

SYMMETRY ANALYSIS OF TWO-DIMENSIONAL  
PATTERNS FOR COMPUTER VISION

Technical Note 186

June 1979

By: Robert C. Bolles, Computer Scientist  
Artificial Intelligence Center

---

SRI Project 8487

To be published in the Sixth International Joint  
Conference on Artificial Intelligence, Tokyo, Japan.  
20 - 23 August 1979.

SYMMETRY ANALYSIS OF TWO-DIMENSIONAL PATTERNS  
FOR COMPUTER VISION\*

Robert C. Bolles  
SRI International,  
333 Ravenswood Avenue,  
Menlo Park, California 94025

A system is described that automatically determines the rotational and mirror symmetries of two-dimensional patterns. The properties of patterns that determine their symmetries are delineated, a representation scheme based on these properties is defined, and algorithms to perform the symmetry analysis are presented. An implementation based on the SRI vision module is described.

1. INTRODUCTION

Two fundamental properties of a two-dimensional pattern are its rotational and mirror symmetries. These properties are important in the recognition and location of patterns, because they can be used to locate key features that determine the orientations of the patterns and differentiate them from those that are similar. These properties are especially important in performing industrial vision tasks, in which symmetrical, or almost symmetrical, parts are common. For example, Figure 1 is a part to be picked up and added to a subassembly. It is two-fold rotationally symmetric (i.e., its appearance is unchanged by a rotation of 180 degrees). If it is occasionally presented upside down, it would be important to know that it is not mirror-symmetric (the relative orientation of the holes with respect to a line joining their centers is different in the mirror image). Since the pattern is not mirror-symmetric, a vision system could detect upside-down parts and instruct the manipulator to turn them over before adding them to the subassemblies.

Current industrial vision systems, like SRI's vision module [5] and Bausch and Lomb's OMNICON [2], recognize patterns on the basis of such global features as the area and perimeter of a pattern, because statistical pattern recognition techniques can easily model such features. Since these simple vision systems do not use local features, such as holes and corners, or their symmetries, they cannot automatically determine the orientation of the pattern in Figure 1 or distinguish it from its mirror image.

A few artificial intelligence programs have performed symmetry analysis. Evans' program, which worked geometric-analogy problems, tested the primitive figures to see whether they were mirror-symmetric about a horizontal or vertical axis [3]. Gips' program analyzed two 3-dimensional strings of cubes to determine whether they were rotational or mirror

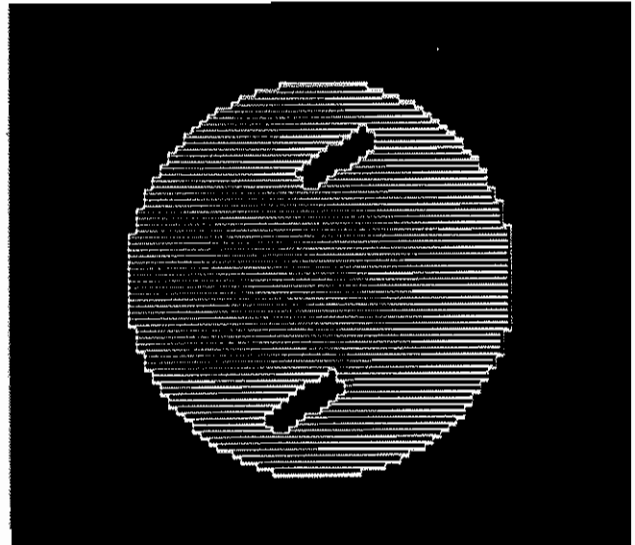


FIGURE 1 PICTURE OF A SAMPLE PART

transformations of each other [4]. Perkins' program [7] used a form of correlation to determine whether a pattern was rotationally symmetric. That program, like its counterparts, was not intended to perform a general-purpose symmetry analysis aimed at understanding local features, structures of local features, and their properties. It is precisely this kind of comprehensive understanding that is the purpose of the research reported in this paper.

In a recent paper Wechsler [8] describes an algorithm that decomposes two-dimensional patterns into mirror-symmetric components. His approach employs the same mathematical descriptions and tests for symmetry as are referred to in this paper, but his goal is to describe regions rather than characterize structures of local features.

\* This work was supported by NSF under grant APR75-13074.

## 2. REPRESENTATION SCHEME

We shall now define a representation scheme based on a set of structural primitives for patterns. The intent is to identify important constructs, their properties and relationships, and to define a convenient way to describe patterns.

Pattern symmetries are determined by the properties of groups of "similar" features. Features are defined as similar if (1) they have the same description (i.e., type, size, and shape), (2) they are equidistant from the pattern's center of gravity, and (3) they have the same orientation with respect to a line from the pattern's center of gravity to their own. Therefore, if two features are similar, the pattern can be rotated about its center of gravity so that one feature is repositioned at the other feature's original position and orientation. For example, the two holes in Figure 1 are similar.

