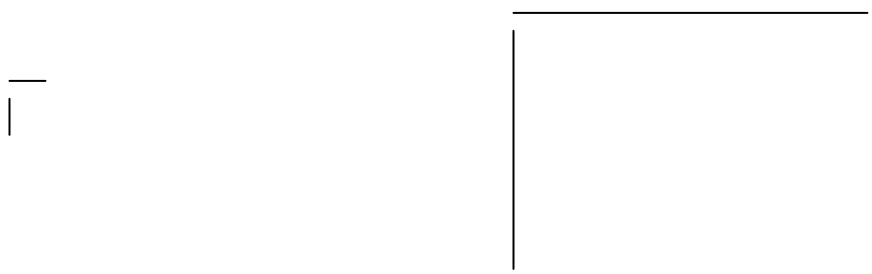


Figure 5.12 Drawing resulting in potential contradictions

We have not implemented a mechanism to detect and correct such contradictions. Currently, the only solution for the user is to draw carefully, keeping in mind the magnitude of the tolerance thresholds. The MAX_OFFSET threshold is indicated by the size of the grid cells. The smallest primitives and distances in the symbol should be larger than the grid size. And the level of noise, like accidental gaps and misalignments, should be smaller than the grid size.

If the physical size of the pixels on the device is too small, it may be hard to keep the noise under the MAX_OFFSET (which is specified in pixels) when drawing. We allow the user to change this constant, which will be reflected visually in the grid size.

Absolute noise thresholds may be somewhat unnatural. Consider the lines in Figure 5.13:



a) b)

Figure 5.13 a) Short lines. b) Long lines

Although the distance between the endpoints of the two lines is the same in both cases, the lines in Figure 5.13b are much more likely to be perceived as connected than in the lines in Figure 5.13a. That means that the maximum tolerance for line connectivity could be larger for longer lines.

We do decrease the noise threshold if primitives are small, which is achieved by the mandatory minimum SIZE_TO_OFFSET_RATIO. This constant always limits the noise threshold to less than a third of the primitive size. Yet the system does not increase the noise threshold beyond MAX_OFFSET if primitives get larger.

We chose to have an absolute maximum threshold for all primitive sizes, so that it is clearer to the user what the system's maximum noise tolerance is. We believe that this would make it easier to determine how carefully one should draw, though we have not verified this assumption in user studies. In the future work, if the system includes contradiction resolution, size-dependent noise thresholds will probably be more appropriate.

Although there is no generic mechanism for contradiction detection, we have included several routines to correct one type of common mistake with relative length and size constraints. These routines enforce the transitive closure in "same-length" and "same-size" constraints and remove the "longer" and "larger" constraints that contradict the closure.

Consider the triangle in Figure 5.14a. In Figure 5.14b the sides of the triangle are aligned to show their relative length.

a) b)

Figure 5.14 a) Triangle. b) Lengths of sides of the triangle

Suppose, for example, that the system considers the length difference for line pairs (l1 l2) and (l2 l3) negligible and records the constraints “same-length: (l1 l2) (l2 l3)” and “longer: (l3 l1).” Using transitive closure the system finds that for consistency with “same-length: (l1 l2) (l2 l3),” lines l1 and l3 also need to have the same length. Hence, it removes the “longer” constraint and replaces it with “same-length (l1 l3)”. A similar mechanism is used for relative oval size.

Clearly, the limitation of this mechanism is that it may interpret a series of very gradually increasing lines to be the same length, even if the length of the first and the last line in the sequence are significantly different.

5.2.3 Example result of identifying all constraints

Figure 5.15 shows a drawing of a military symbol with strokes segmented into geometric primitives. We label the primitives for convenience.

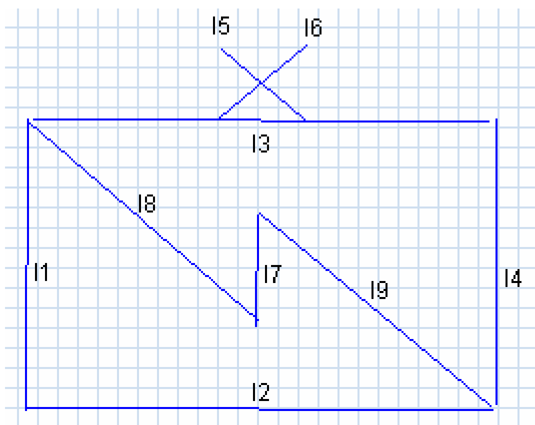


Figure 5.15 Military symbol

The table below shows 122 constraints that the system finds in the symbol.

connects: (l4.p1 l3.p2) (l4.p2 l2.p2) (l4.p2 l9.p2) (l3.p1 l1.p1) (l3.p1 l8.p1) (l2.p1 l1.p2) (l2.p2 l9.p2) (l1.p1 l8.p1) (l9.p1 l7.p1) (l7.p2 l8.p2) meets: (l6.p1 l3.cp1) (l5.p2 l3.cp2) intersects: (l6 l5) horizontal: (l3) (l2) vertical: (l4) (l1) (l7) pos-slope: (l6) neg-slope: (l5) (l9) (l8) above: (l5 l9) (l3 l8) (l9 l2) (l8 l2) (l6 l9) right: (l4 l8) (l4 l9) (l9 l1) (l9 l7) (l9 l8) (l8 l1) (l7 l8) below: (l2 l9) (l9 l3) (l8 l3) left: (l1 l8) (l1 l9) (l9 l4) (l8 l4) (l8 l7) (l8 l9) (l7 l9) upper-right: (l5 l1) (l5 l8) (l4 l2) (l3 l1) (l6 l1) (l6 l8)	upper-left: (l5 l4) (l3 l4) (l3 l9) (l1 l2) (l6 l4) lower-right: (l4 l3) (l4 l5) (l4 l6) (l2 l1) (l2 l8) (l9 l5) (l9 l6) lower-left: (l2 l4) (l1 l3) (l1 l5) (l1 l6) (l8 l5) (l8 l6) above-centered: (l6 l2) (l6 l3) (l6 l7) (l5 l2) (l5 l3) (l5 l7) (l3 l2) (l3 l7) (l7 l2) right-centered: (l4 l1) (l4 l7) (l7 l1) parallel: (l5 l8) (l5 l9) (l9 l8) perpendicular: (l5 l6) (l8 l6) same-length: (l5 l6) (l2 l3) (l1 l4) (l9 l1) (l9 l4) (l8 l1) (l8 l4) (l8 l9) (l7 l5) (l7 l6) longer: (l4 l5) (l4 l6) (l4 l7) (l3 l1) (l3 l4) (l3 l5) (l3 l6) (l3 l7) (l3 l8) (l3 l9) (l2 l1) (l2 l4) (l2 l5) (l2 l6) (l2 l7) (l2 l8) (l2 l9) (l1 l5) (l1 l6) (l1 l7) (l9 l5) (l9 l6) (l9 l7) (l8 l5) (l8 l6) (l8 l7)

5.3 Tension lines

The next processing step is to find tension lines – the horizontal and vertical alignments of primitives in the symbol. The system starts by creating a list of *tension points*. The list includes all line centers and endpoints and points on the top, bottom, left, right, and center of the ovals. The horizontal or vertical alignment of two or more tension points defines a tension line (Figure 5.16).

These alignments are found by a horizontal and vertical sweep through the list of tension points sorted by y and x coordinates respectively. Each group of consecutive tension points for which the maximum vertical (or horizontal) difference between point coordinates is less than MAX_ OFFSET corresponds to a different tension line. This means that the maximum misalignment of points on a tension line is MAX_OFFSET, consistent with the overall noise threshold in the system.

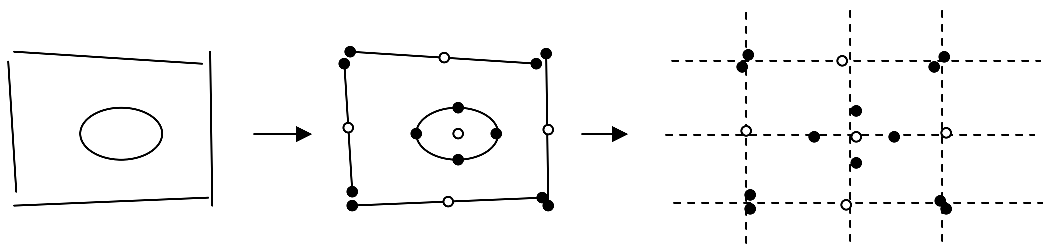
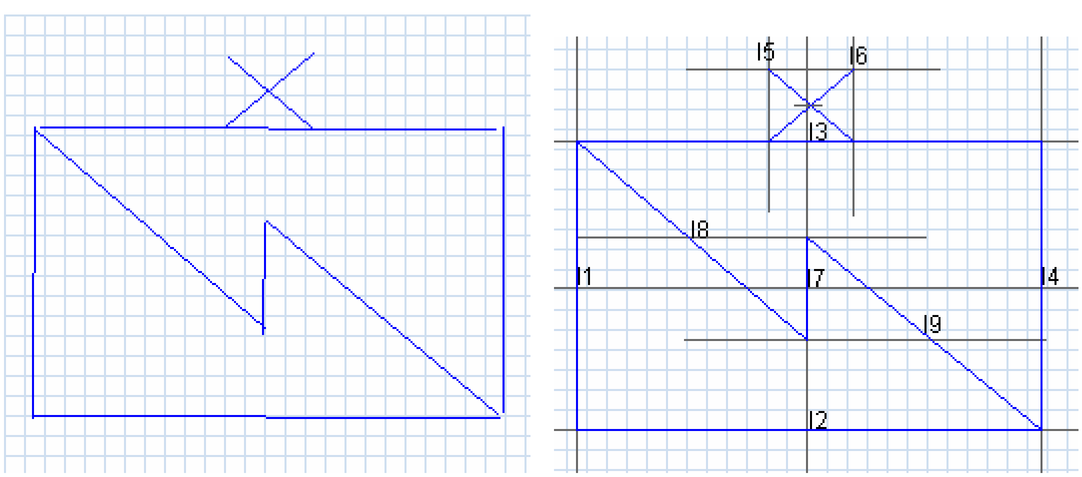


Figure 5.16 Tension lines defined by groups of tension points

Grey lines in Figure 5.17b show tension lines for the military symbol in Figure 5.17a.



a) b)

Figure 5.17 a) Symbol b) Tension lines for the symbol

5.4 Obstruction

After finding all constraints and tension lines the system proceeds to calculate obstruction. The obstruction value for each pair of primitives is roughly the number of other primitives between the pair. This section explains how obstruction values are calculated.

Consider Figure 5.18. There are 4 lines between the lines l1 and l6.

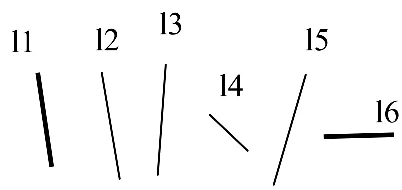


Figure 5.18 Four lines separate lines l1 and l6

Notice, however, that it is not always clear whether a primitive is "between" a given pair. If we look at lines l1 and l3 in Figure 5.19, it is hard to decide whether line l2 is between them.

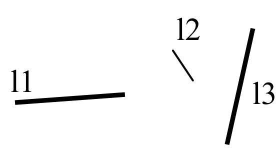


Figure 5.19 Is line l2 "between" lines l1 and l3?

In Figure 5.19 line l2 does not completely separate l1 and l3, but it creates some obstruction. In this case we would like to assign an obstruction value that is somewhere between 0 (as, for example, in Figure 5.20a) and 1 (as in Figure 5.20b), so we use non-integer obstruction values.

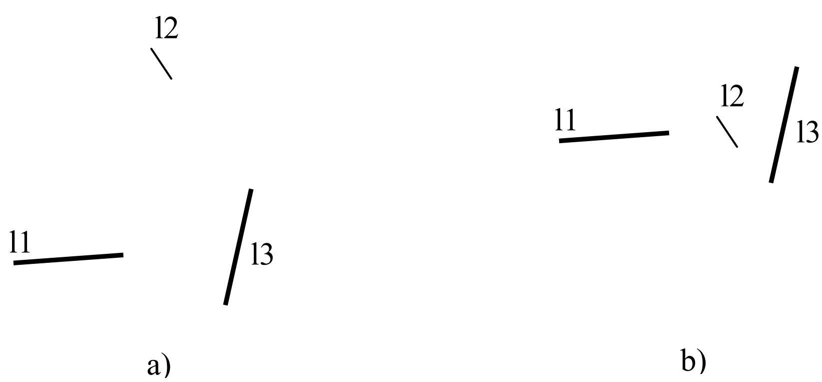
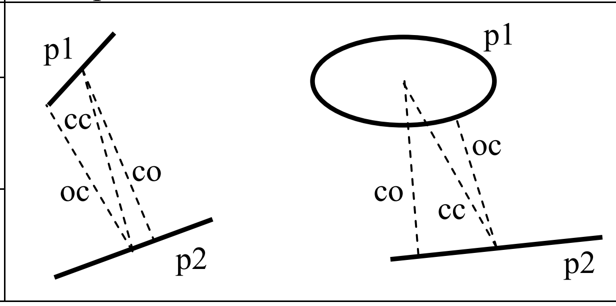


Figure 5.20 a) Line l2 creates no obstruction for the pair (l1 l3). b) Line l2 is clearly between l1 and l3

To calculate obstruction values for a pair of primitives, we define three special lines connecting them:

Connecting line	Examples
A line connecting the centers of two primitives (cc)	
A line connecting the center of the first primitive to the closest point on the second primitive (co)	
A line connecting the center of the second primitive to the closest point on the first primitive (oc)	

The contribution of every remaining primitive p_i in the symbol to the obstruction value for the pair (p_1,p_2) is an exponentially decreasing function of the distance between p_i and each of the connecting lines. This distance is taken relative to the size s of the smaller primitive in the pair.

$$O(p_1, p_2) = \begin{cases} \sum_{i \neq 2,3} (\alpha^{\text{distance}(p_i, cc)/s} + \alpha^{\text{distance}(p_i, co)/s} + \alpha^{\text{distance}(p_i, oc)/s})/3, \\ \\ 0, \text{if } p_1 \text{ and } p_2 \text{ connect, meet, overlap, touch, are tangent, or one is inside} \\ \text{the other} \end{cases}$$

where $s = \min(\text{size}(p_1), \text{size}(p_2))$ and α is set to 0.2.

We examine obstruction calculation for the pair of lines (p1, p2) in Figure 5.21.

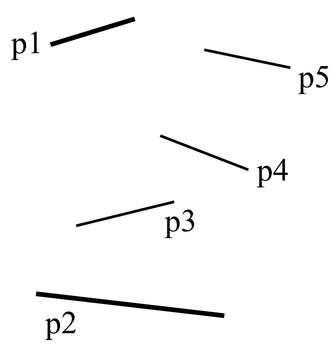


Figure 5.21 Example lines

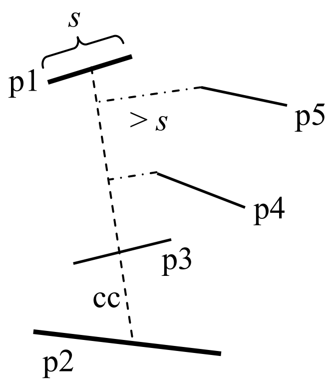


Figure 5.22 Positions of primitives relative to the connecting line cc

Figure 5.22 shows the special line cc connecting the centers of lines p1 and p2. In this case s is equal to the length of p_1, since it is the smaller of the two lines. Line p3 intersects cc, so distance(p_3, cc) / s = 0 and $\alpha^{\text{distance}(p_3,\, cc)\,/\,s} = \alpha^0 = 1$. Line p4 causes less obstruction: $\alpha^{\text{distance}(p_4,\, cc)\,/\,s} = \alpha^{0.5} = 0.45$. When distance($p_3$, cc) exceeds s the exponent becomes greater than 1, and the obstruction will become less than α = 0.2, which is relatively small.

