Transformational Form Perception in 3D: Constraints, Algorithms, Implementation

Dana H. Ballard
Computer Science Department
University *of* Rochester

Hiromi Tanaka
Department of Control Engineering
Osaka University

Abstract

Most of the geometric information in a scene is captured by local coordinate frames oriented according to local geometric features. In polyhedral worlds these features are faces, vertices and edges. Such features lead naturally to parallel algorithms for building such a scene description from stereo input that are insensitive to noise and occlusion. This representation can be used for object location, object recognition, and navigation.

1. Introduction

The critical problem of form perception is that of picking the right representation. In previous papers we have argued that if *frame primitives* are picked as the underlying geometric representation, then many problems related to form perception can be solved in an elegant way (2,3,4,5,6,7,23 |. Frame primitives are geometric coordinate frames that can be extracted from more primitive image features. These primitives play a dual role: thev can be- Regarded as features in then own right and used in the form matching process directly, or they can be used to specify transformations between themselves and other features.

In this paper, frame primitives are developed with respect to a polyhedral model of the geometric environment. The advantage of a polyhedral model is that the polyhedral primitives arc intimately related to the frame primitives. Howevei. any substrate related to coordinate frames such as symmetries |8| may be used as well.

frame primitives express the fundamental nature of rigidity: two shapes are equivalent if there exists a rigid transformation that maps one into the other. This idea can also be extended to the matching of a prototype with portions of a scene. A portion of a scene is said to represent an -instance of a prototype it there exists a rigid transformation mapping the prototype into portions of the scene. 1 he- use of rigidity distinguishes the approach from topoplogical matching I1.12324.

The problem of matching a 3d prototype to an image can be hierarchically organized into: (1) the recovery of 3d lines from stereo image data (for monocular approaches, see |?I.II|); (2) the construction of a 3d polyhedral scene model: and (3) the matching of portions of that model to a library of stored prototypes. This hierarchical strategy is similar to that of [9| and has several advantages over the methods that trv to match the image to the 3d prototype in one step for example. [19.20] try to match the 3d prototype directly with the 2d line drawing.

The computation is implemented in a connectionist architecture, motivated by biological information processing systems (2|. The complete processes of extracting 3d structure and matching is carried out by a parallel probabilistic relaxation algorithm

2. Basic Concepts

2.1. Geometrical Primitives

A *3d plane* is defined by $aX + bY + cZ + d = 0$ where $\mathbf{n} = (a, b, c)$ is a unit normal to the plane surface and d is the distance of closest approach to the origin.

A *3d line* is defined by the equation $\mathbf{x} = \mathbf{D} + s\mathbf{e}$ where $\mathbf{D}$ is the closest point to the origin that the line passes through, $\mathbf{e}$ is the unit vector direction of the line, and $s \geq 0$ is a scalar parameter

Each of the above can be seen as the *partial specification of local coordinate frames* where the frame parameters are synonymous with geometric features. In other words, these geometrical entities provide *natural* choices for the orientation of local geometric features.

2.2. Frame Descriptions

A coordinate frame is defined by a transformation with respect to a *base coordinate frame* which has vectors $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) = ((1,0,0)(0,1,0)(0,0,1))$. This transformation is specified by scale factor s, a rotation of the frame by an angle θ about a unit vector $\mathbf{w}$ with respect to a base coordinate frame, and a translation of the origin by $\mathbf{x}_0$. Notationally we will refer to the complete frame as F the rotation part as E. Thus $F = (E, \mathbf{x}_0)$, where $E = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ is an orthonormal basis.

A virtue of the geometric primitives is that the transformation itself allows one to change the point of view. Thus to make an arbitrary frame $F = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{x}_0)$ the base frame, one need only apply the transformation in reverse to all the frame primitives. That is, $F^{-1} = (\mathbf{w}, -\theta, -\mathbf{x}_0)$, where $\mathbf{w}$ and θ are determined from $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ (we show how this is done in Section 3.3).

This extremely important point is the main virtue of frame primitives: *frame primitives contain all the information necessary to change the point of view.* Any particular primitive may be chosen as a reference for all the other primitives by applying the reference transformation to each primitive.

