N89-26604

A Connectionist Model for Dynamic Control

Kevin C. Whitfield Sharon M. Goodall James A. Reggia [†]

Department of Computer Science
University of Maryland
College Park, MD 20742

1. INTRODUCTION

This report describes the application of a connectionist modeling method known as competition-based spreading activation to a camera tracking task. The work described here is one part of an ongoing research project being conducted at the University of Maryland. The overall project is exploring the potential for automation of control and planning applications using connectionist technology. The emphasis of this effort is on applications suitable for use in the NASA space station and in related space activities. However, the results are quite general and could be applicable to control systems in general.

The technology offered by connectionist methods has several potential advantages over more conventional computational methods. For example, because connectionism is based on the tenet that useful computation may arise as an emergent property of local interactions between the nodes of a network, most connectionist models are suited for parallel processing implementations. Such parallel implementations may result in substantial reductions in processing time requirements over more conventional sequential implementations. This characteristic is desirable in real time applications, such as those intended for the space station. In addition, many connectionist models have a large degree of fault tolerance, which provides for graceful degradation in performance: only a partial loss of capability is experienced when operating in a defective condition. This property is essential to systems which operate remotely or in locations which are hard or costly to reach.

In the arena of connectionist models, the competition-based activation method provides several advantages over other techniques [5]. Of practical significance is the fact that with this technique there is often a substantial reduction in the number of links required for implementation. In many connectionist models, competitive interactions between nodes are implemented by using inhibitory (negatively-weighted) connections. However, with competition-based spreading activation, competitive interactions are implemented through the competitive allocation of node output. Thus no inhibitory connections may be needed to implement competition between nodes in a network using this method.

Competitive activation mechanisms have recently been developed successfully in a number of applications. For example, they have been used to implement a system for medical diagnosis in the combined domains of neurology and psychiatry [4, 7]. This connectionist model usually identifies the (Bayesian optimal) set of disorders with the highest posterior probability for a given set of manifestations. A competitive activation method has been used to implement a 2000 node, 12000 connection system which performs the transformation of a printed word into a corresponding phonetic representation [6]. Finally, in the field of approximation theory, a connectionist model using competitive activation has been used to obtain approximate solutions to the NP-hard minimum vertex cover problem [3]. This approach yielded high accuracy and significantly outperformed a more conventional "greedy" approximation algorithm.

The goal of this work was to develop a prototype dynamic control system for camera tracking. The motivations for this task were a) to investigate the applicability of competition-based spreading activation to this general problem area, and b) to test the capabilities of the MIRRORS/II

Acknowledgements: Supported in part by NASA Award NAG1-885 and by NSF Award IRI-8451430 with matching funds from AT&T Information Systems.

† Also with Dept. of Neurology (UMAB) and University of Maryland Institute for Advanced Computer Studies.

software environment for supporting this work. This is the first application of competitive activation mechanisms to a dynamic control system.

The next section describes the problem scenario. Section 3 details the connectionist representation of the problem. It contains a description of the node sets involved, the connections between the node sets, and the activation methods used with each node set. Section 4 then describes the MIRRORS/II and CRYSTALS utilities used for implementation, and gives an example of simulation output from the connectionist model. Section 5 details the model's evaluation, along with the results. Finally, Section 6 summarizes the research results and offers suggestions for future research in this area.

2. THE CAMERA TRACKING PROBLEM

The connectionist model developed in this research is that of a simplified camera tracking controller. The system contains multiple cameras which move on a linear track and are used to photograph a field of oncoming targets. The objective of the control system is to minimize the number of targets "missed"—the number of targets which propagate out of the target field without being photographed.

Figure 1 gives a pictorial representation of the camera tracking problem considered here. In this version, there are three tracking cameras which can move horizontally along the bottom edge of a 15 x 20 array of locations (cells), each of which can potentially contain a photographic target. The targets propagate as shown from the top edge of the array (row 1) towards the bottom edge (row 20), within one of the fifteen vertical columns. The targets move downward at the rate of one location per tick of the simulation clock, simulating entities whose photograph is desired as they approach and pass under a spacecraft. When the tracking cameras determine that they should move, they move horizontally in the appropriate direction at the rate of one location per tick. In this camera tracking problem, each camera photograph covers a three row by three column field of view. Photographic targets in rows 18, 19 and 20 of the target grid are potentially within a camera's field of view (depending on the camera's column position); the cameras are situated over row 19 of the target grid.

3. THE CONNECTIONIST MODEL

This section describes a connectionist model controller for the camera tracking problem described above. The node sets are described first, followed by the connections (links) and the activation methods used. Finally, the mechanisms of camera motion and picture-taking are described.

3.1. Nodes

There are three sets of nodes: the Terrain Cell node set (Tcell), the Camera Cell node set (Ccell), and the Position Cell node set (Pcell). The possible target locations of Figure 1 are represented by Tcells as shown in Figure 2. There are 300 Tcells arranged in a 15 x 20 array. The presence of a target is represented via a non-zero Tcell activation value. Target propagation is accomplished through Tcell-to-Tcell communication of activation. An activation level of 1.0 for a Tcell indicates that a photographic target is currently located at the position represented by that Tcell.