The analysis is analogous for the connecting lines oc and co. We divide the obstruction values obtained for each of the connecting lines by 3, so that if some primitive intersects all three of them the total value would come to 1 (Figure 5.23):

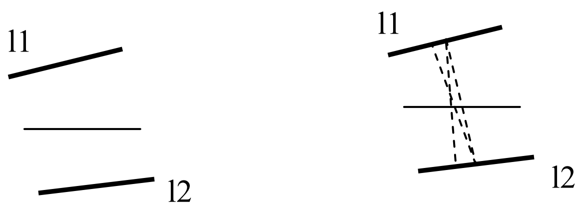


Figure 5.23 One line separates lines l1 and l2

Notice, however, that there is a problem with defining obstruction in terms of the distance to the connecting lines. Consider the example in Figure 5.24:

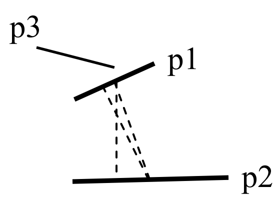


Figure 5.24 Line p3 should not obstruct p1 and p2

Line p3 should not obstruct the pair (p1, p2), but it is very close to the connecting lines so the obstruction formula would give a value close to 1. To deal with this problem we remove from consideration all the primitives that are behind what we call the *boundary infinite lines* for the primitives p1 and p2. These lines narrow down the region where a primitive can obstruct the pair (p1, p2) (

Figure 5.25):

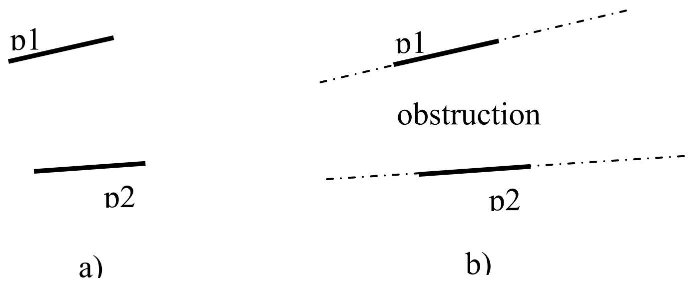


Figure 5.25 a) Pair of lines. b) Boundary lines for the pair (p1, p2) and the obstruction region

The obstruction values are calculated only for primitives that are fully or partially contained in the obstruction region between the boundaries. For each of the primitives in a pair (p1, p2), the boundary is defined depending on the relative orientation of the primitive and the line cc connecting the centers of p1 and p2. The goal is always to keep the boundaries close to parallel. We define two cases, depending on the acute angle α between the primitive and the line cc:

- $\boldsymbol{\alpha \geq 72^o}$**:** This means that the primitive is close to being perpendicular to the connecting line cc. The boundary in this case is simply the extension of the line:

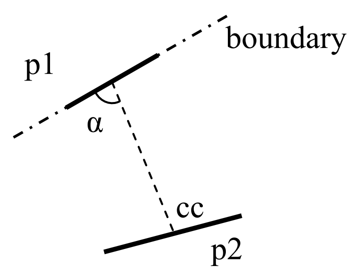


Figure 5.26 Boundary for the primitive p1

- $\boldsymbol{\alpha < 72^o}$**:** In this case the primitive is close to facing the other primitive in the pair with its endpoint. The boundary is perpendicular to the line cc and passes through the endpoint of the primitive p1, with a small offset (MAX_OFFSET). The offset is included so that lines connected to this endpoint would not be considered behind the boundary:

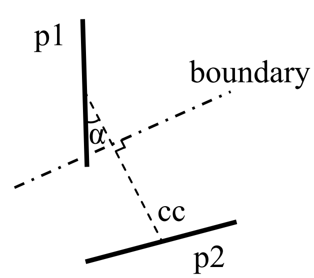


Figure 5.27 Boundary for the primitive p2

For ovals, the boundary is perpendicular to the line cc. As in the previous case, there is a small offset (MAX_OFFSET) that exposes part of the oval, so that a line tangent or meeting the oval at that part would not fall behind the boundary:

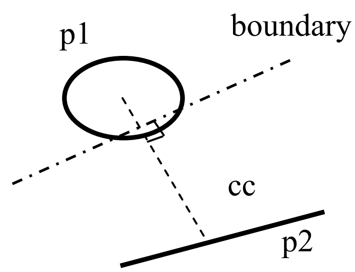


Figure 5.28 Boundary for the oval

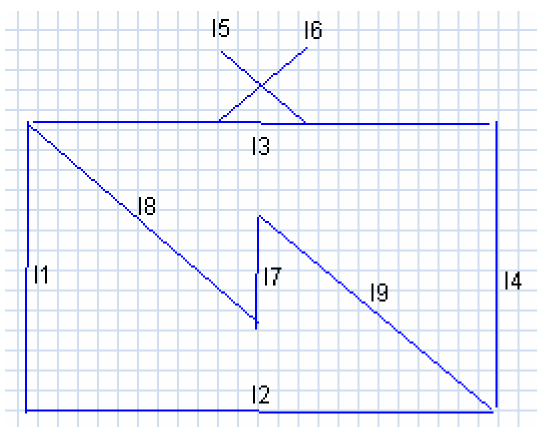


Figure 5.29 Military symbol

The table below shows obstruction values for the symbol in Figure 5.29. As we can see, the obstruction value for the pair (l3, l2), for example, is 3.9. It is caused by the lines l7, l8, and l9, and somewhat by the lines l4 and l1. As defined by the obstruction equation, when two primitives touch, the obstruction will be zero, as for the pair (l3 l1), for example.

	l1	l2	l3	l4	l5	l6	l7	l8	l9
l1	0								
l2	0	0							
l3	0	3.9	0						
l4	4.8	0	0	0					
l5	3.2	4.7	0	1.5	0				
l6	2.5	4.7	0	2.3	0	0			

l7	1.5	0.5	0.6	1.5	2.4	2.4	0		
l8	0	0.8	0	3.9	2.6	1.7	0	0	
l9	3	0	0.8	0	1.9	1.9	0	0	0

5.5 Grouping

This is the final processing step before relevance scores can be calculated for all the constraints. We support two grouping principles: connectedness and familiarity. The system produces candidate groups by segmenting the drawing into connected components and identifying previously learned symbols as drawing subparts. It then combines these groups into a hierarchy and merges any groups that share the same primitives. This section describes these steps in detail.

5.5.1 Connected components

Any two primitives that touch in some way are considered to be in the same connected component. To compute the components, the system constructs a graph in which nodes are primitives and an undirected edge exists for any pair of primitives constrained by “connects”, “meets”, “intersects”, “touches”, “overlaps”, or “tangent.” The system performs a depth-first search on this graph to find its connected components, which correspond to the connected components in the symbol.

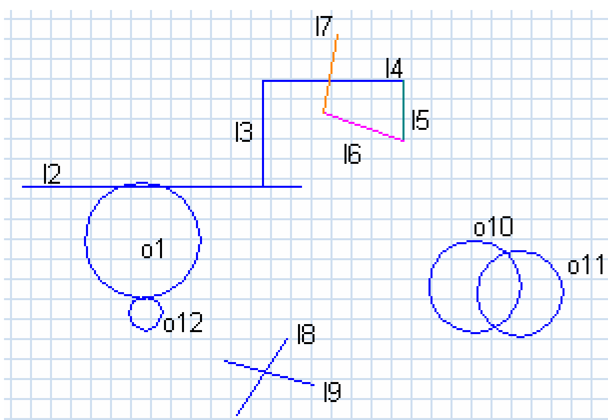


Figure 5.30 Examples of touching primitives

The system identifies three connected components in Figure 5.30:

Component 1: l6, l7, l4, l5, l3, l2, o1, o12
Component 2: o11, o10
Component 3: l9, l8

5.5.2 Previously learned symbols

To identify the second set of candidate groups the system looks for previously learned symbols as subparts of the new symbol. For each stored symbol it searches for a mapping of primitives that makes its constraints a subset of all the constraints in the new symbol.

For example, the primitives l7, l6, l8, and l5 in Figure 5.31b satisfy the constraints of the rectangle symbol in Figure 5.31a, given the mapping: (l1→l8) (l2→l5) (l3→l6) (l4→l7).

Figure 5.31 a) Rectangle. b) New symbol

Identifying previously learned symbols is a subgraph isomorphism problem on the symbol graphs, where the primitives are nodes and constraints are edges. We use Ullman's algorithm to compute the isomorphism [Ullman, 1976]. It proceeds by trying one mapping pair at a time and checking edges given the pairs so far, until it fails or finds the compete mapping. For example, if the algorithm is looking for the rectangle from Figure 5.31a in the symbol in Figure 5.31b, it can try setting (l1→l8). The "horizontal l1" constraint is satisfied for l8, so it proceeds to set the mapping for l2, now trying to ensure that the mapped primitives in the new symbol satisfy the same constraint as l1 and l2 in the rectangle symbol, and so on.

The running time of this algorithm is exponential in the number of primitives and linear in the number of previously learned symbols. We find that in practice it runs reasonably fast because most symbols have a small number of primitives and because mappings are quickly pruned when constraints involve only a few primitives.

Previously learned symbols may be related. For example, an isosceles triangle is a subclass of a triangle in general. The isosceles triangle has more constraints. The system keeps track of the subclass relationships between the learned symbols in a multiple-inheritance domain graph. Figure 5.32 shows such a graph for different kinds of triangles.

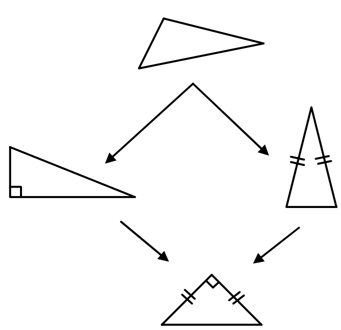


Figure 5.32 Domain graph for different types of triangles

The lines l8, l9, l10 in Figure Figure 5.31b would match all of these triangles. In such cases the system chooses the most specific interpretation, i.e. the one with most constraints. To achieve this, the matching process starts from the bottom of the domain graph. Once a set of primitives is matched to a symbol in the domain graph, there is no need to match this set to the ancestors of the symbol. We know that they are all guaranteed to match because they contain fewer constraints.

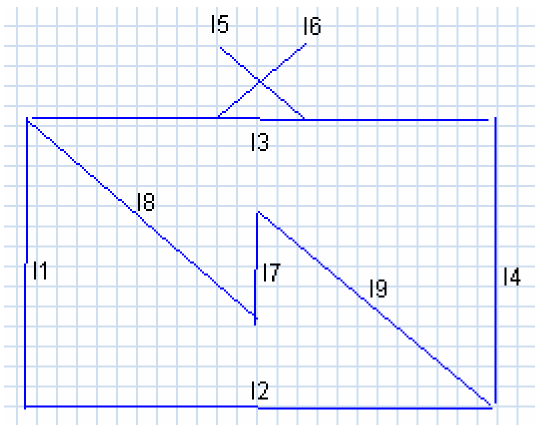


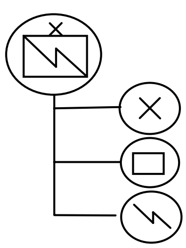
Figure 5.33 Military symbol

The system identifies previously learned cross and rectangle symbols as subparts of the symbol in Figure 5.33.

5.5.3 Combining grouping factors

The system combines candidate groups – connected components and previously learned symbols – into a group hierarchy.

Figure 5.34 shows the group hierarchy for the symbol in Figure 5.33.



Group g1 connected-component

Group g2 symbol – cross

Group g3 symbol – horizontal rectangle

Group g4 other (the remaining primitives in the connected component)

Figure 5.34 Group hierarchy of the symbol in Figure 5.33

If one group shares primitives with another group but cannot be its child or parent in the hierarchy, the two groups are merged into one. For example, in Figure 5.35 the system would find a triangle and a rectangle (the whole figure). They share the same primitive, so they will be merged into one group.

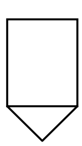


Figure 5.35 Symbol with competing groupings

This approach does not always produce the most salient grouping hierarchy. For example, Figure 5.36b shows the grouping hierarchy for the symbol in Figure 5.36a:

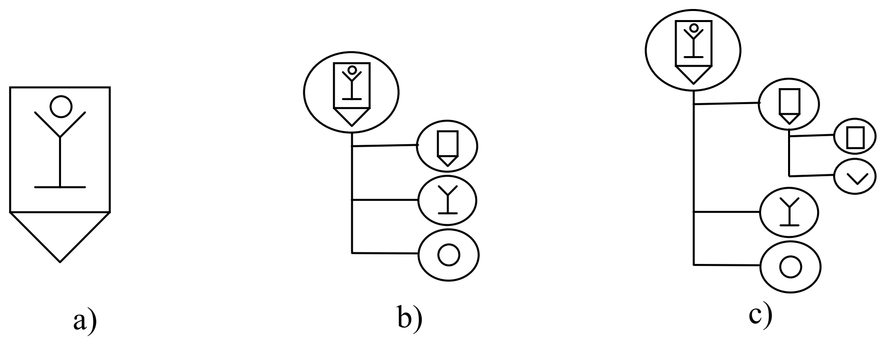


Figure 5.36 a) Symbol. b) Grouping produced by the system. c) Alternative grouping

As a result of merging, the grouping in Figure 5.36b does not recognize the rectangle as a salient part of the symbol. Consequently, the system will not record, for example, constraints like "inside (circle rectangle)", which would be more concise than specifying interactions of the circle with each of the primitives in the rectangle instead. The grouping in Figure 5.36c would be more appropriate.

A potential approach to this problem would be to resolve competitions between groups by picking a "winner" that gets to keep the shared primitives, rather than merging the groups. The winner could be defined, for example, as the group with the largest number of primitives. With that approach the system would produce a grouping shown in Figure 5.36c.

5.5.4 Group constraints

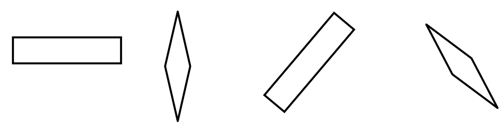
The system finds constraints between every two groups in the hierarchy that do not have an ancestor-descendant relationship. We currently support aspect ratio, orientation, relative position and relative size constraints, which are defined similarly to constraints on ovals and lines:

- **Aspect ratio:** The aspect ratio of a group is defined by the aspect ratio of the group's smallest-area bounding box (which does not have to be axis-parallel). This constraint is only identified for closed shapes. The group is "non-elongated" if the ratio of its length to its thickness is less than 1.5. Otherwise the group is elongated.

Group constraint	Example

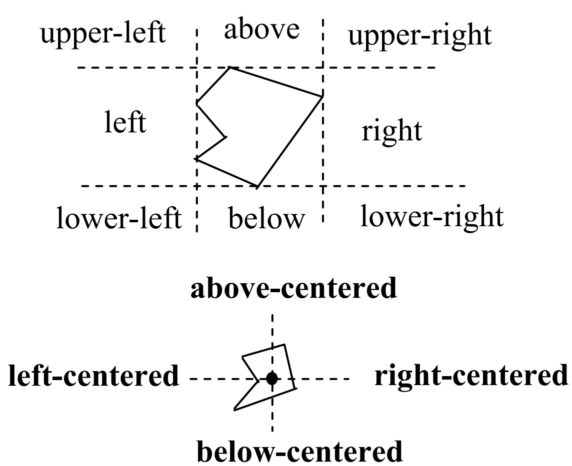
Group constraint	Example
Non-elongated, Elongated	

- **Orientation:** The constraint applies only to elongated groups. The orientation of the group is defined by the orientation of the longer axis of the smallest-area bounding box of the group. So it is computed as defined for lines in section 5.2.1.1:

Group constraint	Example
Horizontal, Vertical, Positive slope, Negative slope	

- **Relative position:** Position constraints are defined the same way as for lines and ovals in section 5.2.1 using the axis-parallel bounding box of the second group and the center of the first group (defined as the geometric center of the smallest-area bounding box). The position of the center relative to the bounding box determines the constraint (see the table below).
  "Inside" and "inside-centered" constraints are identified only if the outer group is a closed shape. "Inside-centered" holds if one group is inside another and the difference between the center coordinates of the groups is less than MAX_OFFSET and satisfies the SIZE_TO_OBJECT_RATIO. The size of the group is defined as the length of the smallest-area bounding box of the group. "Inside" constraints hold in the same loose sense as we mentioned for lines and ovals. The primitives of the inner group are allowed to touch the boundary of the outer group as long as the system does not identify "intersects" constraints.

Group constraint	Example



Group constraint	Example
Inside, **Inside-centered**	

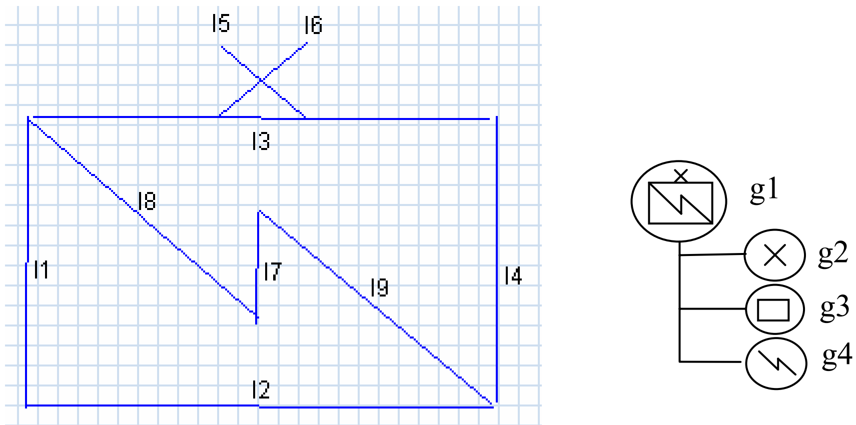


Figure 5.37 Military symbol

For the symbol in Figure 5.37 the system finds 4 group constraints:
above-centered: (g2 g4)
inside-centered: (g4 g3)
elongated: (g3)
horizontal: (g3)

5.6 Assigning relevance scores

A relevance score between 0 and 1 is computed for every constraint. This section explains how relevance scores are calculated based on:

- Default scores
- Obstruction
- Tension lines
- Grouping

5.6.1 Default scores

The default score for every constraint type is selected to approximate the relative perceptual relevance of the type:

Constraints	Default relevance score
Connects	1.0
Meets, Intersects, Tangent, Inside, Inside-centered	0.95

Touches, Overlaps	0.9
Horizontal (lines), Vertical (lines)	0.8
Positive slope, Negative slope, Position constraints (except inside), Parallel, Perpendicular	0.7
Horizontal (ovals), Vertical (ovals), Elongated, Non-elongated, Same-length, Same-size	0.6
Longer, Larger	0.55

We obtained these scores by ordering different types of constraints by their perceptually saliency, based on our introspection with various symbols, and assigning scores spread out in the interval between 0.5 and 1.0 according to this ordering.

