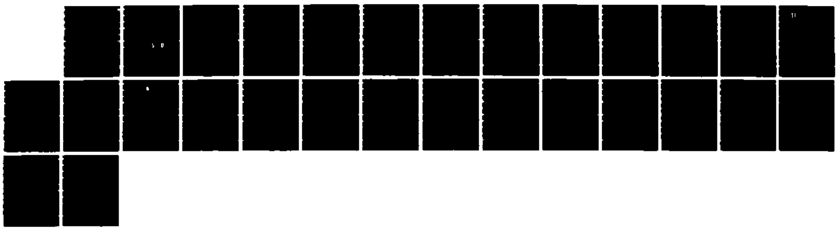
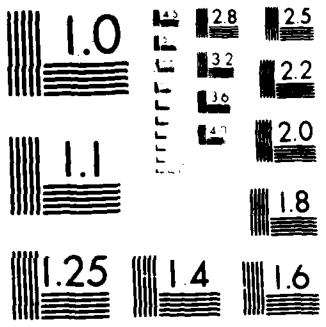
AD-A179 408   TINKER TOY RECOGNITION FROM 2D CONNECTIVITY(U)   1/1
ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE S HOLLBACH
OCT 86 TR-196 N00014-84-K-0655
UNCLASSIFIED   F/G 9/4   NL




1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

(12)

AD-A179 408

Tinker Toy Recognition from 2D Connectivity

Susan Hollbach
Department of Computer Science
The University of Rochester
Rochester, New York 14627

TR 196
October 1986

rochester

Department of Computer Science
University of Rochester
Rochester, New York 14627

87 4 21 267

12

Tinker Toy Recognition from 2D Connectivity

Susan Hollbach
Department of Computer Science
The University of Rochester
Rochester, New York 14627

TR 196
October 1986

Abstract

An end-to-end system has been built as a preliminary investigation into structure based object recognition in the tinker toy world. The system successfully perfoms constant time topological matching of TV input to a multiple-object model base.

This work was supported in part by NIH Public Health Service research grant number 1 R01 NS22407-01 and in part by ONR research contract number N00014-84-K-0655.

0. Introduction

Object recognition can occur without reconstruction of the component surfaces, as evidenced by the recognizability of line drawings and silhouettes. We are investigating the use of object structure as a recognition cue [Lowe, Coooper-86a]. For example, many people will first view Figure 1 as an arbitrary collection of line segments. But once having been told that this is a sketch of a bicycle, the structural information attached to bicycles is sufficient to make this a recognizable rendition.

Having accepted the premise that object recognition is based on structural cues, the need arises for a more precise definition of the term "structure". For the purposes of this document, we define object structure to be the spatial relations between the primitive components making up the object. The need also arises for a domain in which the primitive components of an object are easily extracted. One such domain is the tinker toy world. Tinker toys consist of wheels connected by rods that fit into the eight radial holes and two axial holes in the wheels (see Figure 2). In analysing pictures of tinker toys, it is only necessary to locate the position and

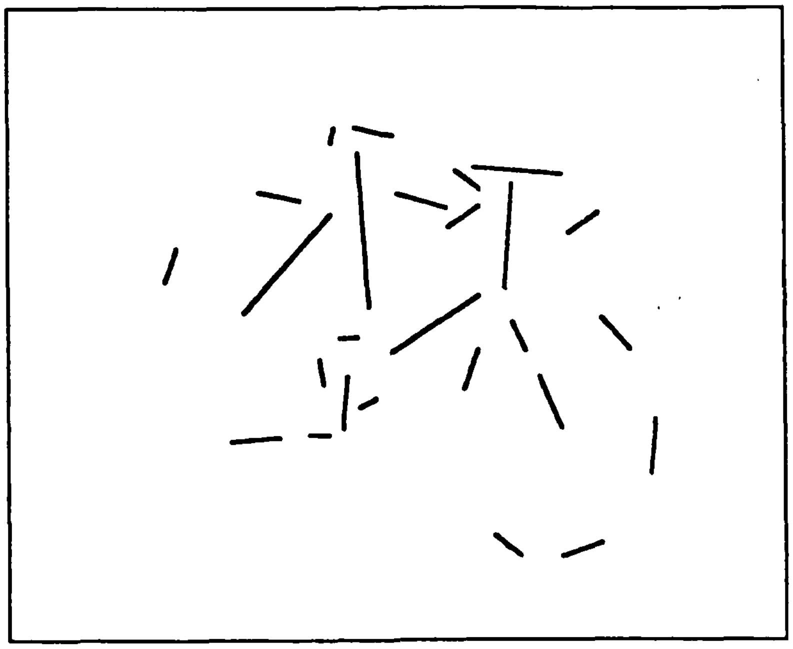


Figure 1: a bicycle (from D. Lowe)

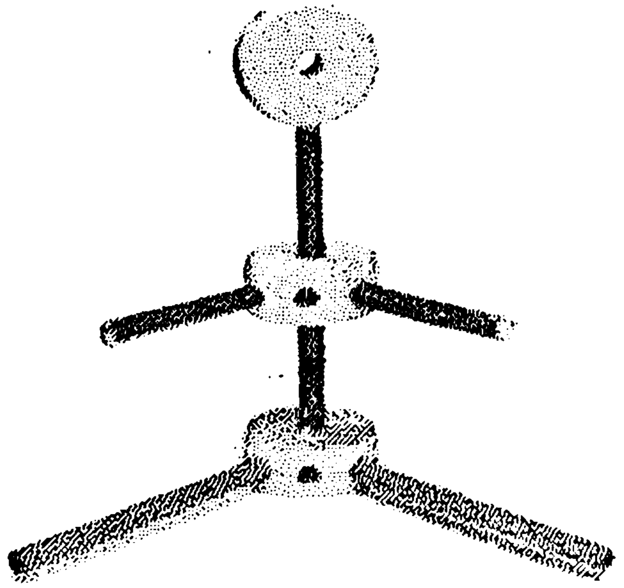


Figure 2: a tinker toy

connectivity of each primitive. There is no surface reconstruction, since primitives can be distinguished from one another on the basis of purely structural cues (see Figure 3).