There are three tracking cameras represented by the Ccell. Associated with each camera is a 3 x 3 field of view (FOV), which is located within the bottom three rows of the target field (rows 18, 19 and 20). Each camera moves independently along the bottom edge of the field although each camera has a weighted preference for certain Tcell columns. For Ccells, a non-zero activation value is used to indicate that a photograph was taken during the previous tick of the simulation clock.

The only node set in Figure 2 not depicted in Figure 1 is the Pcell node set (three rows in lower half of Figure 2). The Pcell nodes are used to compute the level of target activity in the area which would eventually be covered by a camera at the corresponding position. The level of activation of a Pcell indicates the level of demand for camera coverage from that location. There are three rows of Pcells, one for each tracking camera. For fifteen columns of Tcells, a camera's FOV may centered at any location along the camera edge from within the thirteen central positions.

Hence, there are thirteen Pcells in each row (indexed by 2 through 14).

3.2. Connections

There are links from each Tcell (except for the last row of Tcells) to the Tcell directly beneath it. Designating a Tcell's position as (row,column), a Tcell at (i,j) connects to its neighbor at (i+1,j). These links are used to propagate activations representing targets toward the camera edge. Due to the nature of the activation method for Tcells (see below), the weight on these links is largely irrelevant, although it must be non-zero. For convenience, these links were implemented with unit weights.

In order to transfer photographic target information to the cameras, there are links from Tcell nodes to Pcell nodes. There is a link with the weight CTRWT to each of the three Pcells in the column directly beneath a Tcell, as well as links with the weight SIDWT to the six immediately adjacent Pcells. Of course, these links are only present when those Pcells actually exist. For example, Tcells in the first column only have three links to Pcells: a link of weight SIDWT to the three Pcells located in column 2. Some of these links are shown in Figure 2.

In order to provide photographic target location information to the tracking cameras, the bottom three rows of Tcells have links to the Ccells. The Ccells use this information from rows 18, 19 and 20 to determine if any of the Tcells in these rows both: a) have non-zero activations, and b) are located within the Ccell's current field of view[1]. Some of these links are shown in Figure 2.

The Pcell to Ccell connections are used to communicate the coverage demands from the Pcells to the Ccells. Within each column of the Pcell set, the weights to the Ccell set sum to 1.0, the maximum possible weight on a link.

In Figure 3, the three tracking cameras are shown fully covering first the leftmost and then the rightmost portion of the target field. When positioned fully to the left, Camera 1 is centered on column 2, Camera 2 is centered on column 5, and Camera 3 is centered on column 8. To avoid field of view overlap, Camera 1 should be the only camera ever centered on any of the columns 2-4. As a result, Pcells in columns 2 through 4 of the first Pcell array row have non-zero links only to the first Ccell, with the maximum possible link weight, 1.0. Similarly, Camera 1 never needs to be centered on any of the columns 9-14, so links from the Pcells in columns 9 through 14 of the first Pcell row to Ccell 1 have zero weight. In between those positions, a linear function of column position for weights on links from Pcells in the first row to the first Ccell was used. The weighting scheme for weights on links from Pcells in the third row of the Pcell array to the third Ccell mirrors those in the first row of Pcells to the first camera. Any difference between 1.0 and the sum within a Pcell column of links to first and third Ccells was used as the value for the weight of the remaining link from the Pcell in the second row to the second Ccell. A summary of these weights is found in Table 1.

Finally, there are connections from each Ccell to each of the Tcells in the bottom three rows of the target field. This connection is used by each Ccell to inform the Tcell that its target has been photographed and that the associated activation should not be propagated to the next Tcell. This process is described in Section 3.4.

3.3. Network Activation

This section details the activation methods used by the different node sets. An activation method (or rule) is a local computation carried out by a node based on the input signals it is receiving.

3.3.1. Tcell Activation

The nodes of the Tcell set use a different activation function to output activity to each set to which it is connected. Each Tcell outputs its activation to the neighboring Tcell in the next row of the same column, across the unidirectional link to that Tcell (Tcell nodes in row 20 of the Tcell array do not connect to any other Tcells).

[1] In the implementation, this information was not output by the Tcells, but rather was "collected" by the Ccells using links going the opposite direction. This was done to avoid excessive use of memory to maintain copies of Tcell activations at each Ccell.

Tcells in the bottom three rows (rows 18, 19 and 20) also output their activation to the three nodes of the Ccell set. For both of these outputs, output on a link from a Tcell is determined solely by the Tcell's activation level; the weight on these links is always 1.0 and hence is not a factor.