2.3. Matching Different Frame Descriptions

An instance of an object in the viewer centered frame may be related to a prototypical internal representation in an object-centered frame bv a *viewers: transforminations.* but this problem is generally underdetermined [5]. Furthermore, the image usuallv contains many features that belong to different objects, and these lend to confound the perception of a particular shape,

A key simplifying assumption is that the internal representation contains only a single object. In this case the viewing transformation can generally be computed and parts of the object in the image can be identified despite other image clutter |3,6|. The task of determining if a known object is in an image is posed as: is there a transformation of a subset of image features such that the transformed subset can be explained as the object? If the answer to this question is no, then the object is not present. If yes. then the transformation provides all the necessary information about the object.

The principal feature of our method is that it decouples the computations for orientation and translation. In other words, the orientation of the object can be detected without knowing its translation.

3. Specification of Constraints

3.1. Matching Two-Dimensional Stereo Lines

A line in the image plane defines a plane which passes through the focal point ($Z = 0$) as well as the line itself (this development is similar to that of [12]). Consider the intersection of this plane with the image plane. At the intersection locus: $X = x$, $Y = y$, and $Z = 1$ so that the equation of the line in the image plane is

$$ax + by + c = 0$$

The different imaging geometries of stereo images will produce two different lines, and correspondingly, two different planes. In order to relate these two planes, they must be expressed in the same coordinate frame. For the simple case of parallel views separated by Δ the equation of the second plane (a_2, b_2, c_2) in terms of the first frame is simply

$$a_2X + b_2Y + c_2Z - a_2\Delta = 0$$

The two planes, when intersected, define a 3d line $(\mathbf{D}, \mathbf{e})$. Given the two normals $\mathbf{n}_1$ and $\mathbf{n}_2$, $\mathbf{e}$ can be readily calculated as

$$\mathbf{e} = \langle \mathbf{n}_1 \times \mathbf{n}_2 \rangle \tag{3.1}$$

where the operator $\langle\ \rangle$ normalizes its vector argument. The vector from the origin $\mathbf{D}$ may be calculated by solving

$$\mathbf{D} = N^{-1}\mathbf{d} \tag{3.2}$$

where

$$N = \begin{bmatrix} n_{1x} & n_{1y} & n_{1z} \\ n_{2x} & n_{2y} & n_{2z} \\ e_x & e_y & e_z \end{bmatrix} \quad \text{and} \quad \mathbf{d}^T = (0, d_2, 0)$$

3.2. Building Polyhedral Scene Descriptions

Given a collection of 3d lines, the problem now becomes one of recovering additional 3d structure. To do this one can exploit an instance of Barlow's notion of "suspicious coincidences." Applied to a polyhedral world, this idea is as follows. *It is unlikely that two three-dimensional lines will happen to be coplanar or meet at a vertex accidentally.* Thus we consider constraints between all pairs of 3d lines, assuming that intersections usually reflect rigid polyhedral structure. The constraints are simple applications of vector geometry of R^3 [7,23] and are as follows.

I. Two planes define a concave or convex edge. Pairs of planes $P_1 = (\mathbf{n}_1, d_1)$ and $P_2 = (\mathbf{n}_2, d_2)$ produce edges which have an associated coordinate frame $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ where

$$(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) = (\langle \mathbf{n}_1 \times \mathbf{n}_2 \rangle,\ \mathbf{e}_3 \times \mathbf{e}_2,\ \langle \mathbf{n}_1 + \mathbf{n}_2 \rangle) \tag{3.3}$$

The distance of closest approach of the edge to the origin, $\mathbf{D}$, can be calculated from the following three equations.

$$\mathbf{n}_1 \cdot \mathbf{D} = d_1,\ \mathbf{n}_2 \cdot \mathbf{D} = d_2,\ \mathbf{e}_1 \cdot \mathbf{D} = 0$$

