63000-C.R.
19P.

A Final Technical Report
Grant No. NAG 5-597
August 1, 1985 - October 31, 1986

VISUAL MONITORING OF AUTONOMOUS LIFE SCIENCES EXPERIMENTATION

Submitted to:

National Aeronautics and Space Administration
Applied Engineering Division
Engineering Directorate
Goddard Space Flight Center
Greenbelt, MD 20771

Attention: Mr. Raymond G. Hartenstein
Code 735

Submitted by:

G. E. Blank
Graduate Research Assistant

W. N. Martin
Assistant Professor

Report No. UVA/528244/CS87/101

April 1987

SCHOOL OF ENGINEERING AND APPLIED SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF VIRGINIA
CHARLOTTESVILLE, VIRGINIA 22901

A Final Technical Report
Grant No. NAG 5-597
August 1, 1985 - October 31, 1986

VISUAL MONITORING OF AUTONOMOUS LIFE SCIENCES EXPERIMENTATION

Submitted to:

National Aeronautics and Space Administration
Applied Engineering Division
Engineering Directorate
Goddard Space Flight Center
Greenbelt, MD 20771

Attention: Mr. Raymond G. Hartenstein
Code 735

Submitted by:

G. E. Blank
Graduate Research Assistant

W. N. Martin
Assistant Professor

Department of Computer Science
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
UNIVERSITY OF VIRGINIA
CHARLOTTESVILLE, VIRGINIA

Report No. UVA/528244/CS87/101
April 1987

Copy No. 4

Report On NASA Grant Number NAG 5-597

G.E. Blank
and
W.N. Martin

Department of Computer Science
Thornton Hall
University of Virginia
Charlottesville, Virginia 22901

Introduction

The intent of this research was to investigate the design and implementation of a computerized visual monitoring system to aid in the monitoring and control of life sciences experiments on board a space station. Toward this end this we have taken the following steps: we have chosen a likely multiprocessor design, defined a plausible life sciences experiment with which to work, considered the theoretical issues involved in the programming of a visual monitoring system for this experiment on our multiprocessor, designed a system for monitoring the experiment, and implemented simulations of such a system on a network of Apollo workstations. In the following sections we will describe in greater detail each of the above steps and the results we achieved.

The Pipelined Pyramid

The multiprocessor configuration we chose as a starting point is a multi-resolution image processing design known as a "pipelined pyramid." A pipelined pyramid[1] consists of a series of "image layers," each of which comprises a $2^K \times 2^K$ matrix of processors (K=0,1,..,L). A pyramid system begins with a base level composed of $2^L \times 2^L$ processors connected in the array

topology, described below. Each processor corresponds to a single pixel so that $2^L \times 2^L$ is the highest resolution image in the pyramid. In the standard array topology each processor is directly connected to its immediate 8-neighbors. For the pyramid system, in addition to those intra-level connections the base level is partitioned into sets of 2×2 adjacent elements with all four processors in each set connected to a single "parent" processor at the next higher level in the pyramid. That next higher level is connected in a intra-level array, having $2^{L-1} \times 2^{L-1}$ elements, and is further connected, in the 2×2 manner, to another higher level in the pyramid. This construction is continued until L+1 levels have been generated with the highest level being a single element with no parent.

Associated with the pyramid is a loosely-coupled network of processors, each having arbitrary access to the pyramid. By a "loosely-coupled" network we intend that the system conforms to the definition given by Smith in which the network nodes in the system "spend a far greater percentage of their time in computation than in communication with other nodes."[2] This is intuitively appealing since we want the extra processing power to be used primarily for image analysis and not for system coordination and housekeeping. Also, extensive message traffic can create bottlenecks that seriously degrade system performance. There is a trade-off for having this loosely-coupled property, however, since the quality of the system's cohesiveness depends upon the amount of information about the system as a whole is available to each node.