More accurate relevance ordering could potentially be obtained through looking at a large variety of symbols, using the same approach as in Goldmeier's similarity experiments. In such an experiment the subjects would look at a symbol and two variations of it, produced by changing two constraints that we want to compare. The subjects would be asked which of the variations looks more similar to the original symbol and their choice would indicate which of the two compared constraints is less important. The constraint varied to produce the more similar symbol is the less perceptually relevant of the two, because changing it altered the perception of the symbol less. Section 3.1 provides such an experiment for comparing the importance of the degree of curvature to the importance of line straightness (Figure 3.3).

Three factors – obstruction, tension lines, and grouping – are used to increase and decrease the default scores of relative position, size, length, and orientation constraints. The score of all "touch" (i.e. connect, intersects, etc.), individual orientation, aspect ratio, and group constraints is not changed. As a result, these constraints will always remain in the description. In Chapter 7 on future work, we discuss why it may still be useful to rank these constraints by relevance and what could be done to enable the system to learn that in certain cases even these constraints may be irrelevant.

Each of the three adjustment factors pushes the relevance of a constraint up or down depending on the strength and direction of influence δ of this factor. For the relevance score r, the new score r' after adjustment will be:

$$r' = \begin{cases} r + \delta(1 - r), \text{if the factor } \delta \text{ is positive and hence increases the relevance.} \\ r + \delta r, \text{if the } \delta \text{ is negative.} \end{cases}$$

This formula achieves an asymptotic approach towards both 0 and 1.

The factors are applied in the order of:

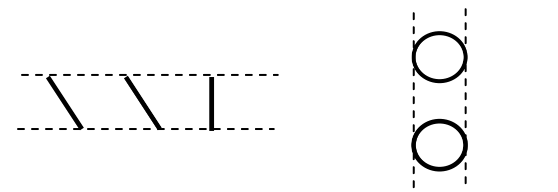
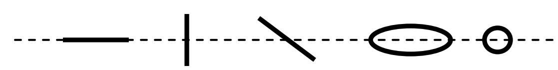
1. Obstruction
2. Tension lines
3. Grouping

5.6.2 Obstruction

An obstruction value is calculated for each pair of primitives, corresponding roughly to the number of primitives between the pair. The relevance of relative orientation, position, length, and size constraints for this pair will be decreased according to the amount of obstruction $O(p_1, p_2)$. This is intended to mimic the psychological observation that the more primitives are between a given pair the less we pay attention to the constraints for it. The influence constant for this factor is $\delta_{ob} = -0.15\ O(p_1, p_2)$.

5.6.3 Tension lines

Tension lines represent salient alignments of the primitives in a symbol. This factor increases the relevance of the relative position, length, and size constraints violating which would prevent the formation of identified tension lines. We deal with cases where the pair of primitives supports either one or two tension lines:

Relevance increased	Example
Affected constraints: Above- , below-, right-, and left-centered; Same-length; Same-size. **Condition:** The constraint is between two primitives that have endpoints on two parallel tension lines (formed by these or other primitives).	
Affected constraints: Above- , below-, right-, and left-centered. **Condition:** The constraint is between two primitives the centers of which are on the tension line with at least one more center point of another primitive.	 The relevance of the "right-centered" constraint for all of these pairs will be increased.

The influence constant for tension lines is $\delta_{tl} = +0.5$.

5.6.4 Grouping

Grouping affects relative orientation, position, length, and size constraints. The factor approximates people's tendency to pay attention only to aggregate properties of the grouped primitives and to ignore the individual interactions of primitives in different groups.

The system decreases the relevance of the constraints between a pair of primitives if they belong to two different groups. Examples of such primitives are shown in Figure 5.38 in bold:

a) b)

Figure 5.38 a) Two primitives in different connected components. b) Two primitives in previously learned shapes

The influence constant for a pair of primitives in different groups when neither of the groups is a previously learned symbol is $\delta_{dg} = -0.2$. If one or both of the groups is a previously learned shape we expect the attention to individual primitives to be even less so the constant is $\delta_{ds} = -0.4$.

5.6.5 Example

After applying the obstruction, tension lines, and grouping factors to adjust the default relevance scores, the system removes constraints with scores that ended up below the 0.5 threshold.

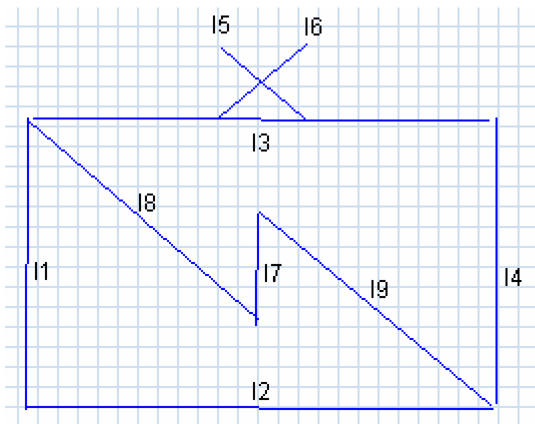


Figure 5.39 Military symbol

67 low-scoring constraints were removed from the initial list of 122 constraints for the symbol in Figure 5.39. Examples include:

parallel: (l5 l8)
same-length: (l7 l6)
upper-right: (l6 l1)
upper-left: (l5 l4)

5.7 Removing redundancies

The descriptions for previously learned symbols are available in the domain graph, so there is no need to list the constraints for those symbols in the new description. To produce the final description the system filters out all such constraints that pertain to the previously learned symbols that are part of the new symbol.

The final description for the symbol in Figure 5.39 contains 26 constraints, after removing 29 constraints related to the descriptions of the cross and the rectangle:

GROUP HIERARCHY: Group g1 connected-component: l6 l3 l5 l4 l2 l1 l8 l7 l9 Group g2 symbol - cross: l6 l5 Group g3 symbol - horizontal rectangle: l3 l4 l2 l1 Group g4 other: l8 l7 l9 CONSTRAINTS: elongated: (g3) above-centered: (g2 g4) inside-centered: (g4 g3) connects: (l4.p2 l9.p2) (l3.p1 l8.p1) (l2.p2 l9.p2) (l1.p1 l8.p1) (l7.p1 l9.p1) (l7.p2 l8.p2)	meets: (l5.p2 l3.cp2) (l6.p1 l3.cp1) vertical: (l7) neg-slope: (l9) (l8) right: (l9 l7) (l9 l8) (l7 l8) upper-right: (l6 l8) upper-left: (l3 l9) (l8 l2) above-centered: (l5 l3) (l5 l7) (l3 l7) (l7 l2) (l6 l3) (l6 l7) right-centered: (l4 l7) (l7 l1) parallel: (l8 l9) same-length: (l4 l9) (l1 l8) (l8 l9) longer: (l3 l7) (l3 l8) (l2 l7) (l2 l9) (l9 l7) (l8 l7)

Applying the mechanisms inspired by the studies on human perception and removing redundant information has allowed the system to reduce the number of constraints for this symbol from the initial 122 to 26.

Figures below demonstrate the variations of the symbol in Figure 5.39 that would and would not fit the description:

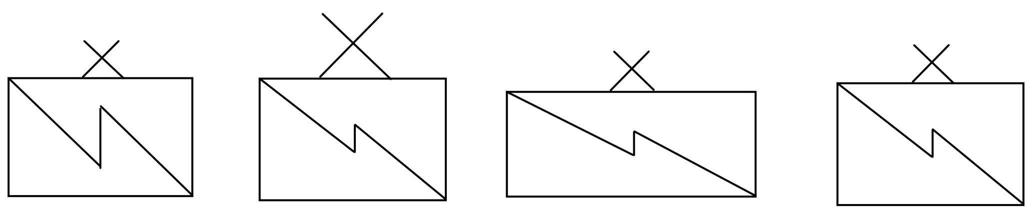


Figure 5.40 Examples of variations that would fit the description

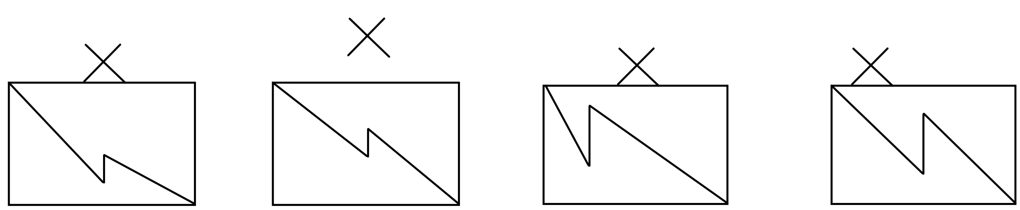


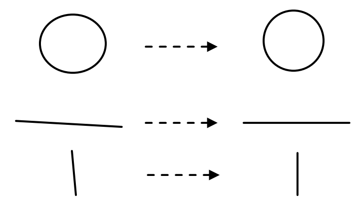
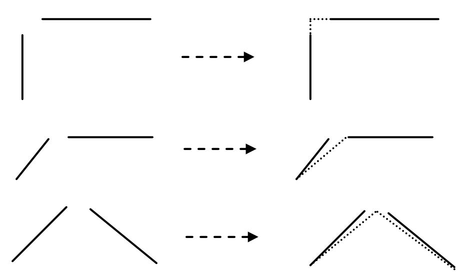
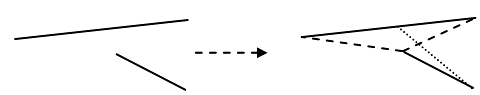
Figure 5.41 Variations that would not fit the description

5.8 User interface

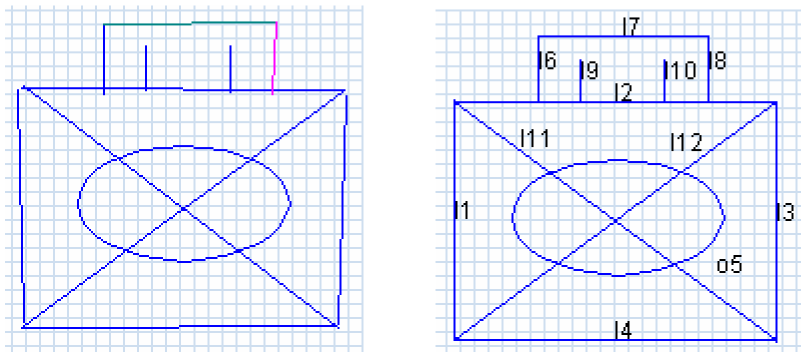
We would like the user to be able to check descriptions output by the system without having to read the text. We have taken initial steps towards creating a suitable interface for this purpose. It combines straightening the symbol to enforce some of the constraints in the description and displaying the rest of the constraints using simple graphical notation similar to the conventions in geometry textbooks.

5.8.1 Straightening the symbol

The system attempts to straighten the primitives in the symbol and enforce the constraints from the description. Currently, only orientation, aspect ratio, connects, and meets constraints are taken into account. The system proceeds through four steps:

Step	Example
1. Straighten individual primitives: Ovals satisfying the "non-elongated" constraint are turned into perfect circles. Lines that the system identified as horizontal or vertical are rotated through the center to achieve perfect alignment with the axes.	
2. Align collinear primitives: Axis-parallel lines that have the same orientation and satisfy "connects" constraints are made collinear.	
3. Enforce connections: Endpoints of lines satisfying "connects" constraints are adjusted in three ways: • If both lines are not slanted, their endpoints are extended to the point of intersection. • If one of the lines is slanted, its endpoint is connected to the other. • If both lines are slanted, the connection point is set to be the midpoint.	
4. Enforce meets constraints: The endpoint of the line that should "meet" the other line is adjusted to be on that line in such a way that the ratio of distances from the endpoint of the first line to the endpoints of the second line is preserved.	

Steps three and four are performed for each constraint without consideration of whether the transformation may break other constraints, so it is possible that not all of these constraints will hold in the final drawing. In practice, however, this algorithm works reasonably well. Figure 5.42b shows the straightened version of the symbol in Figure 5.42a.



a) b)

Figure 5.42 a) Original primitives. b) Straightened symbol

We have also explored an alternative way to straighten the symbol using tension lines (though chose to keep the first method for the current system). For all tension points on a given horizontal tension line the system sets the same y coordinate (calculated from their average). The same happens for the x coordinates for the points on the vertical tension lines:

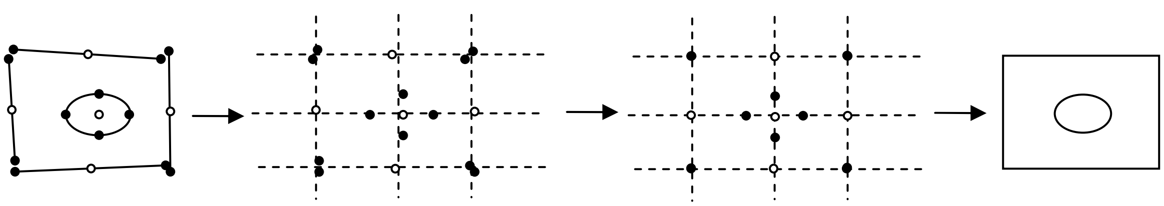
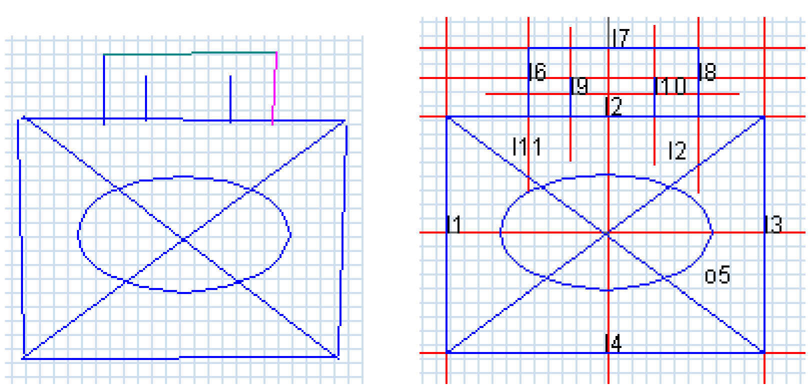


Figure 5.43 Straightening the symbol using tension lines

This mechanism usually produces more accurate results than the straightening algorithm described previously. Consider the example in Figure 5.44b for the symbol in Figure 5.44a.

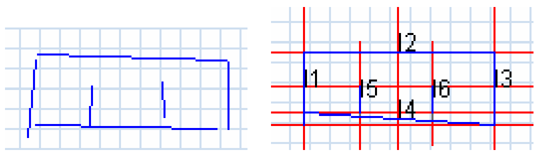


b)

a)

Figure 5.44 a) Original primitives. b) Straightened symbol

Unfortunately, the identified tension lines are sometimes contradictory. For example, several points from a short vertical line may appear on the same horizontal tension line because of the tolerance for noise. This makes straightening out much less reliable. Consider the result in Figure 5.45 below:



a) b)

TENSION LINES:
Horizontal: l2.p1 l2.center l1.p1 l2.p2 l3.p1
Horizontal: l6.p1 l5.p1 l3.center l1.center l6.center
Horizontal: l5.center l6.p2 l4.p1 l5.p2 l4.center
Horizontal: l3.p2 l4.p2 l1.p2

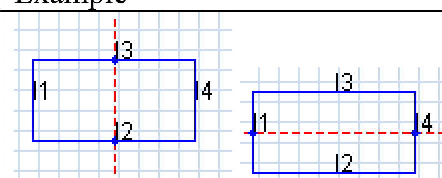
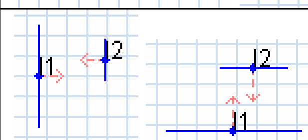
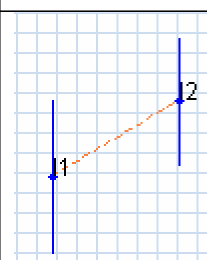
c)

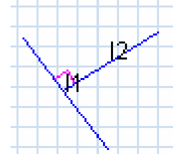
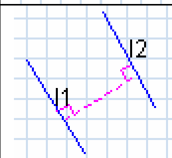
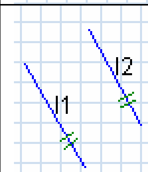
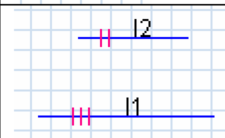
Figure 5.45 a) Original primitives. b) Drawing straightened according to tension lines. c) The list of horizontal lines the system identified in the drawing

We have not yet implemented a mechanism to remove such contradictions, so we mostly rely on the first method to straighten the symbol.

5.8.2 Graphical notation

In addition to straightening out the symbol we display some constraints graphically. For certain constraints, like same length and perpendicular, there are established conventions, like the ones used, for example, in geometry textbooks. For others we have created our own notation. We mark only the less obvious constraints, i.e. the ones that may not be evident from straightening the symbol:

Constraint	Notation	Example
Above-centered, below-centered	Centers of the primitives marked by dots. Dashed-line through the centers	
Above, below, left, right	Centers of the primitives marked by dots. Dashed arrow (axis-parallel) in the direction of the other primitive	
Upper-right, upper-left, lower-right, lower-left	Centers of the primitives marked by dots. Dashed line connecting the centers.	

