Optimization in Model Matching and Perceptual Organization: A First Look

Eric Mjolsness, Gene Gindi, and P. Anandan

Research Report YALEU/DCS/RR-634
June 1988

Optimization in Model Matching and Perceptual Organization: A First Look

Eric Mjolsness[†], Gene Gindi[‡], and P. Anandan[†]

[†]Department of Computer Science
[‡]Department of Electrical Engineering

Yale University,
New Haven, CT 06520

June 28, 1988

Abstract

We introduce an optimization approach for solving problems in computer vision that involve multiple levels of abstraction. Specifically, our objective functions can include compositional hierarchies involving object-part relationships and specialization hierarchies involving object-class relationships. The large class of vision problems that can be subsumed by this method includes traditional model matching, perceptual grouping, dense field computation (regularization), and even early feature detection which is often formulated as a simple filtering operation. Our approach involves casting a variety of vision problems as inexact graph matching problems, formulating graph matching in terms of constrained optimization, and using analog neural networks to perform the constrained optimization. We will show the application of this approach to shape recognition in a domain of stick-figures and to the perceptual grouping of line segments into long lines.

This work was supported in part by ONR grant N00014-86-0310, by AFOSR grant AFOSR-85-0344, and by DARPA grant DAAA15-87-K-0001.

1 Introduction

Optimization has been one of the more powerful and successful methods in many areas of computer vision. Examples include most of the work in relaxation labeling (see [Hummel-83] for a review), the various uses of optimization in surface reconstruction [Grimson-85,Terzopoulos-86] and optical flow computation [Horn-81,Anandan-88], some of the early attempts at graph matching via template-spring models [Barrow-71,Ballard-82, Fischler-73], and some of the recent formulations [Geman-84] of image restoration as a stochastic optimization process.

Much of the past computer vision work involving optimization can be divided broadly into three categories: (i) dense field computation typically involving a smoothness constraint, (ii) relaxation formulations, especially for edge and region labeling problems, and (iii) simple model matching problems which are transformed into graph matching and are solved using a type of "template-spring" approach. An attractive feature common to all three classes of approaches is that they are easily amenable to "neural" network implementation.

The most notable examples of the use of optimization for dense field computation are the surface reconstruction algorithms [Grimson-85,Terzopoulos-86], the optical flow algorithms [Horn-81,Anandan-88], and the shape-from-X algorithms – e.g., the shape from shading technique of Horn [Horn-86]. For this class of techniques, which are referred to as "regularization" techniques [Poggio-85], the input is usually a retinotopic map of low-level measurements (involving image intensity and its various spatio-temporal derivatives), and the output is another retinotopic map which is the dense field being computed. The optimization problem typically involves two terms which may be called the approximation error and the smoothness error. The approximation error measures the amount of inconsistency between the estimated dense field and the measured data, and the smoothness error measures the amount of spatial variation in the dense field. Sometimes the smoothness term is modified [Terzopoulos-86] to allow discontinuities in the field.

The relaxation approach for labeling has a long history in computer vision and includes the blocks world algorithms of [Waltz-75] and edge and region labeling algorithms [Rosenfeld-76,Faugeras-81]. The problem is formulated typically as that of choosing a label

(from a discrete set of labels) for a set of objects so that certain compatibility constraints between the labels of neighboring objects are satisfied. The edge and region labeling algorithms are of particular interest to us, because they appear more readily amenable to an optimization formulation. In fact, Faugeras and Berthod [Faugeras-81] provided an explicit optimization formulation of the labeling problem. Later, Hummel and Zucker [Hummel-83] unified a variety of relaxation algorithm into a single optimization framework, and provided an extensive theoretical analysis of relaxation labeling.

The third class of approaches using optimization in vision has fewer examples. Fischler and Elschlager [Fischler-73] proposed a template-spring model for matching an object, represented as a graph, with image data. The nodes of the graph are the parts of the object and the arcs between the nodes represent the constraints which the nodes must satisfy. A function consisting of three types of terms is formulated to express the degree of match between the model graph and the data. The three terms are a "template cost" which measures the degree of mismatch between a model part and a piece of the data, a "spring cost" which measures the degree to which a constraint between two model nodes is not satisfied by the corresponding data nodes, and a "missing" cost, which penalizes for missing as well as redundant nodes. Davis [Davis-79] used a similar formalism for solving a curve matching problem. In their MSYS system, Barrow and Tenenbaum [Barrow-76] propose a relaxation algorithm called M^* (based on traditional heuristic search methods such as A^*) for the consistent labeling problem. This algorithm is somewhat more general in that it combines a parallel relaxation algorithm with heuristic sequential search. Recently, von der Malsburg and Bienenstock [von der Malsburg-86,von der Malsburg-88] have formulated a graph matching problem in terms of optimization and have discussed its relevance for pattern recognition. Although the search algorithm and the meaning of consistency differ from technique to technique, the idea of translating the model matching problem into a graph matching problem and constructing a functional indicating the degree of match between model and data graphs is a common and recurrent theme.

Each of the three categories of approaches listed above are homogeneous in the sense that all the objects and models involved are at a single level of abstraction. For example, the surface interpolation schemes (even when they include explicit representation of discontinuities) do not include explicit abstractions representing objects whose surface

characteristics may in fact determine the expected degree of smoothness. On the other hand, the relaxation labeling (and the model matching) approaches rely on an unspecified low-level algorithm that delivers the objects to be labeled. This low-level algorithm must determine the objects by processing raw intensity data without benefit of the emerging results of the labeling process. While it would seem highly beneficial to allow objects at multiple levels of abstraction to simultaneously interact, surprisingly, none of the optimization approaches used in computer vision appear to have done this.

In order to incorporate interactions involving objects at multiple levels of abstraction, it may be necessary to include hierarchical organization, as it is typically done in non-optimization approaches to computer vision [Hanson-86]. Two types of hierarchical organizations are important: compositional hierarchies involving object-part relationships and specialization hierarchies involving object-class relationships. The advantage of hierarchical organization is that it makes the search process involved in image interpretation easier to express and more efficient. In this paper, we will introduce methods for obtaining these improvements in programmability and efficiency by incorporating hierarchical organizations into the optimization paradigm.

2 Casting Model Matching as Optimization

Our approach to the use of optimization in vision involves casting a variety of problems as inexact graph matching problems and formulating the graph matching problem as the minimization of an objective function. We are interested in analog neural network solutions to this optimization problem. Many advantages of such networks, including speed, parallelism, and biological plausibility have been noted [Hopfield-85,Poggio-85]. Network formulations of vision problems involving multiple levels of abstraction have been pursued by Ballard [Ballard-86] and others. We, too, are concerned with network formulations of problems involving multiple levels of abstraction, but in the context of objective functions for graph matching. The large class of vision problems that can be subsumed by our method includes traditional model matching, perceptual grouping, dense field computation (regularization), and even early feature detection which is often formulated as a simple filtering operation.

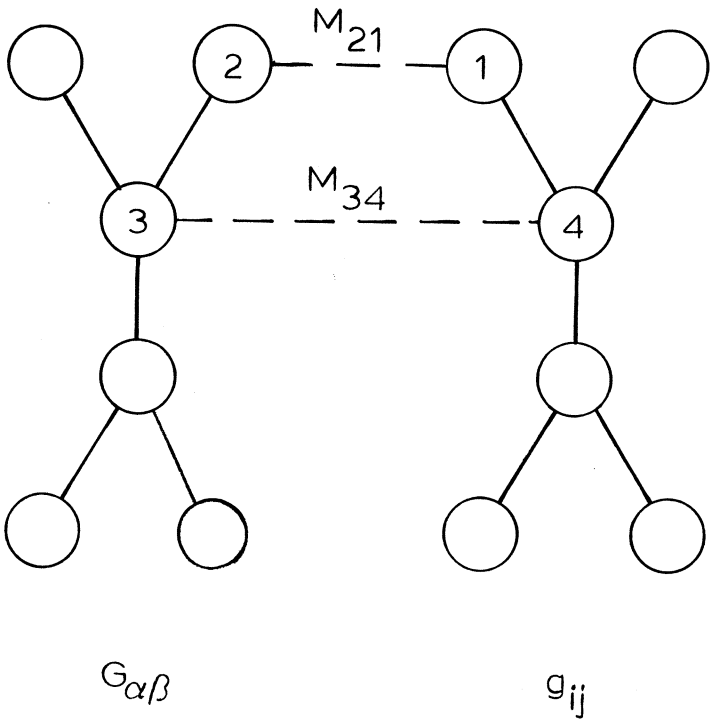


Figure 1: **Graph Matching** – The circles represent graph nodes, the solid lines graph arcs and the dotted lines matches between nodes in two different graphs. The nodes 1,2,3 and 4 form a consistent rectangle.

2.1 Model Matching as Graph Matching

Consider the following problem of pure graph matching (see Figure 1). Given two graphs each consisting of N nodes, let α $(1 \leq \alpha \leq N)$ index the nodes of one of the graphs, and i $(1 \leq i \leq N)$ index the nodes of the other. Let $G_{\alpha\beta}$ be the connection matrix of one of the graphs, i.e. $G_{\alpha\beta} = 1$ if the nodes α and β are connected by an arc; otherwise $G_{\alpha\beta} = 0$. Likewise g_{ij} is the connection matrix of the other graph. A sparse match matrix M $(0 \leq M_{\alpha i} \leq 1)$, of dynamic variables represents the correspondence between nodes α and i. Note that the entries in the graph connection matrices and those in the match matrix represent the pointers that are typically used in symbolic graph matching, but the numerical encoding has the advantage that it allows for concise expression of objective functions.

A simple objective function for this match matrix just maximizes the number of consistent rectangles[Hopfield-84a] (see Figure 1):

$$E(M) = -\sum_{\alpha\beta}\sum_{ij} G_{\alpha\beta} g_{ij} M_{\alpha i} M_{\beta j}. \tag{1}$$

This expression may be understood as follows: Given nodes α and i, the match value $M_{\alpha i}$

is to be increased if the neighbors of α match to the neighbors of i. If nodes β and j are neighbors of α and i, respectively, (i.e., $G_{\alpha\beta} = 1$ and $g_{ij} = 1$), then a nonzero $M_{\beta j}$ is evidence of such a neighborhood match, hence $M_{\alpha i}$ should be increased. By symmetry, a non-zero $M_{\alpha i}$ causes an increase in $M_{\beta j}$. As shown in Figure 1, nodes 2 and 3 on the model side and 1 and 4 on the data side form the corners of a "rectangle". For a binary-valued match matrix, E as defined in equation 1 merely counts the number of consistent rectangles in the system. Thus minimizing E is the same as maximizing the number of consistent rectangles.

Note that E as defined above can be trivially minimized by setting all the elements of the match matrix to unity. However, to do so will violate the syntactical constraint of having a one-to-one match between nodes. For the case at hand where the two graphs have equal numbers of nodes, such syntactic constraints can be expressed as follows:

$$\begin{aligned} \sum_{\alpha} M_{\alpha i} - 1 \quad &= 0 \\ \sum_{i} M_{\alpha i} - 1 \quad &= 0. \end{aligned} \tag{2}$$

In addition, we may also require that the elements of the matrix be binary-valued, i.e.,

$$M_{\alpha i}(1 - M_{\alpha i}) = 0. \tag{3}$$

In summary, the graph matching problem can be expressed as the problem of minimizing the objective function shown in equation 1, subject to the constraints given in equations 2 and 3.