The Experiment

This experiment, suggested by the work of Professor Best of the University of Virginia Psychology Department, explores the relationship between the output of specific cells in the hippocampus region of a rat's brain and the rat's ability to determine its location[3,4,5]. An electrode

is implanted near a given cell and its output is sampled and recorded several times per second while the rat attempts to locate food on a "radial maze." The idea is to determine the role of the given cell in the rat's awareness of its location and orientation. For example, does the cell "fire" when the rat is in a particular region, or pointed in a particular direction, or close to some object, or does it serve some other purpose? In order to answer these types of questions, it is also necessary to record other information about the rat in addition to the cell output. Hopefully, a causal relationship can be found between this information and the cell output leading to a theory about the purpose of the cell with regard to location and orientation awareness in the rat. Ideally, we would like to have access to all of the sensory input available to the rat in order to find the exact relationship between the cell functioning and the rat's experience. Since this is obviously impractical, the next best thing would be to observe the rat and try to infer (from the direction it appears to be looking, for example) what its sensory input is. As it is presently implemented, this experiment only monitors the position and direction the rat is facing in conjunction with the cell output data.

The Monitoring System

The monitoring system consists of a camera and/or various sensors that collect data. The advantage of this approach over some others which record "raw" data is that the computer will be able to interpret the data as it arrives and look for significant events. For example, in one approach to the above experiment, the direction the rat is facing is determined as follows: a red light is fixed to the posterior of the rat and a blue light is attached to its neck; these two points form a vector which is detected by sensors which in turn record this data over time. Clearly this is a gross approximation of the desired data and could be improved upon by a camera feeding data to a program capable of determining which direction the rat is looking.

Theoretical Issues

In the experiment described above the monitoring system must locate various features of the rat in the image data, and then track these features over time. These include such things as the location of the rat, its orientation, whether its eyes are open or closed, the direction its eyes are looking, etc. The primary challenge is to be able to perform the necessary image analysis in reasonable time. Thus, the major theoretical issue we addressed in our research was how to minimize the average running time of multiprocessor algorithms. The reasoning behind the two approaches we have adopted is explained below.

The following expression determines the amount of time needed to perform any computation,

$$T = \frac{W}{NR}$$

where T = time (in seconds), W = work (in machine instructions), N = average number of processors, and R = average processor work rate (in machine instructions/second). This immediately suggests three ways of reducing the time, T:

(1) reduce W, the total amount of work necessary to accomplish the task;

(2) increase N, the average number of processors working on the problem, or

(3) increase R, the average processor work rate.

If we assume our multiprocessor consists of identical processors, number (3) above need not be considered since R is already fixed (if the system had different processors, the more powerful ones would demand the highest utilization). Since our system consists of a loosely-coupled network of identical processors, we will attack the problem of minimizing T using the first two options above.

Reducing W corresponds to what we shall call focus of attention. Focus of attention refers to the process of concentrating a system's computational efforts in areas most likely to yield

results, thus reducing the amount of unproductive computation This focus of attention is accomplished by providing the system with knowledge of the input domain, in this case the rat experiment environment, and of general problem-solving techniques. This knowledge will be in the form of heuristics, since it is often difficult to state precisely whether a computation will yield useful results prior to performing the computation. The following examples will help illustrate these ideas.

If the task is to locate the position of the rat's eyes within a 256 by 256 array of pixels, the brute-force approach would be to consider all possible locations in which the eyes could possibly be, match each against a "template," and choose the best match. However, if we already know the locations of the rat's tail and nose, we can drastically reduce our search by considering only a rectangular subimage containing these two points. Thus, we have provided our system with some knowledge of the input domain, namely the relationship between the nose, eye, and tail locations of a rat, and this knowledge will enable the program to focus its efforts on activities that are likely to yield results.

For another task, locating the rat given that we have recorded its location and velocity in a previous time unit, a brute-force algorithm, similar to the one described above, could be applied to the image. However the system can narrow the search by defining a region in which the rat should be if it continued moving with its last velocity observed. The size of the region can be made such that it would very likely contain the rat even if it had stopped or changed direction, etc. In any case the procedure need not be foolproof--the defined region simply directs the program where to look first. Further search can be initiated if a sufficiently good match is not found within the region.

The above examples give a common-sense idea of how to achieve focus of attention. In order to get a more precise measure of a program's efficiency, we have researched and applied some results of information science. These are presented in the section on system design.

The second approach to our problem--increasing N in the expression for total time, T--corresponds to exploiting parallelism. The distributed system model upon which this project is

based is a loosely-coupled network of processors attached to the specialized computer vision multiprocessor, the "pipelined pyramid." In order to exploit parallelism we have chosen to use a method called the Contract Net Protocol (described below).