Perpendicular	Square at the line intersection.	
Parallel	Squares at the corners of the lines and a line perpendicular line to them	
Same-length	Two short dashes through both lines	
Longer	Three dashes on the longer line and two dashes on the shorter line	

All the constraints related to a given primitive are displayed whenever the user clicks on it. The drawing may get cluttered if constraints of all types are displayed at the same time, so we provide a set of check boxes to specify which constraints should be shown:

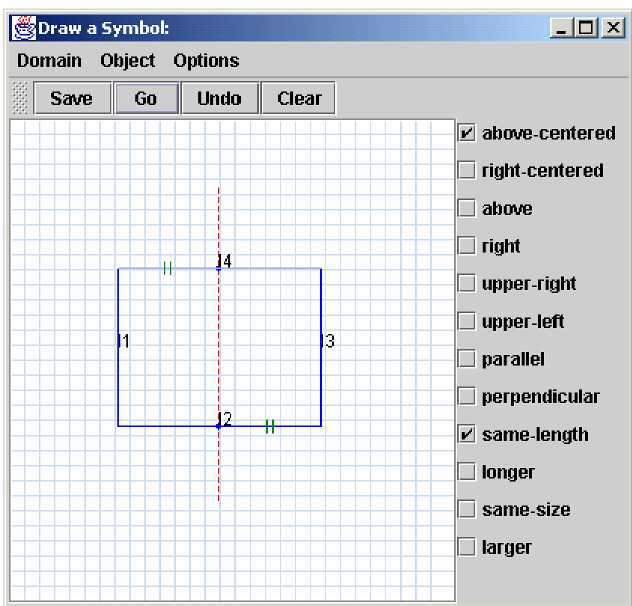


Figure 5.46 Choosing constraint types to display

Chapter 6 Evaluation

The ideal evaluation of the system would be to use the produced descriptions in a sketch recognition engine and test the recognition accuracy. The user would teach a symbol to the system and then draw multiple variations of it for the engine, noting false positives, false negatives, and the correct answers. As the recognition engine in the Design Rationale Group is still under development, the learning system had to be evaluated in isolation.

Our primary goal was to test whether the system accurately generalized the symbols using knowledge about human perception of geometry. We wanted to verify that the system captured the same properties that a person would pay attention to when learning a new symbol. To do this, we conducted a user study where subjects were shown an unknown symbol and several variations of it and asked whether each variation should be recognized as the original symbol. We tested whether the users accepted and rejected the same variations that would be accepted or rejected using the system's description.

In some domains, people may use domain-specific information to decide what properties are important. For example, in electric circuit symbols we know that the lines representing wires can have arbitrary length. Because the system only uses geometric information, we picked symbols from military planning – a domain the subjects were not likely to be familiar with and where symbols have little resemblance to the actual objects they represent.

We describe the procedure and the results of the study in the next sections.

6.1 Data set and study procedure

We used 9 symbols from the military planning domain (Figure 6.1). We chose the symbols to have a varying number of primitives and a varying number of contained known shapes.

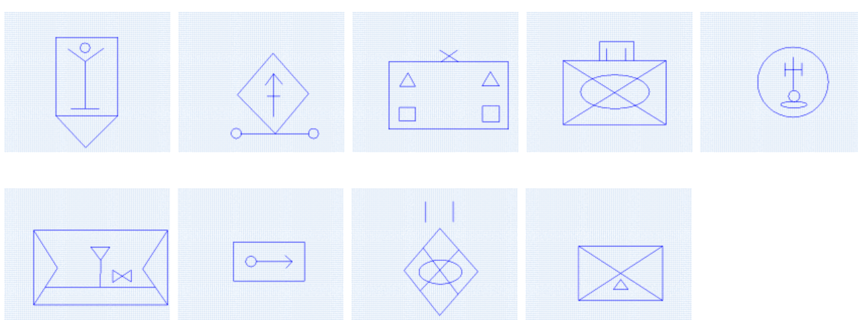


Figure 6.1 Test symbols

We examined the descriptions produced for these symbols and designed 20 variations for each one: 10 variations that would be accepted and 10 variations that would be rejected by the description produced by our system. Here is the procedure we used to do this:

The goal in constructing the variations was to approximate a uniform distribution of the changes across different properties and across degrees of change. To produce each variation we randomly picked to change one of the eight parts of the description (preferably without violating other constraints):

- Touch constraints: connects, meets, intersects, etc.
- Orientation
- Relative position
- Relative length and size
- Relative orientation
- Group aspect ratio
- Group relative position
- Number of primitives

We also randomly chose either a large or a small degree of change.

The original symbols and the variations were drawn with very low levels of noise (i.e. satisfying all constraints almost perfectly) so that people would not attribute the variations to sloppy drawing, but rather see them as routine changes to the original symbol. The variations that the system would accept or reject were randomly mixed. Appendix B presents the complete data set.

The subjects completed the study online. The drawings were shown one-at-a-time, each drawing occupying the whole browser window. They had to vote on 20 variations for each of the 9 original symbols. Before voting on each variation, the subjects were shown the original symbol, so that they would remember what it looked like. They were asked whether the variation should be recognized as the original symbol. Only "yes" and "no" options were provided. The subjects could take as much time as they needed to decide on the answer. We also provided the option to look at the original symbol again by pressing the "Back" button on the browser, in case the subjects were uncertain. The order of the original symbols was randomized for each subject to average out potential order effects. We surveyed 33 subjects getting judgments for a total of 180 variations (20 for each symbol).

6.2 Results

Before evaluating the agreement of the system with human judgment it is important to see whether the subjects agreed with each other. For each variation, we recorded the majority answer and the percentage of people who gave that answer (majority percentage). The chart in Figure 6.2 gives an assessment of the agreement levels.

The y-axis shows the proportion of the total of 180 variations for different levels of majority percentage on the x-axis. For almost 40% of the variations the subjects had high agreement – the majority percentage was above 90%. On more than half of data set the majority percentage was higher than 80%. Appendix B gives the detail on the votes and majority percentages for each variation in the data set.

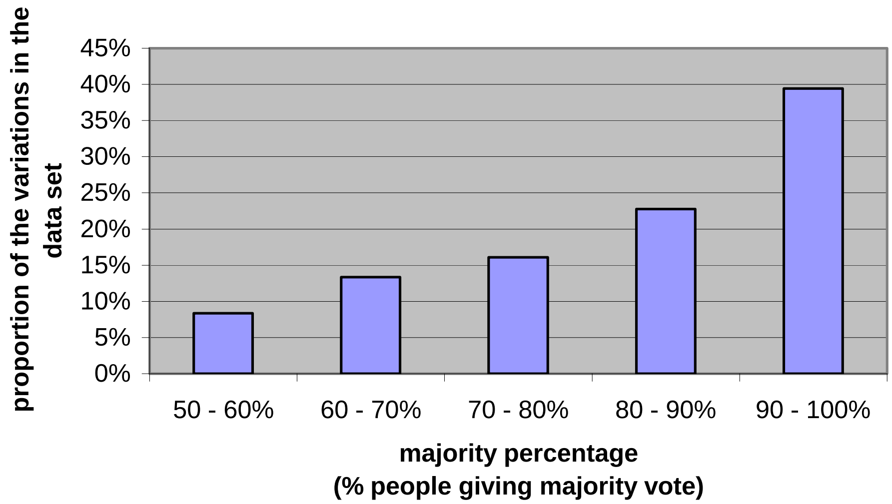


Figure 6.2 Levels of agreement for different variations

The chart shows that there were still a substantial number of cases (more than a third) where the subjects did not reach agreement, i.e. the opinions were strongly divided. Examples include:

(The question was: should the variation be recognized as the original symbol?)

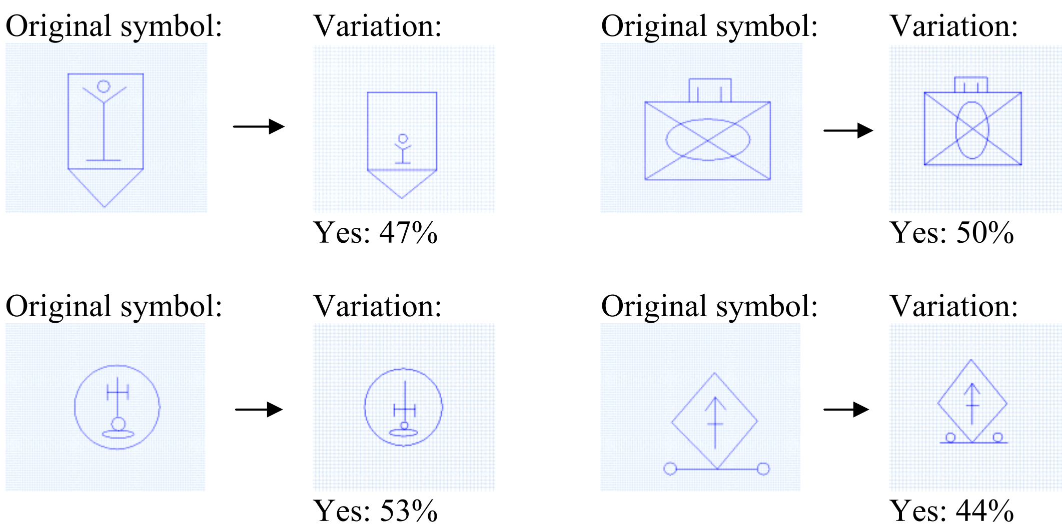


Figure 6.3 Variations that caused divided opinions.

We think that it is reasonable to expect divided opinions in some cases. The degree of perceptual similarity is a continuous property, yet we were forcing the subjects to make a binary decision. Subjects may differ on the exact threshold for when they consider a variation to be dissimilar enough from the original symbol to be rejected.

For such borderline cases, it makes less sense to evaluate the performance of the system (i.e. level of agreement with people) since people did not even agree with each other. Hence, we report the results not only for the complete data set, but also for the subsets of variations with high agreement (with majority ≥ 80% and majority ≥ 90%).

The chart in Figure 6.4 shows the evaluation results. We measured the proportion of times that the system agreed with the majority answer. For the whole data set the system achieved 77%. For the subset of the variations with higher intra-subject agreement (majority percentage ≥ 80%) the system achieved 83%. For an even smaller subset of data with the highest agreement (majority percentage ≥ 90%) the performance was 95%. Notice that the baseline performance is 50%. The system would agree with people half of the time if it guessed randomly.

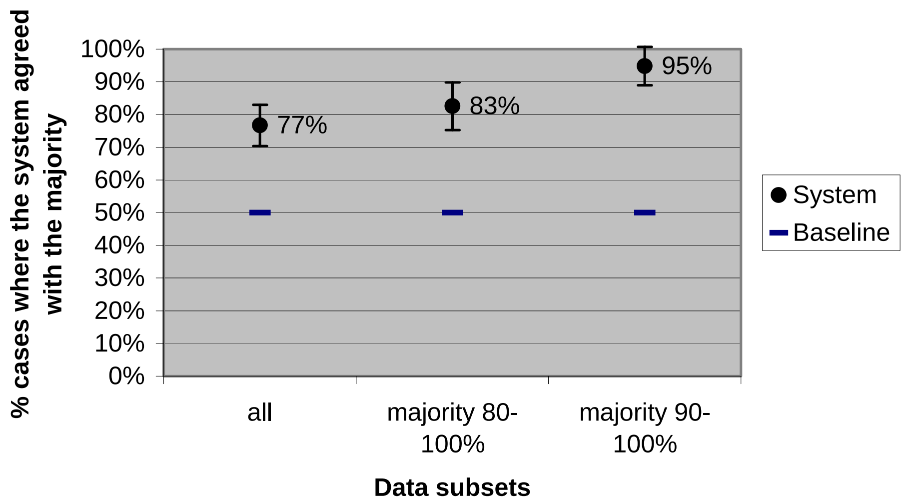


Figure 6.4 Percentage of cases where the system agreed with the majority answer

The system captures enough relevant information about the symbol to perform significantly above chance level. Yet there is still a lot of room left for improvement. In the next section we analyze the kinds of mistakes the system makes in order to assess what would be required to achieve better performance.

Notice that the data set was created to reflect variations that are produced by picking changes uniformly over all properties in the description. This set in not necessarily representative of the variations that people would be likely to produce when intending to draw the original symbol in a sketch. So we do not think these results are an accurate assessment of recognition accuracy. This is only an assessment of agreement in perceptual judgment. To measure potential recognition performance, a better way to construct the data set would be by showing people the original symbol and then asking them to draw it several times.

6.3 Analysis of disagreements

The system has produced both false positives and false negatives, though there were significantly fewer false positives.

6.3.1 False positives

These are cases where the variation fit the description, but the majority vote was not to recognize it as the original symbol. These cases fall into two categories.

In the first category the variation introduces connects, intersects, meets, or touches constraints that originally were not in the description. For example:

Original symbol: Variation: Original symbol: Variation:

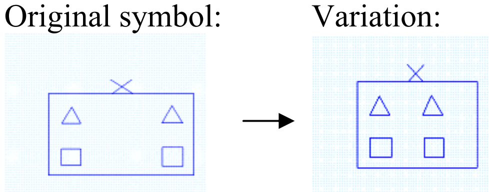
No: 72% No: 89%

Figure 6.5 Variations accepted by the description but rejected by the majority of the subjects

These examples fit the description, because the symbols are represented in the system by specifying which constraints should hold, rather than which constraints should not hold. Yet the majority of the subjects reject the variation because, perceptually, the symbol is altered significantly. To correct this kind of error the system would have to be extended to support "must-not" constraints. We think that these constraints would only be relevant for "touch" properties, like "connects", "meets", "intersects", "touches", "overlaps", etc., since these are most perceptually salient and can strongly alter the perception of the symbol.

The second type of disagreement is caused by the lack of explicit symmetry detection in the system. The variation below satisfies the description of the original symbol, even though it lacks symmetry. The majority, however, rejects the example (though this is only a slight majority).

Original symbol: Variation:



No: 53%

Figure 6.6 Variation that fits the description but is rejected by the majority

In summary, false positives arise because the system does not capture some properties of the symbol that have high perceptual relevance. The system does not look for these properties due to limited description vocabulary.

6.3.2 False Negatives

80% of the errors the system made were false negatives. These examples represent cases where variations of the symbol violate some description constraints, but the majority of the subjects still consider them similar enough to the original symbol to be recognized.

One type of false negatives occurs when the aspect ratio of a subpart of the symbol is changed, but people do not consider this change of the symbol significant:

Original symbol: Variation: Original symbol: Variation:

Yes: 89% Yes: 92%

a) b)

Figure 6.7 Changes in aspect ratio

Figure 6.7a, for example, is described by the system as having a "vertical rectangle," hence the rectangle in the variation of the symbol does not fit the description. Yet, for the subjects, it seems sufficient to just see a rectangle, regardless of the aspect ratio. We think that this effect may be related to the number of primitives in the symbol. When there is a lot of other detail in the symbol, people seem to generalize the representation on the composing sub-shapes more. The system could attempt to mimic this by recording more general versions of the previously learned shapes from the domain graph (Figure 6.8), if symbol containing the shape has a large number of other primitives. Currently the system always prefers the most specific versions.

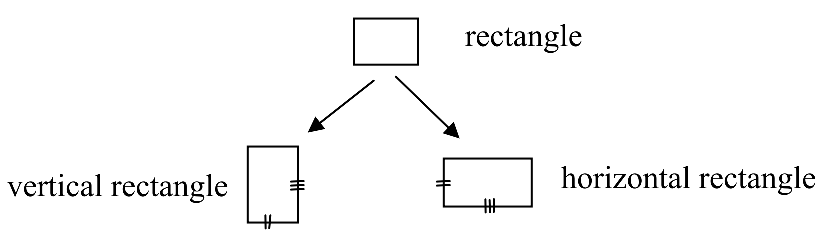
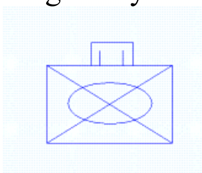


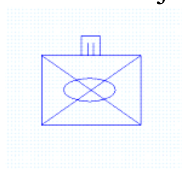
Figure 6.8 Domain graph that the system searches for previously learned shapes

Another type of false negatives we encountered were a few cases where the system found "longer" constraints to be important and included them in the description, yet the majority of the subjects accepted the variation with these constraints violated, for example:

Original symbol:



Variation rejected by the system:



Yes: 84% (the majority accepted the variation)

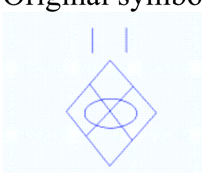
Figure 6.9 Changes in relative length constraints

Notice that in the variation of the symbol, the small top part of the symbol is no longer horizontally elongated as it is in the original symbol. This causes it to violate the "longer" constraints that were established between the "top" and the "sides" of this part.

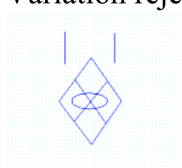
Perceptually this is similar to the aspect ratio problem that we described for the previous example. The difficulty, however, is that the system does not identify a separate aspect ratio property for the top part of the symbol that it could reason with. It also does not have a mechanism to downgrade importance of individual "longer" constraints in one part of the symbol due to the rest of the symbol containing a lot of primitives. These are important problems to investigate in the future.

The system also made one error related to position constraints:

Original symbol:



Variation rejected by the system, but accepted by majority:



Yes: 74%

Figure 6.10 Changes in position constraints

The system records, for example, that top-left side of the diamond in the original symbol is to the lower-left of the short vertical line above. When the two vertical lines are moved apart enough, the constraint no longer holds. The perceptual change is not very significant, however. It would have been enough to record that the vertical lines are above the top sides of the diamond.

All the examples above are composed of several high-level shapes: diamond, oval, rectangle, etc. It seems that the most perceptually relevant feature is the combination of these high-level shapes, and people pay less attention to the individual detail. The system needs to include more mechanisms for decreasing relevance of constraints on the primitives that constitute detail, when multiple previously learned shapes are present.