Groups of similar features are important because they form structures of features that limit the potential number of rotational symmetries for the pattern. In particular, if a pattern is M-fold rotationally symmetric and it contains a structure of features that is N-fold rotationally symmetric, M must be a divisor of N.

The representation scheme describes patterns in terms of these structures of features. Since the features in a structure are similar, the structures can be represented by one example of the feature and a list of angular positions. Individual features can either be simple, such as a corner, or complex, such as the entire pattern in Figure 1. Complex features are described in terms of structures of other features. Therefore, the representation scheme describes patterns as trees in which the nodes represent individual features or structures of features. Figure 2 shows the two types of nodes. The "descriptions" of individual features correspond to the global features used by simple vision systems.

FEATURE	STRUCTURE OF FEATURES
POSITION	POSITION
ORIENTATION	ORIENTATION
ROTATIONAL SYMMETRY	ROTATIONAL SYMMETRY
DESCRIPTION: TYPE	POINTER TO SAMPLE FEATURE
SIZE	NUMBER OF OCCURRENCES
SHAPE	POINTER TO LIST OF ANGULAR POSITIONS
POINTER TO LIST OF STRUCTURES	

FIGURE 2 NODES IN THE REPRESENTATION

Every node in a tree that represents a pattern has a position, orientation, and rotational symmetry. The "leaf" nodes, which correspond to simple (or primitive) features, have inherent values for these properties. The values for the other nodes are functions of their inherent values and the values of the nodes directly beneath them. An important part of the representation scheme is the set of conventions that specify these functional relationships. Unfortunately, these conventions cannot be described for lack of space, but may be found in [6].

## 3. SYMMETRY ANALYSIS

In this section we briefly describe algorithm for performing the symmetry analyses.

### 3.1 Rotational Symmetry Analysis

Given a pattern, the rotational symmetry analysis builds a tree to describe it and, in the process, determines its rotational symmetry. The algorithm is as follows:

```

RotSymm(PATTERN) =
  Create a new feature node for the PATTERN
  Fill in the description of the PATTERN
  Set the node's list or structures to empty
  If the PATTERN is complex then
    For each subpattern, call RotSymm(subpat)
    Group together similar subpatterns
    For each group
      Create a new structure node
      Add it to the list of structures
      Select a sample feature for the struct.
      Fill in the number of occurrences and
      angular positions of the features
      Delete the unused feature nodes
      Compute the position, symmetry, and
      orientation of the structure
    Compute position, symmetry, and orientation
    of the PATTERN
  Return a pointer to the PATTERN's node
  
```

The rotational symmetry of the pattern is the symmetry of the topmost feature node in the tree description.

### 3.2 Mirror Symmetry Analysis

Given a tree description of a pattern, the mirror symmetry analysis builds a tree to describe the mirror image of the pattern and, if the two trees are similar, declares the pattern to be mirror-symmetric.

Two short recursive procedures can build the mirror image tree. One produces the mirror image of a feature node, the other produces a mirror image of a structure-of-features node. Since the similarity of two patterns is independent of their orientation, the mirror image of a pattern can be formed by reflecting it about any axis, not necessarily an axis of mirror symmetry. Therefore, it is not necessary to locate an axis of mirror symmetry to determine whether or not a pattern is mirror-symmetric. The mirror image of a feature node can be formed by reflecting its position and orientation about the X-axis and replacing each structure in the list of substructures by its mirror image. The mirror image of a structure can be formed by reflecting its position and orientation about the X-axis, replacing the sample feature by its mirror image and reflecting all the angular positions in the list of occurrences about the X-axis.

The mirror symmetry analysis could be performed without constructing the mirror image tree, but the algorithm would be more complicated.

#### 4. EXPERIMENTAL SYSTEM

The experimental system is an extension of the vision system developed by Agin at SRI International [1], which performs a connectivity analysis of binary pictures and produces a tree description of the "blobs" in the picture. (A blob is a connected region of uniform color.) The symmetry analysis converts blob descriptions into the symmetry representation and performs the mirror symmetry analysis. The features it uses are holes, elongations (i.e., axes with the minimum second moments), and minimum and maximum radii. The analysis automatically locates these features, groups them into structures of features, and determines their symmetries. Sample results are shown in Figure 3. The pattern in Figure 3a is not rotationally symmetric, but it is mirror-symmetric. The pattern in Figure 3b is three-fold rotationally symmetric, but not mirror-symmetric. The pattern in Figure 3c is two-fold rotationally symmetric, but not mirror-symmetric.