The Contract Net Protocol

The Contract Net Protocol (CNP) is an abstract problem solving model developed by Smith[2]. This method of distributed problem solving is based on the metaphor of a group of cooperating experts: overworked experts look to "contract-out" tasks to cohorts who have the necessary expertise, free time, or less important work that can wait. In an actual system each processor will be supplied with some relevant expertise, the ability to both bid for and contract out tasks, and heuristics for evaluating jobs for which it should bid and for evaluating contractors to which a job should be awarded. Note that the processors are programmed to look for work when idle and distribute work when overburdened, so the workload is naturally spread over the network. This property is independent of the number of processors in the network, thus making the CNP adaptable to networks of varying sizes. Another advantage of this distributed problem solving method is the following: since there is negotiation between the invoked and invoking processes, information from both is available to bear upon the decision of what action to take next. This negotiating is distinct from traditional languages and rule-based programming, in which the invoked process is passive.

System Design

The system consists of a CNP problem solving network having arbitrary access to a multi-resolution pipelined pyramid. Each node will be provided with low-level image-processing routines and other knowledge. One node is responsible for maintaining a repository of data that is the system's current interpretation of the image data. This repository is called the "scene description model" (SDM). The SDM consists of a series of "slots" in which the information about the various features are kept. These include: the gross location of the rat, the location of

where the tail attaches to the body, the open/closed status of the eyes, the location and direction of the eyes, the direction the rat is pointed, whether the rat is on all fours or upright, the position of the ears, whether the rat is sniffing or feeling its way around, etc. If the system is unsure about its interpretation of the image data, multiple hypotheses are maintained until all but one are eliminated. Specifically, hypothesis H_0 is eliminated when the probability of H_0 being true (given other hypotheses and data) falls below some minimum value,

$$Pr(H_0 \mid H_1, \ldots, H_n, data) < P_{\min}$$

implies that H_0 should be eliminated.

CNP Design

The CNP system will consist of M abstract problem solving nodes called "contract nodes" implemented on a network of R processors. The implementation of a contract node could be anything from a group of processors within a node to a group of processes within a single processor. Functionally, the design of a contract node consists of a task process, a message process, and a local knowledge base (see Figure 1). The task process is in charge of actually executing the "contracts" or jobs awarded to the node; this may include optionally contracting subtasks to other nodes. The message process controls the inter-node communication and decides whether to bid on a given contract. There is a formal communication language known to all nodes in the system. The types of messages available include: task announcement, contract award, contract bid, information request, node available, and result report. We will illustrate the use of these with an example that will also help demonstrate the workings of the CNP.

Assume that contract node Z is working on a task and decides that part of the job should be reallocated to another node. A task announcement message is created in which a subcontract is defined (typically just the name of a subroutine and a list of parameters). The message process broadcasts the task announcement to the other nodes. The message processes on the other nodes receive and interpret the announcement, and then decide if they wish to submit a bid for the contract. Node Z, called the "manager" with respect to this subcontract, waits for the bids to arrive

and then decides which node will be allocated the subtask. An award message is broadcast over the network, fulfilling two functions--first, it informs the node awarded the contract to begin work, and second, it notifies the other nodes that the contract has already been awarded to a different node. Upon completing the subtasks, the node awarded the contract will collect the results into a report message which will be sent back to the manager node. In the meantime, the manager node can continue working on other parts of the task that generated the subtask, or suspend the original task and continue processing another task until the result report arrives.

As an alternative to making a task announcement, node Z can check its mailbox for a node available message, indicating that another node is idle and available for work. If that node required code or other information in order to execute the task, it would send an information request message.

Besides the message and task processes, each contract node contains a third major component: the knowledge base. Contained within the knowledge base are the following: local information about the state of the system, task queues, and rule lists. We have chosen to implement both the task process and message process as rule-based systems. This enforces a separation between the underlying problem solving system and the particular domain on which we are working (since both the message and task processes are just general ''inference engines'' and the particulars of the domain are contained in the rules). Also, rules are a convenient way of including discrete ''chunks'' of domain knowledge into the program -- new rules can be added one by one without major re-writes of the code. Similarly, with meta-rules we can alter the priorities of the rules without affecting large parts of the system. The task queues consist of linked lists of records containing information about the task. There are three types of queues: ready tasks, suspended tasks, and terminated tasks. The ready task queue consists of jobs that are ready to be run, that is, jobs that were just awarded or came from the suspended task queue. The suspended task queue consists of jobs that require subcontractor reports or other information before they can proceed. The terminated task queue consists of finished tasks to be deleted from the node.