II. Two edges can determine a plane. Coplanar 3d lines with orientations $\mathbf{e}_{11}$ and $\mathbf{e}_{12}$ must lie in a plane $(\mathbf{n}, d)$, where

$$\mathbf{n} = \mathbf{e}_{11} \times \mathbf{e}_{12} \tag{3.4}$$

and d is specified by

$$\mathbf{n} \cdot \mathbf{D}_1 = \mathbf{n} \cdot \mathbf{D}_2 = d \tag{3.5}$$

III. Two edges can determine a vertex. If two coplanar 3d lines intersect, then the intersection points can be determined by solving two equations from

$$\mathbf{D}_2 - \mathbf{D}_1 = \alpha\mathbf{e}_{12} - \beta\mathbf{e}_{11} \tag{3.6}$$

IV. Two planes and a 3d line can define an edge. A 3d line only has a direction $\mathbf{e}$, but additional orientation information can be computed in the case where the line is also the intersection locus of two planes. In this case a frame can be specified that has the orientation given by (3.3) where $\mathbf{n}_1$ and $\mathbf{n}_2$ are the normals of the two planes.

3.3. Matching a Scene Frame with a Prototype

I. Rotational Constraint. To develop the rotational constraint, we show how given two orientation frames F_p and F_s from the prototype and scene, we can compute the rotation $\boldsymbol{\theta}$ about a unit vector $\mathbf{w}$. The mathematics of quaternions, e.g., [25], states that the rotation of one unit vector $\mathbf{u}$ into another unit vector, $\mathbf{v}$, around a unit vector axis, $\mathbf{w}$, is given by:

$$R = \sqrt{-(\mathbf{w} \times \mathbf{v})(\mathbf{w} \times \mathbf{u})} \tag{3.7}$$

We will use the orientation vectors $\mathbf{e}_1$ from both the model and image as $\mathbf{u}$ and $\mathbf{v}$, respectively. How do we define $\mathbf{w}$? Since $\mathbf{w}$ must be perpendicular to both $\mathbf{u}$ and $\mathbf{v}$, it can be defined in terms of the two orientation frames as

$$\mathbf{w} = (\mathbf{e}_{1p} - \mathbf{e}_{1s}) \times (\mathbf{e}_{2p} - \mathbf{e}_{2s}) \tag{3.8}$$

In the special case of where $\mathbf{e}_{1p}$ equals $\mathbf{e}_{1s}$ and $\mathbf{e}_{2p}$ equals $\mathbf{e}_{2s}$, we arbitrarily set $\mathbf{w} = (1, 0, 0)$ and $\boldsymbol{\theta} = 0$. This procedure and (3.7) specify the axis of rotation.

The next step is to compute $\boldsymbol{\theta}$ for the general case. The rotation from $\mathbf{u}$ to $\mathbf{v}$ is also given by

$$R = \cos(\boldsymbol{\theta}/2) + \sin(\boldsymbol{\theta}/2)\mathbf{w} \tag{3.9}$$

where $\boldsymbol{\theta}$ is the angle of rotation around the unit vector axis $\mathbf{w}$. Thus the angle $\boldsymbol{\theta}$ can be computed from (3.7) and (3.9).

II. Rotating the Model. Once the rotation between the prototype and scene has been established, then the origins of the prototype frames can be appropriately rotated by (for details, see [25]):

$$\mathbf{x}'_p = R\,\mathbf{x}_p R^{-1} \tag{3.10}$$

This results in a new set of origins that only differ from their counterparts in the scene by a translation vector.

III. Translational Constraint. The translational constraint is trivially computed if the correspondence between a scene frame origin $\mathbf{x}_s$ and a rotated prototype frame origin $\mathbf{x}'_p$ is known. The answer is the difference vector

$$\Delta\mathbf{x} = \mathbf{x}'_p - \mathbf{x}_s \tag{3.11}$$

4. Algorithms

4.1. Energy Minimization

The parallel algorithm uses a form of relaxation developed for binary threshold units by [15,14]. Each unit has a state, s, which is either on ($s = 1$) or off ($s = 0$). In the case of ternary constraints, a unit k will receive a weighted input from pairs of other value cells i and j in an amount given by $w_{ijk}s_is_j$. Each unit turns on according to the following computation:

1) Compute the input

$$p_k = \sum_{i,j \text{ connected to } k} w_{ijk}s_is_j$$

2) Substract a threshold

$$p_k = p_k - \theta_k$$

3) Turn s_k on if $p_k > 0$, else turn it off.