This paper describes an end-to-end system that takes a picture of a tinker toy, performs a structural analysis of the picture and matches the resulting structure to a model base of tinker toy principal views (see Figure 4). The principal views are determined by noting that for a fixed camera tilt, the topology of the figure remains invariant under rotation over a range of camera angles. Thus for each figure there is a reasonably small set of topologies which, taken together, constitute the principal views of a figure. (Reasonaly small means fewer than 10). These topologies are determined empirically, then added by hand to the model base. A match to one of the principal views can be taken as evidenc for a match to the model. The matching procedure is based on the topology of connectivity, not the geometry. This means

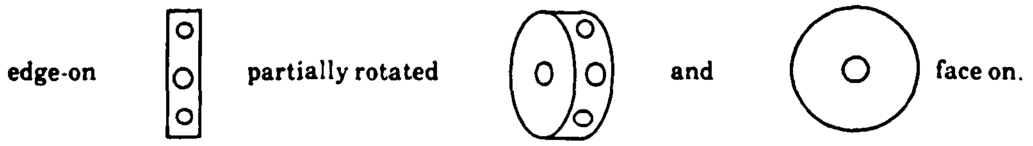


Figure 3: the three principal views of a wheel

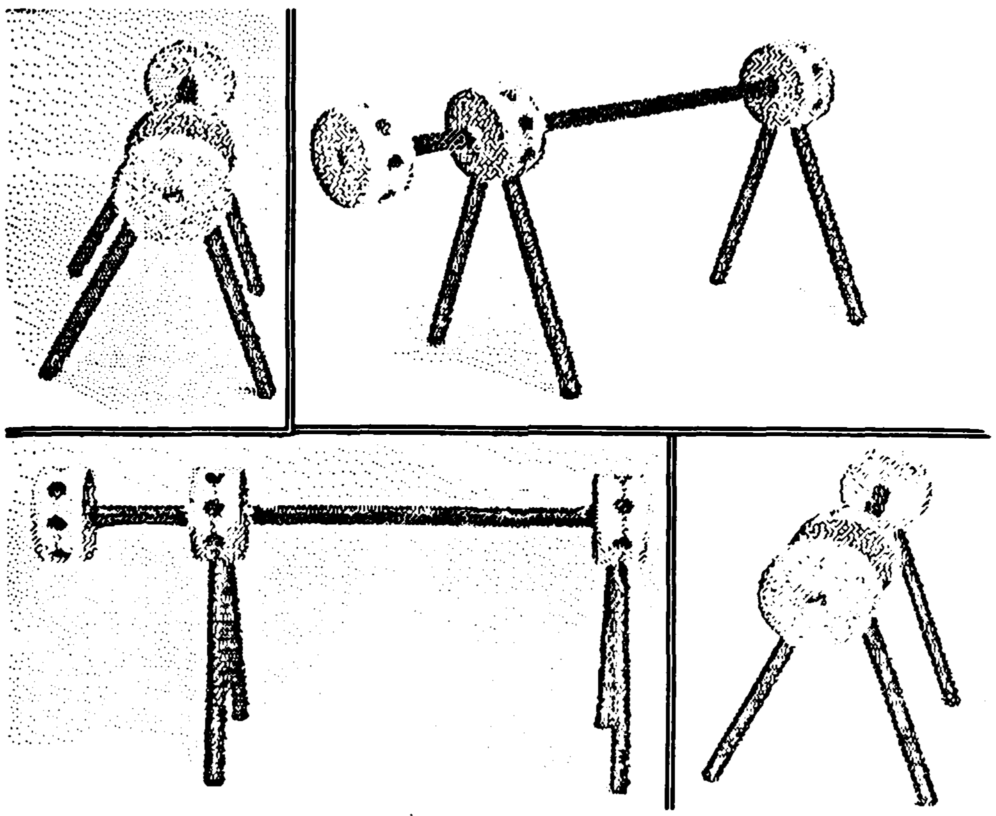


Figure 4: four principal views of a horse

that it is possible that a topological match to a given principal view could be taken as evidence for a match to more than one model, since two figures that are geometrically quite dissimilar may match topologically, as demonstrated in Figure 5. Clearly, the work described in this document constitutes little more than a preliminary investigation into the problem. For a detailed proposal on how to extend the work presented here, see [Coooper-86a].

The structure of the system is summarized in Figure 6. The system consists of four stages: 2D primitive acquisition, 3D primitive acquisition, instance building and model invocation. The 2D primitive acquisition stage yields circles and line segments from a greylevel image of a tinker toy. The 3D primitive acquisition stage