Focus of Attention

Attacking the problem of how to focus the system's efforts has led us into the realm of information science. The original idea was that a program can be thought of as accepting as input symbolically encoded "information," and through formal manipulations, repeatedly derive more information, finally outputting the most relevant portions of this information. Thus, when we talk about a program's "efficiency," we are referring to the rate at which information is derived. In this way the nodes can choose the action which will be best for the system as a whole with regard to efficiency. For more information about the following section, see Yovits, Rose, Abilock[6].

Within the context of a decision-maker (DM), information is something that affects the DM's perception of what courses of action are possible and the probabilities for the occurrence of various outcomes from these actions. "Positive information" increases the DM's certainty of its options or the results ensuing from these actions, thus increasing its certainty of which course of action to follow. "Negative information" does the reverse. More formally, for each situation, S, we have an N by M probability matrix, P, where the N rows correspond to the N possible courses of action and the M columns correspond to the M possible outcomes,

$$P[i,j] = Pr(j^{th}\ outcome \mid i^{th}\ action\ executed).$$

Information can affect either the perception of the possible courses of action (the number of rows of P) or the probabilities in P. In addition to P, there is a "value" matrix, V, associated with each situation, S, that quantifies the value to the DM of each outcome given each course of action, that is,

$$V[i,j] = Value(j^{th}\ outcome \mid i^{th}\ action\ executed).$$

What we are trying to attain is to have each node perform the best course of action in each situation. A good approximation of which action is best is the expected value, thus giving us a vector of N expected values. These are calculated by the following

$$EV[i] = \sum_{j=1}^{M} P[i,j]\,V[i,j].$$

If we were absolutely certain of our P and V matrices, i.e. the probabilities and values associated with each (action, outcome) pair, then the results in the long run can be maximized simply by choosing that action with the largest EV. However, since there may be considerable uncertainty about both P and V, a more complex decision strategy must be adopted. Specifically, we might choose an action whose EV was not optimal in order to obtain some feedback with which to adjust our EV. Also, in the CNP system it will be advantageous to have a "mix" of course-of-action decisions so that not all nodes contend for the same contract.

Because of this, a reasonable decision strategy would be to associate a probability with each course of action as given by

$$Pr(\textit{choose action } i) = \frac{EV[i]^C}{\sum_{k=1}^{m} EV[k]^C},$$

where C = a confidence factor ranging from 0 to ∞. A little algebra will show that the above strategy has the following four desirable characteristics:

(1) $\sum_{k=1}^{N} Pr(\textit{choose action } k) = 1,$

i.e., the DM will choose one of the N actions;

(2) If EV[i] > EV[k], then Pr(choose action i) > Pr(choose action k),

i.e., the DM will prefer a course of action that has a superior EV;

(3) If the DM has no confidence in the current knowledge (C = 0),

then Pr(choose action i) = 1/M for all i,

i.e., the DM will choose an action at random; and

(4) If the DM has total confidence in the current knowledge (C = ∞)

and there is a single action, i, having the best EV,

then Pr(choose action i) = 1, and Pr(choose action k) = 0 for all k not equal to i.

Within the CNP, the above decision strategy will be implemented as follows: included in the task announcement will be a number indicating how much work this piece of information will

save the system as a whole; this quantity less the expected amount of work needed to obtain this information will be the expected value to the node of winning the contract; the expected value is calculated for each of the announcements and other possible courses of action open to the node; the probabilities associated with each action are calculated; and then, a random number generator is used to pick one of these.

System Operation

In the following section we demonstrate how the above concepts have been used in our design of the experiment monitoring system. Recall that for the purposes of the experiment we wish to observe and record the following features: rat location, rat orientation, open/closed status of eyes, direction eyes looking, upright/all-fours status, whiskers making sniffing motion (olfactory cues), feet making scratching or feeling movements (tactile cues), ears' positions (auditory cues), nose location, tail attachment location, and velocity.