Tcell activity is competitively distributed to the nodes of the Pcell set. Unlike other output from Tcell nodes, Tcell output to a Pcell is proportional to the Tcell's activation level, the weight on the link between the Tcell node and the Pcell node *and* the activation of the Pcell node receiving the Tcell output. This activation method is *competition-based* since the Pcell nodes actively compete for a Tcell's activity [5]. A Pcell node's competitive strength is determined by its activation level–link weight product, relative to its competitors. The exact output at time t from Tcell node i to Pcell node j is formulated as follows:

$$out_{ji}(t) = \frac{a_j(t)\ wt_{ji}}{\sum_k a_k(t)\ wt_{ki}} \frac{a_i(t)}{1.596d^2}$$

where $a_j(t)$ is the activation level of Pcell node j at time t and wt_{ji} is the weight on the link from Tcell node i to Pcell node j, and d is the Tcell node's distance from the photographic edge of the target array. In the special case when all of the destination Pcells have an activation level of zero, the output at time t becomes:

$$out_{ji}(t) = \frac{a_i(t)\ wt_{ji}}{1.596d^2}$$

By definition, the distance from the cameras for Tcells in row 1 is 20 and the distance for Tcells in row 20 is 1. The constant $1.596 = \sum_1^{20} \frac{1}{d^2}$ is actually arbitrary, since camera motion is determined from relative activation levels. It was chosen for convenient verification of Tcell weights, in that if all Tcells are clamped to an activation value of 1.0 when the Pcells are at rest (zero activation), the net input to each Pcell is 1.0 from neighboring Tcells.

Tcell nodes receive input from three possible sources: a neighboring Tcell, a Ccell or external input. Tcell nodes in the first row of the Tcell array receive input solely from external input. A non-zero external input indicates that a target is to be propagated down this column in the Tcell array. All other Tcell nodes receive input from a preceding Tcell. Tcell nodes in rows 18, 19 and 20 also receive input from the Ccell set; a negative activation is sent to a Tcell node when a Ccell node takes a picture of that Tcell[2]. A Tcell node i updates its activation at time $(t+1)$ using the following function:

$$a_i(t+1) = \begin{cases} in_{Ccell} & \text{if } in_{Ccell} \neq 0 \\ 1.0 & \text{if } in_{Tcell} > 0 \\ 0.0 & otherwise \end{cases}$$

where $in_{Ccell} = \sum_{i \in Ccell} in_i$, $in_{Tcell} = in_{ext} + \sum_{i \in Tcell} in_i$.

The Tcell activation function can be thought of as a simple threshold function if the negating Ccell input is ignored. If the Tcell input is above the threshold level zero the Tcell assumes its maximum level of activation of 1.0 during the next tick. If the input value is at or below the threshold, the Tcell assumes its minimum level of activation (0.0) during the next tick of the simulation clock.[3]

3.3.2. Ccell Activation

The activation of the Ccells is used simply to indicate that a picture has been taken. Each camera cell receives input from the bottom three rows of Tcells, and from a row of Pcells. If any of the Tcells in the camera's current field of view are active, the camera takes a picture of these targets. This is indicated by a positive Ccell activation level.

$$a_i(t) = \begin{cases} 1 & \text{if } photograph\ taken \\ 0 & otherwise \end{cases}$$

When a camera cell is active, it outputs negative activity to each Tcell in its current field of view using the following function:

$$out_{ji}(t) = \frac{-a_j(t)\ d^2}{(d+1)^2}$$

[2] The negative activation is not actually *sent* by the nodes of the Ccell set to the Tcell nodes; see Section 3.4.

where $a_j(t)$ is the destination Tcell's current activation and d is its distance from the photographic edge of the target array. Each Ccell also receives input from a row of Pcell nodes in the Pcell array. This input is used to determine whether to change the camera's position and if so, in which direction. This processing is described in Section 3.4.

3.3.3. Pcell Activation

Pcells have relatively simple output and activation update functions. Each Pcell outputs its current activation to a single Ccell on each advancement of the simulation clock[3]. Its activation level is determined by its input from the nodes of the Tcell set:

$$a_j(t+1) = \sum_{i \epsilon Tcell} in_i(t)$$

3.4. Ccell Processing

In order to determine if the camera's current position should be modified, a Ccell identifies the Pcell among those to which it is connected with the highest Pcell activation–link weight product. The camera then moves one location horizontally towards that Pcell's position, if it is not already there. Since this process occurs during each tick, it follows that the maximum camera speed is one column position per tick. In the implementation, each Ccell actually looks at each neighboring Pcell's activation level and calculates this product, since the Ccell also needs to know the position attribute of the Pcell with the highest activation–weight product to determine which direction to move.

During the output portion of a simulation tick, each Ccell searches its current 3 x 3 field of view to check for target activations. If at least one target is discovered, the camera will take a photograph during that tick. In addition, the Ccell modifies the current activation value of any positively activated Tcells in the field of view to be $-\frac{1}{1.596\,(d+1)^2}$, where d is the distance associated with the activated Tcell. Using this modified activation, the Tcell will subsequently output contributions approximately equal in magnitude to those contributions made for that target during the previous tick, but with a negative sign. This output is intended to have the effect of "cancelling" Pcell activation attributable to that target during the previous tick. In addition, the resultant negative sign of the Tcell activation inhibits propagation of the target to its Tcell neighbor, causing the target to disappear after it has been photographed.