A similar objective function for pure graph matching has been found independently by von der Malsburg and Bienenstock [von der Malsburg-86]. Also, Cooper and Hollbach [Cooper-87,Cooper-88,Feldman-88] have described a neural network expressing much the same graph-matching strategy,

The transition from pure graph matching to model matching is achieved by first identifying the graph $G_{\alpha\beta}$ as a model, and identifying g_{ij} as the data. However, to complete the transition the above graph needs to be augmented in a number of ways. To represent complex visual objects, the nodes of the data graph must include quantitative parameters that record information about shape, size, position, etc. In addition to checking the structural

similarity between model and data graphs, the matching process should also measure the degree to which the data-side parameters satisfy the requirements imposed on them from the model side. Since object recognition must be independent of transformations such as translation, rotation, and scaling, parameters must be checked in a manner independent of these transformations. Although it may be possible to precompute and store the parameter checks in tables, geometric transformations (which typically occur in shape recognition) are sufficiently complex, and the incoming data may vary over a sufficiently wide range, that the table will be prohibitively large. This implies that the necessary arithmetic should be done dynamically.

As discussed above, we also require that nodes on the data side must be capable of representing objects at different levels of abstraction. To introduce the compositional hierarchy, there must be nodes representing wholes and nodes representing parts, and a special graph representing the relationship between parts and wholes. Since a single whole is usually connected to several parts, the part-whole graph is restricted to be a tree (or generally a Directed Acyclic Graph or DAG). Given a model node and a data node which matches it, all nodes which represent the parts of the data node must find unique matches among the parts of the model node.

In the simple graph matching example given above, the arcs between the nodes on the data side were provided as input. However, in most recognition and grouping problems, the dynamic determination of part-whole relationships and group membership is itself a significant and unavoidable task. The matching process should compute such relationships rather than assume them as input. The dynamic character of the grouping problem also implies that a node representing a group may not even exist until at least some of its parts have been determined. At that time the node may be allocated.

The motivation for incorporating specialization hierarchies includes the search efficiency provided by using a discrimination tree to index the models, and the notational convenience provided by the inheritance of properties from the general to the more specialized models. In particular, the addition of a specialized model requires only the verification of new properties not shared with the more general model; all the properties which are shared with the general model are automatically inherited. In order to achieve this, our model base should incorporate a specialization hierarchy (in the form of a tree or a DAG), the

data nodes should be allowed to simultaneously match to a model and at most one of its specializations, and the match process should include property inheritance.

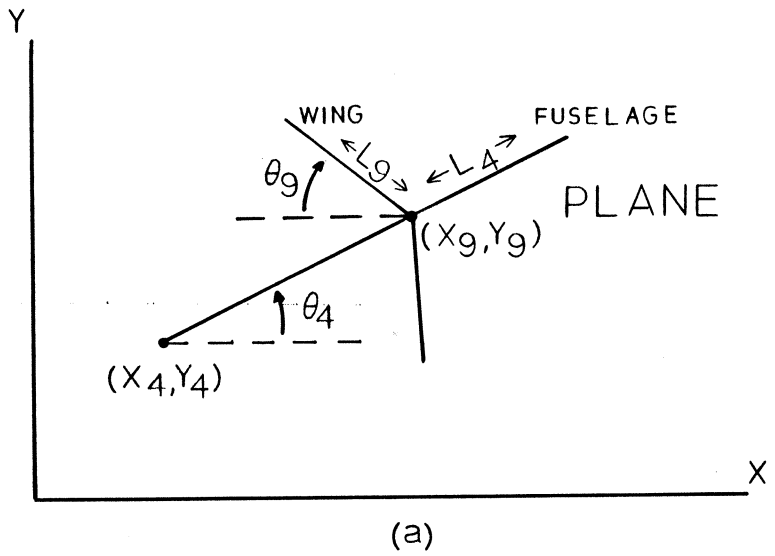
In summary, in order make the transition from pure graph matching to model matching and perceptual grouping, we have identified the following design requirements: parameter checks and arithmetic operations involving parameters, a compositional hierarchy with nodes representing objects at different levels of abstraction, dynamic allocation, and a specialization hierarchy with property inheritance.

2.2 Frames and Objective Functions

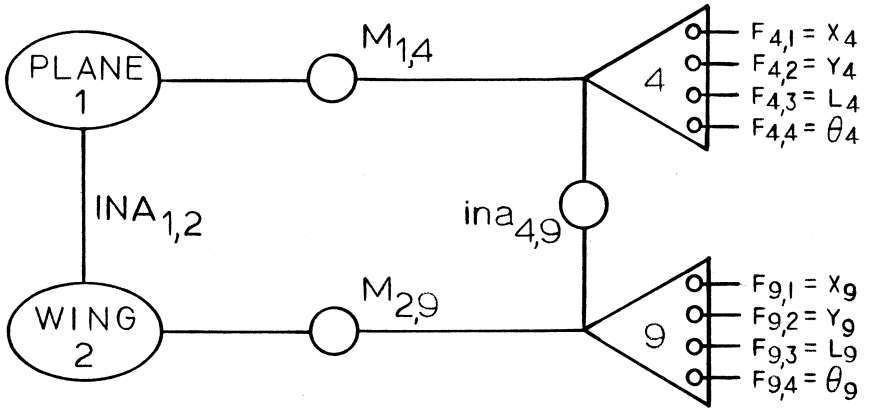
The introduction of compositional hierarchy to the pure graph matching problem is straightforward. We simply let the connection matrices represent part-whole relationships between objects and denote them as $INA_{\alpha\beta}$ on the model side and as ina_{ij} on the data side. The need for dynamic allocation of part-whole relationships implies that the data side *ina* values cannot be predetermined. This means that ina_{ij} will, in general, be a variable quantity. Also, the dynamic determination of "groups" implies that the parameters of the data node representing the group cannot be predetermined; hence, these parameters must also be variable quantities.

The addition of variable parameters to the data nodes suggests a natural organization: we package the parameters into "frames" which are bundles F_i of variable parameters F_{is}, where s indexes the different parameters (or "slots") of a frame. Frames can be regarded as representing visual abstractions or perceptual organizations in terms of a few parameters. The process of dynamically employing a frame to represent an abstraction is called "allocating" a frame. We will refer to such a network of frames and models as "Frameville". Our use frames is motivated by the conventional use of the same idea as established in [Minsky-75,Fahlman-79]. Our frames contain slots for parameters, and are connected to other frames via *ina* links. No information concerning the match criteria or information about the control flow is included in the frame. Our match criteria are expressed as objective functions while our control process is based on the particular choice of a technique for minimizing the objective function.

We introduce the necessary modifications to the objective functions and constraints in order to perform model matching with the aid of a Frameville example (Figure 2). The



(a)



(b)

Figure 2: **Example of Frameville rectangle rule.**(a) Example object, a plane, consists of parameterized parts fuselage and wings. (b) shows the rectangle relationship between frames (triangles) representing a wing and the fuselage. The open circles symbolize dynamic variables that interact according to Equations 4 and 5. The ovals represent the models (plane and wing) and the arc connecting them represents the *INA* relationship between them.

rectangle shown in Figure 2b is the extension of the rectangle shown in the pure graph matching problem (see Figure 1). Equation 1 becomes the "rectangle" rule:

$$E(M) = -\sum_{\alpha\beta}\sum_{ij} INA_{\alpha\beta} ina_{ij} M_{\alpha i} M_{\beta j}. \tag{4}$$

Whereas in the case of graph matching the sole syntactic constraint was that of one-to-one matches between data and model nodes, with the introduction of the compositional (and specialization) hierarchies, more complex syntactic constraints are needed. The new syntactic constraints will be discussed below. In Figure 2b, the ovals on the left represent the models while the triangles on the right are the data side frames. The circles inside the triangles represent the parameters (slots) of the frame. The line connecting models 1 and 2 indicates the part-whole relationship between the two models. The circles labeled M are the match variables. The circle labeled *ina* on the data side is the variable indicating the dynamic part-whole relationship between the frames. We will use circles to denote dynamic variables, ovals to denote models, and triangles to denote frames.

In this example, the parameters of the line representing the fuselage are also used as the parameters of the entire plane. In Figure 2a, the parameters (x_4, y_4) denote the coordinates of the end point of that line, θ_4 represents the orientation of that line with respect to an arbitrary but fixed coordinate system, and L_4 represents the length of that line. Similarly $(x_9, y_9, \theta_9, L_9)$ are the parameters of the line representing a wing of the plane. As in the case of the pure graph matching, the number of consistent rectangles should be maximized. This is expressed as the minimization of the term given above in equation 4.

Additionally, in order for the rectangle $(1, 4, 9, 2)$ to be consistent, the parameters F_{4s} and F_{9s} should satisfy the criterion indexed by models 1 and 2. Such a constraint generally results in the addition of the following term to the objective function:

$$\sum_{i,j,\alpha,\beta} INA_{\alpha\beta} ina_{ij} M_{\alpha i} M_{\beta j} H^{\alpha\beta}(F_{i1}, F_{i2}, \ldots, F_{j1}, F_{j2}, \ldots) \tag{5}$$

where the term $H^{\alpha\beta}(.,.)$ measures the deviation of the parameters of the data side frames from what is demanded by the models. Equation 5 expresses a general principle of Frameville, namely that objective functions can be model specific. This is an extension of the usual use of distance metrics in pattern recognition. Here the metric to be used is

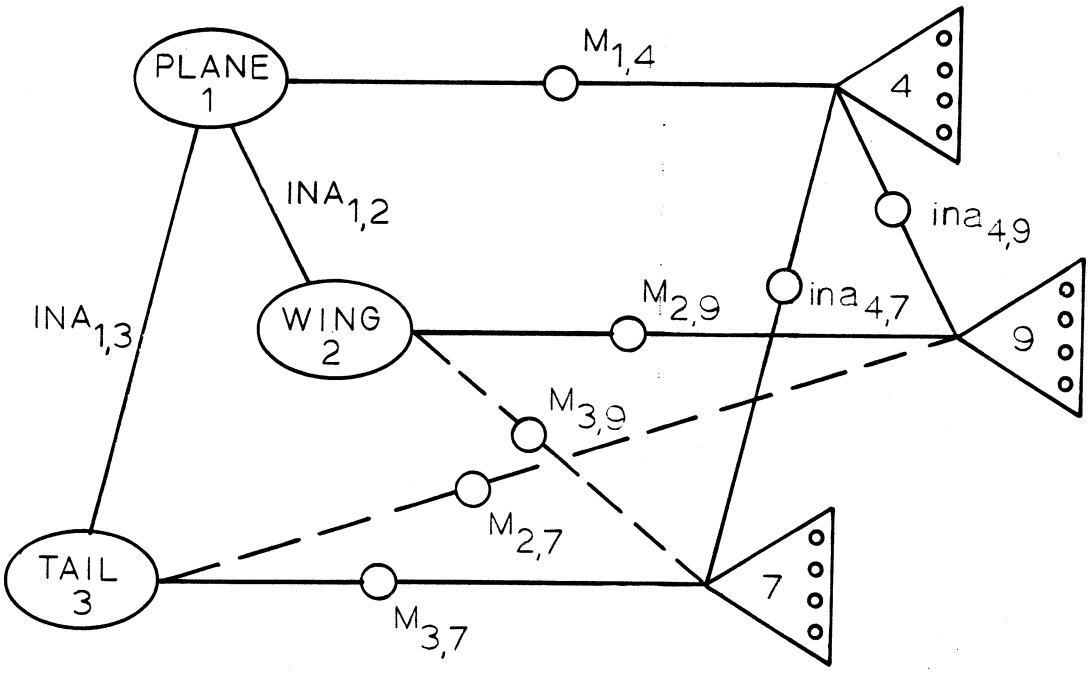


Figure 3: **Frameville sibling competition among parts.** The match variables along the dotted lines ($M_{3,9}$ and $M_{2,7}$) are suppressed in favor of those along the solid lines ($M_{2,9}$ and $M_{3,7}$)

.

context dependent - indeed there is an entire database of metrics $H^{\alpha\beta}$. In the example in Figure 2 we specifically require that the end point of the line representing the wing should lie on the line representing the plane (fuselage), and that the wing should be at some reasonable angle for being a wing, e.g., the relative angle of the wing to the fuselage should be in the range $(45 - 90°)$.