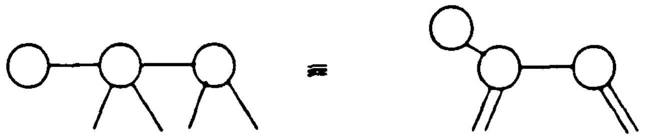


Figure 5: topologically indistinguishable figures

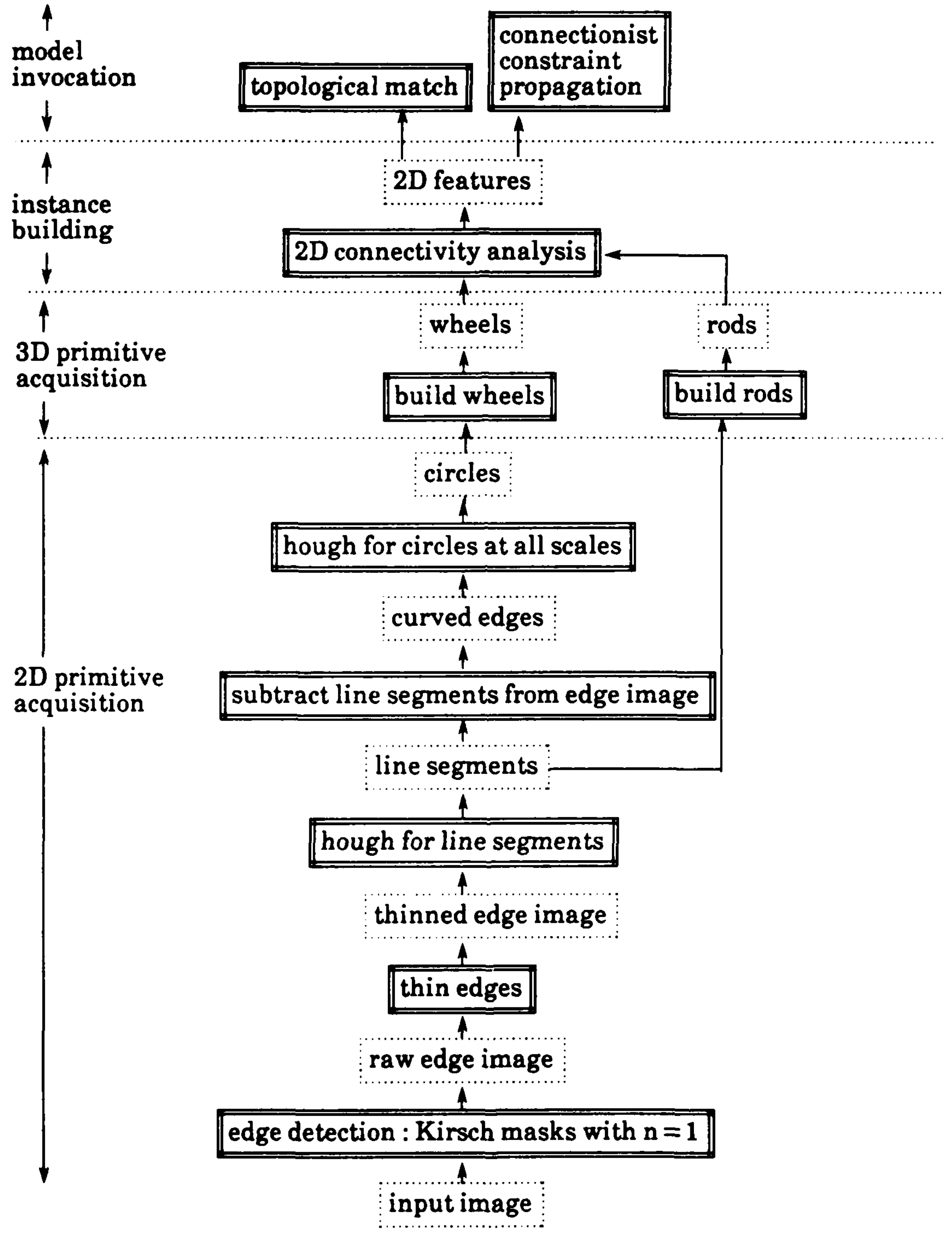


Figure 6: Summary of system structure

groups the line segments into rods and the circles into wheels. A tinker toy instance is then built from these primitives by establishing the topological connectivities between the rods and wheels. This instance is ultimately matched to the model base in one of two ways, one using backtracking search, the other using parallel constraint propagation. The rest of this paper documents the operation of each stage in the system.

1. 2D Primitive Acquisition

1.1 The Kirsch Edge Operator

The first step in the process of 2-D feature finding is to apply an edge operator to the image, resulting in magnitude (strength of response) and gradient (direction of steepest "slope", or direction of maximum intensity change). The edge operator chosen is the kirsch operator, consisting of four masks.

The response of this edge operator is not optimal, in that it produces multiple responses per edge. For an ideal step edge, the operator produces two identical responses, because of the strip of 0's down the center of each mask. For an ideal ramp edge, the operator produces a whole series of identical responses, as the edge operator "travels" up the slope. Fortunately, however, real world image edges tend to have a point of steepest gradient corresponding to the point of highest contrast of the edge, in which case the operator will produce a maximum value at this point.

1.2 Edge Thinning

The tendency of real images to have a point of steepest gradient led to the development of a simple edge thinning algorithm, that edits the magnitude image for multiple responses to an edge, by suppressing all but the local maximum in the direction of the gradient and bolstering the confidence of edges with similar neighbours in the dirction perpendicular to the gradient. The pseudo-code algorithm for this process appears in Appendix A.

1.3 Finding Line Segments

The thinned edges are then fed into a line segment finder. The algorithm used is a variant on the canonical Hough technique to find straight lines [Ballard&Brown].

The usual approach is first to obtain the gradient magnitude and direction information for each point in the image. This amounts to tabulating the values for two functions, *mag*(e) and *dir*(e), the gradient magnitude and direction at point e. Each edge point e in the image such that *mag*(e) is greater than some global threshold then casts its vote for the existence of the straight line passing through that point with slope corresponding to *dir*(e).

One problem with the basic scheme is that the votes cast are for parameterized, infinite straight lines, with no notion of end points. The endpoints of the image line segment could always be found through back projection and search. But it seems simpler to adopt a more sophisticated voting strategy, one that will not only obviate the need to go back to the image to search for endpoints, but is also insensitive to the quantization of the gradient direction. This strategy is an extension of pairwise voting [Ballard-86]. In pairwise voting, pairs of edge points vote for the line passing through them both, on the condition that $dir(e_1) = dir(e_2)$ and the slope of the line passing through both points falls within the range of this gradient direction. The main problem encountered in this scheme is the crosstalk that occurs between edge points belonging to neighbouring parallel edges. The current quantization of the gradient is too coarse to filter out these erroneous matches successfully. The problem of imprecise gradient direction information is skirted by using collinear triples instead of pairs as evidence for the existence of a line segment. Any arbitrary pair of points defines a straight line; the presence of a third collinear point, however, effectively rules out the possibility of cross talk (assuming reasonably well-behaved data, eg. sharp, high contrast edges). The collinearity requirement entails a certain amount of local search for corroborative evidence prior to casting a vote. For each significant edge point, a small area is searched in the two directions orthogonal to the gradient direction at that point. If the search turns up significant edge points of similar gradients in both directions, a vote is cast for the line segment passing through these three points. If a vote has already been cast in the target bin for (what will necessarily be a different) line segment, the four endpoints are sorted by their x component if the slope of the line segment is less than 1, by the y components otherwise, and the maximum and minimum are retained as the new endpoints (see Figure 7).

Once all the voting for line segments is over, there are three post-processing stages. The first compares each line segment found to all others and merges those

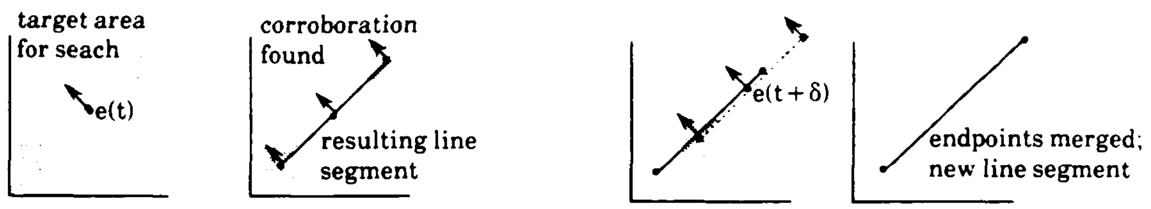


Figure 7: search strategy for line evidence

that are approximately collinear and that have the same gradient, provided that the end points are interlaced. The merging process repeats until no further merges are possible. The second stage systematically perturbs the endpoints of each line segment and back projects against the image to find the best global fit, to correct for the inaccuracy introduced by curvature at the ends of line segments. The third stage checks each segment against the input image for breaks, creating new line segment descriptions as required. This corrects for the inadvertent merging of collinear but distinct line segments.

The edge image is then edited to remove the edges corresponding to the line segments found. This step is designed to reduce the search space for the next stage.

1.4 Finding Circles

The significant edge elements remaining in the edited image mostly belong to 2D projections of wheels, shown in Figure 3.