Note that this process must be done *before* Tcell output for that tick is performed. If it were done *after* Tcell output, there would be no way of halting the target propagation, since it would already have been propagated. Thus, it is required that, for the output phase of a simulation tick, processing will sequence through the node sets one at a time. Within the individual sets, of course, nodes are processed in parallel. This does not violate the parallel nature of the network, in that the processing time requirement for this processing still remains independent of the network size (assuming that the node set sizes are changed proportionately). This aspect of the camera processing suggests a performance improvement which will be addressed later in this paper.

4. IMPLEMENTATION AND TESTING SCENARIO

This section briefly describes the implementation of the connectionist network described in the previous section. It then describes network testing and performance measurements.

4.1. Implementation

The camera tracking connectionist network was implemented using the general–purpose network simulation system MIRRORS/II [1]. In addition, a high–level front–end processor for MIRRORS called CRYSTALS was used [2].

The basic input to the MIRRORS simulator consists of a network specification and a control specification. The network specification details the node sets and their members, along with

[3] This activation value is not actually *sent* to the Ccells in the

various default node parameters for the set (maximum activation level, decay rate, etc.). This is followed by a description of node-specific information (including connections emanating from a node, overrides to the default parameters, etc.). The second section of input to the MIRRORS simulator is the control specification. This details such things as the length of the simulation, a schedule of external influences (called *events* in MIRRORS) that take place during the simulation, and the items to be recorded during the simulation. It also specifies in what order the node sets are to be updated and output.

Even for large networks, the control specification is usually generated relatively easily by hand. However, manual generation of a network specification quickly becomes tedious for reasonably large networks. This motivated the development of the CRYSTALS preprocessor, which allows the user to use predefined high-level, topologically regular structures, such as lines, spheres, etc., to specify network components. CRYSTALS also gives the user access to Franz Lisp functions to perform such things as iteration during the process of network generation. An example showing portions of the CRYSTALS-generated MIRRORS input file used for this research appears in Appendix I of this paper. The contents of this appendix corresponds to the network simulation described in Section 5.3 of this paper.

4.2. Testing Scenario

Each of the test runs made for this research had a duration of 1000 ticks of the simulation clock. During a simulation, photographic targets were randomly generated at various constant rates (densities) in the first row and propagated toward the cameras. Targets were randomly generated according to a uniform distribution; an identical random seed was used for each test run. When a target was photographed, a message recording the event was output to the screen. In a similar manner, a message was output indicating that a target was missed if that target left the last row without being photographed. The counts of the two types of messages comprise the performance measurements.

5. RESULTS

This section details the individual tests run with the MIRRORS implementation of the connectionist model. Included are the test specifics, the data collected, and a brief discussion of what the results of each test mean.

5.1. Link Weight Determination

The first set of test runs were done to collect data for comparison between different values of the parameters CTRWT and SIDWT, which are the weights on the links from the Tcell node set to the Pcell node set (see Section 3 above). This was done to empirically determine good values for these parameters, which would be utilized in further testing. All of these tests were done with an incident target density of 8%. The results from these tests are shown in Table 2.

As can be seen from this table, performance was best with the links from the Tcells to the Pcells all equally weighted (SIDWT = CTRWT = 0.333). These were the values for CTRWT and SIDWT used in further testing.

5.2. Initial Network Performance

After selecting values for CTRWT and SIDWT parameters, a set of tests was run to measure network performance under various target densities (from 1% to 50%). The results of these tests are shown in Table 3. The performance results vary from a low of about 83% with a 50% target density, to over 99% with a 1% target density. Considering the fact that the success rate for statically positioned cameras would be around 60% at any density, these performance results appear to be a substantial improvement over statically positioned cameras. There would, however, be some improvement with randomly moving cameras over statically positioned ones, due to the three row thickness of the camera field of view.

implementation; see discussion in the following section.

5.3. A Performance Improvement

Even though the performance results given in Table 3 are good, there is a way to obtain even better performance. It lies in the way in which the node sets are ordered in their respective update and output cycles. In the simulations of the previous test, the Ccells performed their processing using the Pcell activations from the *previous* tick. If, instead, this processing could access more recent Pcell activations, one would expect a performance improvement, since the propagation delay would be reduced by one tick.

This technique was used in the second set of test runs. Using the MIRRORS Order command, the specification file was modified to ensure that, in any tick, the Ccell node set update processing occurred after both the Pcell and the Tcell update processing (see Appendix I). Camera motion processing was then modified to refer to the most recent Pcell activation values.

This sequencing does not destroy the applicability of a parallel implementation for this model, as the processing time requirement still remains independent of the network size (assuming that the node sets are increased in size proportionately). The only effect is that each tick of the simulation clock must effectively be divided into two phases rather than a single phase. The results of this set of simulations is shown in Table 4. In the table, the improvement is shown in the change of the percent of photographed targets from the corresponding percent photographed given in Table 3. The results in Table 4 indicate that the modified network performed better in every test case. Indeed, at the lowest density measured, a perfect record was achieved. Because the modified network was an obvious improvement, it will be used for comparison in the remaining tests, which involve degraded performance.

5.4. Fault Tolerance

Once a reasonable network performance had been obtained, the model's performance under partially disabled conditions was studied. Since the Ccells were deliberately designed to be localized in their motion, it was expected that incapacitating one of them would drastically reduce performance. This was not explored during this research. Instead, an investigation into the fault tolerance of the network in the face of Pcell incapacitation was undertaken. Several test simulations were run with various subsets of the Pcell node set disabled (clamped to zero). These tests are outlined in the following subsections.