Chapter 7 Future Work

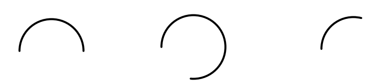
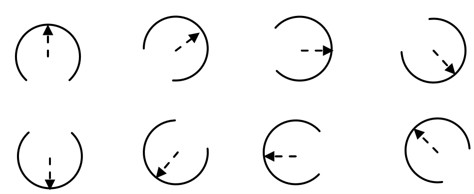
This chapter describes our ideas on improving the system's descriptive ability, achieving better relevance ranking by using domain information, and alternative approaches to the user interface.

7.1 *Extending the system's descriptive ability*

To represent a larger variety of symbols the system would need support for arcs, curved elements, and symbols that contain an arbitrary number of certain elements (like a resistor, or a dashed line). In addition, many symbols could be described more concisely if the system used higher-level constraints that include more than two primitives. The next sections outline potential steps towards reaching these goals.

7.1.1 Arcs

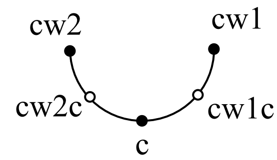
Incorporating arcs into the system would require defining a set of constraints that correspond to singular and non-singular arc properties. The table below shows a possible list of such properties:

Properties (singular ones shown in bold)	Example
Arc angle: **half-arc**, >half-arc, <half-arc	
Arc orientation: **top**, top-right, **right**, bottom-right, **bottom**, bottom-left, **left**, top-left	

Constraints defined similar to those for lines and ovals could also be used with arcs:

- Connects, meets, intersects, touches.
- Position constraints (referring to the center of the bounding box): above, right, left, below, upper-right, upper-left, lower-left, lower-right, above-centered, below-centered, left-centered, right-centered, inside, inside-centered
- Same-size, larger (referring to largest dimension of the bounding box)

Parameterized constraints like meets, connects, intersects, and touches would refer to the points on the arc in the table below:

Part	Notation	Example
First point on the arc in clockwise direction	**cw1**	
Any point between cw1 and the center of the arc curve	cw1c	
Center of the arc curve	**C**	
Any point between cw2 and the center of the arc curve	cw2c	
Second point on the arc in clockwise direction	**cw2**	

The descriptive power of these properties and constraints would have to be tested on a variety of symbols.

7.1.2 Curve representation

A large number of symbols contain spirals, waves, and other curved elements:

Figure 7.1 Symbols with curved elements

In many systems curves have been represented by parameters that do not easily capture the important perceptual characteristics. For example, consider Bezier curves. The small circles in Figure 7.2 shows the four Bezier control points for the drawn curve. Two of them do not lie on the curve and it would be hard for a person to judge the positions of these points when looking at a given curve.

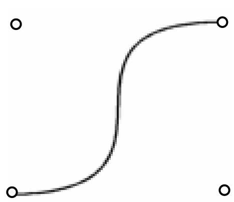


Figure 7.2 Control points for the Bezier curve

Bezier control points are not the perceptually salient elements of the curve. The positions of the endpoints, the existence of an inflexion point, and the “angular distance” traversed by the two segments separated by the inflexion point are probably more perceptually relevant. A description in these terms would capture the perceptual similarity between different curves in Figure 7.3, even though some of them are composed from more than one Bezier curve segment or from two arcs:

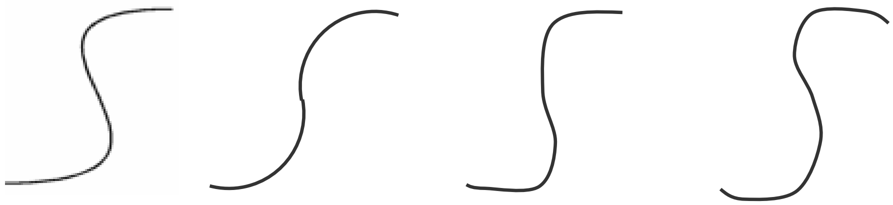


Figure 7.3 Perceptually similar curves

Future work should explore perceptually salient properties of curves to create a qualitative vocabulary for describing curved symbols.

7.1.3 Arbitrary number of elements

Symbols often have components that can be repeated an arbitrary number of times:

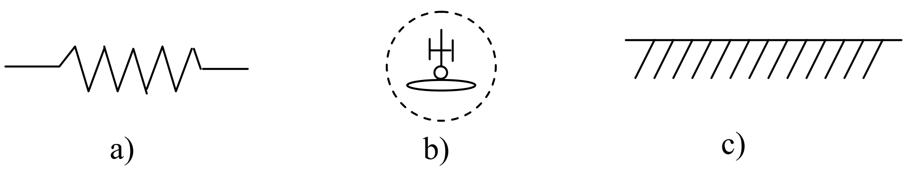


Figure 7.4 Symbols with varying number of primitives. a) Resistor symbol. b) Symbol from military planning. c) Symbol for ground or surface in mechanical engineering

Learning such configurations presents two challenges. The system first has to be able to identify a group of repeated components and, second, decide whether an arbitrary number of them is acceptable. Goldmeier's studies provide some hints on how this may be done. He distinguishes the geometric elements perceived by people as either *material* or *form*. Consider two experiments in Figure 7.5. Which of the b and c is more similar to a?

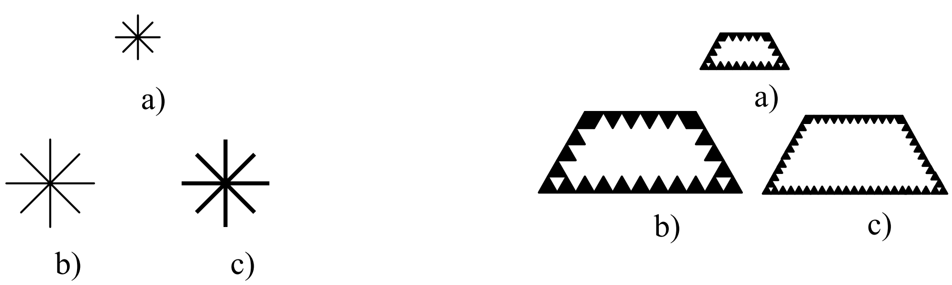


Figure 7.5 Which of b and c is more similar to a?

Even though uniform scaling of the symbol should not, supposedly, affect similarity, most of the subjects pick the example where the line width or the size of the small triangles remains the same (answers b and c respectively), i.e. the symbol that is not a uniformly scaled version of the original. Goldmeier argues that the lines of a certain width or the small triangles are perceived as material that makes up a larger shape (form). For the symbol to remain perceptually more similar, he claims that "the form is best preserved by proportional enlargement; material properties are best preserved by keeping the measurements of the material elements constant" [Goldmeier, 1972] However, ask yourself the same question for Figure 7.6:

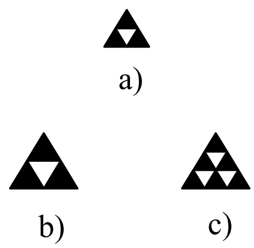


Figure 7.6 Which of b and c is more similar to a?

Most subjects choose b. In this case smaller triangles are not considered material. The difference between the cases when repeated elements can be viewed as material and when they should be viewed as form is best illustrated by Figure 7.7:

Figure 7.7 Which of b and c is more similar to a?

In the first experiment most subjects have picked c, treating the lines as material. However, in the second experiment they chose b. The presence of exactly three lines is perceived as a salient part of the form (structure) of the symbol.

According to Goldmeier, when the repeated elements are small compared to the size of the symbol and there is a large number of them, people start perceiving them as material rather than form and hence become insensitive to the variation in number of such components. The difficult task is defining quantitatively the terms "small relative to the symbol size" and "large number of elements."

7.1.4 Higher-level constraints

Due to the current restriction of the vocabulary to binary constraints, the system cannot capture certain constraints, even though they are perceptually salient. For example, the system does not represent symmetry constraint, which has been sown by Goldmeier to be a very important property [Goldmeier, 1972].

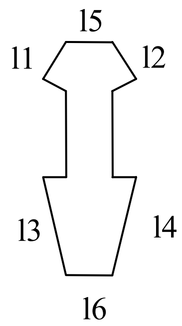


Figure 7.8 Symmetrical symbol

Tension lines, however, increase the relevance of some constraints violating which would break the symmetry. This sometimes helps implicitly capture the horizontal or vertical symmetry requirement. In Figure 7.8, for example, the tension line heuristic causes the system to increase the relevance of constraints "same-length: (l1 l2) (l3 l4)" and "above-centered (l5 l6)." In general, any two primitives symmetrical across the vertical or horizontal axis will form one or more tension lines, helping increase the relevance of constraints on their relative position and sometimes length:

Figure 7.9 Symmetrical segments form tension lines

However, currently there is no mechanism to require that the two elements should be equidistant from the symmetry axis or that they should have the same absolute slope. Hence, the system produces the same constraints for the symmetrical and non-symmetrical symbols in the pair of examples below, missing the fact that there is an important perceptual difference between them.

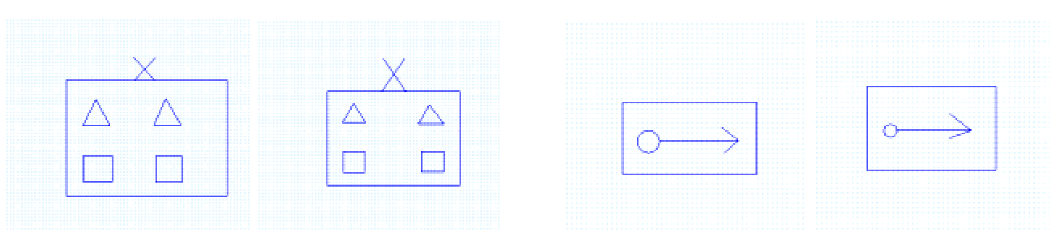


Figure 7.10 Pairs of symbols that would result in the same description

In addition to the non-binary symmetry constraint, the system would also benefit from adding constraints like interval equality between pairs of lines and alignment of several endpoints of different primitives. With these constraints symbols like the ones in Figure 7.11 could be described more concisely:

a) b)

Figure 7.11 a) Symbol requiring interval equality constraints. b) Symbols requiring alignment constraints

The only way the system currently allows constraining more than two primitives at a time is through group constraints. Improving grouping would help identify more accurate global constraints. The system supports only two grouping principles: connectedness and familiarity of shape. Proximity, similarity, continuity, and closure factors identified by gestalt psychologists [Wertheimer, 1923] need to be added to better approximate perceptually relevant grouping of the primitives within the symbol. Drawing order may possibly provide additional clues for grouping since we think that people will be more likely to draw perceptually salient components consecutively, without overlap.

7.2 Knowledge of Other Symbols in the Domain

Using knowledge about other symbols in the domain would help the system produce more adequate descriptions. Consider a simplified example, for the sake of explanation: assume that the description produced for the capacitor symbol in Figure 7.12a below did not include the constraint "same-length l2 l3."

a) b)

Figure 7.12 a) Battery symbol. b) Capacitor symbol

The subsequently presented battery symbol in Figure 7.12b would then match the capacitor description. The system should compare descriptions of different symbols in the domain and ensure that they have different descriptions by updating them appropriately. The correct action to take in this case would be to include the constraint "same-length l2 l3" to the capacitor description and "longer l3 l2" to the battery description. However, the challenge is that this is not the only constraint that distinguishes these symbols – there is also the relative length of lines l1 and l4, for example. Which distinguishing constraints should the system choose to include in the updated descriptions? We believe that it is important to explore the use of perceptual ranking of constraints for making such choices.

Note, for example, that the obstruction value for lines l4 and l1 is higher than for lines l2 and l3, hence the system would rank constraints between lines l2 and l3 higher on perceptual importance. It is those constraints that are better candidates for inclusion in the description.

7.3 Improved user interface

We have only started to explore the user interface for verifying the correctness of the descriptions produced by the system. We have experimented with displaying the constraints graphically, so that the user does not have to read the description. This method still requires a lot of concentration from the user and the notation quickly gets cluttered when the symbol has a lot of primitives.

The next section describes an alternative approach to checking produced descriptions that is based on variations of the symbol.

7.3.1 Automatic generation of potential "near misses"

Instead of displaying constraints graphically, the system could show different variations of the symbol that fit and do not fit the description and ask the user to accept or reject them. Then it would modify the description based on the responses.

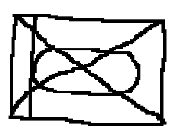


Figure 7.13 Military planning symbol

In Figure 7.13, the horizontal elongation of the rectangle and the oval may or may not be a required constraint. One way to verify that would be to ask the user whether the following examples should be recognized as the symbol:

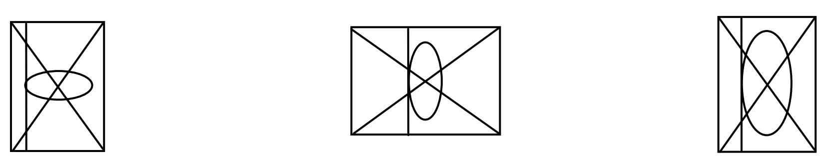


Figure 7.14 Examples with questionable constraints removed

The system would remove the constraints that are violated in the accepted examples and include missing constraints that differentiate the original symbol from the rejected examples.

The space of variations may be too large to explore exhaustively. For example, if a description contains 30 constraints and the option is to drop or keep each constraint, there may be up to $2^{30} \sim 1$ billion variations. Even if we assume that it is enough to check each constraint individually, the user would still have to look at 60 symbols. The main challenge is to generate only the few variations that the system could benefit from, i.e. the variations that explore the constraints that the system is "not sure" about.

The system could take advantage of relevance scores to identify such constraints, as they approximate the degree of perceptual salience. For example, there is no need to check the constraints that have a high score (like connects or meets). Removing those constraints would most likely produce a symbol that is significantly different and that the user would reject. That would give no new information to the system. On the other hand, varying constraints with scores near the filtering threshold is more likely to provide "near misses" that the system can learn from, because its judgment may differ from that of the user.

7.4 Relevance ranking for recognition robustness

A generic recognition engine will use the system's descriptions to identify symbols in user's sketches. If relevance scores were included in the description, the engine could use them for error-tolerant matching, making the recognition potentially more robust in the cases when the description is too constraining. Consider, for example, one of the constraints for the symbol in Figure 7.15.

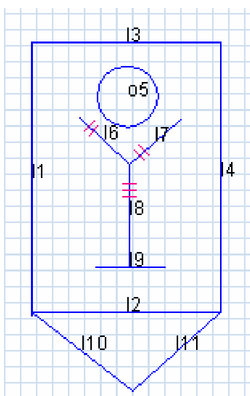


Figure 7.15 Military symbol

The system decides that line l8 should be longer than l6 and l7. Now assume that it is in fact incorrect, i.e. the user still wants the system to recognize the variations of this symbol where these constraints do not hold. The system gave these constraints relevance scores of 0.55, which are only slightly above the filtering threshold and lower than the scores of most other constraints (e.g. connects has a score of 1.0 and meets has a score of 0.9). Error-tolerant recognition would proceed by computing the matching error by summing the number of discrepancies between the input sketch and the constraints in the description, weighted by their relevance scores. Any input with a total error below a certain threshold would be considered to fit the description. When the description is incorrectly overconstrained, the engine may still recognize the input symbol, as long as the constraints that are required by the description but are missing from the input have low relevance.

Chapter 8 Conclusion

We have presented a system for learning shape descriptions from a single example of a symbol. By explicitly putting in knowledge about human perception we attempt to guide the generalization process. The generalization power derives from two sources:

1. Qualitative vocabulary of constraints based on perceptual singularities:

The vocabulary contains singular and non-singular terms, reflecting the property values that people attend to (singularities) and aggregating values that they ignore (non-singularities). This aggregation is an important initial generalization step.

In spite of the qualitative nature, the vocabulary is adequate for describing a large variety of symbols because it captures perceptually salient properties that we expect to be the basis for creation of graphical languages.

2. Perceptually inspired mechanisms for ranking constraints by relevance:

Constraints are assigned default relevance scores, based on their average perceptual importance. In addition, obstruction, tension lines, and grouping mechanisms that take into account the particular configuration of the primitives in the symbol cause these scores to be increased or decreased. These mechanisms reflect the observation that people pay attention to global properties of the symbol and that perceptual relevance of constraints is context-dependent.

As shown on several examples the system is capable of adequately describing complicated symbols with a lot of detail. We measure the success of the system in learning a new symbol by how well it captures the properties that people would pay attention to. The user study has shown that the system performs reasonably (83%) on the examples where the subjects agreed among each other.

Future work on the system would include improving its descriptive ability by providing support for curves and symbols with an arbitrary number of elements and by extending the constraint vocabulary to support higher-level constraints like symmetry, interval equality, and multiple alignments. As we have shown, knowledge about perception may provide further clues on how to achieve these extensions.

References

[Hammond and Davis, 2003] T. Hammond and R. Davis. LADDER: A Language to Describe Drawing, Display, and Editing in Sketch Recognition. To appear in Proceedings of IJCAI 2003

[Alvarado and Davis, 2002] C. Alvarado and R. Davis. A Framework for Multi-Domain Sketch Recognition. AAAI Spring Symposium on Sketch Understanding, 2002.

[Hammond and Davis, 2002] T. Hammond and R. Davis. Tahuti: A Geometrical Sketch Recognition System for UML Class Diagrams. 2002 AAAI Spring Symposium on Sketch Understanding, 2002.

[Alvarado and Davis, 2001] C. Alvarado and R. Davis. Resolving ambiguities to create a natural sketch based interface. Proceedings of IJCAI, 2001.