To illustrate the dynamic character of the matching variables M and the "allocation" variables *ina* we include an additional part (the tail) to our example (see Figure 3). We impose the additional constraints that a single frame (say 7) cannot match simultaneously the wing (model 2) and the tail (model 3), and that a single model (say 2) cannot match simultaneously two different frames (say 7 and 9) which are constituents of the same parent frame (say 4). More generally, two distinct parts of the same whole on the model side must match two distinct frames and vice-versa. This is expressed by the two equations given

below:

$$INA_{\alpha\beta}M_{\alpha i} - \sum_j ina_{ij}M_{\beta j} = 0 \quad (6)$$

$$ina_{ij}M_{\alpha i} - \sum_\beta INA_{\alpha\beta}M_{\beta j} = 0. \quad (7)$$

Obviously, the correct assignment of the frames to models will be determined by the most consistent parameter checks.

We illustrate the incorporation of a specialization hierarchy by modifying the example shown in Figure 2 to the one shown in Figure 4. Specifically, two competing specializations, "propeller-plane" and "jet-plane" are introduced, along with the corresponding specializations of the wing model. We incorporate a specialization hierarchy by introducing a link on the model side called $ISA_{\alpha\beta}$, a binary matrix equal to unity if model β is a specialization of a model α. A frame may simultaneously match to a model and at most one of its specializations. Thus, in this example we require that Frame 4, which matches to the plane (model 1) should also match uniquely to one of jet-plane (model 4) and propeller-plane (model 6). Similarly Frame 9, which matches the wing (model 2) also matches uniquely to one of models 5 and 7 (prop-wing and jet-wing respectively). In general, matches between *ISA* siblings compete for matches to a given frame; it is this competition which makes the network act as a discrimination tree, and is expressed as

$$M_{\alpha i} - \sum_\beta ISA_{\alpha\beta}M_{\beta i} = 0. \quad (8)$$

Note that property inheritance is automatically achieved by allowing the same frame to match to a model and its specialization. The additional verification of properties specific to the specialization is simply expressed as additional model-specific constraints involving the parameters. For example, in order for a plane to specialize to jet-plane, its wing must specialize to a jet-wing, which involves verifying more stringent constraints on the relative angle of the wing to the fuselage (e.g., that it should in the range 45-60 degrees). We need not reverify that the jet-wing satisfies all requirements for a general wing model, as this is verified by the plane-wing consistency rectangle (i.e., the rectangle involving models 1 and 2 and frames 4 and 9).

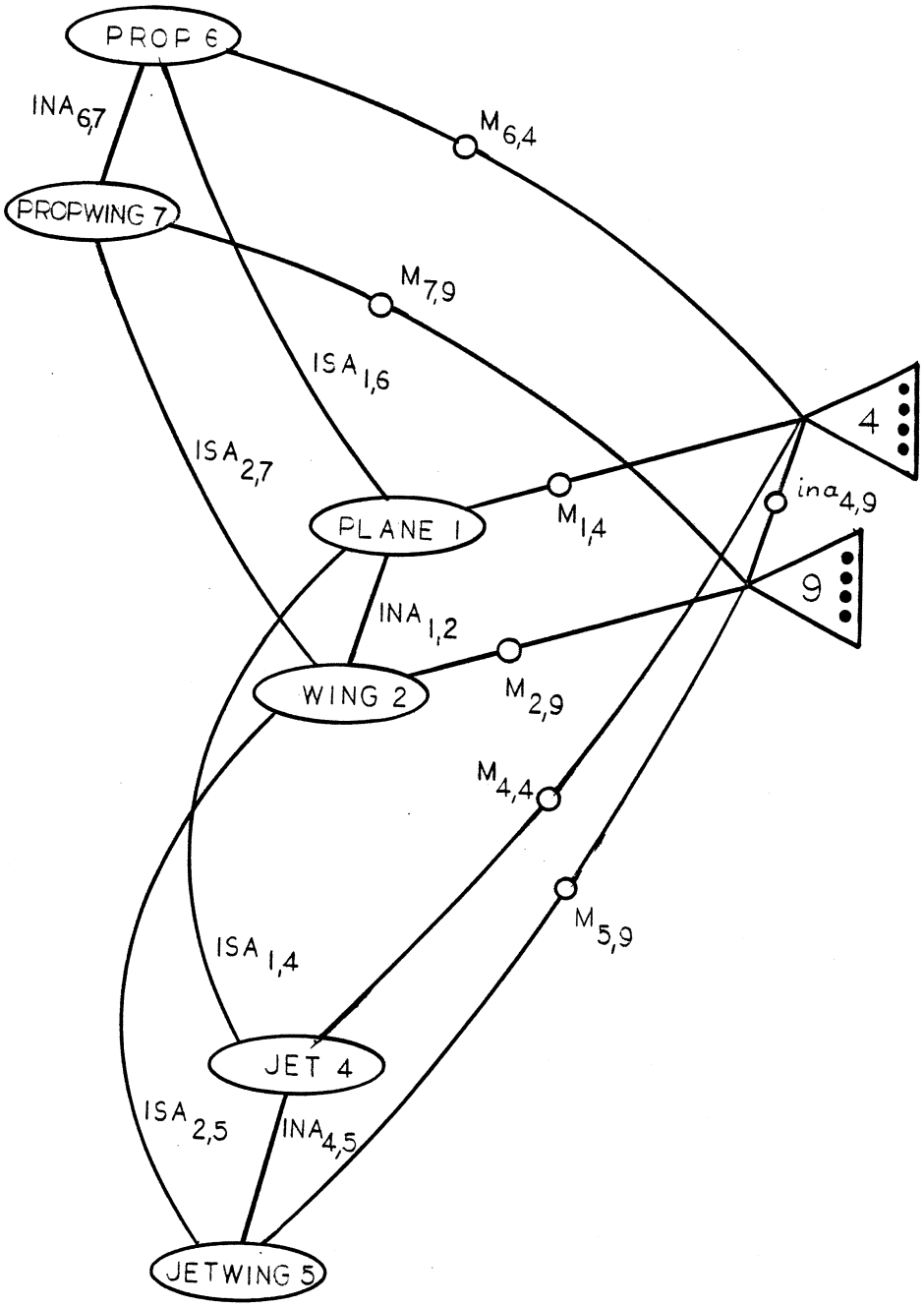


Figure 4: **Frameville specialization hierarchy**. The plane model specializes to a propeller or a jet plane and correspondingly the wing model specializes to prop-wing or jet-wing. Match variables $M_{6,4}$ and $M_{4,4}$ compete as do $M_{7,9}$ and $M_{5,9}$.

Figure 4 also illustrates a general syntactic requirement on the *ISA* and *INA* matrices. If, for example, three of the four links of the "diamond" formed by the plane, wing, propeller, and propeller-wing models are present then the fourth must also be present. Note that this is a syntactic requirement on the constant matrices of the model side and not a rule involving the dynamic variables.

Finally, we include the constraints that the match M and the ina variables should stabilize to binary values (0 or 1), although they may take intermediate values during the course of the optimization. Thus, we require the following:

$$\begin{aligned} M_{\alpha i}(1 - M_{\alpha i}) &= 0 \\ ina_{ij}(1 - ina_{ij}) &= 0. \end{aligned} \tag{9}$$

The parameters F_{is} should, however, be allowed to take real values; currently, we use an analog variable to represent such parameters. This contrasts with the value-unit representations used in [Ballard-86] and other traditional connectionist approaches. Our choice is motivated by the fact that complex arithmetic relationships, such as those involved in coordinate transformations, will be relatively inefficient if value-unit representations are used.

3 Optimization Techniques

Having outlined some of the basic objective functions and constraints involved in our optimization problem, we now consider various approaches for performing the optimization. The optimization problem at hand can be stated as that of minimizing $E(\mathbf{x})$, subject to constraints

$$h_i(\mathbf{x}) = 0, \quad \forall i, 1 \leq i \leq m \tag{10}$$

$$g_j(\mathbf{x}) \leq 0, \quad \forall j, 1 \leq j \leq p \tag{11}$$

where $\mathbf{x}$ is the collection of all dynamic variables M, ina, and F_{is} in Frameville, and h_i are the previously listed syntactic constraints and g_j express the restrictions on the ranges of some of the dynamic variables.

3.1 Unconstrained Optimization – Hopfield Networks

One can generally approximate a constrained problem as an unconstrained problem by incorporating the constraints as additive "penalty" terms in the objective function. Each constraint results in a term of the form $c_i h_i^2(\mathbf{x})$ where c_i is a constant. For example, Equation 8 specifying the constraint of sibling competition in a specialization hierarchy becomes the additive term

$$c(M_{\alpha i} - \sum_{\beta} ISA_{\alpha\beta} M_{\beta i})^2. \tag{12}$$

Thus the Frameville objective function to be minimized consists of the original rectangle rule plus penalty terms to reflect constraints.

Hopfield [Hopfield-84b] has proposed a neural network implementation for unconstrained optimization problems similar to ours. In [Hopfield-84b] the dynamic variables are identified with neurons; the architecture (connection weights between neurons) follows from the form of the objective function; and a descent procedure specifies the update rule for each neuron. As discussed above, constraints are incorporated as penalty terms in the objective function [Hopfield-85], and, in addition, an "analog gain" term is added to constrain the dynamic variables to lie within a certain range. This analog gain term leads to a network which includes the familiar nonlinear sigmoidal mapping beween the internal state of a neuron and its output.

For a Frameville version, the resulting equations of motion are

$$\begin{aligned} \frac{du_i}{dt} &= -\frac{\partial E}{\partial V_i} \\ V_i &= g(u_i) \end{aligned} \tag{13}$$

where E is the objective function, and, for notational convenience, the dynamic variables M, ina, and F_{is} are identified with a singly subscripted variable V_i, the output of the ith neuron. Also, u_i is the internal state of the ith neuron and g is a sigmoidal mapping between V and u. Unlike the familiar quadratic objective functions used by [Hopfield-85] and others, our objective functions are of higher order (> 2) as seen from the form of the rectangle rule, Equation 4, and the constraints in Frameville. Also, the nonlinear analog gain term applies only to M and ina.

Finally, it should be noted that for neural nets in general, the objective function should be of a certain simple form, typically polynomial combinations of dynamic variables, in order that the objective function be easily mapped into a network. For Frameville, this means that the parameter check functions $H^{\alpha\beta}$ of Equation 5 should be of polynomial form.

3.2 Constrained Optimization With Networks

The advantage of approximating the constrained optimization problem by an unconstrained problem is that it is possible to use descent methods to solve the unconstrained problems, and such methods naturally give rise to network implementations. However, sometimes we may wish to retain "hard" constraints, i.e., require that a constraint be strictly satisfied. In some cases, this may even be a requirement; for example, when performing arithmetic it may be useful to have intermediate variables which are related to other variables via hard constraints (see line grouping example in section 4).

A common method to solve a constrained optimization problem with "hard" constraints is the method of Lagrange multipliers. This involves the formulation of the Lagrangian [Luenberger-84]

$$l(\mathbf{x}, \lambda_1, \lambda_2, \ldots) = E(\mathbf{x}) + \sum_j \lambda_j h_j \tag{14}$$

where λ_j are the Lagrange multipliers. The necessary conditions for a solution to the original constrained problem are,

$$\begin{aligned} \nabla_{\mathbf{x}} l(\mathbf{x}, \lambda_1, \lambda_2, \ldots) &= 0 \\ h_j(\mathbf{x}) &= 0. \end{aligned} \tag{15}$$

A search technique, itself possibly an optimization, is generally needed to solve for these conditions.