Units are turned on and off acording to steps 1 through 3 until convergence is achieved. Formally, if all the weights w_{ijk} are symmetrical ($w_{ijk} = w_{jik} = w_{kij}$) then the converged set of states will minimize the energy function $E = -\Sigma\, w_{ijk}\, s_i\, s_j\, s_k$. In practice, the examples that we have tried are well conditioned, so that convergence is achieved in one iteration.

If the algorithm takes one iteration to converge, it is roughly equivalent to *correlation*. If more than one iteration is required, i.e., a sequence of states s^k converges to a fixed point s^* then the algorithm specified by steps 1-3 is equivalent to *gradient search*. Gradient search will work if the constraints have a single minimum. The case where E has more than one local minimum can be handled by *simulated annealing* [13,14,10].

4.2. Value Cells

The constraints are represented using special connectionist architecture based on *value cells*. In the value cell approach, a particular vector variable is represented as a discrete collection of cells where the central location of the cell is a particular value for the variable and the width of the cell is its accuracy. For example, representing edges in an image requires a three-dimensional cell type with location (x_0, y_0, θ_0) and width $(\Delta x, \Delta y, \Delta\theta)$. Collections of these cells cover the range of all possible edge positions and orientations.

Since value cells are described in terms of threshold units, they can be either *off* or *on*. An on value signifies the presence of an edge at the associated position and orientation. Additional accuracy and the avoidance of certain computational problems can be achieved through the use of overlapping cells. The technical details of overlapping cells strategies are discussed in a separate paper [22]. Value cells representing different variables are related via constraints. In the familiar example of line detection an edge cell (x_0, y_0, θ_0) is related to the line cell (ρ_0, θ_0) through the constraint $x_0\cos\theta_0 + y_0\sin\theta_0 = \rho_0$.

The previous line constraint was binary in that a value cell for a specific edge value was connected to a cell for a specific line. A special kind of constraint is a *ternary constraint* in which three value cells are related. A ternary constraint can improve the accuracy of line detection by using two edges (x_1, y_1, θ_1) and (x_2, y_2, θ_2), that are on the same line iff $\theta_1 = \theta_2 = \arctan(y_2 - y_1, x_2 - x_1)$. If this constraint is satisfied, then the (ρ, θ) cell is determined by

$$\theta = \arctan(y_2 - y_1, x_2 - x_1) \qquad (4.1)$$

$$\rho = x_1\cos\theta + y_1\sin\theta$$

4.3. The Overall Network

Representative value cells in the overall network are shown in the following figure. Part A) denotes the prototype frame primitives. B) shows the cells related to the view transformation. ( ) shows the polyhedral network, and 1>) shows the stereo network. Ternary constraints are denoted with an arc connecting pairs of inputs. In each parameter network, only representative cells are shown.

5. Implementation

5.1. Data Structures

The overall algorithm is conceptually simple; asynchronously, each cell evaluates its input and turns on or off. When the entire network converges, the on units represent the solution to the particular problem. This simple description can lead to inefficient implementations, since most of the cells are off in anv given instant Thus when a processor checks its input pairs (in the case of a ternary constraint), most of the cells will be off. To take advantage of this, we use pairs of *on* units to calculate incremental inputs, and when all such pairs have been considered for any network, subtract thresholds and determine whether to turn the units on or off. This strategy is repeated for all the ternary constraints in the network The only differences are: (1) the different constraints that relate different value cells: and (2) the set of" on units at any given instant. The above strategy requires a data structure that only records on cells.

We use hash tables with collision resolution via chaining, e.g., (16, pp. 462-469).