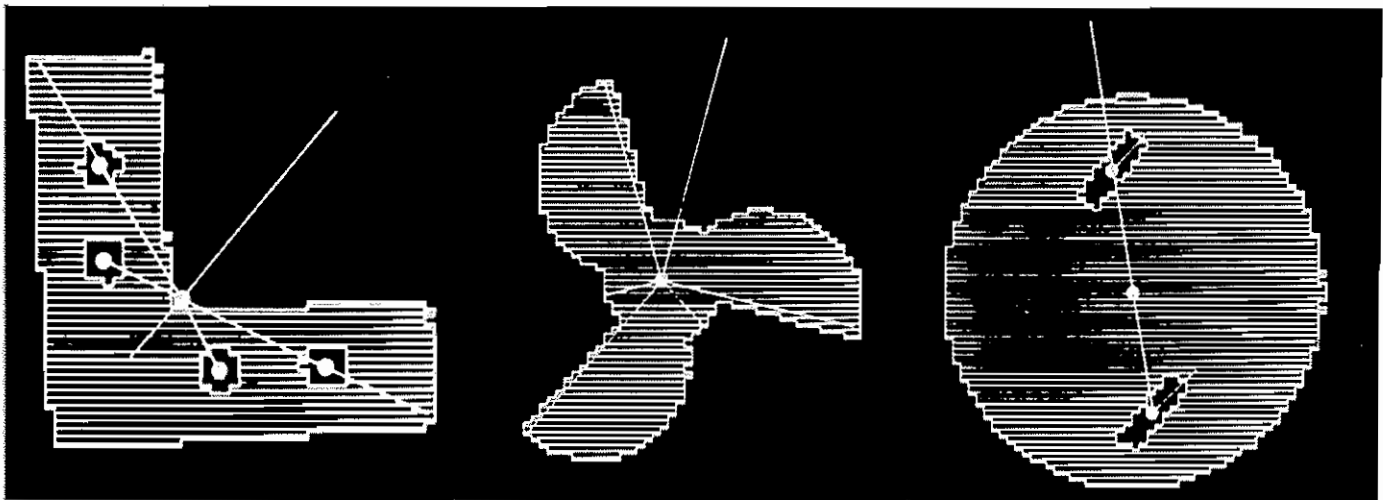
Since the algorithms analyze digital images, thresholds are required for making some of the decisions. For example, how close to perfect symmetry does a set of features have to be for the program to call it symmetric? The program automatically establishes these thresholds on the basis of the pattern being analyzed and a model of the expected pixel errors. These thresholds, however, can be interactively overridden.

#### 5. DISCUSSION

One of the basic goals of this research is to simplify the process of producing programs for recognizing and locating patterns. Ultimately this means the development of a system that can be trained by showing it examples. Such a system would determine the symmetries of a pattern, locate key features, choose the most cost/effective features to use at run time, and produce a program to perform the task. In contrast thereto, the current program is limited to determining the symmetries and locating some key features.

#### REFERENCES

- [1] G. J. Agin and R. Duda, "SRI Vision Research for Advanced Automation," Second USA-Japan Computer Conference, Tokyo, Japan (August 1975).
- [2] "Bausch and Lomb Omnicon Pattern Analysis System," Analytic Systems Division Brochure, 820 Linden Avenue, Rochester, New York, (1976).
- [3] T. G. Evans, "A Program for the Solution of a Class of Geometric-Analogy Intelligence-Test Questions," Semantic Information Processing, pp. 271-353, (MIT Press, Cambridge, Massachusetts 1968).
- [4] J. Gips, "A Syntax-Directed Program that Performs a Three-dimensional Perceptual Task," Pattern Recognition, Vol 6, pp. 189-199 (1974).
- [5] G. Gleason and G. J. Agin, "A Modular Vision System for Sensor-Controlled Manipulation and Inspection," 9th Intl. Symposium on Industrial Robots, Washington, D.C., pp. 57-70 (March 1979).
- [6] D. Nitzan, G. Agin, R. Bolles, G. Gleason, J. Hill, D. McGhie, R. Prajoux, A. Sword, and W. Park, "Machine Intelligence Research Applied to Industrial Automation," SRI Report to NSF, SRI Intl., Menlo Park, California (August 1979).
- [7] W. A. Perkins, "A Model-Based Vision System for Industrial Parts," IEEE Trans. on Computers, Vol. C-27, pp. 126-143 (February 1978).
- [8] H. Wechsler, "A Structural Approach to Shape Analysis Using Mirroring Axes," Computer Graphics and Image Processing, Vol. 9, No. 3, pp. 246-266 (March 1979).



(A)

(B)

(C)

FIGURE 3 RESULTS OF THE SYMMETRY ANALYSIS