Recently, a network implementation [Platt-87] has been developed for a continuous descent algorithm [Platt-87,Arrow-58] to solve the search problem given above. In addition to the original set of dynamic variables, the descent algorithm also updates the Lagrange multipliers. Thus, in Platt's implementation there are nodes denoted V_i in the network

corresponding to each of the dynamic variables and other nodes denoted λ_j to the Lagrange multipliers. The equations of the motion are,

$$\begin{aligned} \frac{dV_i}{dt} &= -\frac{\partial E}{\partial V_i} - \sum_j \lambda_j \frac{\partial h_j}{\partial V_i} \\ \frac{d\lambda_i}{dt} &= h_i(V_1, V_2, \ldots). \end{aligned} \tag{16}$$

Platt [Platt-87] has demonstrated good results in applying this method to the solution of the Travelling Salesman Problem and an analog decoding problem. Since some of our examples involve the use of hard constraints, this may be a promising technique to consider.

Finally, while there are a number of traditional optimization techniques, such as the conjugate gradient method, Newton's method, and heuristic search techniques such as Branch and Bound (e.g., [Barrow-76] uses this technique for a labeling problem), it is not clear whether these techniques are suitable for network implementation.

4 Examples

4.1 Recognition of Simple Stick Figures - Stickville

An early implementation these ideas was object recognition in Stickville, a precursor to Frameville that contained many of its important aspects, but involved quadratic optimization instead of the higher order terms typical of Frameville. Stickville is a domain of stick figures composed of connected assemblages of linear segments or "sticks". As shown in Figure 5 a model part at the lowest level corresponds to a single stick. Abstraction in Stickville is interpreted as representing a collection of parts by a single main part. Thus the "plane" model is abstracted by "fuselage", the main part of "plane".

An important restriction in Stickville is that the data groupings are not dynamic; they are precomputed. Thus ina_{ij} is a constant and not a dynamic variable. In Stickville, the ina matrix is defined so that $ina_{ij} = 1$ if data sticks i and j are connected and zero otherwise. Also, our input data was restricted to consist of a set of trees, i.e., each stick in a connected assemblage of sticks (except for one root stick) had the property that one of its ends abutted the side of one other stick. Figure 5a shows the stick figure of a plane

and its *ina* links, and 5b shows a model representation for a plane and its *INA* links. The tree structure may be observed in figure 5a.

Another important restriction involves parameters on the data side. Like *ina* links, parameter checks are also precomputed; hence there are no dynamic variables corresponding to frame slots. For Stickville, three quantities are precomputed for each pair of connected sticks i and j. These parameters, illustrated in figure 5c are: r_{ij}, the size of stick j relative to i, θ_{ij}, the relative angle, and y_{ij}, the location of the attach point of stick j to stick i in a coordinate system whose y-axis ($0 \leq y \leq 1$) is coincident with stick i [Gindi-86]. If stick i abuts stick j, then $y_{ij} = 0$ (e.g., $y_{14} = 0$ in figure 5c). Thus in place of a frame with slots, a vector of fixed parameters is associated with each connected ($ina_{ij} = 1$) pair of data nodes.

A specialization hierarchy is included in Stickville. Figure 6 shows a model-base typical of the ones used in our simulations. In Stickville, specialization is achieved by the inclusion of new parts and by the restriction of allowed parameter ranges on pairs of other parts. For example, in order to specialize "plane" to "jet", four new parts (two engines, two tails) must be found, and the angle of the wing relative to fuselage must fall within a subset of the allowed wing-fuselage angle for plane, i.e the wings must be swept back. The specialization hierarchy is encoded in a binary matrix $ISA_{\alpha\beta}$ as described earlier.

The rectangle rule for Stickville becomes

$$\begin{aligned} E(M) &= -\sum_{\alpha,i,\beta,j} INA_{\alpha\beta} ina_{ij} M_{\alpha i} M_{\beta j} H^{\alpha\beta}(\mathbf{J}_{ij}) \\ \mathbf{J}_{ij} &= (r_{ij}, \theta_{ij}, y_{ij}). \end{aligned} \tag{17}$$

Note that the only dynamic variables are the match neurons M. Unlike Equation 17 where the parameters F_{is} depend on one frame i and a slot number s, the precomputed parameters J_{ij} here depend on two data sticks i and j. The parameter check term H is a stored function that evaluates the vector $\mathbf{J}$ and returns a large number if the data side parameters satisfy conditions specified by $H^{\alpha\beta}$ on the model side. ($H^{\alpha\beta}$ is a sum of three quadratics, one for each of r, θ, and y). The choose rule, Equation 8, retains its form.

Because the number of *INA* links to a model α may not equal the number of *ina* links to a stick i, a match between α and i cannot require a strict one-to-one match between the

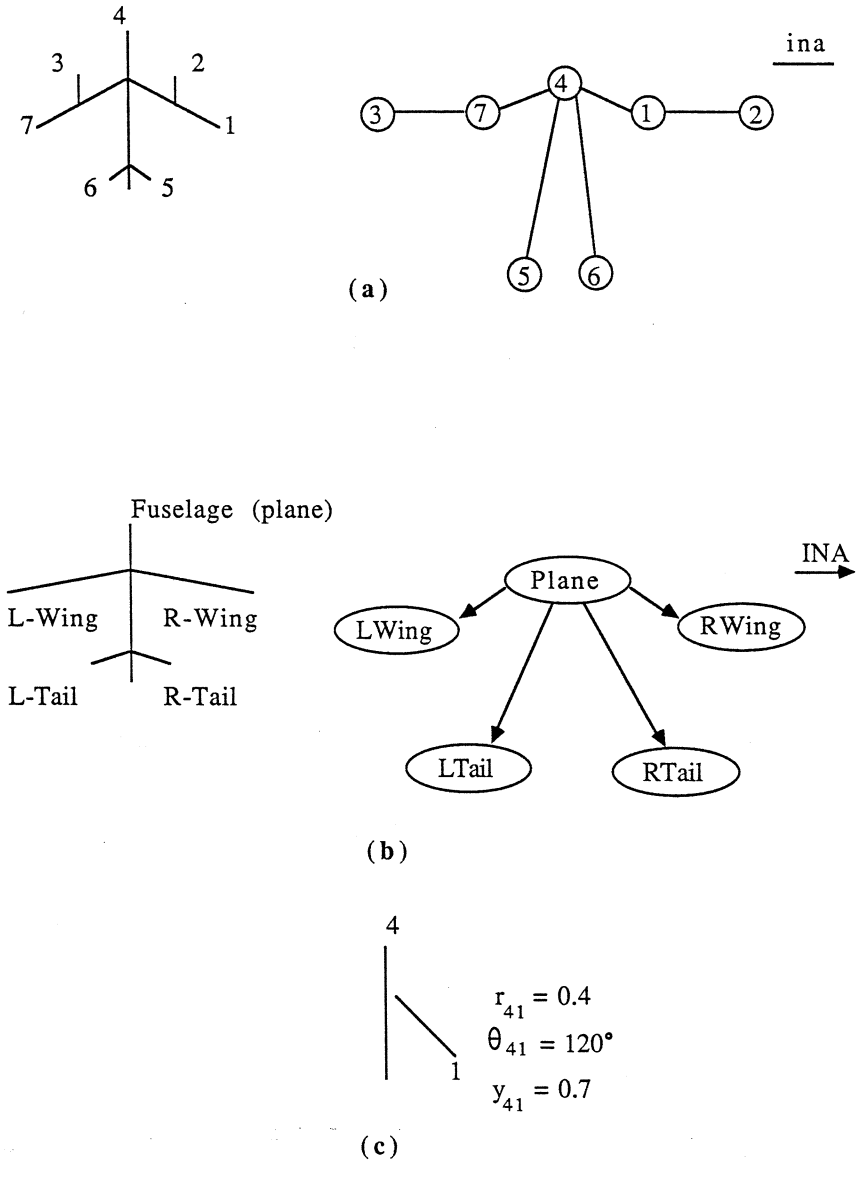


Figure 5: **Definitions in Stickville.** (a) A stick figure of a plane, and its representation as a graph with *ina* links (which are symmetric in Stickville). (b) A plane model and its representation as a graph with asymmetric *INA* links. (c) Parameters characterizing the relationship between two data sticks.

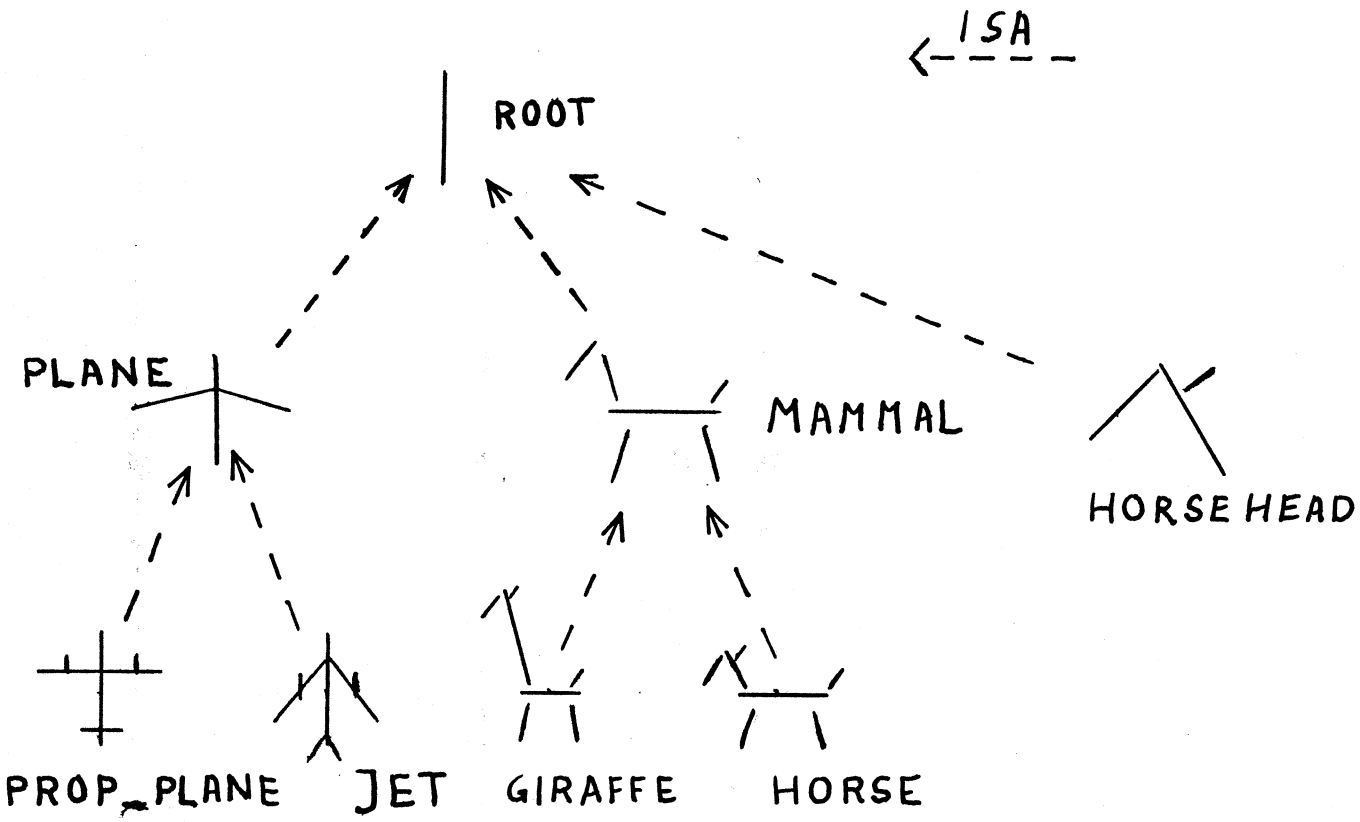


Figure 6: **Typical specialization hierarchy for Stickville.**

parts of model α and parts of stick i (i.e., those sticks connected to it). Thus in Stickville, where the data-side *ina* links are fixed *a priori*, Equations 6 and 7 cannot both be exactly satisfied.