5.4.1. Test One

In this set of runs, all of the "even-columned" Pcells—those Pcells in columns 2, 4, 6, 8, 10, 12, 14—were disabled. This represents a 46% operational Pcell node set—a substantial level of degradation.

From Table 5 it can be seen that there was, as a result of the highly incapacitated Pcell node set, a substantial degradation in the performance. However, it must be noted that in disabling the Pcells in columns 2 and 14, all information concerning targets in columns 1 and 15 was lost. This corresponds to around $\frac{2}{15} = 13.33\,\%$ of all incident targets. Thus, even if *all* targets in columns 2 through 14 were photographed, one would not expect performance much above 86.7%. Some number of targets in columns 1 and 15 must have been photographed serendipitously.

Of course, one way to ameliorate this high degree of degradation would be to have 15 rather than 13 Pcells in each row. However, since having a camera centered on column 1 or 15 implies only having a 2 x 3 field of view on the target field, additional processing may be needed to avoid those positions when operating under normal (nondegraded) conditions.

Another interesting aspect of this set of tests is that performance degraded more at lower densities. One possible explanation depends on a property of the target generation formula: runs using a higher target density include all those targets found in runs using lower target densities, plus some additional targets. It could be that some targets which turn out to be difficult to photograph at lower densities are easier to capture with the additional targets at the higher densities.

5.4.2. Test Two

In this set of test runs all of the "odd-columned" Pcells—those Pcells in columns 3, 5, 7, 9, 11, 13—were disabled. This was done for two reasons. Firstly, it avoids the difficulty discovered in the previous test scenario in that the loss of those columns does not obliterate the presence of certain targets from the Pcells. Also—considering now the proposed solution to the problem identified in the previous test set—this scenario corresponds to the case where not only odd-columned Pcells are disabled, but all of the proposed additional Pcells are incapacitated as well. Thus this set of runs is applicable both to the network as it currently is, and as it would be if it were modified. For the current network, this represents a 54% operational Pcell node set. The results of these runs are included in Table 6.

As can be seen from the table, the high level (46%) of incapacitation of the Pcells has resulted in little or no degradation in the system's performance. Not only is the performance loss due to "invisible" targets no longer present, but in one case performance was actually improved. One possible explanation for a performance increase is that for some targets, while incapacitation of the even-columned Pcells made that target harder to photograph (in that it registered one column farther from the camera than it would normally), incapacitation of the odd-columned Pcells made that target easier to photograph (in that it registered one column closer to the camera). In any case, it seems that this improvement is probably coincidental; additional runs with different random number generator seeds are needed before any conclusion can be reached.

5.4.3. Test Three

In this final set of test runs, the Pcells were incapacitated in a "good-bad-bad" sequence (i.e., the Pcells in columns 3, 4, 6, 7, 9, 10, 12 and 13 were disabled). This avoided "invisible" targets while bringing the Pcell node set down to a 38.5% operational state. The results of these runs are shown in Table 7.

As can be seen, the level of degradation in system performance is only slightly larger than that which was present in the previous set of runs (see Table 6). Considering that over 61% of the Pcells are inoperable, this seems to represent a significant degree of fault tolerance.

6. CONCLUSIONS AND FUTURE RESEARCH

When fully operational, the connectionist model performed well with respect to the problem addressed. The fact that it performed so well suggests that competition-based connectionist models may have potential for control applications. This is supported by the fact that the model performed well under a range of degraded conditions. In the area where performance was lowest, feasible alternatives were found which would enhance performance. Of course, only specific types of degradation were tested -- it would be advantageous in designing an actual hardware implementation to orient the design so that the probability of other more catastrophic failures is minimized. This property of selective, tolerant degradation is essential in the design of systems which are targeted for remote operation.

The application addressed in this research was admittedly of limited scope, both in the size of the problem, and in the detail of the implementation. However, the intent was not to design a fully operational system -- it was simply to investigate the applicability of competition-based connectionist models to control applications. In this respect the goal was achieved, in that no problems developed which could be attributed to the connectionist approach or to the competition-based spreading activation method.

To fully investigate the degree to which this solution is appropriate to the problem addressed the design and simulation of a more conventional algorithmic solution would be needed. This would be useful at least for the purpose of comparison. In addition, an expansion of the size of the target field, along with a reduction in target density would make the problem more closely approximate real world problems — e.g. satellite reconnaissance. From the theoretical aspect, much work remains to be done. Perhaps the problem could be viewed as a nonlinear optimization problem, where an attempt could be made at deriving local minimization methods through which an appropriate global minimization goal could be approximately achieved. Also, an attempt to derive bounds on optimum performance could be made.

Currently, we are analyzing a more challenging formulation of this problem. Instead of simulating cameras with a three by three field of view, we are now simulating cameras which have a one by one field of view, on the same fifteen column wide target area. In preliminary simulations, it appears that the competitive activation method performs well. Fault tolerance tests have not yet been performed.