The method of disk detection currently being used is to first hough for circles, then create groupings of circles by mutual intersection. This is based on the empirical observation that an unoccluded view of a wheel will generally be characterized by a tightly knit cluster of mutually intersecting circles, so the center of a pattern of circular activity is a reasonable approximation to the position of a disk. The center is found by drawing a bounding box around the pattern of mutually intersecting circles and taking the inscribed circle as the approximate location of the disk. In cases of occlusion only one disk is reported when there are in fact two. This inaccuracy is corrected for by incorporating the evidence for a single wheel directly into the appropriate principal view of the model.

2. 3D Primitive Acquisition

Stage 2 involves creating a set of pseudo-3D features, called wheels and rods, from the input set of 2D structures, line segments and disks.

2.1 Data structures

The tinker toy analog of a winged-edge representation [Ballard & Brown] has been adopted as the most flexible and complete data structure for the connectivity information. A wheel includes 10 pointers to rods, and a rod 2 pointers to wheels. A rod is defined by a pair of cartesian coordinate pairs (the endpoints), and the width. The endpoints are distinguished into start point and end point, where the start point is the 'leftmost' (i.e. smaller x-ordinate), except for vertical lines, where it is the 'lowest' (i.e. smaller y-coordinate). This means that the angle subtended from the x axis by the rod will have the range $-90° < \alpha <= 90°$ (see Fig. 8). Wheels are equally simple, consisting of width, height, a pair specifying the center, and ten pointers to rods. All the wheels and rods found are stored in two statically allocated arrays, and the array indices serve as symbolic pointers to items in the tables. The relevant pseudo-code definitions are included as Appendix B. The semantics of rod ends are given in Figure 8; although the semantics of the wheel slot orderings has not as yet been worked out, it is immaterial for topological matching.

2.2 Geometry of rod building

The rods are put together by considering each pair of straight lines in the image. If the gradients are not exact opposites, the possibility of a match is ruled out immediately. If they are opposites, they must also be consistent with the information about the relative contrast in the image. A histogram of the original image will indicate whether the background is light or dark; the rods are assumed to

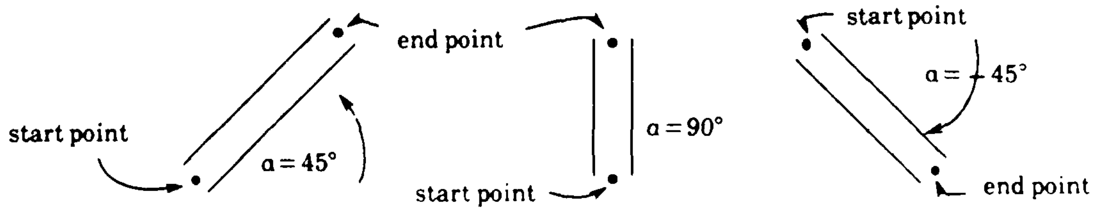


Figure 8: examples of rods at various orientations

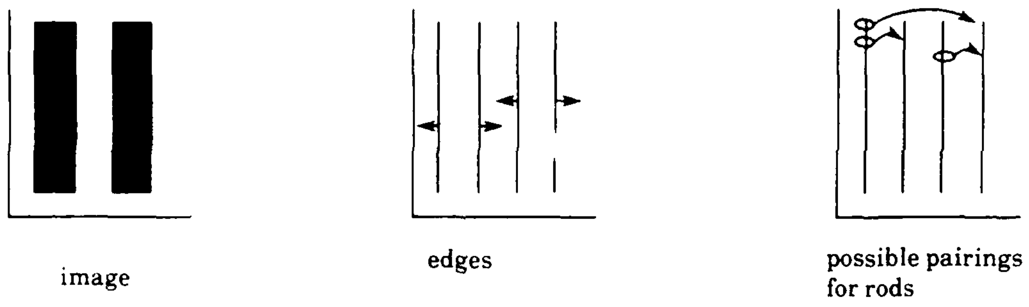


Figure 9

be of the contrasting colour. Thus in an image with a white background, a pair of lines whose gradients "point towards" each other (see Figure 9) will fail the initial match test.

The geometry of the test is depicted in Figure 10. Given a pair of parallel line segments **ab** and **cd**, with associated gradients, define a line $l_1 = \mathbf{a} + s\mathbf{p}$, where **a** is one endpoint of the first line segment and **p** is the vector associated with that line segment's gradient. In figure 10, for example, $\mathbf{p} = (-1, 0)$, since the gradient of line segment **ab** points west. Also define line $l_2 = \mathbf{c} + t[\mathbf{d}-\mathbf{c}]$, which is simply the parametric line passing through the line segment **cd**. The point of intersection of these two lines can be found by setting $l_1 = l_2$ and solving for the values of parameters s and t:

$$l_1 = \mathbf{a} + s\mathbf{p}$$
$$l_2 = \mathbf{c} + t[\mathbf{d} - \mathbf{c}]$$
$$l_1 = l_2 \Rightarrow \mathbf{a} + s\mathbf{p} = \mathbf{c} + t[\mathbf{d}-\mathbf{c}]$$
$$\Rightarrow s\mathbf{p} - t[\mathbf{d}-\mathbf{c}] = [\mathbf{c}-\mathbf{a}]$$

$$\Rightarrow \begin{bmatrix} p.x & c.x-d.x \\ p.y & c.y-d.y \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} c.x-a.x \\ c.y-a.y \end{bmatrix}$$

which is in the familiar and easily solvable form $\mathbf{Mx} = \mathbf{b}$. Back substitution of these values into their respective line equations would give the coordinates of the point of intersection; however, the information we need is simply the sign of the parameter s. If s is positive, then the gradients of the two line segments point toward each other, and if negative, they point away from each other.

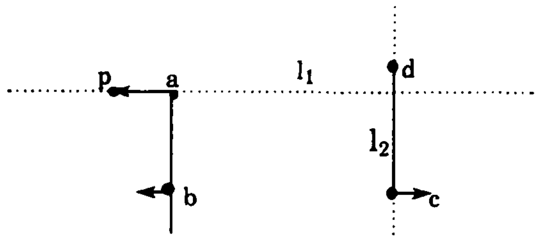


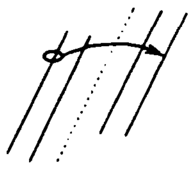
Figure 10

The rod-building process uses the following algorithm:

1) initial match -- a filter to create the initial set of possible pairings (described above in detail).
2) Make all the unique pairings, where both lines can only match with each other.
3) Analyse these groupings, taking the mode as the true rod width.
4) Throw away any "rods" not having about this width (this disposes of highlights and of parallel rods) and use this information to further restrict the possibilities list.
5) Handle collinear rods by requiring that the line perpendicular to the rod and passing through the midpoint of one line segment intersect the other. (see Fig. 11) The geometry of this test is similar to that of the *initial match test*.