Although a higher-order term can be used to solve this problem, we were interested in finding a quadratic term for reasons of economy. Therefore, for the present implementation, we used the following three constraints which can be exactly satisfied.

$$\sum_{\beta j} INA_{\alpha\beta} ina_{ij} M_{\beta j} - C_{\alpha i} M_{\alpha i} = 0 \tag{18}$$

$$\sum_{i,j} \overline{ina}_{ij} M_{\alpha i} M_{\alpha j} = 0 \tag{19}$$

$$\sum_{\alpha,\beta} \overline{INA}_{\alpha\beta} M_{\alpha i} M_{\beta i} = 0 \tag{20}$$

where

$$C_{\alpha i} = \min(\sum_{\beta} INA_{\alpha\beta}, \sum_{j} ina_{ij})$$

and $\overline{ina}$ is the transitive closure of ina. Since INA is not symmetric, $\overline{INA}$ is defined as the transitive closure of $INA + INA^T$.

Equations 19 and 20 express the notion that a model must uniquely match to one element of a connected assemblage of sticks and vice versa. We used Equation 19 despite the undesirable consequence that it prohibits multiple occurrances of the same model within a connected assemblage of sticks. Equation 18 enforces the requirement that the number of matches between parts of α and parts of i should be as expected.

We used the Hopfield neural net model, Equations 13, as an optimization method in our Stickville simulation. The constraints were expressed as additive penalty terms and a nonlinearity was applied to nodes $M_{\alpha i}$ in order to keep the values of the match neurons within range. We used the forward Euler method to simulate the differential equations as a sequence of discrete time steps.

Results are displayed in Figures 7 and 8 in the form of the match matrix. Here, each circle is a match neuron $M_{\alpha i}$ and its value is proportional to the radius of the shaded portion. The rows are labelled with symbols corresponding to model parts (e.g. jet, jet-left-wing, right-tail, mammal, foreleg, etc.), and each column corresponds to a single data

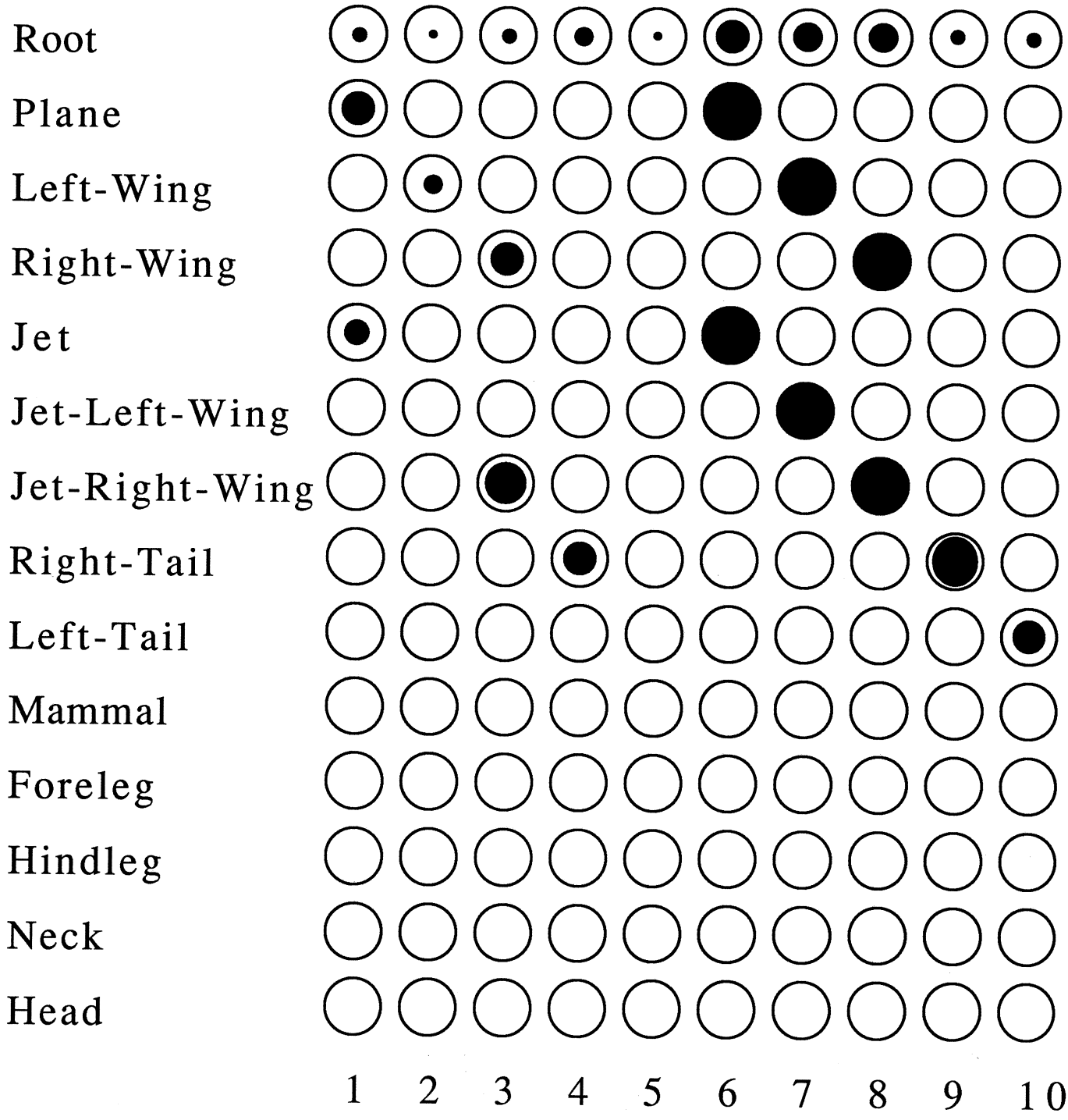


Figure 7: **Stickville match matrix after 28 time steps.** The circles represent the dynamic variables whose value is encoded by the radius of the shaded portion. The different columns represent the different data sticks, and the different rows represent different models. The model-base consists of plane, jet, mammal and their parts, and a root model. The data consists of two jets.

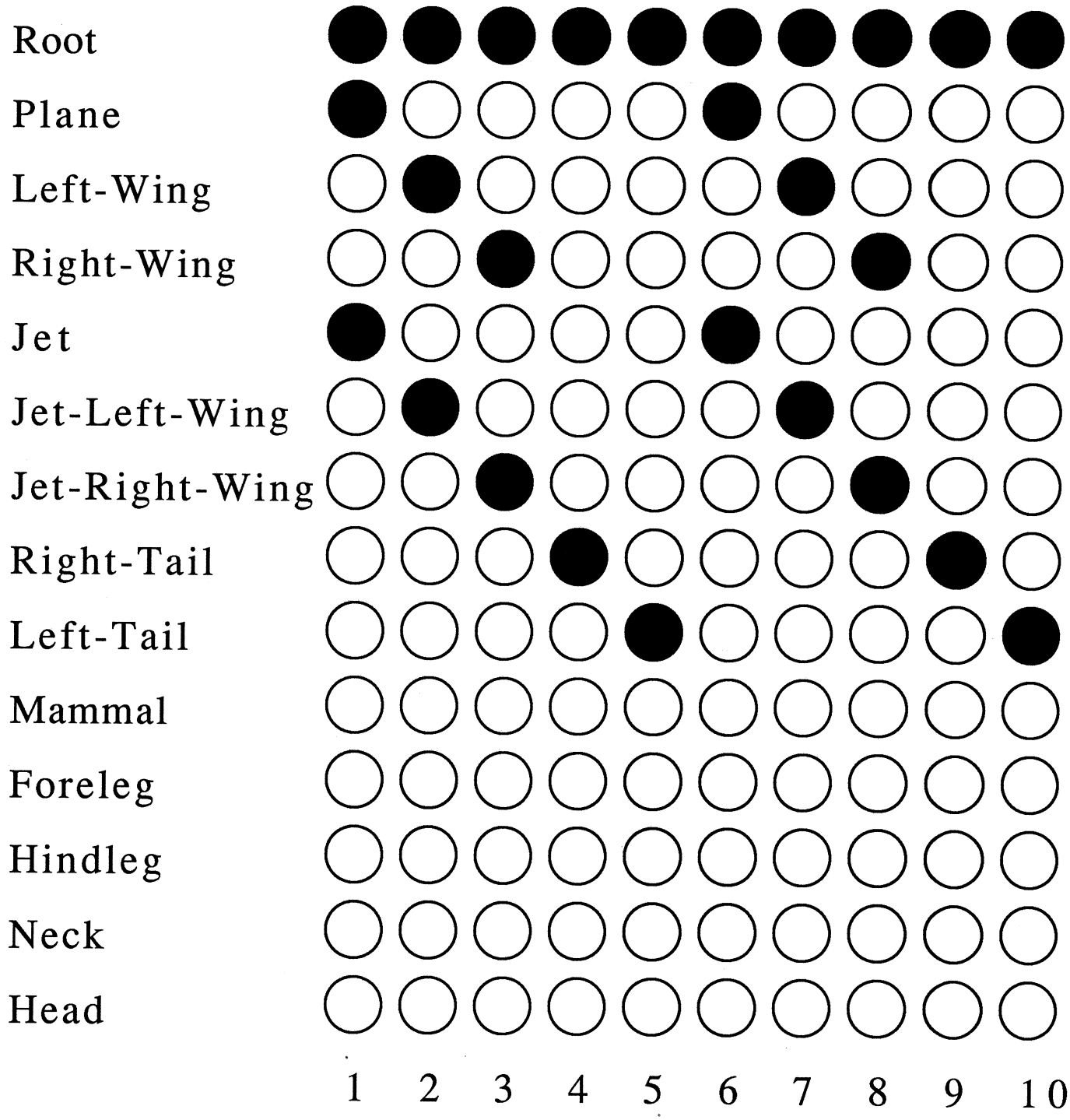


Figure 8: **Stickville match matrix achieves the correct fixed-point after 70 time steps.** Note that a stick matches to a model and all of its generalizations (more than one neuron on in a column) and that both the jets have been found (more than one neuron on in a row).

stick. For the results shown, the input consisted of 10 sticks comprising two separate jets, and the model base consisted of a subset of that shown in Figure 6.

Figure 7 shows the state of the system after 28 time steps and Figure 8 shows the system after it has stabilized at about 70 time steps. Both jets are recognized. Note that multiple instances of the same model are recognized and that the network recognized both the plane model and its specialization, the jet model.

4.2 FLville, A Frameville Implementation of a Compositional Hierarchy.

Experiments were conducted in a Frameville implementation of FLville, a domain constituting a two-level compositional hierarchy. Figure 9 illustrates the domain on the data side. At the lowest level are unit-length line segments parameterized by location x, y and orientation θ, corresponding respectively to slots $(F_{is}, s = 1, 2, 3)$. We allow only horizontal $(\theta = 0)$ and vertical $(\theta = \pi/2)$ orientations. There are two high-level models, "F" and "L", composed of appropriate sets of four low-level segments. The figure illustrates an "F" (darkened segments) in the center of the array, and an "L" at the top right. Also shown are extraneous "noise" segments. To simplify matters, we designated one group of frames as "high level" in that they could possibly match only to high-level models, and a separate group as correspondingly "low-level" frames. The simple task here is to use the Frameville methodology to recognize instances of "F", "L", and their parts. The high-level models are parameterized by the parameters of a designated main part, in this case, the upper vertical segment of an "F" or "L".

On the model side, there are seven low-level models as shown in Figure 10. These correspond to seven possible positional roles that a segment may assume in the context of a composite figure. These positions are illustrated iconically inside the model nodes in Figure 10 and correspond to the positions of segments in the familiar seven-segment LED display. The high-level models "F" and "L" are then specified by the set of *INA* links. Shown in the Figure are *INA* links for an "F".