[Sezgin et al., 2001] M. Sezgin, T. Stahovich, and R. Davis. Sketch Based Interfaces: Early Processing for Sketch Understanding. Proceedings of PUI, 2001.

[Sezgin, 2001] Metin Sezgin. Feature Point Detection and Curve Approximation for Early Processing of Free-Hand Sketches. M.S. thesis, MIT, 2001

[Ullman, 1976] J. R. Ullman. An Algorithm for Subgraph Isomorphism. Journal of the ACM, Vol. 23, No. 1, January 1976, pp. 31-42.

[Goldmeier, 1972] Erich Goldmeier. *Similarity in Visually Perceived Forms*. Psychological Issues, Vol. 8, No. 1, 1972.

[Goldmeier, 1982] Erich Goldmeier. *The Memory Trace: its formation and its fate.* 1982.

[Arnheim, 1974] Rudolf Arnheim. *Art and Visual Perception*. University of California Press, 1974.

[Ullman, 1990] D. G. Ullman. The Importance of Drawing in the Mechanical Design Process. Computers & Graphics. Vol. 14, No. 2, pp. 263-274, 1990.

[Gross *et al.,* 1996] Gross, M. D. and E. Do. *Demonstrating the Electronic Cocktail Napkin. Conference Companion*. ACM Conference on Human Factors in Computing (CHI ‘96), pp 5-6. 1996.

[Gross, 1996] M. D. Gross. The Electronic Cocktail Napkin – a computational environment for working with design diagrams. Design Studies 17 (1996) 53-69. Elsevier Science.

[Gross and Do, 2000] Gross, M. D., E. Y.-L. Do, 2000. Drawing on the back of an Envelope: a framework for interacting with application programs by freehand drawings. Computers & Graphics 24(6): 835-849.

[Rubine, 1991] Rubine D. *Specifying Gestures by Example.* Computer Graphics, Vol. 25, No. 4, July 1991.

[Winston, 1970] Patrick Winston. *Learning Structural Descriptions from Examples.* Ph.D. thesis, MIT, 1970.

[Miller et al., 2000] E. G. Miller, N. E. Matsakis, P. A. Viola. Learning from One Example Through Shared Densities on Transforms. Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 2000.

[Arvo and Novins, 2000] J. Arvo. and K. Novins. Smart Text: A symthesis of Recognition and Morphing. In Proc. of AAAI Spring Symposium on Smart Graphics, pp. 140-147, 2000.

[Arvo and Novins, 2000b] J. Arvo. and K. Novins. Fluid Sketches: Continuos Recognition and Morphing of Simple Hand-Drawn Shapes. In ACM Symposium on User Interface Software Technology, pp. 73-80, 2000.

[Igarashi et al., 1997] T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive Beautification: A Technique for Rapid Geometric Design. ACM Annual Symposium on User Interface Software and Technology, pp.105-114, 1997.

[Di Fiore and Van Reeth, 2002] F. Di Fiore and F. Van Reeth. A Multi-level Sketching Tool for "Pencil-and-Paper" Animation. AAAI Spring Symposium on Sketch Understanding, 2002.

[Lipson and Shpitalni, 2002] H. Lipson and M. Shpitalni. Correlation-Based Reconstruction of a 3D Object From a Single Freehand Sketch. AAAI Spring Symposium on Sketch Understanding, 2002.

[Fonseca et al., 2002] M. J. Fonseca, C. Pimentel, and J. A. Jorge. CALI: An Online Recognizer for Calligraphic Interfaces. Proc. AAAI Spring Symposium: Sketch Understanding Workshop, 2002.

[Caetano et al., 2002] A. Caetano, N. Goulart, M. Fonseca, and J. Jorge. JavaSketchIt: Issues in Sketching the Look of User Interfaces. Proc. AAAI Spring Symposium: Sketch Understanding Workshop, 2002.

[Landay and Meyers, 2001] J. A. Landay, B. A. Meyers, Sketching Interfaces: Toward More Human Interface Design. IEEE Computer, 2001. 34(3): pp. 56-64.

[Shilman et al., 2002] M. Shliman, H. Pasula, S. Russel, R. Newton. Statistical Visual Language Models for Ink Parsing. In Proc. AAAI Spring Symposium: Sketch Undestanding Workshop, 2002.

[Feguson and Forbus, 1999] R. W. Ferguson, K. D. Forbus. GeoRep: A Flexible Tool for Spatial Representation of Line Drawings. Qualitative Reasoning Workshop, 1999.

[Connell, 1985] J. H. Connell. Learning Shape Descriptions: Generating and Generalizing Models of Visual Objects. MIT AI Lab Tech Report 853, 1985.

[Calhoun et al, 2002] Calhoun, T. F. Stahovich, T. Kurtoglu, L. B. Kara, 2002. Recognizing multi-stroke symbols. In AAAL 2002 Spring Symposium: Sketch Understanding Workshop.

[Kurtoglu and Stahovich, 2002] T. Kurtoglu and T. F. Stahovich. Interpreting Schematic Sketches Using Physical Reasoning. In AAAI 2002 Spring Symposium: Sketch Understanding Workshop.

[Forbus and Usher, 2002] K. D. Forbus, J. Usher. Sketching for Knowledge capture: A progress report. IUI'02, January 13-16, 2002

[Lin et al, 2002] J. Lin, M. Thomsen. J. A. Landay. A Visual Language for Sketching Large and Complex Interactive Design. CHI 2002. Vol. 4. No. 1, 307-314

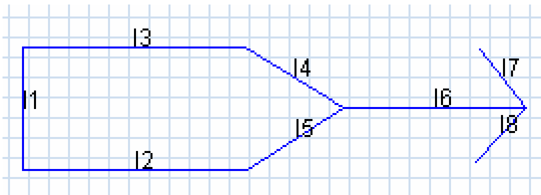
[Saund et al, 2002] E. Saund, J. Mahoney, D. Fleet, D. Larner, and E. Lank. Perceptual Organization as a Foundation for Intelligent Sketch Editing. In AAAI 2002 Spring Symposium: Sketch Understanding Workshop.

[LeCun et al, 1995] Y. LeCun, L.D. Jackel, L. Bottou, A. Brunot, C. Cortes, J.S. Denker, H. Drucker, L. Guyon, U.A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of Learning Algorithms for Handwritten Digit Recognition. In F. Fogelman and P. Galinari, editors, Inter. Conf. on Artificial Neural Networks, pp. 53-60, Paris, 1995.

[Wertheimer, 1923] M. Wertheimer. Untersuchungen zur Lehre von der Gestalt. Psychologische Forschung, 4, 301-50. Translation in W. D. Ellis (ed.) A Source Book of Gestalt Psychology. New York: H. B. J., 1938.

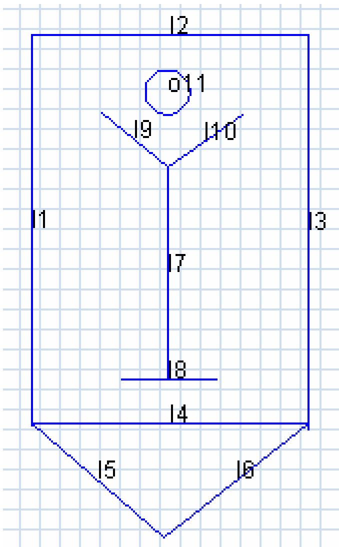
Appendix A

Initial Constraints for the Symbol



connects:(l5.p2 l4.p2) (l5.p1 l2.p2) (l5.p2 l6.p1) (l4.p2 l5.p2) (l4.p1 l3.p2) (l4.p2 l6.
p1) (l3.p2 l4.p1) (l3.p1 l1.p1) (l2.p2 l5.p1) (l2.p1 l1.p2) (l1.p1 l3.p1) (l1.p2
l2.p1) (l8.p1 l7.p2) (l8.p1 l6.p2) (l7.p2 l8.p1) (l7.p2 l6.p2) (l6.p1 l5.p2) (l
6.p1 l4.p2) (l6.p2 l8.p1) (l6.p2 l7.p2)
horizontal:(l3) (l2) (l6)
vertical:(l1)
pos-slope:(l5) (l8)
neg-slope:(l4) (l7)
above:(l7 l6) (l6 l8)
right:(l5 l1) (l4 l1) (l8 l1) (l7 l1)
below:(l8 l6) (l6 l7)
left:(l1 l5)
upper-right:(l5 l2) (l4 l2) (l3 l1) (l8 l2) (l7 l2) (l7 l5) (l6 l2) (l6 l5)
upper-left:(l4 l6) (l4 l8) (l3 l4) (l3 l5) (l3 l6) (l3 l7) (l3 l8) (l1 l2) (l1 l8)
lower-right:(l5 l3) (l4 l3) (l2 l1) (l8 l3) (l8 l4) (l7 l3) (l6 l3) (l6 l4)
lower-left:(l5 l7) (l2 l4) (l2 l6) (l2 l7) (l2 l8) (l1 l3) (l1 l4) (l1 l7)
above-centered:(l4 l5) (l3 l2) (l7 l8)
right-centered:(l8 l5) (l7 l4) (l6 l1)
perpendicular:(l5 l7) (l8 l7) (l7 l5) (l7 l8)
same-length:(l5 l1) (l5 l4) (l4 l1) (l4 l5) (l3 l2) (l2 l3) (l1 l4) (l1 l5) (l8 l7) (l7 l8)
longer:(l5 l7) (l5 l8) (l4 l7) (l4 l8) (l3 l1) (l3 l4) (l3 l5) (l3 l6) (l3 l7) (l3 l8)
(l2 l1) (l2 l4) (l2 l5) (l2 l6) (l2 l7) (l2 l8) (l1 l7) (l1 l8) (l6 l1) (l6 l4)
(l6 l5) (l6 l7) (l6 l8)

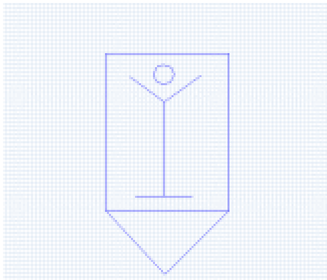
Initial Constraints for the Symbol



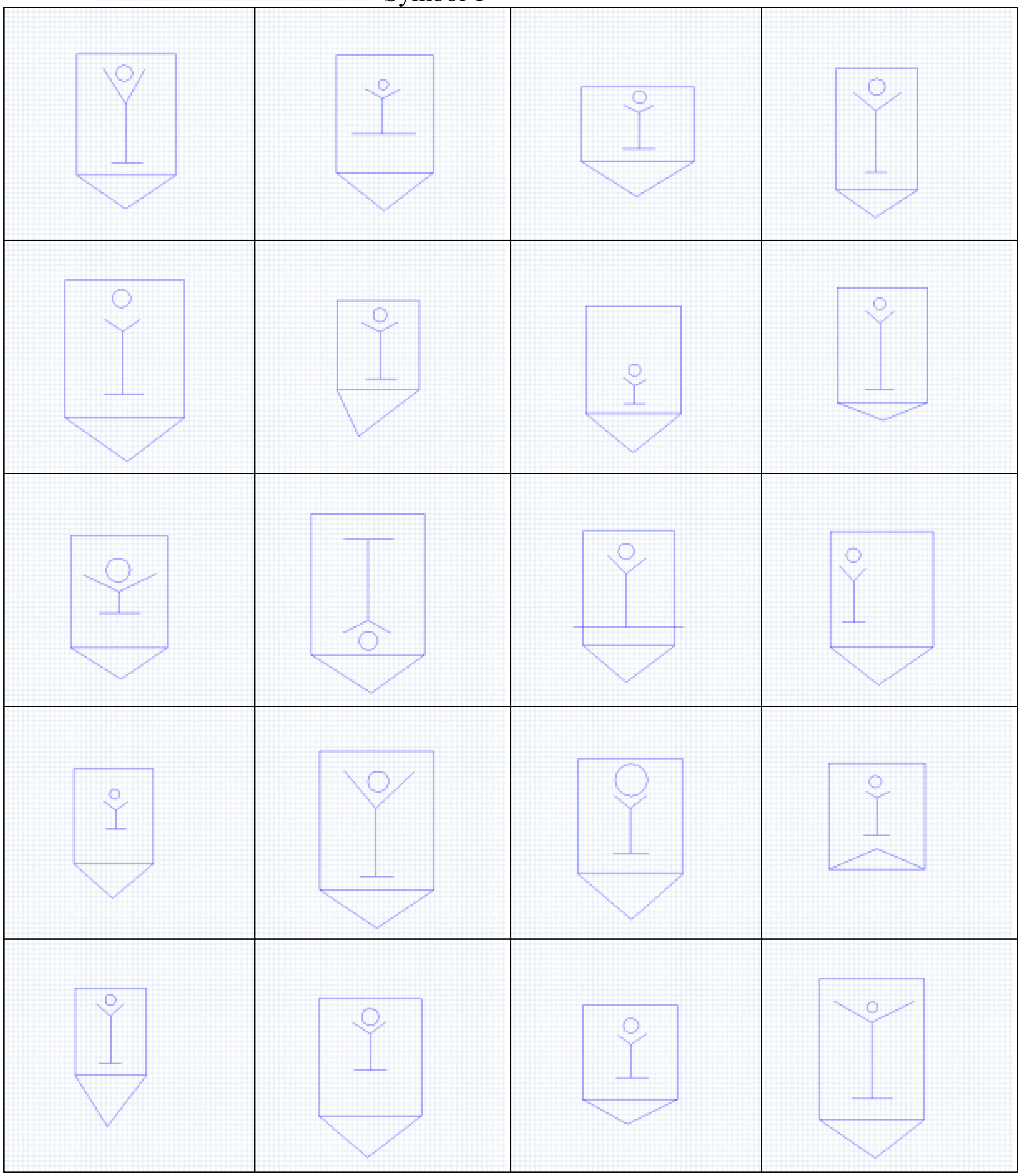
connects:(l6.p1 l5.p2) (l6.p2 l4.p2) (l6.p2 l3.p2) (l5.p2 l6.p1) (l5.p1 l4.p1) (l5.p1 l1.p2) (l4.p2 l6.p2) (l4.p1 l5.p1) (l4.p2 l3.p2) (l4.p1 l1.p2) (l3.p2 l6.p2) (l3.p2 l4.p2) (l3.p1 l2.p2) (l2.p2 l3.p1) (l2.p1 l1.p1) (l1.p2 l5.p1) (l1.p2 l4.p1) (l1.p1 l2.p1) (l10.p1 l9.p2) (l10.p1 l7.p1) (l9.p2 l10.p1) (l9.p2 l7.p1) (l7.p1 l10.p1) (l7.p1 l9.p2)
meets:(l7.p2 l8.c)
non-elongated:(o11)
horizontal:(l4) (l2) (l8)
vertical:(l3) (l1) (l7)
pos-slope:(l10) (l6)
neg-slope:(l5) (l9)
above:(l4 l5) (l4 l6) (l2 l6) (l2 l10) (l10 l4) (l10 l6) (l10 l8) (o11 l6) (o11 l10) (l9 l4) (l9 l5) (l9 l8) (l8 l6) (l7 l6)
right:(l3 l7) (l10 l1) (o11 l1) (l9 l1) (l8 l1) (l7 l1)
below:(l6 l2) (l6 l4) (l6 l10) (l5 l2) (l5 l4) (l4 l10) (l10 l2) (l9 l2) (l8 l10)
left:(l1 l7) (l10 l3) (o11 l3) (l9 l3) (l8 l3) (l7 l3)
upper-right:(l10 l5) (l10 l7) (o11 l5) (o11 l9) (l3 l4) (l3 l5) (l3 l6) (l3 l8) (l2 l1) (l2 l5) (l2 l9) (l8 l5) (l7 l5)
upper-left:(l2 l3) (l1 l4) (l1 l5) (l1 l6) (l1 l8) (l9 l6) (l9 l7)
lower-right:(l5 l1) (l10 o11) (l4 l1) (l4 l9) (l3 l2) (l3 l9) (l3 l10) (l3 o11) (l8 l9) (l7 l9) (l6 l1) (l6 l7) (l6 l8) (l6 l9) (l6 o11)
lower-left:(l5 l3) (l5 l7) (l5 l8) (l5 l9) (l5 l10) (l5 o11) (l4 l3) (l1 l2) (l1 l9) (l1 l10) (l1 o11) (l9 o11)
above-centered:(l2 l4) (l2 l7) (l2 l8) (l2 o11) (o11 l4) (o11 l7) (o11 l8) (l8 l4) (l7 l4) (l7l8)
right-centered:(l6 l5) (l3 l1) (l10 l9)
parallel:(l5 l9) (l10 l6)
same-length:(l6 l5) (l5 l6) (l4 l2) (l3 l1) (l2 l4) (l1 l3) (l10 l8) (l10 l9) (l9 l8) (l9 l10) (l8 l9) (l8 l10)
longer:(l6 l8) (l6 l9) (l6 l10) (l5 l8) (l5 l9) (l5 l10) (l4 l5) (l4 l6) (l4 l7) (l4 l8) (l4 l9) (l4 l10) (l3 l2) (l3 l4) (l3 l5) (l3 l6) (l3 l7) (l3 l8) (l3 l9) (l3 l10) (l2 l5) (l2 l6) (l2 l7) (l2 l8) (l2 l9) (l2 l10) (l1 l2) (l1 l4) (l1 l5) (l1 l6) (l1 l7) (l1 l8) (l1 l9) (l1 l10) (l7 l5) (l7 l6) (l7 l8) (l7 l9) (l7 l10)

Appendix B

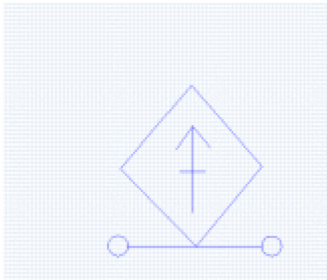
This appendix describes the test set that was used for the evaluation of the system. For each symbol (at the top of the page) there are 20 variations, shown in the same order as they were presented to the subjects. For each variation the subjects were asked whether it should be recognized as the original symbol. The subsequent table shows the answer according to the description produced by the system and the answer given by the subjects. It also includes the information on the majority percentage. The entries for the variations on which the subjects disagreed with the system are highlighted.