2.3 Geometry of Wheel Building

Ideally, at this low level one should keep as much of the image information as possible, since early approximations can introduce later errors. For example, it would be nice to be able to distinguish the three principal views of a wheel, depicted in Figure 2. In practice, however, the disks found by grouping mutually intersecting circles suffice to determine 2D connectivity, the next stage in the process.


virtual rod discarded in step 4

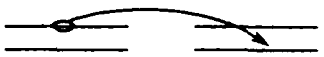

false match discarded in step 5

Figure 11

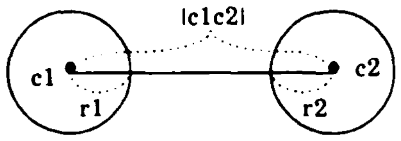


Figure 12

The geometry of this task is extremely simple. Measure the length of the line segment connecting the centers of the two circles being compared. If this length exceeds the sum of the two radii, the circles do not intersect (see Figure 12).

3. Instance Building

3.1 Connectivity Analysis

Having grouped the edges into rods and the circles into wheels, the next step is to establish the connectivity between these two sets of features. The obvious approach is to examine each of the $n \times m$ possible connections between the n wheels and m rods, using intersection as the connectivity metric. Unfortunately there is no gurarantee that a rod will actually intersect the wheel it is attached to; see Figure 13. In order to circumvent this problem, the image space is partitioned into regions that correspond to the presence of distinct features. Since the feature space is quite sparse when compared to the image space, a quad tree type structure was chosen.

The first stage is to build a quad tree for the image from the wheels found by the circle grouper. The goal of this stage is to classify coarsely the various regions in the image according to the (unique) disk occupying that region (see Figure 14). The rods are then added to the appropriate cells, where a rod is deemed to belong to a cell if one of its endpoints lies in the cell.

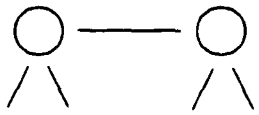


Figure 13: a disjoint but 'connected' figure

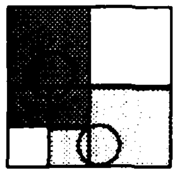


Figure 14

Finally, the quad tree is traversed, and the connectivity information transferred to the data structures of appendix B.

4.0 Model Invocation

4.1 Topological Matching

The task of model recognition can be made arbitrarily complex, depending on the goals of the matching process. If the model base is viewed as defining structural equivalence classes, however, the task simplifies to topological matching, which can be quickly and elegantly performed using recursive backtracking. The idea is to use the system stack to record tentative match assignments, which are kept only if the recursive call to solve the rest of the problem succeeds. The result of this procedure is a unique one-to-one correspondence mapping primitives of the input to primitives in the matching model. Ties are broken arbitrarily, leaving no indication of the existence of multiple choices for a match. Furthermore, there is no notion of scoring possible near misses to select the best of the alternatives; only exact matches are reported.

4.2 Constraint Propagation

A considerably more interesting solution to the problem has been developed on the Rochester Connectionist Simulator [Fanty-86]. There are two objects available to the programmer, units (neurons) and links (synapses). For an overview of the philosophy and methods of connectionist models, see [Feldman-82].

The design of the connectionist version of the model invocation task was originally outlined in [Goddard-86], with a clarification in [Cooper-86a]. It is slightly weaker (and hence more flexible) than the topological matching procedure outlined above.

The basic idea is to match the input to all models in parallel, taking the highest scoring match to be the winner. Separate units are created for each possible rod-rod, wheel-wheel and joint-joint matching between the figure and each model in the model base. The group that reports the highest activation is taken to be the winner.

The matching criterion for the wheel-wheel and rod-rod matches is the degree of connectivity of the primitive: for wheels, this is the number of attached rods (vice versa for rods). That is, a wheel with four attached rods will match any and all model wheels with four attached rods. The effect of taking the cumulative score of such a match is essentially to count the number of these unary constraint matchings between figure and model. A joint-joint match is more complex, incorporating the connectivity not only of the joint wheel, but also the connectivity of the wheel at the other end of the joint rod. The values of these matches are not factored into the cumulative score, but rather are used to edit the unary constraint matches.

The actual network consists conceptually of k subnets, each of which compares the input to a different model in the model base (*for a model base of size* k). The cumulative results of these comparisons are examined by the "grandmother unit" for the network, and the model whose subnet scored highest is declared to be the best match. For a graphical depiction of the network structure, see Figure 15.

Each tinker toy, whether model or input, is expressed uniformly as *n* wheel units, *m* rod units where *n* and *m* are respectively the (variable) number of wheels and rods of that toy, and $n \times m$ binary constraint units.

The structure of each of the k subnets consists of three square arrays of 'binder' units, one for rods, one for wheels, and one for the binary constraints. Each entry in an array (binder unit) corresponds to a different component or constraint matching. Inputs to binder unit u_{ij} come from the i^{th} figure component and the j^{th} model conponent. The semantics of this arrangement is clear for the rod and wheel binders: the i^{th} rod/wheel in the figure matches to the j^{th} of its counterparts in the model. The semantics of a constraint-constraint (joint-joint) match differs slightly, since they exercise an editorial influence over the rod and wheel binders. Each constraint binder unit is connected to all the component binder units it could possibly influence.