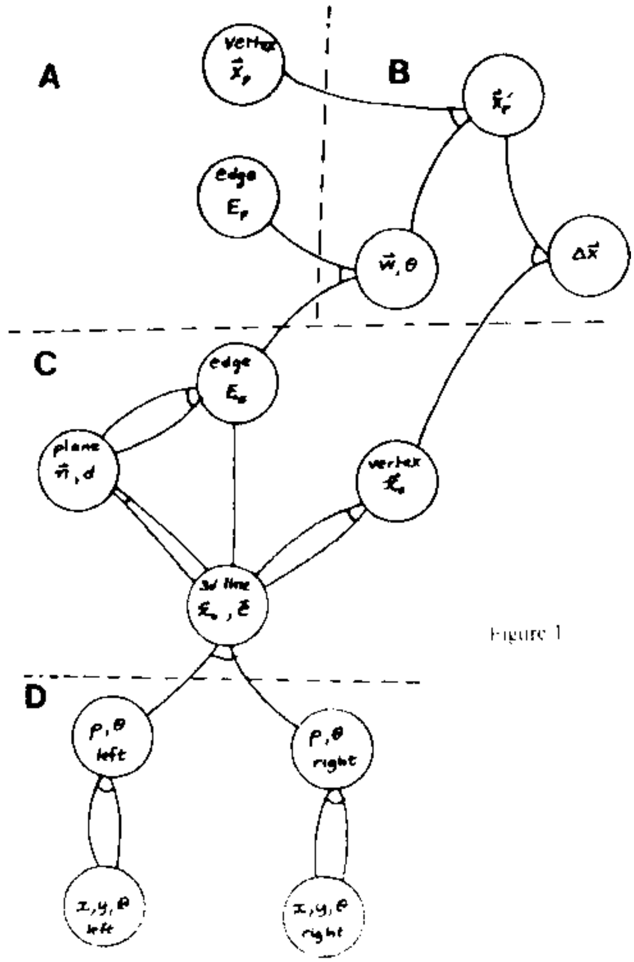

Figure 1

5.2. Generic Relaxation Structures

For each ternary constraint, two tables must be indexed to compute the index of a third. Then the input is added to the appropriate cell, as follows.

```
Foreach u in T1 do
  Foreach v in T2 do
    {
      w := rule (u, v, T1, T2)
      Increment (w, T3)
    }
```

The function of *Increment* is to add the appropriate weight to the p field of the entry in the Linetable. If there is no previous entry, an appropriate cell is added. Next the entries are thresholded, and if below threshold, deleted from the table:

```
Foreach u in T do
   if u < threshold then
      delete (u);
```

Given the generic format, we can specify the network of computations by specifying collections of three tables, each group having an associated update rule. The following summary table represents this information

T_1	T_2	T_3	rule
edge	edge	line	Eq. 4.1
line	line	3d line	Eqs. 3.1, 3.2
3d line	3d line	plane	Eqs. 3.4, 3.5
3d line	3d line	vertex	Eq. 3.6
plane	plane	possible edge	Eq. 3.3
possible edge	3d line	edge	Eq. 3.3, Sec. 2B
scene edge	prototype edge	orientation	Eqs. 3.8, 3.9
orientation	model vertex	rotated vertex	Eq. 3.10
rotated vertex	image vertex	translation	Eq. 3.11

5.3. Implementation

The status of the current implementation is that the constraints lor matching an object in (wo coordinate frames ha\e been shown to work by |23|. The stereo portion of the constraints are currently being implemented, Figure 2 shows the result for a simulated three-dimensional wrench. In these first tests, the data for the wrench is rotated and translated to obtain a scene copy and there is no sell occlusion.

This is then matched against the original using the constraints described in Section 3.3 implemented as described in Sections 4 and S. the figure shows: A) the wrench, B) \alues for the direction of rotation(magmtude not shown) and C) values for the direction of tratislation(magnitude not shown). The multiple values are die result of false pairings between scene frame primitive and prototype. Although the grey scale does not emphasize this, the correct transformation is found easily in this case.

6. High-Level Control

The prototype frames form a generic basis set. In order to represent a particular object, an appropriate subset of value units must be turned on. One way to do this is to represent objects as specific links between an object token space and the prototype frame space. This arrangement forms a basic architecture that can be used in several different ways.