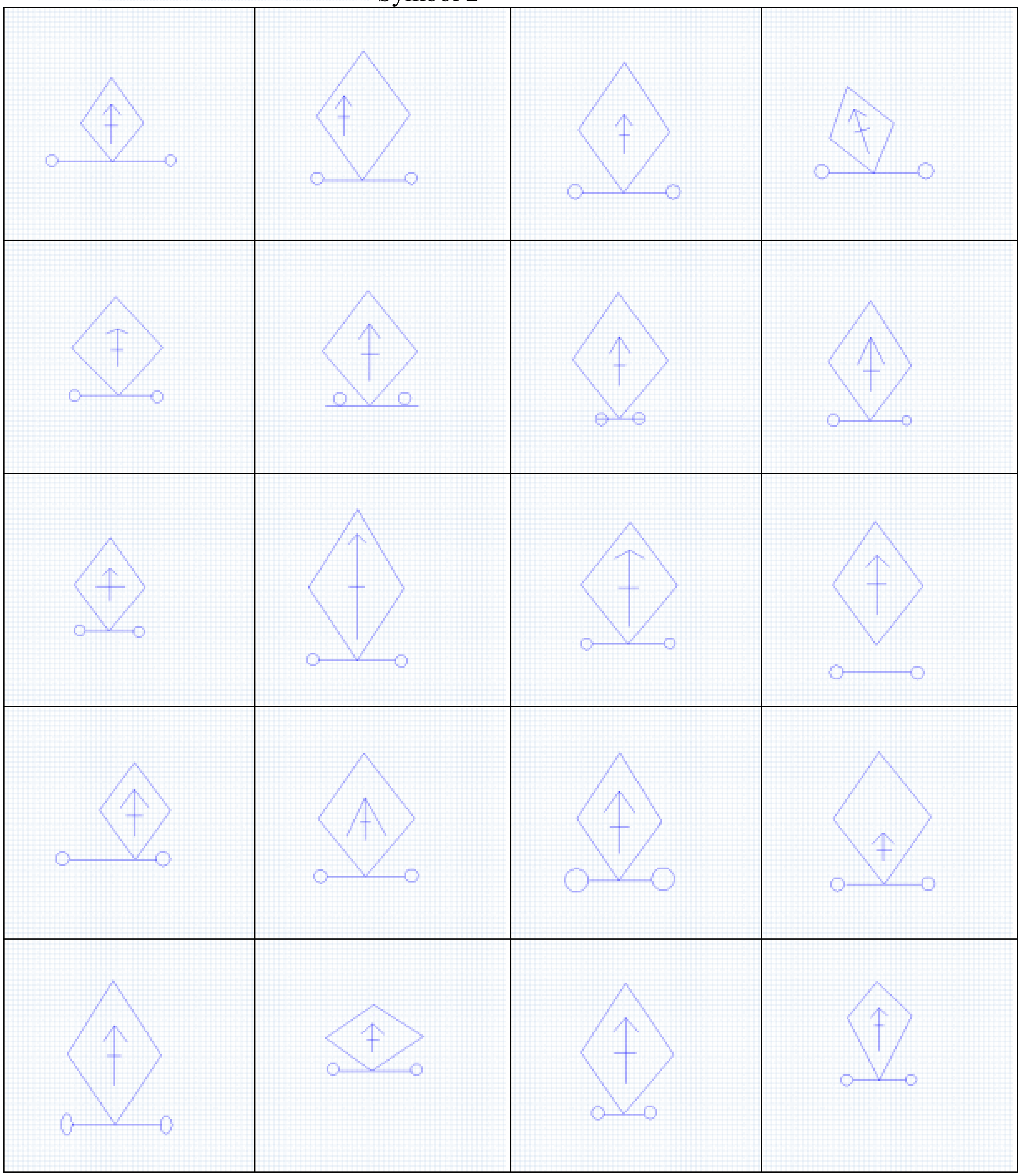
Symbol 1



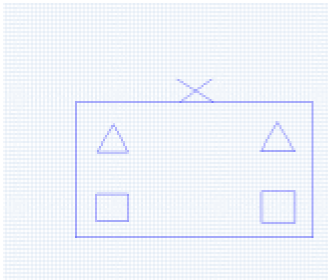
System: Majority: Majority %:	YES YES 97%	YES YES 89%	NO YES 89%	YES YES 100%
System: Majority: Majority %:	YES YES 66%	NO NO 63%	NO NO 53%	YES YES 89%
System: Majority: Majority %:	NO NO 50%	NO NO 92%	YES NO 89%	NO NO 76%
System: Majority: Majority %:	YES YES 74%	YES YES 89%	NO YES 55%	NO NO 92%
System: Majority: Majority %:	NO YES 92%	YES YES 84%	NO YES 89%	YES YES 79%



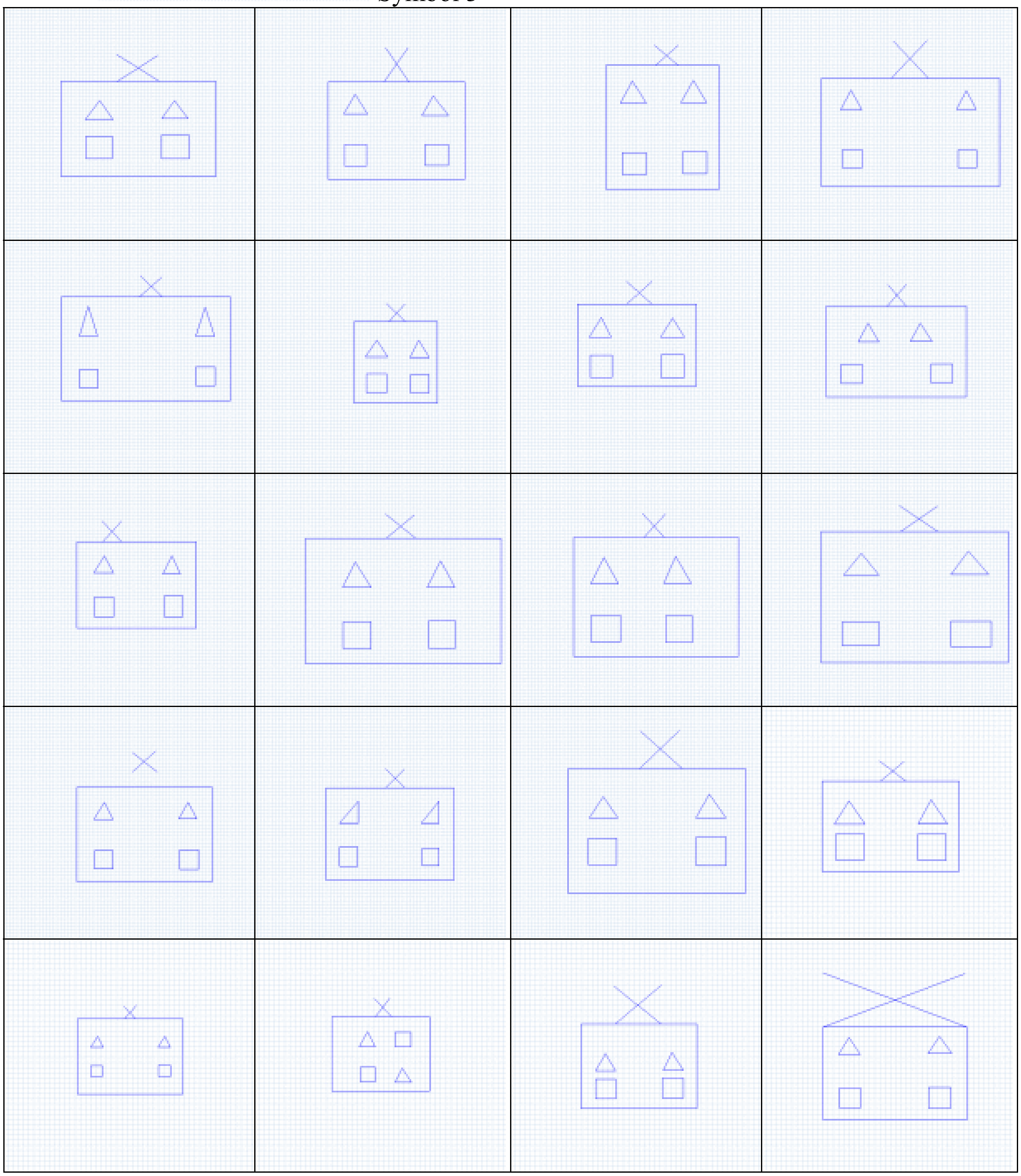
Symbol 2



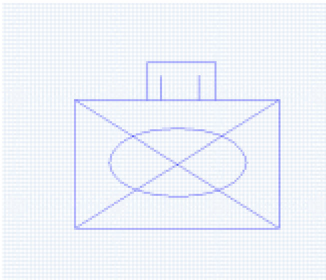
System:	YES	NO	YES	NO
Majority:	YES	NO	YES	NO
Majority %:	100%	76%	93%	73%
System:	YES	NO	NO	YES
Majority:	YES	NO	NO	YES
Majority %:	71%	56%	54%	80%
System:	YES	YES	YES	NO
Majority:	YES	YES	YES	NO
Majority %:	78%	76%	85%	83%
System:	NO	NO	YES	NO
Majority:	NO	NO	YES	NO
Majority %:	80%	90%	100%	76%
System:	NO	YES	YES	NO
Majority:	YES	YES	YES	YES
Majority %:	93%	73%	85%	59%



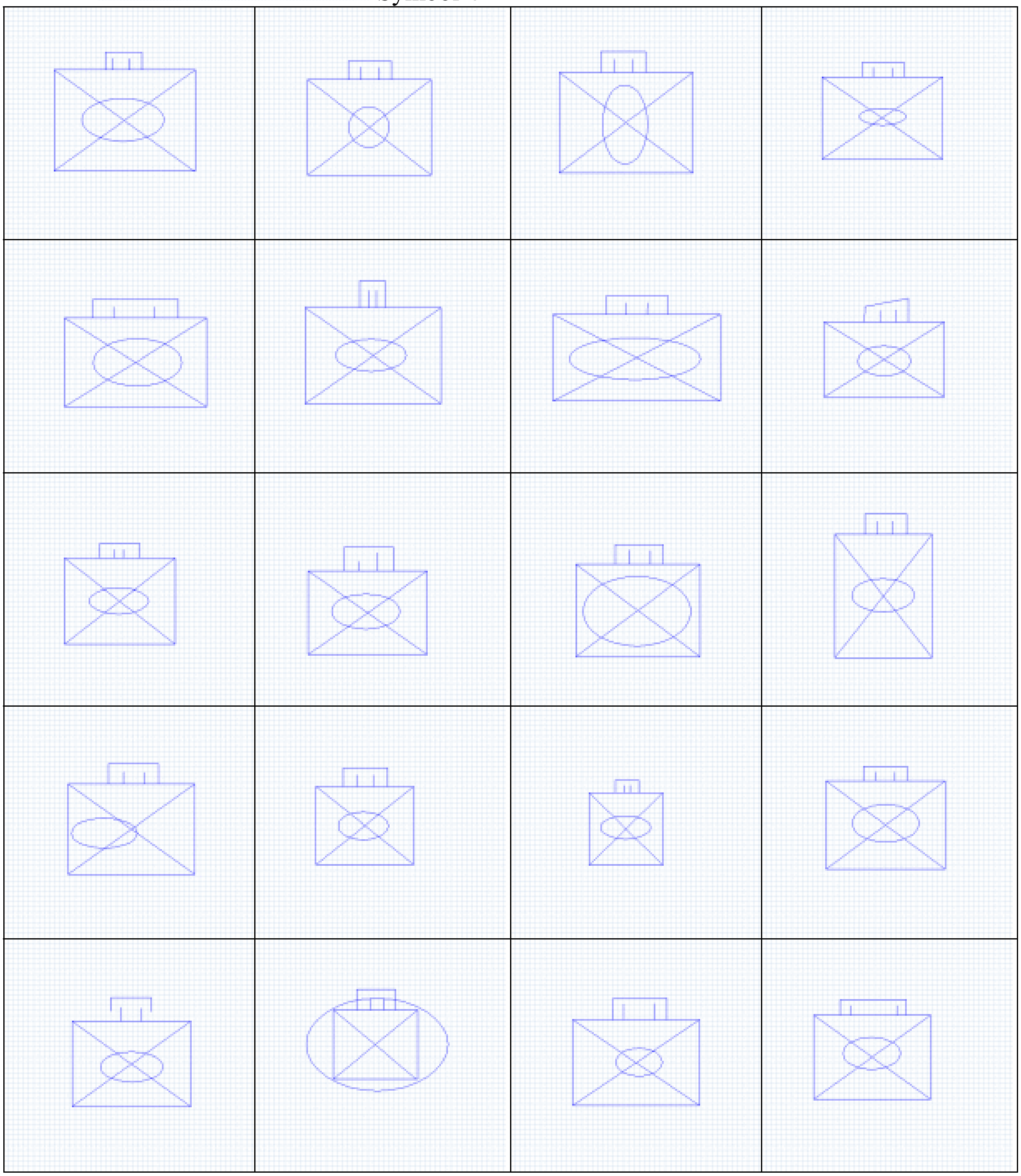
Symbol 3



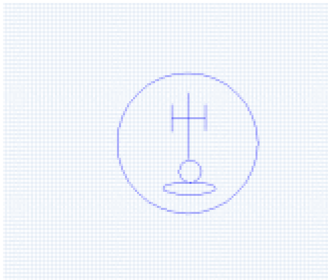
System:	YES	YES	NO	YES
Majority:	YES	YES	YES	YES
Majority %:	95%	95%	75%	95%
System:	NO	NO	YES	NO
Majority:	YES	YES	YES	NO
Majority %:	83%	75%	100%	85%
System:	NO	YES	YES	NO
Majority:	NO	YES	NO	YES
Majority %:	83%	85%	53%	90%
System:	NO	NO	YES	YES
Majority:	NO	NO	YES	YES
Majority %:	75%	93%	90%	93%
System:	YES	NO	YES	NO
Majority:	YES	NO	NO	NO
Majority %:	90%	93%	50%	98%



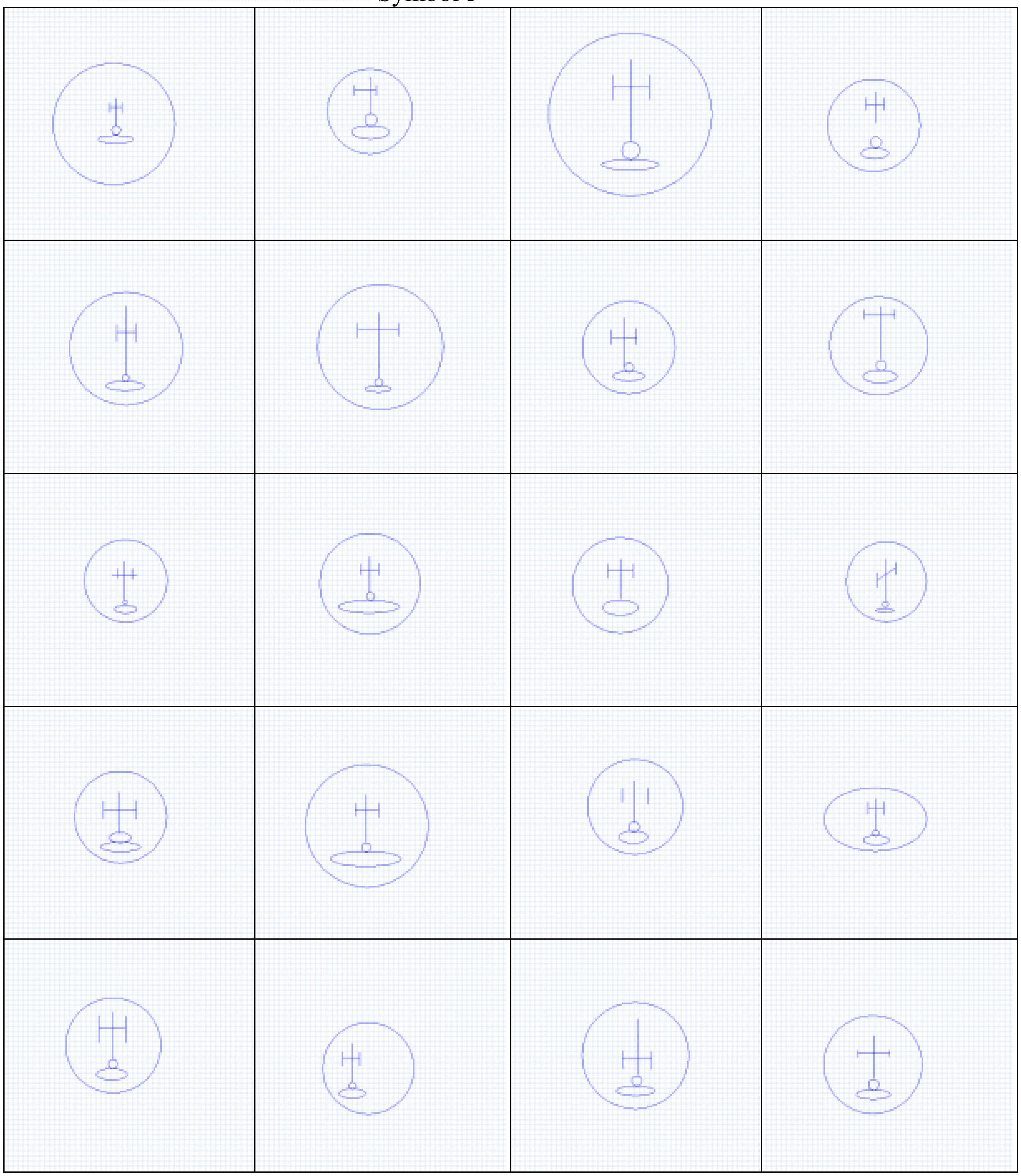
Symbol 4



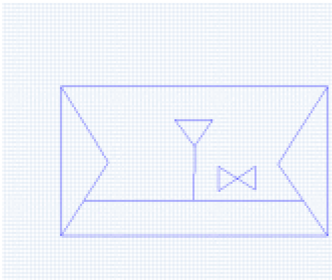
System: Majority: Majority %:	YES YES 95%	NO YES 79%	NO YES 50%	YES YES 84%
System: Majority: Majority %:	YES YES 84%	NO YES 84%	YES YES 100%	NO NO 61%
System: Majority: Majority %:	YES YES 84%	NO NO 55%	YES YES 84%	NO YES 63%
System: Majority: Majority %:	NO NO 97%	YES YES 84%	NO YES 82%	YES YES 97%
System: Majority: Majority %:	NO NO 92%	NO NO 100%	YES YES 89%	YES YES 74%