The rectangle rule for FLville is just that of Equation 5, where the parameter check term $H^{\alpha\beta}$ checks location and orientation of a given part relative to those of a main part. For example, if Frame 3 is matched to model 5 ,"middle horizontal", then its parameters

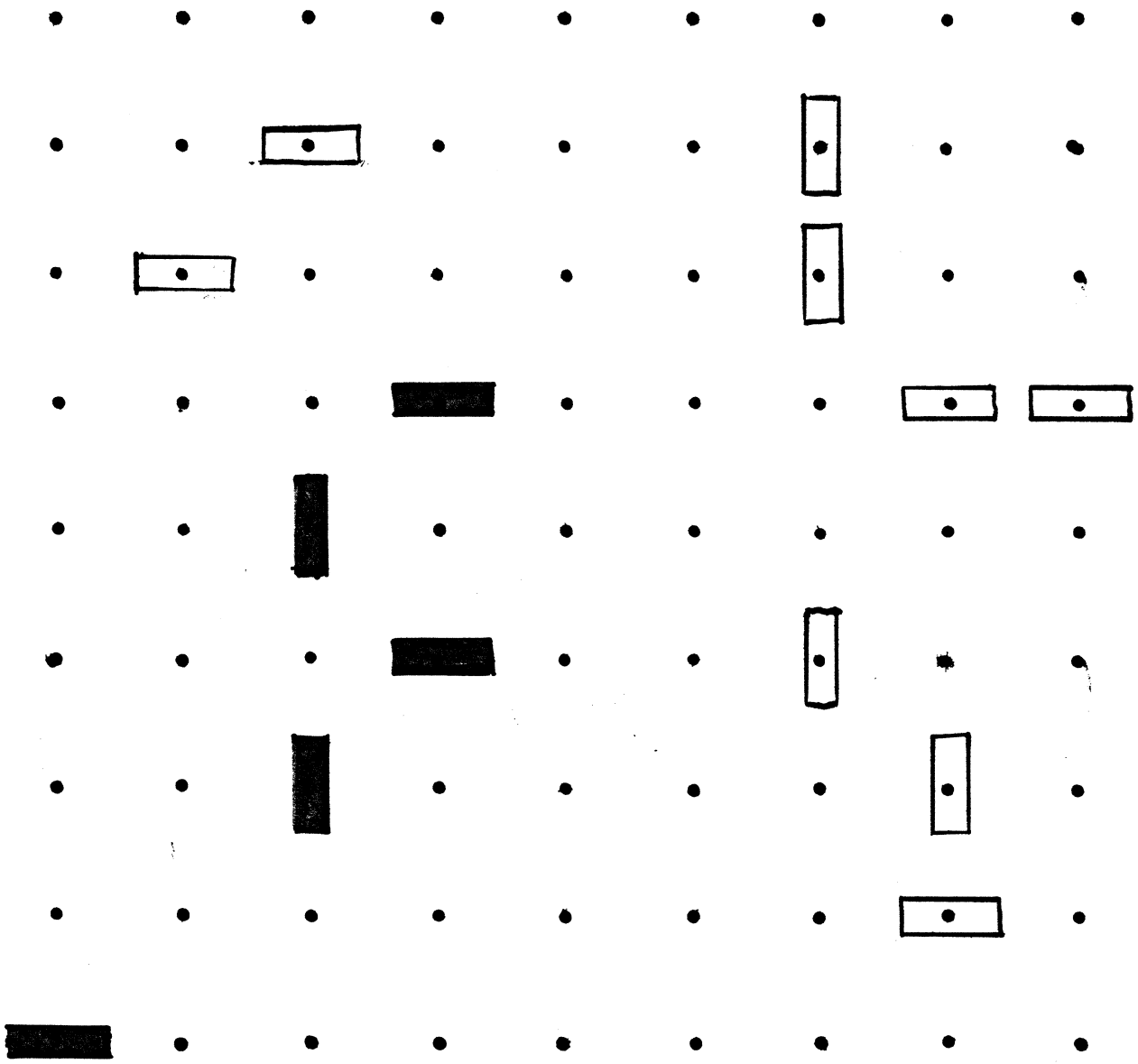


Figure 9: **Input data for FLville.** The rectangular grid of positions for unit length segments oriented horizontally or vertically. The task is to recognize four segments forming an "L" (upper right) and four segments forming an "F" (center) in the presence of noise sticks. The shaded sticks form the input data for the run described in later figures

.

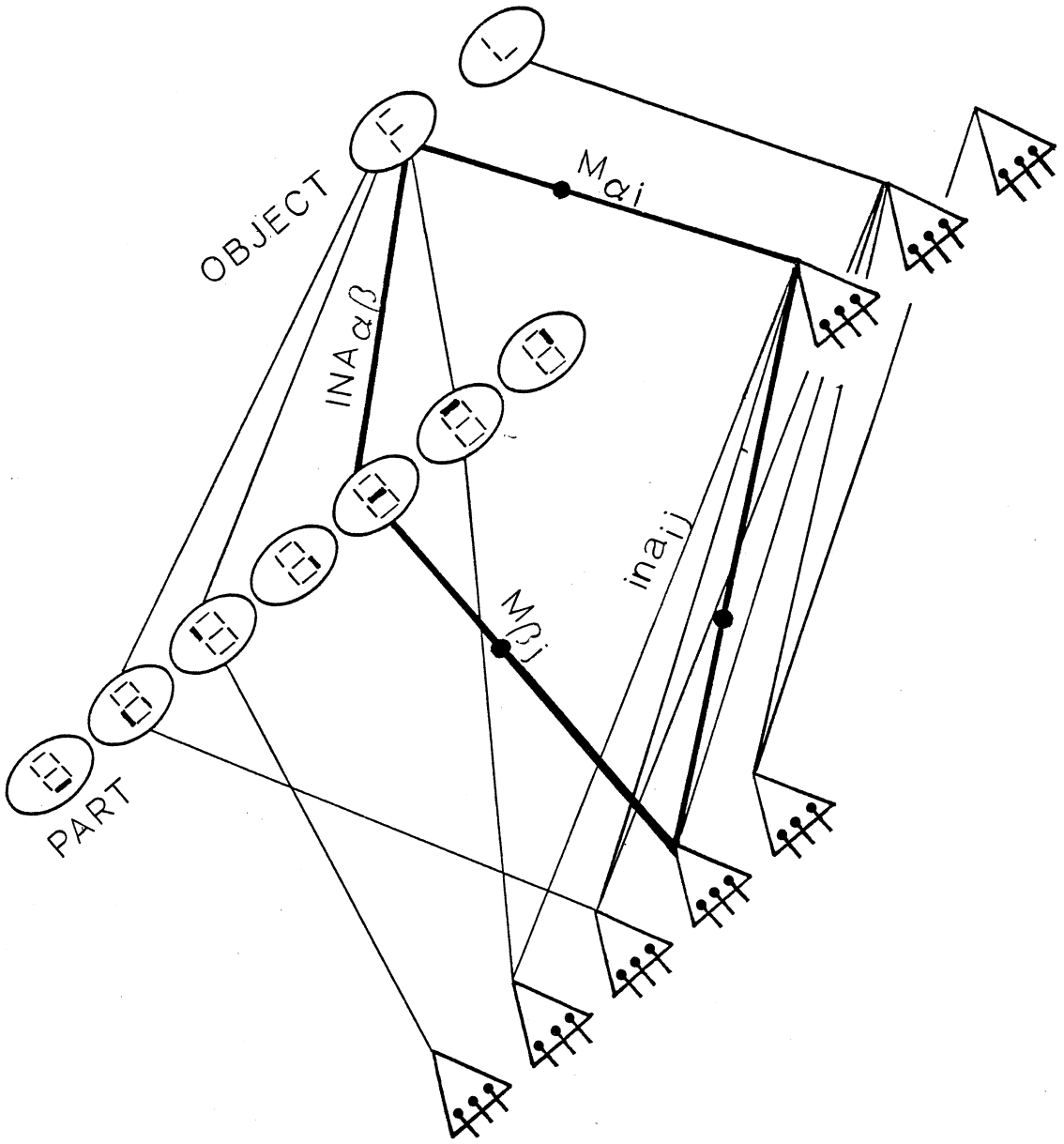


Figure 10: **Structure for FLville.** Models (represented by ovals) occur at two levels, segments and letters. The *INA* links for the letter "F" are shown. Each frame (represented by a triangle) has three slots: two position coordinates and one orientation coordinate. The bold lines highlight a possible consistency rectangle.

$(F_{3,1}, F_{3,2}, F_{3,3} = x_3, y_3, \theta_3)$ must differ from those of a "upper vertical" mainpart by quantities +1,-1, and $\pi/2$, respectively. If Frame 7 is currently matched to model 9, an "F", and contains the parameters of the main part, then an appropriate parameter check term is:

$$H^{9,5}(7,3) = (F_{7,1} - F_{3,1} - 1)^2 + (F_{7,2} - F_{3,2} + 1)^2 + (F_{7,3} - F_{3,3} - \frac{\pi}{2})^2. \quad (21)$$

The quantitities 1,-1, and $\pi/2$ are thus model information that is stored in the objective function. A similar objective function (replace 1, -1 and $\pi/2$ by 0) is used to copy the parameters of a low-level frame matched to a main part into a high-level frame. Note here that a limited form of invariance is achieved by analog computation of relative coordinates; instances "F" and "L" are recognized in a manner invariant to translation.

The other constraints in FLville follow those in generic Frameville with the exception of the syntactic constraints in Equations 6 and 7. In FLville we used a less restrictive form

$$\sum_{\alpha} \sum_{i \neq i'} M_{\alpha i} M_{\alpha i'} = 0. \quad (22)$$

Also included is a term that demands that the fanout of *ina* links from high level to low-level frames equal the corresponding fanout of *INA* links:

$$\sum_{\alpha,i} M_{\alpha i} (\sum_{j} ina_{ij} - \sum_{\beta} INA_{\alpha\beta})^2 = 0. \quad (23)$$

As in Stickville, we used the Hopfield dynamics as an optimization technique. Unlike Stickville, the objective function is not quadratic but of order 5, because the rectangle rule for FLville is of order 5. Figure 11 shows results. Each dynamic variable is displayed as a circle. The large match matrix indicates matches between models and frames at the low level while the smaller one indicates matches between the high-level models and candidate frames. The *ina* matrix indicates matches between the six low-level frames and the three high-level frames. The slots of the three high-level frames are displayed in the 3x3 array of analog neurons. The slots of the low-level frames that represent the parameters for the input data were held fixed and are not displayed. For the match matrices, rows correspond to models and columns to frames.