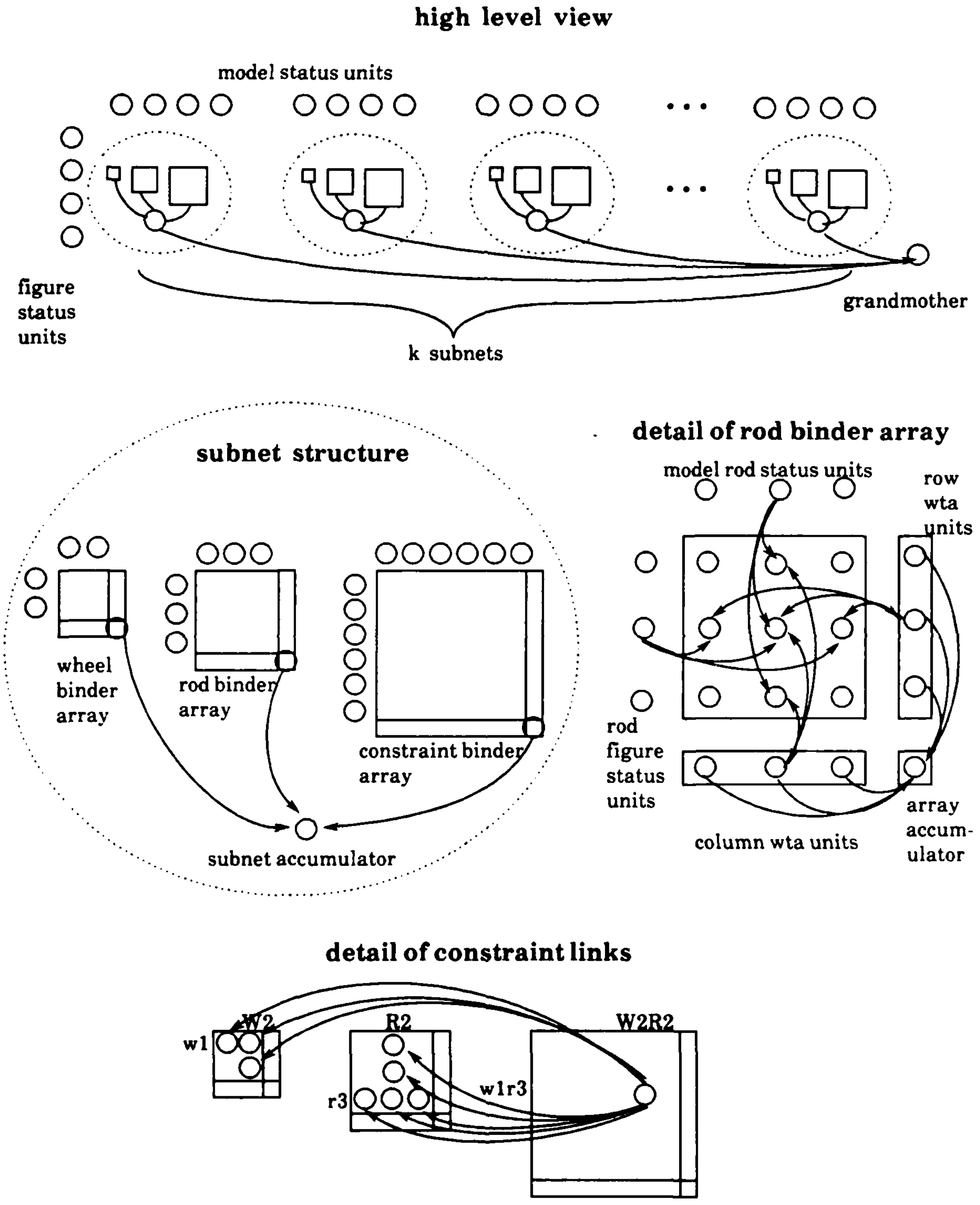


Figure 15: pictorial structure of connectionist network

For example, constraint binder w1r3/W2R2 corresponds to the match between the wheel 1 – rod 3 joint in the input figure with the Wheel 2 – Rod 2 joint in the model. This binder unit is connected to every unit in the w1 row and the W2 column of the weeel binder array, and to every unit in the r3 row and R2 column in the rod binder array. Thus if w1 seems to match W2 on the basis of component infomation, but for no *i* or *j* is the w1r*i*/W2R*j* constraint binder on, this match must be discarded.

The nodes come in five flavours: status, binder, winner take all (wta), accumulation, and decision. Status units represent the information encoded in the tinker toy and serve as inputs to the matching units. Wheel and rod status units encode the connectivity information of each component. Thus activation levels for wheel status units can range from 1 to 10, whereas activation levels for rod status units is limited to the values 1 or 2, since there are only two types of rod, dangling and connecting. There are also status units for binary constraints. This amounts to creating virtual feature nodes at every rod-wheel junction. The activation of these feature nodes is a function of the connectivity of the wheel at the junction and the wheel at the other end of the rod at the junction. The function is a simple one: for the junction of wheel *i* and rod *j*, $C_{ij}(t+1) = W_i(t) + 10 \times W_{R_j}(t)$, that is, the activation (connectivity) of wheel *i* plus 10 times the activation (connectivity) of the wheel attached to the other end of rod *j*. If there is no other wheel attached to rod *j*, i.e. if it is a dangling, rod, $W_{R_j}(t) = 0$, so $C_{ij}(t+1) = W_i(t)$.

In addition to rod, wheel and binary constraint status units, there are also $k + 1$ special purpose status units to count the number of wheels in each tinker toy. Since wheels are considered to be structurally more significant than rods, matches between figure and model having different numbers of wheels are discouraged by allowing the descrepancy to exert a negative influence on the final score for the match.

Binder units are the basic matching units of the network. The state of a binder unit reflects whether of not the inputs from figure and model are equal, that is, if the two components have similar connectivities. There is a separate binder unit to represent each possible match between components in the figure and components in the model base. That is, for a figure of *n* wheels and a model base totalling *m* wheels, there will be $n \times m$ wheel binder units. The constraint binder units encode the possible matches between binary constraints. These are the most numerous of the units: for a figure with *n* wheel and *m* rods, there are $(n \times m) \times (n \times m)$ constraint binder units. A constraint binder encodes whether or not wheel i being connected to rod j in the figure is compatible with wheel k being connected to rod l in the model. This information is fed back to the wheel i-k and rod j-l binder units.

The general formula for updating the activation of binder units is quite complex. Rod and wheel binder units are *on* if their model and figure inputs are equal, and if no evidence to the contrary comes in from the constraint binders. Evidence to the contrary is supplied when fewer than n constraint inputs are *on*, where n is the connectivity of the binder unit (or, more precisely, the connectivity of both figure and model inputs). This is because each active constraint binder indicates an exact match between two binary constraint (rod-wheel junction) nodes. The binary constraint units are grouped by wheel matches. That is, all potential rod matches for the model wheel i – figure wheel j pairing form a group. These groups are subject to a strict winner take all scheme, which means that if ambiguity exists in the matching, eg. between two dangling rods, the tie is broken arbitrarily, leaving only one unit on in every row and column. So if n constraint inputs to a wheel binder are on, there exists a one-to one correspondence between all the rods attached to the two wheels represented by the binder. The constraint binders use the same update rule as the rod and wheel binders, for convenience, but since there are no higher order constraints feeding back to these units, there is never any evidence to the contrary, so the state of constraint binders simply reflects the equality of their inputs.

In addition to the garden variety binder units, there are k special purpose binder units corresponding to the $k+1$ wheel count status units. These binder units represent the discrepancy in wheel count between the figure and each model. The activation function is set, rather arbitrarily, to be -100$\times$ the difference in wheel count, so that even a difference of only one wheel exerts a strong negative influence on the final match. This constant could (and probably should) be adjusted to reflect a more reasonable partial match strategy, but this would require more empirical knowledge of the network's behaviour than is currently available.