I. Object localion. If a particular object is sought, its prototype frame description is turned on by activating its ob|cct token. This (urns on the appropriate frame primitives. Then if a match between the object and a subset of the scene exists, a rotation and a translation unit will be turned on in the transformation network.

II. Object Recognition. If an object has been segmented (by other methods, e.g.. range, color etc.) and its identity is sought, the prototype frame can be loaded with several candidate objects. If the matching process can build a transformation between any of these objects and the segmented object, appropriate rotation and translation units will be turned on.

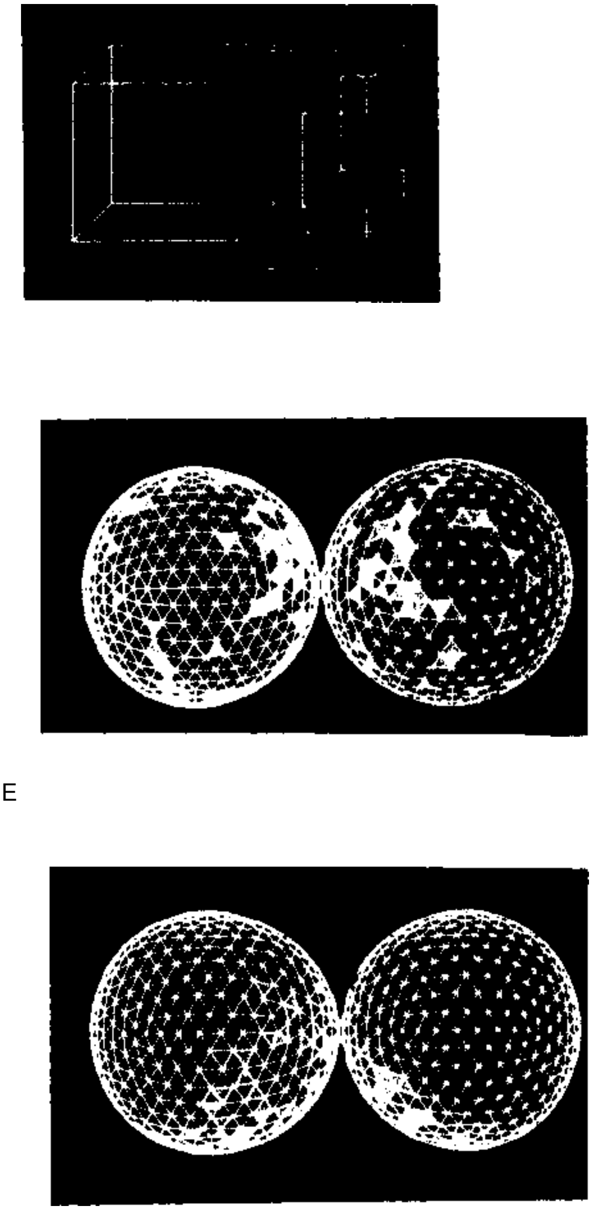

Figure 2.

III. Navigations. *This* architecture can also be used for navigation in the following way. At some initial ume to the current scene is loaded into the prototype frame by activating the identity units in the transformation space. This has the effect of turning on units in the prototype frame that are a copy of the scene units at that instant. Henceforth, as the observer moves around, these units are locked on. The result is that at any instant the transformation units will reflect the transformation between the current scene and that at t_0. The inverse of this transform corresponds to the observer motion.

7. Conclusions

In this paper we have developed a number of interdependent ideas. The main point is that geometric frames provide a natural way of talking about shape. This idea has been present in psychology and differential geometry for a long time. Our contribution has been to develop the particular constraints for polyhedra and show how they lead naturally to algorithms for extracting frame information from the scene and matching it against stored prototypes.

The particular focus of the paper was on the extraction of linear features from edge data and the matching of these features to obtain three-dimensional information.

It is extremely important to note that these algorithms all use relaxation as the computing engine and value cells as the representation. The fact that all these problems can be handled in the same way argues for the generality of the approach.