In the above list we can easily find some work-saving information dependencies. For example, if the rat's eyes are closed there's no point in trying to determine which direction they're looking. This is the common sense idea behind what we discussed in the information section above. In the terms we used in that section we would say that the expected value of the action of determining the open/closed status is greater than that of determining the eyes' direction. Thus there would be a greater probability of selecting this action first. Another example is the fact that in order to determine the rat's velocity we must first have its current location. Therefore, it makes sense to determine the location first.

In addition to the above features, we have added one called the "gross location." This feature will consist of four points forming a rectangle in which the rat will be found. Intuitively, this is a very valuable piece of information because it immediately reduces the search for all other features to a fraction of what they would otherwise be, and can be gotten with little effort. The idea behind it is that at a low resolution level on the pyramid, a white rat will stand out as a white "blob" of pixels. Thus, we can approximately locate the rat with less work. (If there is

more than one "blob," we can look more carefully at a higher resolution in order to eliminate possibilities.) Reasonable information dependencies for the experiment monitoring system are illustrated in Figure 2.

With these preliminaries out of the way we can begin to discuss the system operation. The system starts by giving the node with the SDM the task MONITOR. This activates the following rule:

If MONITOR Then LOCATE-FEATURES TRACK-FEATURES.

This simply sets a state variable within the knowledge base indicating that the system is in "locate mode." The next cycle of the inference engine activates the following rule:

If LOCATE-FEATURES Then TASK-ANNOUNCE-GROSS-LOCATOR
TASK-ANNOUNCE-EYES-LOCATOR
TASK-ANNOUNCE-TAIL-LOCATOR
TASK-ANNOUNCE-NOSE-LOCATOR
etc.

This results in the task announcements for all the features mentioned above. The other nodes in the CNP interpret the announcements and submit bids. The bidding is determined by the decision rule given at the end of the previous section. The node that made the announcements awards the contracts, and the work begins. Because of the extreme importance to the system of knowing the "gross location," it is reasonable that the system devote a lot of computational resources to this task immediately. Thus, within the task process of the node that win the gross location contract is a rule that splits the task and distributes the parts to other nodes. In general, the decision of whether to divide a task is governed by the information scheme in the previous section, that is, we want each node to perform the action that will be most valuable for the system as a whole. Thus, if a given task is sufficiently more valuable than the next best task (in terms of work savings), the system should be willing to "spend" more effort to perform that task. Ideally, the system will continue to devote resources (work) to the best task until the value

of the task (that is, expected work savings minus expected expenditures) is equal to that of the second best task.

Thus, depending on the number of nodes in the CNP, some number of nodes will work together on the gross location of the rat while the remaining nodes work on other features. When the gross location has been determined, a report will be sent to the manager and the information will be put into a slot in the SDM. This information as well as anything else in the SDM can be requested for and used by the other nodes in the network. The rules for executing the various feature-locating routines are capable of adjusting to varying levels of information. Thus, if the gross location is known, the task process executing the ''nose locator'' will search only the rectangle determined by the gross location; otherwise it will search the entire image.

After all the features have been located, the system will go into ''track mode,'' finding and recording the features once every time unit. In the manner described above, the system will have monitored the experiment by concentrating its efforts in promising areas and dividing up the task among the nodes of a cooperating network.

System Implementation

The research described in the above sections has culminated in the realization of two different versions of the monitoring system. One of these is complete, while work on the other is continuing. The former is based on the same general concept of the pipelined pyramid and loosely-coupled network of cooperating processes, except that it utilized a different problem solving method than the CNP. This method assigns the tasks of location and tracking of the various image-features to independent processes called "agents[1]."

Since we do not have access to an actual pipelined pyramid, our implementations are simulations of such a system. We have simulated the pyramid through the use of computer-generated imagery in conjunction with software and data structures that duplicate the multi-resolution feature of the actual hardware. However, since we have a network of Apollo workstations with a parallel language facility called PISCES[7], the system was implemented in an actual

multiprocessor environment with several agents analyzing am image sequence.

The continuing implementation involves establishing the CNP on the Apollo network and installing the rules that affect the decision strategy described in the focus of attention section.