It is hoped that this approach will be generalizable to other areas in real-time control. The NASA space station will consist of a large number of interdependent control systems — many having common properties and goals. A common approach to their design and implementation — perhaps one embodying principles of connectionism — could greatly facilitate the goal of achieving a permanent station in space.

REFERENCES

(1) D'Autrechy C. L., J. Reggia, G. Sutton and S. Goodall, "A General-Purpose Simulation Environment for Developing Connectionist Models", *Simulation* **51**(1):5-19 [1988].

(2) Goodall S., and J. Reggia, "The CRYSTALS Preprocessor for MIRRORS", Dept. of Comp. Science, Univ. of Md. College Park, MD (internal working paper) [1988].

(3) Peng, Y., "A Connectionist Solution for Vertex Cover Problems", Institute for Software, Academia Sinica, Beijing, The People's Republic of China [1988].

(4) Peng, Y., and J. Reggia, "A Connectionist Model for Problem Solving", *IEEE Trans. Systems, Man and Cybernetics* (in press) [1989].

(5) Reggia, J. "Properties of a Competition-Based Activation Mechanism in Neuromimetic Network Models", *Proc. Intl. Conf. on Neural Networks,* II:131-138 [1987].

(6) Reggia, J., P. Marsland, and R. Berndt, "Competitive Dynamics in a Dual-Route Connectionist Model of Print-to-Sound Transformation", *Complex Systems,* (in press) [1988].

(7) Wald, J., M. Farach, M. Tagamets and J. Reggia, "Generating Plausible Diagnostic Hypotheses with Self-Processing Causal Networks", *Jrnl. of Experimental and Theoretical AI,* (in press) [1989].

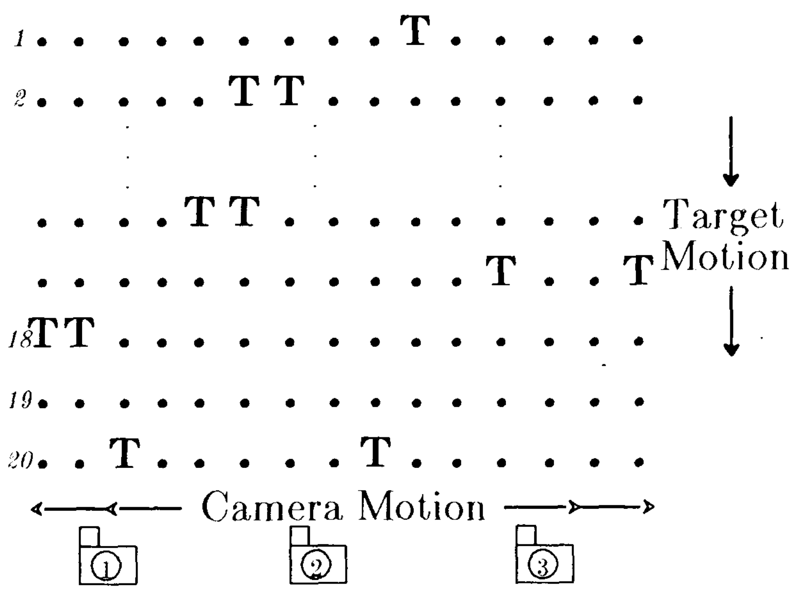

Figure 1. Camera Controller Problem: The problem is defined by a 15 by 20 array of photographic target locations and three horizontally-moving cameras. Photographic targets progress over time from row 1 of the location grid towards row 20, staying within a column. Each camera can photograph a three row by three column area; cameras are situated over row 19 of the grid.