8. Acknowledgements

Shmuel Tomer, Matt Curtis and Owen Kimball provided invaluable programming support in developing the geodesic package and testing two-dimensional versions of the algorithms. Peggy Meeker prepared the many drafts of this manuscript. This research was supported by NSF under Grant MCS 8203290.

9. References

1. Attneave, F., "Some informational aspects of perception," *Psychological Review 61*, 1954.

2. Ballard, D.H., "Cortical connections: Structure and function," TR 133, Computer Science Dept., U. Rochester, July 1984b; to appear, *Brain and Behavioral Sciences*.

3. Ballard, D.H., "Generalizing the Hough transform to detect arbitrary shapes," TR 55, Computer Science Dept., U. Rochester, 1979; *Pattern Recognition 13*, 2, April 1981.

4. Ballard, D.H., "Parameter networks: Towards a theory of low-level vision," TR 75, Computer Science Dept., U. Rochester, 1981; *Proc., 7th IJCAI*, Vancouver, August 1981; *Artificial Intelligence 22*, 235-267, 1984a.

5. Ballard, D.H. and P.M. Hrechanyk, "A connectionist model for shape perception," Computer Vision Workshop, Ringe, NH, August 1982; also appeared as "Viewframes: A connectionist model of form perception," *DARPA Image Understanding Workshop*, Washington, D.C., June 1983.

6. Ballard, D.H. and D. Sabbah, "On shapes," *Proc., 7th IJCAI*, Vancouver, Canada, August 1981; also appeared as "Viewer independent shape recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence 5*, 6, 653-660, November 1983.

7. Ballard, D.H., A. Bandyopadhyay, J. Sullins, and H. Tanaka, "A connectionist polyhedral model of extrapersonal space," *Proc., IEEE Conference on Computer Vision*, Annapolis, MD, 1984.

8. Brady, M. and A. Haruo, "Smoothed local symmetries and their applications," *Int. Journal of Robotics Research 3*, 3, 1984.

9. Faugeras, O.D., "Representation and matching techniques for range data," Rank Price Funds Symposium on Representation and Control in Visual Processing, Malvern, England, April 1985.

10. Geman, S. and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," TR, Brown U., September 1983; also appeared in *IEEE Trans. PAMI*, January 1985.

11. Kanade, T., "Recovery of the three-dimensional shape of an object from a single view," *Artificial Intelligence 17*, 409-460, 1981.

12. Kanade, T., presentation at the University of Rochester, November 1984.

13. Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," *Science 220*, 671-680, 1983.

14. Hinton, G.E. and T.J. Sejnowski, "Optimal perceptual inference," *Proc., IEEE Computer and Pattern Recognition Conference*, 448-453, June 1983.

15. Hopfield, J.J., "Neural networks and physical systems with collective computational abilities," *Proc., Nat'l Acad. Sciences USA 79*, 2554-2558, 1982.

16. Horwitz, E. and S. Sahni, *Fundamentals of Data Structures*. Computer Science Press, 1976.

17. Mackworth, A.K., "Interpreting pictures of polyhedral scenes," *Artificial Intelligence 4*, 2, 121-137, June 1973.

18. Sabbah, D., "Computing with connections in visual recognition of origami objects," to appear, *Cognitive Science*, Special Issue on Connectionism, Winter 1985.

19. Silberberg, T.M., D. Harwood, and L.S. Davis, "Object recognition using oriented model points," CAR-TR-56 and CS-TR-1387, Center for Automation Research, U. Maryland, April 1984.

20. Stockman, G.C. and J.C. Esteva, "3D object pose via cluster space stereo," TR 84-05, Computer Science Dept., Michigan State U., 1984.

21. Sugihara, K., "An algebraic approach to shape-from-image problems," *Artificial Intelligence 23*, 1, May 1984.

22. Sullins, J., "Coarse coding in connectionist networks," forthcoming TR, Computer Science Dept., U. Rochester, 1985.

23. Tanaka, H., D.H. Ballard, M. Curtiss, and S. Tsuji, "Parallel