In experiments to date, correct final states are indeed stable, but these correct states can be reliably reached from random start states only when a single high-level model is

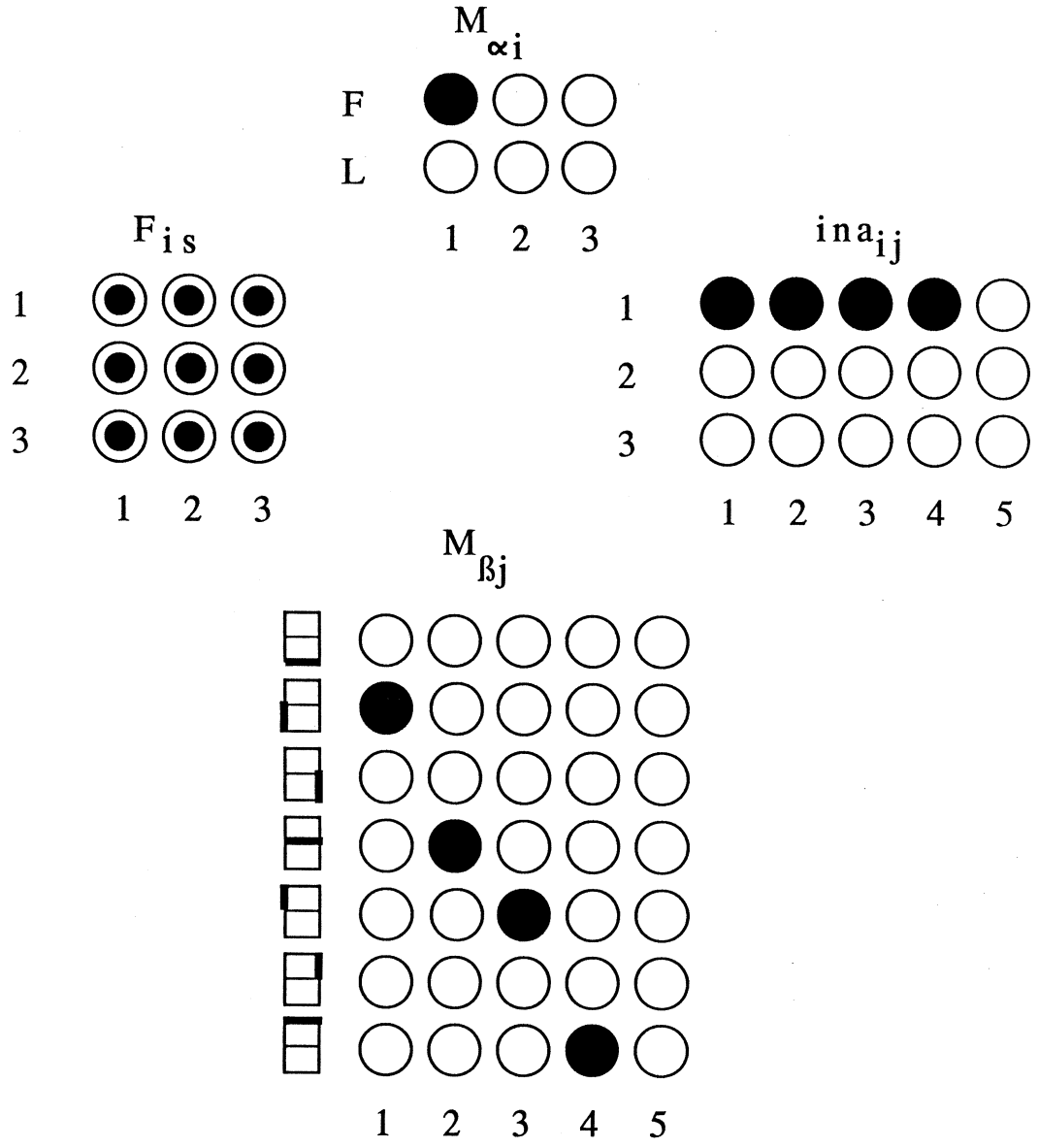


Figure 11: **Match network for FLville.** Each dynamic variable is displayed as a circle. The large match matrix indicates matches between models and frames at the low level while the smaller one indicates matches between the high level models and candidate frames. The *ina* matrix indicates matches between the six low-level frames and the three high-level frames. The slots of the three high-level frames are displayed in the 3x3 array of analog neurons. The slots of the low-level frames that represent the parameters for the input data were held fixed and are not displayed. For the match matrices, rows correspond to models and columns to frames.

present. Convergence to the correct state with two high-level models occurs less than half the time. The display shows a successful match for the input data consisting of the five dark sticks in Figure 9. Four of the five sticks are grouped as an "F". The analog values of the slots all stabilize at a value 0.5, the correct value in this case. (It turns out that slots of other unallocated frames reach 0.5 also, but this is a non-detrimental side effect.) Experimentation with FLville continues.

4.3 Line Grouping

Our third example is the problem of grouping line segments into lines, and further grouping such lines into longer lines, etc. This has generally been recognized as an important example of grouping in early vision, and grouping principles have been identified [Lowe-85, Boldt-87]. Algorithmic solutions to this problem also exist [Boldt-87]. Here we present the formulation of this problem in terms of optimization. We have not yet performed any experimental simulations on this problem, hence no results are available at present.

The set of models which are necessary to do line grouping and their relationships are shown in Figure 12. There are models representing lines, lines at various "levels", and what we call "right" and "left" lines at various levels. The line model represents a line in the image. The lines at different levels are defined recursively as follows: An L_0 line is a short line segment which is directly obtained from the input image via some edge-detection process. An L_l line is is composed of two lines at level $l-1$. The two lines at level $l-1$ which form parts of the line at level l are called "left" and "right" lines at level $l-1$ (LL_{l-1} and RL_{l-1}). The *ISA* and the *INA* relationships between the various line models are indicated in Figure 12.

Each line frame contains six two-dimensional vector-valued parameters and three scalars. Although two points are sufficient to completely specify the line, the use of the additional parameters allows us to simplify the parameter-check terms in the objective function. In the description of the parameters below, we use vectors $\mathbf{e}_i = (x_i, y_i)$ to represent locations of points on the line. The meaning of the parameters and the constraints between them are illustrated by the spring-stick model shown in Figure 13. Four of the vector-valued parameters correspond to the four points marked $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4$ on the higher level (longer) line in Figure 13. In addition, the end points $\mathbf{e}_3$ and $\mathbf{e}_4$ are also copied into slots $\mathbf{e}_L$ and

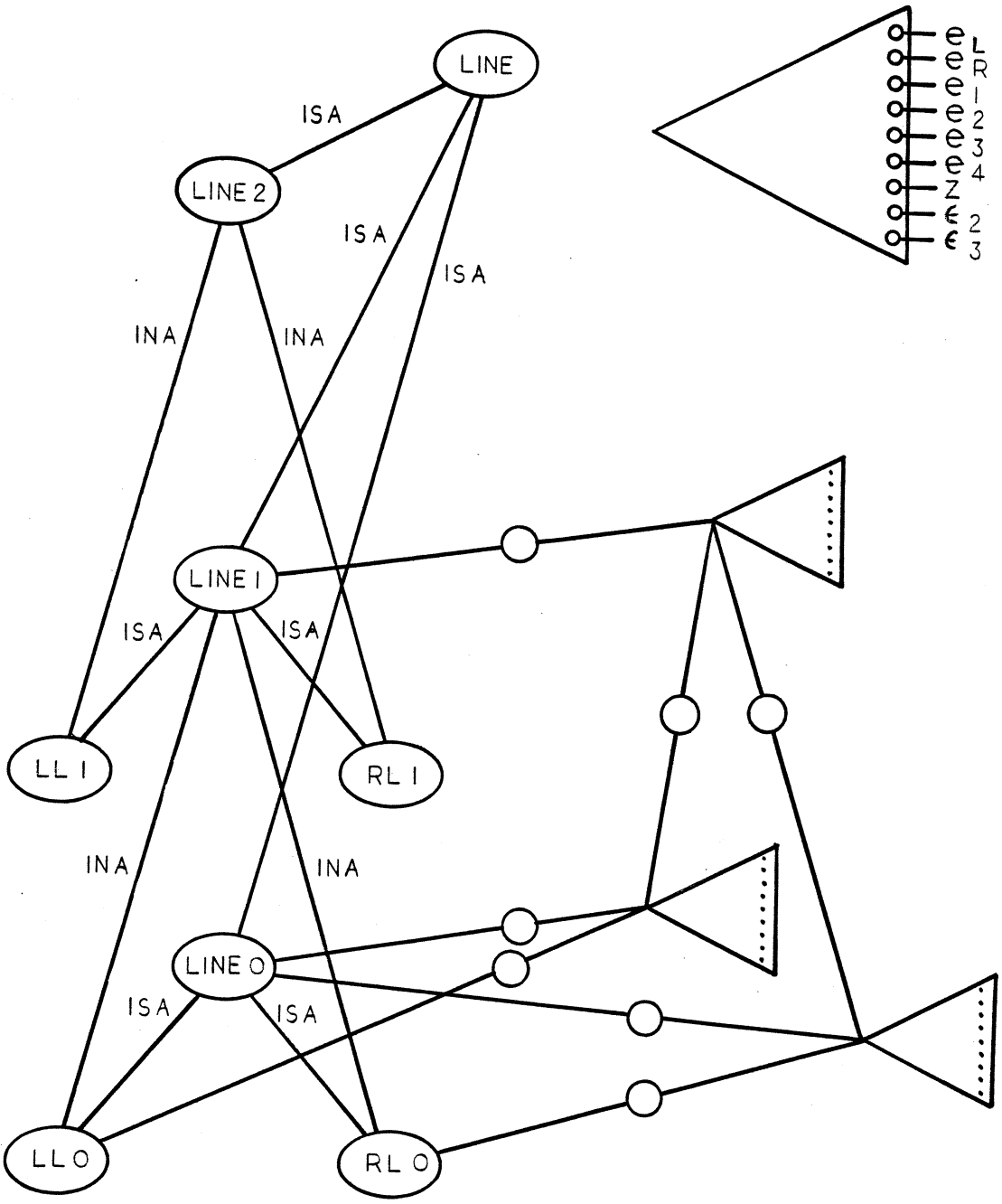


Figure 12: **The models for the line grouping problem.** These models include a line model, its specialization to three levels of a compositional hierarchy, and right and left versions of some of the line models. Each frame has slots for six two-dimensional vectors and three scalars, which are described in the text. The match and *ina* variables are represented by circles.

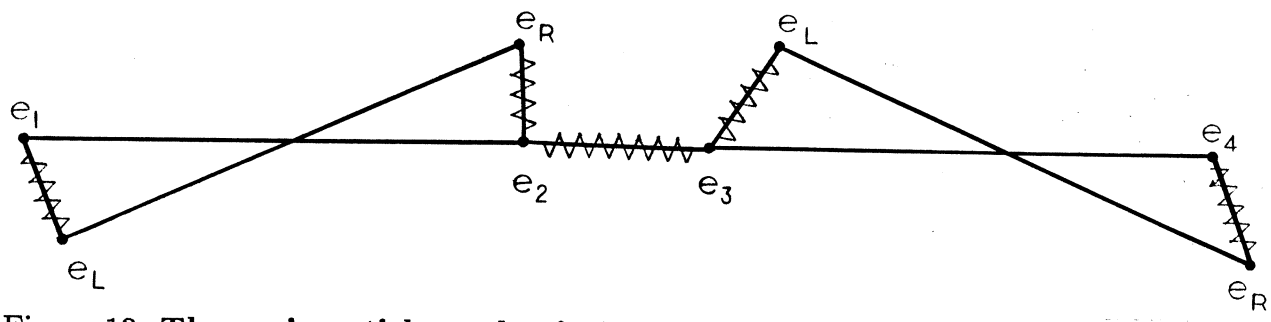


Figure 13: **The spring-stick mechanical model.** Each stick has two end-point vectors. In addition, the longer stick has two intermediate points constrained to lie on it. The outer end points of the small sticks attract the end points of the longer stick. The intermediate points on the longer stick move along that stick to be close to the inner end points of the smaller sticks, and close to each other. The resulting spring energies contribute to the objective function.

$\mathbf{e}_R$, although possibly in the reverse order. This redundant representation is useful to allow the same line-frame to match to either the LL model or the RL model as necessary, i.e., a particular line segment may dynamically group with another line segment on either side of it. The points $\mathbf{e}_3$ and $\mathbf{e}_4$ are required to lie on the line joining $\mathbf{e}_1$ and $\mathbf{e}_2$. This is specified by hard constraints (described below). The scalar parameters ϵ_2 and ϵ_3 measure the distance along line of the points $\mathbf{e}_3$ and $\mathbf{e}_4$ from the end point $\mathbf{e}_1$. Finally, the binary-valued parameter z allows $\mathbf{e}_L$ and $\mathbf{e}_R$ to switch between $\mathbf{e}_1$ and $\mathbf{e}_4$.