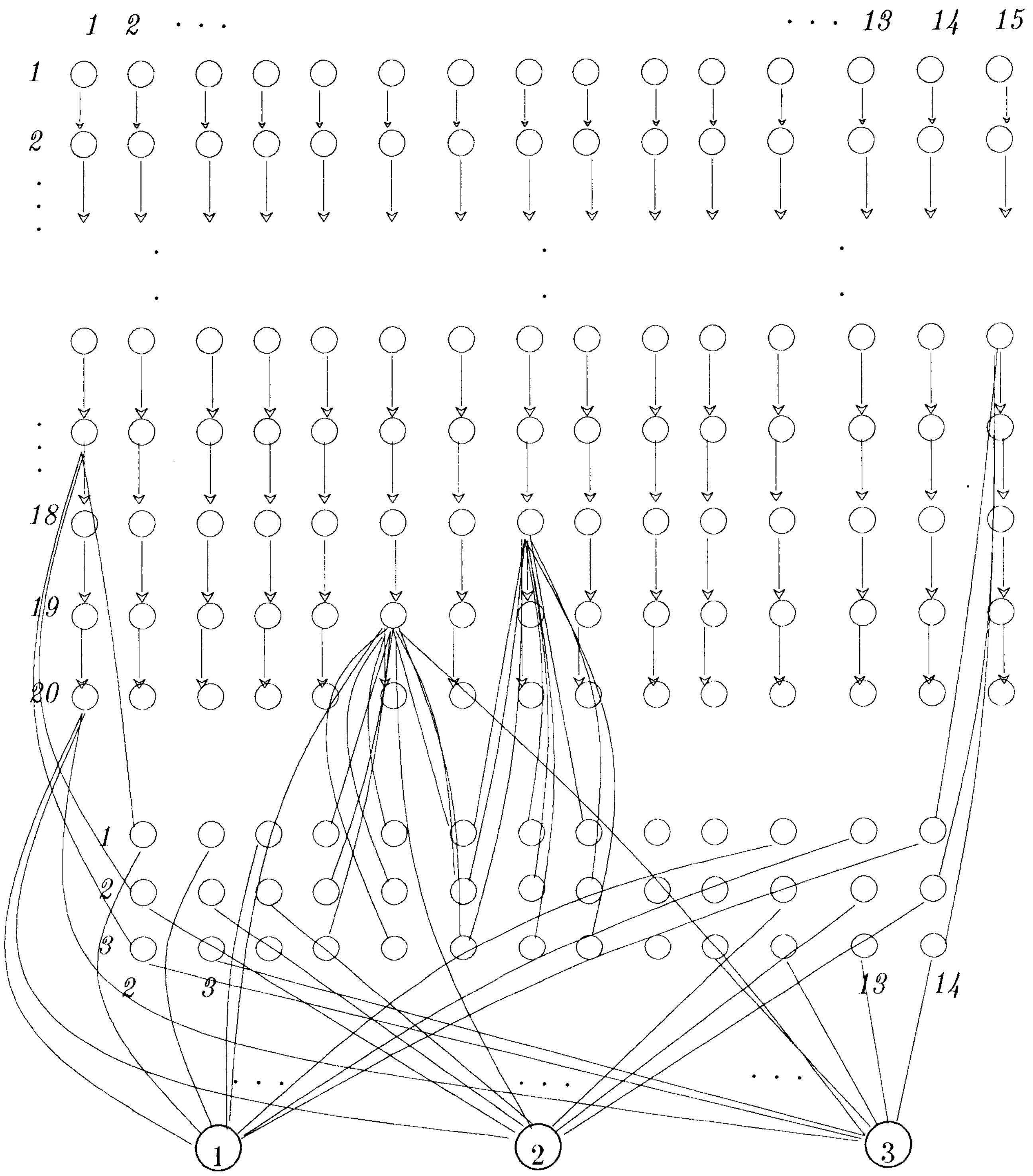

Figure 2. Connectionist Network: Rows 1 through 20 of the figure illustrate the Tcell node set and its connections. Tcells connect to their next neighbor in the Tcell set, Pcells in their column; Tcells in rows 18, 19 and 20 also connect to the Ccells. Middle 39 nodes are Pcell node set. Pcells connect Ccells. Bottom three nodes are Ccell node set. For clarity in the figure, only some of the connections between Tcells, Pcells and Ccells are shown.

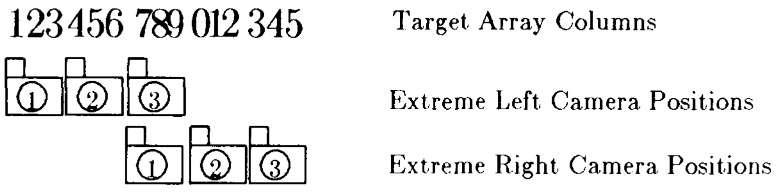

Figure 3. Field of View Overlap Constraints: To avoid overlap of the three cameras' coverage, restrict their movement to be within the range of their extreme left and extreme right column positions.

Table 1. Pcell to Ccell Link Weights

Ccell Number	Column Number Within Corresponding Pcell Row												
	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1.0	1.0	1.0	0.8	0.6	0.4	0.2	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.2	0.4	0.6	0.6	0.6	0.4	0.2	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.4	0.6	0.8	1.0	1.0	1.0

Table 2. Performance Results for Various Tcell to Pcell Link Weights

SIDWT	CTRWT	Number Missed	Number Photographed	Percent Photographed
0.45	0.1	83	1079	92.86
0.35	0.3	82	1080	92.94
0.333	0.333	79	1083	93.20
0.3	0.4	93	1069	92.00
0.25	0.5	97	1064	91.65
0.2	0.6	121	1040	89.58

Table 3. Initial Performance Results

Target Density	Number Missed	Number Photographed	Percent Photographed
50 %	1218	6172	83.52
40 %	937	4950	84.08
30 %	675	3777	84.84
20 %	385	2516	86.73
15 %	221	1928	89.72
10 %	115	1309	91.92
8 %	79	1083	93.20
6 %	37	836	95.77
4 %	14	551	97.52
2 %	4	263	98.50
1 %	1	138	99.28

Table 4. Performance Results With Sequenced Update

Target Density	Number Missed	Number Photographed	Percent Photographed	Change in Percentage
50 %	873	6513	88.18	+4.66
40 %	728	5162	87.64	+3.55
30 %	493	3956	88.92	+4.08
20 %	230	2671	92.07	+5.34
15 %	131	2019	93.91	+4.19
10 %	44	1382	96.91	+4.99
8 %	36	1126	96.90	+3.70
6 %	12	861	98.63	+2.86
4 %	5	560	99.12	+1.59
2 %	2	265	99.25	+0.75
1 %	0	139	100.00	+0.72