Winner take all (wta) nodes are a general purpose construction to implement wta subnets efficiently, where each member of the subnet inhibits all other members. Instead of explicitly creating the n^2 inhibitory links, a wta node is created and bidirectional links established between it and all members of the subnet. The wta node sets its activation level to equal the maximum of its inputs; then if any subnet member receives an activation from the wta node greater than its own, it voluntarily turns itself off. Wta nodes are used in this network to enforce a winner take all policy along the rows and columns of the binder unit arrays, in an attempt to enforce one-to-one component matches. The activation function for wta nodes is the maximum of all inputs to that node.

The results of the wta nodes for each row and column are accumulated in a special accumulation node, to supply an indication of the cumulative strength of the match. The cumulative score for all binder arrays (rod, wheel and constraint) are themselves collected by a second degree accumulation node which represents the total score for the match of the figure with the given model. The wheel count binder units also serve as inputs to the total score accumulator. The activation function for accumulation nodes is the sum of all inputs to that node.

One decision node exists. It looks at all total scores and picks the most likely one. As a side effect, it prints the results of its choice. The activation function for decision nodes is the symbolic name of the maximum of all inputs to that node.

Unique component matches are attained by enforcing a winner take all strategy on each row and column in the binder arrays. A potential problem arises , however, when there is a tie for winner. If the tie is broken arbitrarily, it may or may not matter: see figure 16. Fortunately, the network seems capable of making the correct choices when it does matter. In the example of Fig. 16b, the fact that the horse's shoulders have 2 connecting rods and 2 dangling rods whereas the rump has 1 and 3

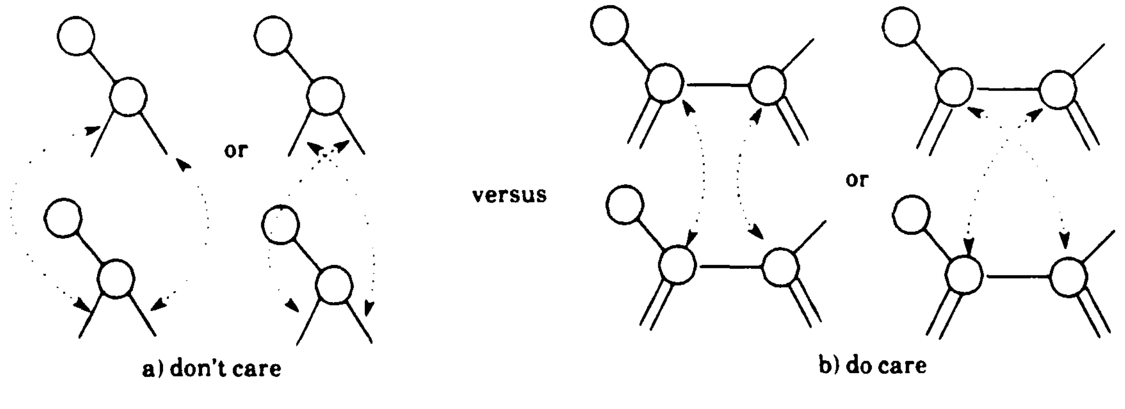


Figure 16

is sufficient to match the wheels unambiguously. The distinction arises between the neck and tail rods: the tail rod constraint unit has an activation of 4, since the rump wheel has 4 attached rods and the tail is a dangling rod, whereas the neck has activation of $14 = 4 + 10 \times 1$. Thus only three constraint nodes will be active for the cross-matching binder unit, whereas four will be on for the correctly matching binder unit. Hence the cross matching binder will turn itself off, while the correct binder remains lit.

One problem the binary constraint feedback loop is unable to solve is the so-called caterpillar/kiwi problem. Given the difficulty of enforcing a unique match, a "kiwi" will match a caterpillar rather than another kiwi (see figure 17). This difficulty is sidestepped by the special purpose wheel counter, whose task is to eliminate matches of figures having differing numbers of rods and wheels, by allowing the discrepancy in the number of wheels to penalize the match proportionally to the magnitude of the discrepancy. The drawback to this approach is that it compromises the ability of the network to report on the best partial match. Once the proportionality constant has been correctly adjusted, however, the connectionist constraint propogation scheme shows itself to be more flexible than the "pure" sequential recursive backtracking topological matcher described in the previous section. The backtracking matcher has no way to score and rank near misses, it can only report success (exact match) or failure.

The network settles into its final state after only seven iterations. Each step of the simulation does the following:

```
for each unit, u, in order
    call the activation function
for each unit u, in order
    update the externally visible outputs for the next step.
```

The first step establishes all the initial settings for the binder units, since the

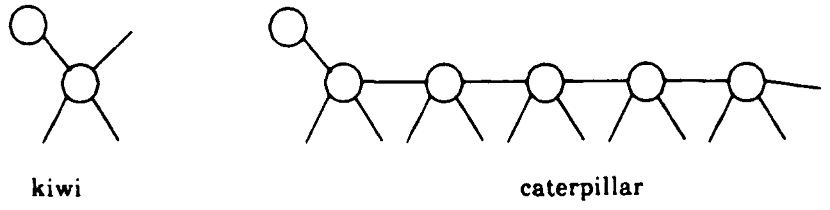


Figure 17: the 4-connected kiwi body matches to each caterpillar segment

activation of the status units is established when the network is built. The binder activations reach their wta nodes in the second iteration, and the wta information goes back to the constraint binders in the same stage, since multiple matches are eliminated by the wta nodes. Thus by the third iteration accurate constraint information is available to the rod and wheel binder units, some of which turn themselves off as a result of inhibitory constraint evidence. Accurate wheel count comparison information is also available at this stage to the decision node, already taking improbable matches out of the running. In the fourth time step, rod and wheel wta nodes receive the updated scores, which get reported to the local accumulator node in stage five. The total score for a given match is available after six steps, and the final decision can be made on the seventh iteration.

The space requirements for this network are linear in the number of models in the model base. An average tinker toy model with three wheels and seven rods requires about 1,000 units and 5,000 links. Thus for a model base with 10 models, 10,000 units and 50,000 links are required. The time requirements are essentially constant, since the results are available after a constant number of iterations. On a Sun Workstation, however, a certain slowdown in execution is noted as the space requirements increase, due to the non-transparency of virtual memory. Thus a network with one model can be built in less than 10 seconds, while a 10 model net requires about 5 minutes to build. The same commments apply to the running time of the simulation: each step requires less than one second in a small network, 1-2 minutes for a large one.

4. Summary and Conclusions

The system's operation is summarized in Figure 18. Figure 18a) shows a digitized picture of a tinker toy, the input to the system. Figure 18b) represents the magnitude of the gradient produced by the kirsch operator (ie. the edge picture). Figure 18c) has both straight lines and circles found by the Hough technique from the edge information of 18b). And 18d) demonstrates the connectivities found between rods and disks: the small circles represent joints between a rod and disk.