For simplicity, we have chosen to combine the rectangle rule with the associated sibling competition constraints as the following terms in the objective function

$$\sum_{i,l}(M_{L_l,i} - \sum_j ina_{ij} M_{LL_{l-1},i})^2 \tag{24}$$

$$\sum_{i,l}(M_{L_l,i} - \sum_j ina_{ij} M_{RL_{l-1},i})^2. \tag{25}$$

These constraints, together with the requirement that the dynamic variables M attain binary values, can be interpreted as "penalty" terms enforcing the following rule: If Frame i matches to a line at level l, then there should be just one Frame j which both matches to the left-line (right-line) model at level $l-1$ and is a part of Frame i. There are two types of specializations to consider: the specialization of the generic line model to the line

models at the various levels, and the specialization of the line model at level l to the left and right line models at the same level. These are also expressed in the form of penalty terms as

$$\sum_i (\sum_l M_{L,i} - M_{L_l,i})^2$$

$$\sum_{i,l} (M_{LL_l,i} + M_{RL_l,i} - M_{L_l,i})^2. \tag{26}$$

The parameter-checks correspond to the energies associated with the springs shown in Figure 13, gated by the appropriate M and ina variables. Specifically, these are

$$\sum_{l,i,j} a_l \; ina_{ij} M_{L_l,i} M_{LL_{l-1},j} (|\mathbf{e}_{1,i} - \mathbf{e}_{L,j}|^2 + |\mathbf{e}_{2,i} - \mathbf{e}_{R,j}|^2)$$

$$\sum_{l,i,j} a_l \; ina_{ij} M_{L_l,i} M_{RL_{l-1},j} (|\mathbf{e}_{3,i} - \mathbf{e}_{L,j}|^2 + |\mathbf{e}_{4,i} - \mathbf{e}_{R,j}|^2)$$

$$\sum_{l,i} b_l \; M_{L,i} (\epsilon_{2,i} - \epsilon_{3,i})^2 \tag{27}$$

where a_l and b_l are scale-dependent spring coefficients (they decrease as l increases, which means a greater tolerance is allowed).

As mentioned above the requirement that $\mathbf{e}_2$ and $\mathbf{e}_3$ should lie on the line joining $\mathbf{e}_1$ and $\mathbf{e}_4$ is expressed as a hard constraints. These are expressed by following equations which specify that the intermediate points should be an interpolation of the two end points.

$$\begin{aligned} \mathbf{e}_{2,i} - \mathbf{e}_{1,i} - \epsilon_{2,i}(\mathbf{e}_{4,1} - \mathbf{e}_{1,i}) &= 0 \\ \mathbf{e}_{3,i} - \mathbf{e}_{1,i} - \epsilon_{3,i}(\mathbf{e}_{4,1} - \mathbf{e}_{1,i}) &= 0 \end{aligned} \tag{28}$$

where ϵ_2 and ϵ_3 measure the distance of $\mathbf{e}_2$ and $\mathbf{e}_3$ from the end point $\mathbf{e}_1$. Similarly, the switching of $\mathbf{e}_L$ and $\mathbf{e}_R$ between $\mathbf{e}_1$ and $\mathbf{e}_4$ are expressed by the following equations.

$$\begin{aligned} \mathbf{e}_{L,i} - z_i \mathbf{e}_{1,i} - (1 - z_i)\mathbf{e}_{4,1} &= 0 \\ \mathbf{e}_{R,i} - (1 - z_i)\mathbf{e}_{1,i} - z_i \mathbf{e}_{4,1} &= 0 \\ z_i(1 - z_i) &= 0 \end{aligned} \tag{29}$$

where the binary-valued parameter z encodes the switching mechanism. Besides these, the usual constraints expressing the fact that M and ina are binary valued are also included.

Thus, we have cast the line grouping problem as a constrained optimization problem. We intend to use either Platt's [Platt-87,Arrow-58] method or a method similar to it to solve this problem.

5 Discussion and Conclusion

The design of the Frameville objective function and constraints raises difficult questions of efficiency and functionality, some of which are addressed here. Full answers must await analysis and extensive experimentation with many particular cases such those outlined in section 4.

Optimization is easier when an objective function is quadratic or "second order" in its independent variables, but our objective functions are third and higher orders. Evaluation of such objective functions or its derivatives is expensive unless the interaction terms are very sparse or highly structured. Often, the high-order objective function may have a structure which allows it to be reduced to a lower order function with a reasonable number of additional dynamical variables. The expense also crucially depends on the sparseness of the dynamic M and ina matrices. If one enforces sparseness constraints during minimization, as well as at the final minimum, then large savings may result from using "virtual" matrix element variables which are created and destroyed dynamically. Such virtual neurons may be governed by objective functions and are related to "windows" of attention [Mjolsness-87].

Order reduction can also be achieved by setting subsets dynamical variables to constant values and simplifying the objective functions accordingly. Large gains in efficiency can result, at the expense of computational flexibility. For example, a frame F_i, whose match neurons $M_{\alpha i}$ are all fixed at definite binary values, is said to be "pre-typed". Likewise if its ina_{ij} variables are fixed then the frame is is "pre-allocated" to a particular position in the compositional hierarchy. Constant M and ina matrices may express efficient data structures for the recognition problem at hand.

Turning to questions of functionality one might wonder whether Frameville-style objective functions are restricted to high-level vision problems. But the inclusion of analog

arithmetic for frame parameters means that one can construct low-level models such as fixed convolution filters, whose kernel is part of the model-specific objective function. Pre-typed and preallocated pixel-frames and filter-frames may allow efficiency to approach that of an ordinary convolution. The Frameville version, however, could be integrated with intermediate-level vision.

It was noted in section 1 that various optimization techniques (e.g., regularization, relaxation labeling, graph matching) have been used to solve problems at single levels of visual abstraction, and it was argued that there should be cooperation between the processes operating at different levels. We have described how to express examples of high-level model matching problems (Stickville and FLville) and intermediate-level grouping problems (line grouping) uniformly in Frameville. Also, we have discussed the possibility of a Frameville version of low-level feature detection. The uniformity with which we have expressed these problems at different levels of abstraction immediately suggests a technique for integrating diverse visual processes. For instance, FLville and line grouping could be integrated simply by identifying the high-level line models as the low-level stick models in FLville. With minor modifications the objective functions for FLville and line grouping can simply be added.

Finally, the optimization paradigm within which we work may itself be questioned, especially when the objective functions are of high order in which case there will generally be many local minima. Of course our objective functions are not "general"; rather, the model-based objective functions are hand-designed and hand-evolved to favor unique correct image interpretations whenever the input data permits. It is not possible, yet, to say whether or not good design disciplines exist for the model base and its objective function which avoid introducing spurious local minima.

In defense of objective functions, one may also say that they form a notation, bordering on a language, of remarkable conciseness and perspicuity for many computations involving mixtures of quantitative and symbolic processing.

Acknowledgements

We wish to thank Joachim Utans for the Frameville simulations and for help with figures, and Tony Zador for the Stickville simulations. We had helpful discussions with Don Delorie,

Art Gmitro, Geoff Hinton, Stan Letovsky, Marty Tenenbaum, and members of the UMass VISIONS project. Thanks are also due to Jim Duncan, Grant Shumaker and Volker Tresp for help in preparing this paper.

References

[Anandan-88] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *International Journal on Computer Vision*, 1988, to appear.

[Arrow-58] K. J. Arrow, L. Hurwicz, and H. Uzawa, editors, *Studies in Linear and Nonlinear Programming*, Stanford University Press, 1958.

[Ballard-82] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice Hall, 1982.

[Ballard-86] D. Ballard, "Cortical connections and parallel processing: structure and function," *Behavioral and Brain Sciences*, vol 9, pp. 67–120, 1986.

[Barrow-71] H. G. Barrow and R. J. Popplestone, "Relational descriptions in picture processing," in D. Mitchie, editor, *Machine Intelligence 6*, Edinborough University Press, 1971.

[Barrow-76] H. G. Barrow and J. M. Tenenbaum, *MSYS: A System for Reasoning About Scenes*, Technical Note, SRI – AI Center, 1976.

[Boldt-87] M. Boldt and R. Weiss, "Token-Based Extraction of Straight Lines," COINS Technical Report, University of Massachussetts, 1987.

[Cooper-87] P. R. Cooper and S. C. Hollbach, "Parallel recognition of objects comprised of pure structure," in *Proc. of DARPA IU Workshop*, Morgan-Kaufmann, 1987.

[Cooper-88] P. R. Cooper, Structure recognition by connectionist relaxation: formal analysis, In *Proc. of DARPA IU Workshop*, pages 981–993, Morgan-Kaufmann, 1988.

[Davis-79] L. Davis, "Shape matching using relaxation techniques," *IEEE Transactions on PAMI*, vol. 1, pp. 60-72.

[Fahlman-79] S. E. Fahlman. *NETL: A System for Representing and Using Real-World Knowledge.* MIT Press, 1979.

[Faugeras-81] O. Faugeras and M. Berthod, "Improving consistency and reducing ambiguity in stochastic labeling: an optimization approach," *IEEE Transactions on PAMI*, vol. PAMI-3, pp. 412-424, 1981.

[Feldman-88] J. A. Feldman, M. A. Fanty, and N. H. Goddard, "Computing with structured neural networks," *IEEE Computer*, pp. 91-103, 1988.

[Fischler-73] M. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computers*, vol. 22, pp. 67–92, 1973.

[Geman-84] S Geman and D Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on PAMI*, vol. PAMI-6, pp. 721–741, 1984.

[Gindi-86] G. R. Gindi and D. H. Delorie, Jr., "Indexing shapes for recognition," In *Proc of the SPIE: Fifth International Conference on Robot Vision and Sensory Controls*, vol. 729, pp. 28-35, 1986.

[Grimson-85] W. E. L. Grimson, "Computational experiments with a feature based stereo algorithm," *IEEE Transacations on PAMI*, vol. PAMI-7, pp. 17–34, 1985.

[Hanson-86] A. R. Hanson and E. M. Riseman, "A methodology for the development of general knowledge-based vision systems," in M. A. Arbib

and A. R. Hanson, editors, *Vision, Brain, and Cooperative Computation*, MIT Press, 1986.

[Hopfield-84a] J. J. Hopfield, *Personal communication*, October 1984.

[Hopfield-84b] J. J. Hopfield. "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Sciences USA*, vol. 81, pp. 3088–3092, 1984.

[Hopfield-85] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.

[Horn-81] B. K. P. Horn and B. W. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[Horn-86] B. K. P. Horn, *Robot Vision*, MIT Press, 1986.

[Hummel-83] R. A. Hummel and S. W. Zucker. "On the foundations of relaxation labeling processes," *IEEE Transactions on PAMI*, vol. PAMI-5, pp. 267–287, May 1983.

[Lowe-85] David G Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, 1985.

[Luenberger-84] David G Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, 1984.

[Minsky-75] Marvin L Minsky, "A Framework for Representing Knowledge," in P. H. Winston, editor, *The Psychology of Computer Vision*, McGraw-Hill, 1975.

[Mjolsness-87] E. Mjolsness, "Control of attention in neural networks," in *Proc. of First International Conference on Neural Networks*, pages 567–574, IEEE, 1987.

[Platt-87] John Platt and Al Barr. "Constrained differential optimization," *Proc. of Neural Inforemation Processing Systems Conference*, 1987.

[Poggio-85] T. Poggio and C. Koch, "Ill-posed problems in early vision," *Proceedings of the Royal Society of London B*, vol. 226, pp. 303-323, 1985.

[Rosenfeld-76] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labeling by relaxation operations," *IEEE Transactions on SMC*, vol. SMC-6, pp. 420-433, 1976.

[Terzopoulos-86] D. Terzopoulos, "Regularization of inverse problems involving discontinuities," *IEEE Transactions on PAMI*, vol. PAMI-8, pp. 413–424, 1986.

[von der Malsburg-86] C. von der Malsburg and E. Bienenstock, *Statistical Coding and Short-Term Synaptic Plasticity: A Scheme for Knowledge Representation in the Brain*, pp. 247–252. Springer-Verlag, 1986.

[von der Malsburg-88] C. von der Malsburg, "Pattern recognition by labeled graph matching," *Neural Networks*, vol. 1, pp.141–148, 1988.

[Waltz-75] D. Waltz, "Generating semantic descriptions from drawings of scenes with shadows," in P H Winston, editor, *The Psychology of Computer Vision*, McGraw-Hill, 1975.