Symbol 5

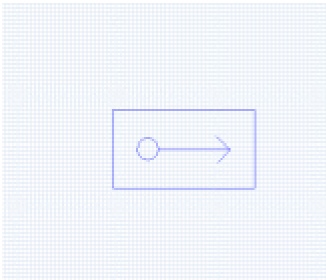


System:	YES	NO	YES	NO
Majority:	YES	NO	YES	NO
Majority %:	90%	53%	98%	88%
System:	YES	YES	NO	YES
Majority:	YES	YES	NO	YES
Majority %:	78%	68%	73%	73%
System:	NO	YES	NO	NO
Majority:	YES	YES	NO	NO
Majority %:	50%	73%	73%	90%
System:	NO	YES	NO	NO
Majority:	YES	YES	NO	YES
Majority %:	85%	80%	100%	78%
System:	YES	NO	YES	YES
Majority:	YES	NO	YES	YES
Majority %:	90%	85%	53%	78%

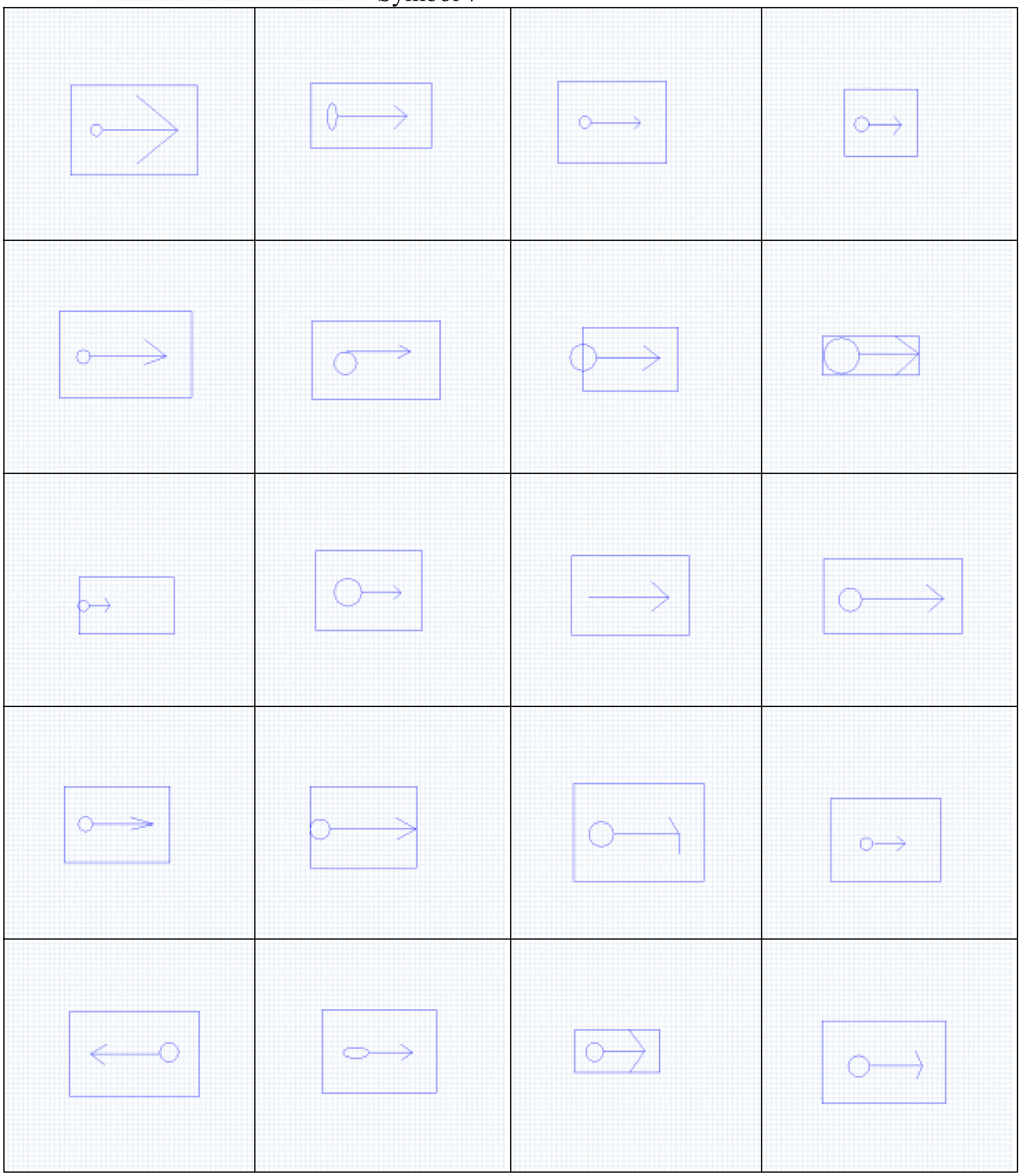


Symbol 6

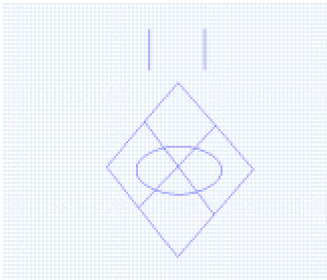
System:	NO	NO	YES	NO
Majority:	YES	YES	YES	YES
Majority %:	79%	92%	82%	64%
System:	NO	YES	NO	YES
Majority:	NO	YES	YES	YES
Majority %:	100%	97%	87%	95%
System:	NO	YES	YES	NO
Majority:	NO	YES	YES	NO
Majority %:	79%	97%	79%	95%
System:	NO	YES	YES	YES
Majority:	NO	YES	YES	YES
Majority %:	69%	92%	97%	69%
System:	NO	YES	YES	NO
Majority:	YES	YES	YES	NO
Majority %:	72%	79%	59%	95%



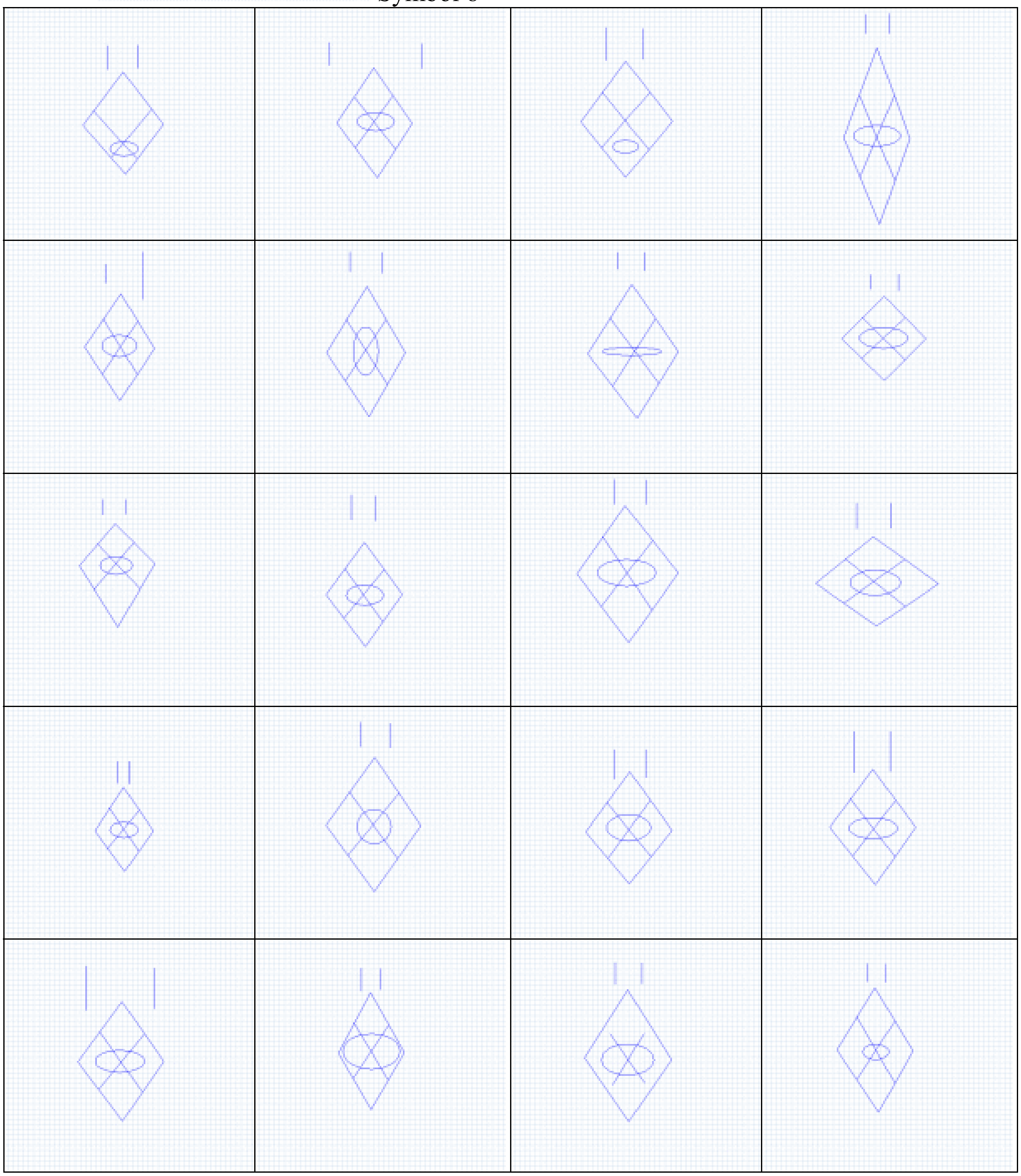
Symbol 7



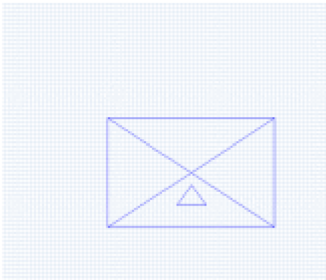
System:	NO	NO	YES	NO
Majority:	YES	YES	YES	YES
Majority %:	62%	82%	97%	85%
System:	YES	NO	NO	YES
Majority:	YES	NO	NO	NO
Majority %:	79%	97%	92%	72%
System:	NO	YES	NO	YES
Majority:	NO	YES	NO	YES
Majority %:	79%	79%	97%	100%
System:	YES	YES	NO	YES
Majority:	YES	NO	NO	YES
Majority %:	85%	51%	82%	79%
System:	NO	NO	YES	YES
Majority:	NO	YES	NO	YES
Majority %:	95%	69%	59%	85%



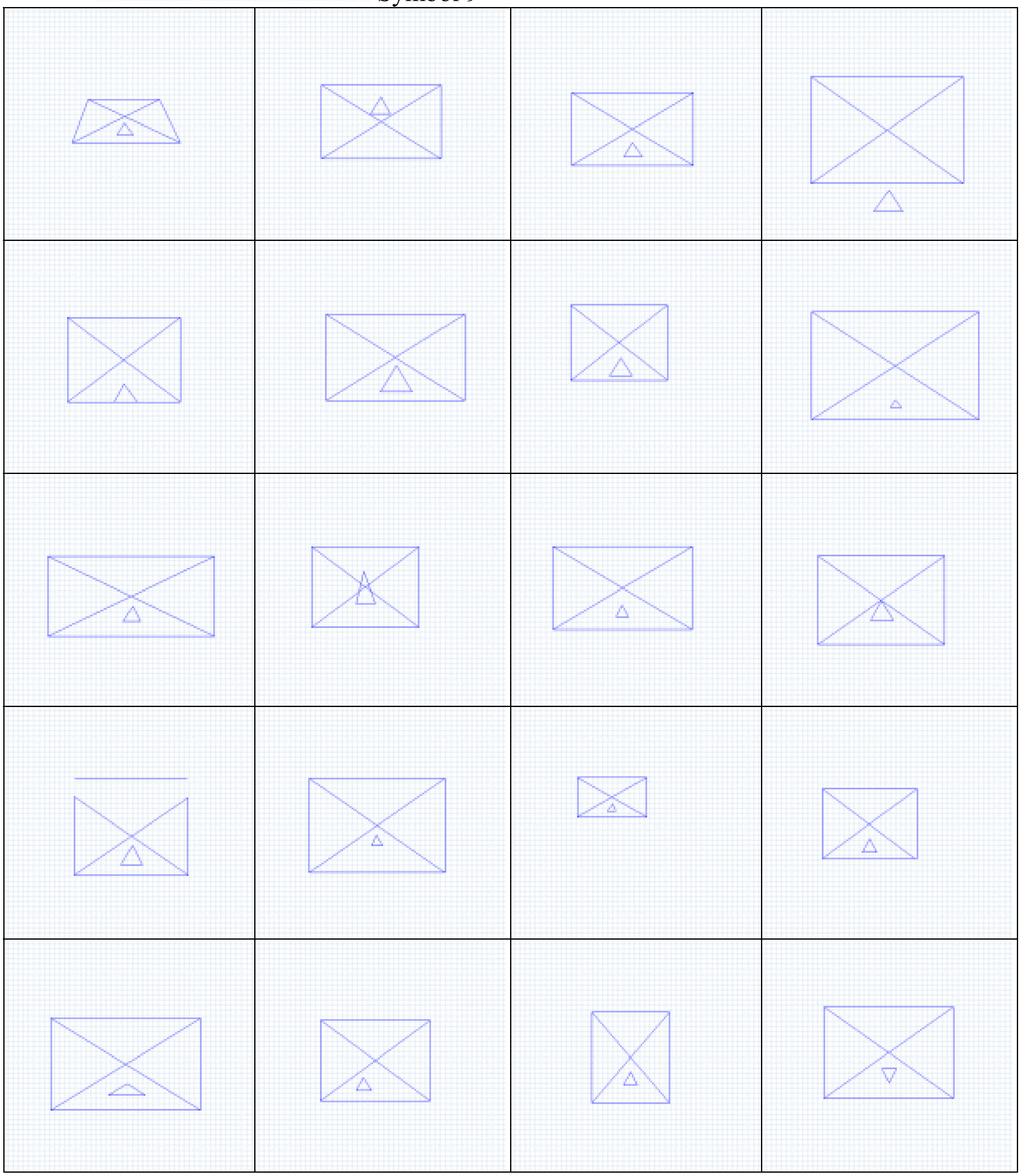
Symbol 8



System:	NO	NO	NO	NO
Majority:	NO	NO	NO	YES
Majority %:	67%	56%	92%	85%
System:	NO	NO	YES	YES
Majority:	NO	YES	YES	YES
Majority %:	56%	90%	59%	95%
System:	NO	YES	YES	YES
Majority:	YES	YES	YES	YES
Majority %:	64%	92%	85%	92%
System:	YES	NO	YES	YES
Majority:	YES	YES	YES	YES
Majority %:	87%	82%	95%	92%
System:	NO	YES	NO	YES
Majority:	YES	NO	NO	YES
Majority %:	74%	54%	79%	82%



Symbol 9



System:	NO	NO	YES	NO
Majority:	NO	NO	YES	NO
Majority %:	73%	100%	97%	97%
System:	NO	YES	YES	YES
Majority:	NO	YES	YES	YES
Majority %:	95%	86%	92%	62%
System:	YES	NO	YES	YES
Majority:	YES	NO	YES	NO
Majority %:	95%	97%	89%	81%
System:	NO	YES	YES	YES
Majority:	NO	YES	YES	YES
Majority %:	95%	86%	95%	95%
System:	NO	NO	NO	NO
Majority:	YES	NO	YES	NO
Majority %:	84%	70%	70%	97%