Table 5. Fault Tolerance Results – Test One

Target Density	Number Missed	Number Photographed	Percent Photographed	Change in Percentage
10 %	185	1240	87.02	-9.90
8 %	155	1006	86.65	-10.25
6 %	125	747	85.67	-12.96
4 %	76	489	86.55	-12.57
2 %	36	231	86.52	-12.73
1 %	21	118	84.89	-15.11

Table 6. Fault Tolerance Results – Test Two

Target Density	Number Missed	Number Photographed	Percent Photographed	Change in Percentage
10 %	60	1366	95.79	-1.12
8 %	44	1118	96.21	-0.69
6 %	24	849	97.25	-1.37
4 %	6	559	98.94	-0.18
2 %	0	267	100.00	+0.75
1 %	0	139	100.00	0.0

Table 7. Fault Tolerance Results – Test Three

Target Density	Number Missed	Number Photographed	Percent Photographed	Change in Percentage
10 %	64	1362	95.51	-1.40
8 %	51	1111	95.61	-1.29
6 %	23	850	97.37	-1.26
4 %	10	555	98.23	-0.89
2 %	1	266	99.63	+0.37
1 %	0	139	100.00	0.00

APPENDIX I: Input File for MIRRORS/II

```
[set Pcell (method PCELL)(connects (Ccell oto))(size 39)
(attribute position optional dynamic (compute POSITION Ppos))]
[set Tcell (method TCELL)(size 300)
(connects (Ccell oto optional) (Tcell oto optional) (Pcell oto))
(attribute position optional dynamic (compute POSITION Tpos))]
[set Ccell (method CCELL) (connects (Pcell oto) (Tcell oto))(size 3)
(attribute position optional dynamic (compute POSITION Cpos)) ]

[implicit (member Pcell)]
[node {1,2} (Ccell ({1} 1.0)) ]
[node {1,3} (Ccell ({1} 1.0)) ]
      .
      .
[node {3,14} (Ccell ({3} 1.0))]
[implicit (member Tcell)]
[node {1,1} (Pcell  ({1,2} 0.333333)) (Tcell ({2,1} 1.0)) ]
[node {1,2} (Pcell  (/ ({1,2} 0.333333)({1,3} 0.333333))(Tcell ({2,2} 1.0))]
[node {1,3} (Pcell (/ ({1,2} 0.333333) ({1,3} 0.333333) ({1,4} 0.333333)) (Tcell ({2,3} 1.0))]
      .
      .
[node
[implicit (member Ccell)]
[node {1} (Tcell (/({20,15} 1.0)({19,15} 1.0)({18,15} 1.0)({20,14} 1.0) ({19,14} 1.0)({18,14} 1.0)
({20,13} 1.0)({19,13} 1.0)({18,13} 1.0) ({20,12} 1.0)({19,12} 1.0)({18,12} 1.0)({20,11} 1.0)({19,11} 1.0)
({18,11} 1.0)({20,10} 1.0)({19,10} 1.0)({18,10} 1.0)({20,9} 1.0) ({19,9} 1.0)({18,9} 1.0)({20,8} 1.0)
({19,8} 1.0)({18,8} 1.0)({20,7} 1.0) ({19,7} 1.0)({18,7} 1.0)({20,6} 1.0)({19,6} 1.0)({18,6} 1.0)
({20,5} 1.0) ({19,5} 1.0)({18,5} 1.0)({20,4} 1.0)({19,4} 1.0)({18,4} 1.0)({20,3} 1.0) ({19,3} 1.0)
({18,3} 1.0)({20,2} 1.0)({19,2} 1.0)({18,2} 1.0)({20,1} 1.0) ({19,1} 1.0)({18,1} 1.0)))
(Pcell (/ ({1,8} 0.2)({1,7} 0.4)({1,6} 0.6)({1,5} 0.8)({1,4} 1.0)({1,3} 1.0) ({1,2} 1.0))]
      .
      .
      .
;
; control specification
;
[control ALTCONTROL]
[events (clamp)(display) (Gen_Targets (parser GENTARG PGen_Targets)
              (handler GENTARG Gen_Targets)]
[order (Output Ccell Tcell)     ; so that Ccells modify Tcells properly
   (Update Pcell Tcell Ccell)] ; so that Ccells get recent Pcells
[Gen_Targets ? (Tcell (& {1,1} {1,2} {1,3} {1,4} {1,5} {1,6} {1,7} {1,8} {1,9} {1,10}
              {1,11} {1,12} {1,13} {1,14} {1,15})) 10.0]
[display ? (Tcell) act nil ((string (%s "Tick: ")(%d *tick*))(string (%s "Tcells: "))(perline 15))]
[display ? (Pcell) act nil ((string (%s "Tick: ")(%d *tick*))(string (%s "Pcells: "))(perline 14))]
[display ? (Ccell) act nil ((string (%s "Tick: ")(%d *tick*))(string (%s "Ccells: "))(perline 3))]
[run 1000]
[exit]
```