This system is interesting mainly in that it tackles the problem of object recognition with multiple figures in the scene and model base. Prior work in object recognition has tended to focus on matching a single figure to a single model, on the assumption that the process can be repeated serially for other figures and models.

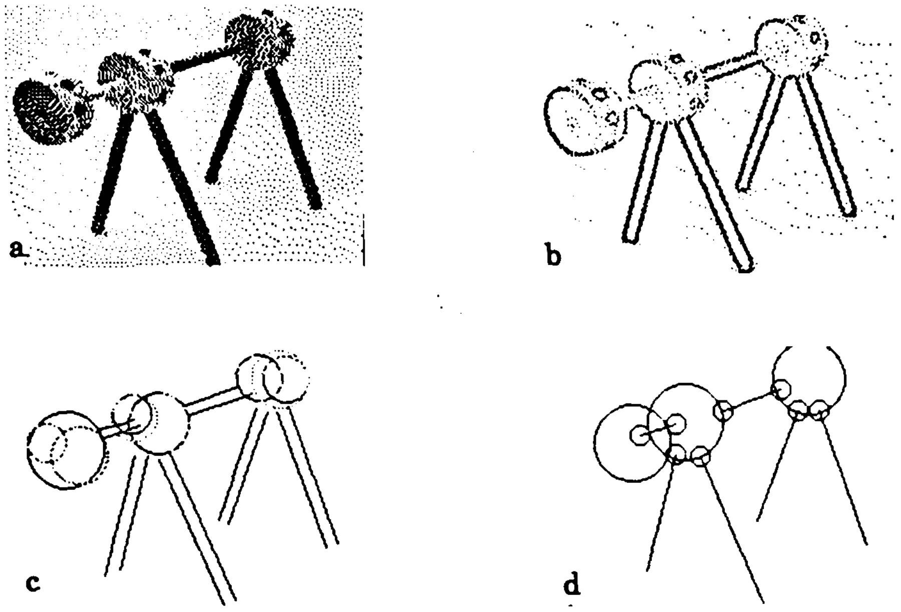


Figure 18

We, however, perform all figure–model matches in parallel, which means that the time complexity of the matching process for each figure is proportional only to the size of the largest model in the model base, not to the number of models. And since matching is perfomed in a constant number of steps, the operating time is in some sense constant. (The space requirements, however, are indeed linear with respect to the size of the model base). Furthermore, partial matches are treated in precisely the same manner as exact matches, namely, as evidence for a given interpretation. Thus multiple objects in the scene and more general forms of occlusion can be dealt with elegantly and robustly.

The tinker toy project as it stands now is merely a preliminary investigation into the structural approach to object recognition. Even at this early stage, however, it seems apparent that the structural approach to object recognition is a fruitful one. It also seems that a relatively simple, constant time connectionist constraint propagation scheme is a surprisingly effective method for topological matching. It has the ability to report partial matches as an advantage over the sequential recursive backtracking matcher, and it operates in a constant number of iterations. This characteristic ensures linear time operation on a sufficiently large parallel machine, such as the BBN butterfly. Future research into the problem will involve developing a geometrical matching scheme based on stereo [Cooper-85], where the 2D geometry of the scene forms a basis for the matching process. This may be implemented as a connectionist net, in which case the topological constraint net described here could serve as a preprocessor.

Acknowledgements

The tinker toy domain was first proposed by Paul Cooper for use in his thesis on structure in object recognition [Cooper-86a]. The initial layout of the connectionist topological matcher was a cooperative effort involving Paul, Jerry Feldman and Nigel Goddard [Goddard-86]. My contribution to this part of the project was to develop appropriate activation functions for the binder units and to implement it all.

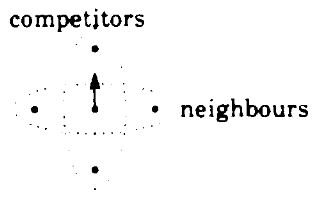
Appendix A

pseuo-code algorithm for edge thinning

```
max = maxumum value of entire image

for all x, y
    initial confidence[x,y] = edge strength[x,y] / max
    find edge type of edge[x,y] (see below)
    case edge type[x,y] of
    competetion: new confidence[x,y] = NIL
    no support: new confidence[x,y] = NIL
    neutral: new confidence[x,y] = initial confidence[x,y]
    support: new confidence[x,y] = 100%
```

To find edge type of edge[x,y]:



```
if      either competitor points in the same direction and has greater strength,
then    type = competition
else if both neighbours point in same direction
then    type = support
else if only one points in same direction
then    type = neutral
else    type = no support.
```

Appendix B

```
type point = record          /* 3-d cartesian coordinates */
          x, y, z: integer;
          end;

     wheel-index = 1..maxwheels;

     rod-index = 1..maxrods;

     wheel-pointer= 0..maxwheels;  /* a value of zero indicates the slot is empty */

     rod-pointer= 0..maxrods;

     view-type = (edge-on, face-on, oblique);

     wheel-type = record
          POINT ur, ll;
          slots: array [1..10] of rod-pointer;
          end;

     rod-type = record
          start, end: point;
          width: integer;
          int n__disks;
          pegs: array [1..2] of wheel-pointer;


var wheels: array [wheel-index] of wheel-type;

    rods: array [rod-index] of rod-type;
```

References

Ballard, D. "Form perception as transformation," TR 148, Computer Science Department, University of Rochester, January 1986.

Ballard, D. and C. Brown. *Computer Vision*, Prentice Hall, 1982.

Cooper, P., D. Friedmann, and S. Wood. "The automatic generation of digital terrain models from sattelite images by stereo," 36th Congress of the International Astronautical Federation, Stockholm, Sweden, October 1985; to appear, *Acta Astronautica*.

Cooper, P. "Connectionist graph matching: A clarification," internal report, Computer Science Department, University of Rochester, 1986a.

Cooper, P. "Object Recognition from Structure", Thesis Proposal, Computer Science Department, University of Rochester, 1986b.

Fanty, M. and N. Goddard."Users manual, Rochester connectionist simulator", TR in preparation, Computer Science Department, University of Rochester, 1986.

Feldman, J.A. and D. Ballard. "Connectionist models and their properties," *Cognitive Science*, 6, 205-254, 1982.

Goddard, N. "Matching asymmetric graphs with dangling edges using a connection network," internal report, Computer Science Department, University of Rochester, 1986.

Lowe, David. *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, Boston Mass., 1985.

Mackworth, A. "Model driven interpretation in intelligent vision systems," *Perception*, vol 5, 349-370, 1977.