References

1. Tan, C.L., and Martin, W.N., "A Distributed System for Analyzing Time-Varying, Multiresolution" Imagery," *Computer Vision, Graphics and Image Processing, Vol. 36, 1987.*

2. Smith, R.G., **A Framework For Distributed Problem Solving,** Ann Arbor:UMI Research Press, 1981.

3. Best, P.J., and Hill, A.J., "Visual and Auditory Cues Support Place Field Activity of Hippocampal Units in the Rat," in **Conditioning: Representation of Involved Neural Functions,** C.D. Woody (ed.), New York:Plenum, 1983, pp. 37-47.

4 Blank, G.E., and Martin, W. N., "Visual Monitoring of Autonomous Life Sciences Experimentation," *Proceedings of the SPIE Conference on Space Station Automation,* Cambridge, MA, 1985, pp. 88-94.

5. Blank, G.E., and Martin, W. N., "Focus of attention in Systems for Visual Monitoring of Experiments," *Proceedings of the SPIE Conference on Space Station Automation,* Cambridge, MA, 1986, pp. 236-242.

6. Yovits, Rose, and Abilock, "Development of a Theory of Information Flow and Analysis," in **The Many Faces of Information Science,** E.C. Weiss (ed.), Boulder:Westview Press, 1977, pp. 19-51.

7. Pratt, T.W., "PISCES: An Environment for Parallel Scientific Computation," *IEEE Software,* Vol. 2, No. 4, 1985, pp. 7-20

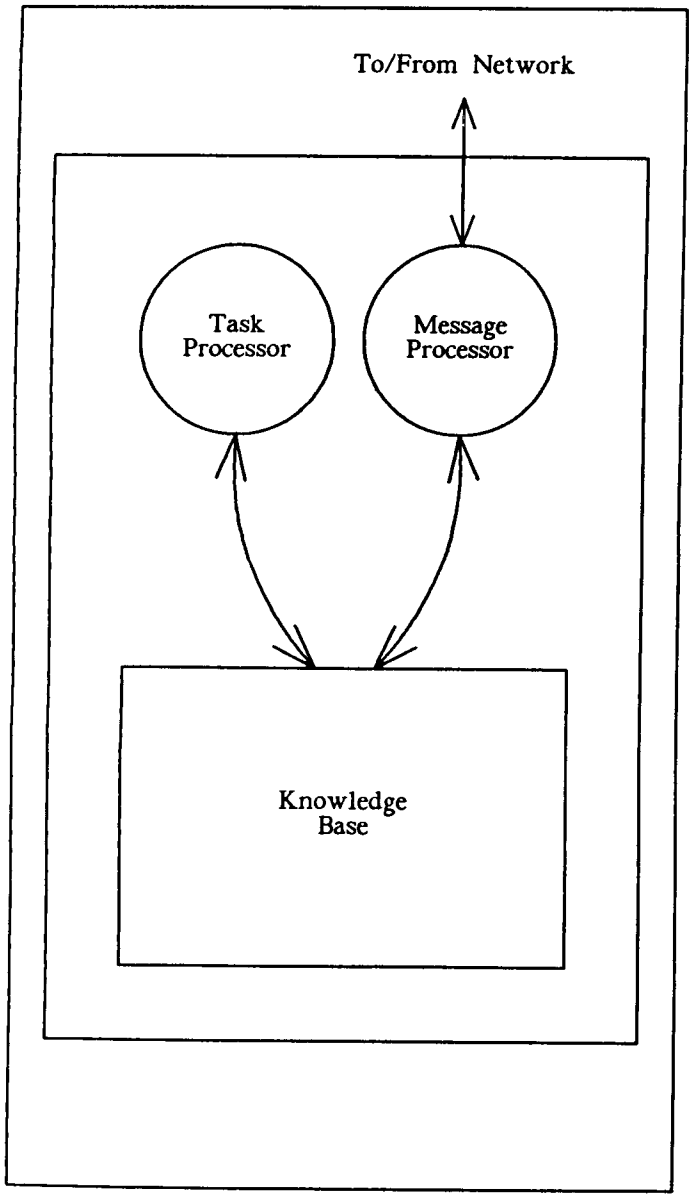

Figure 1. CONTRACT NODE

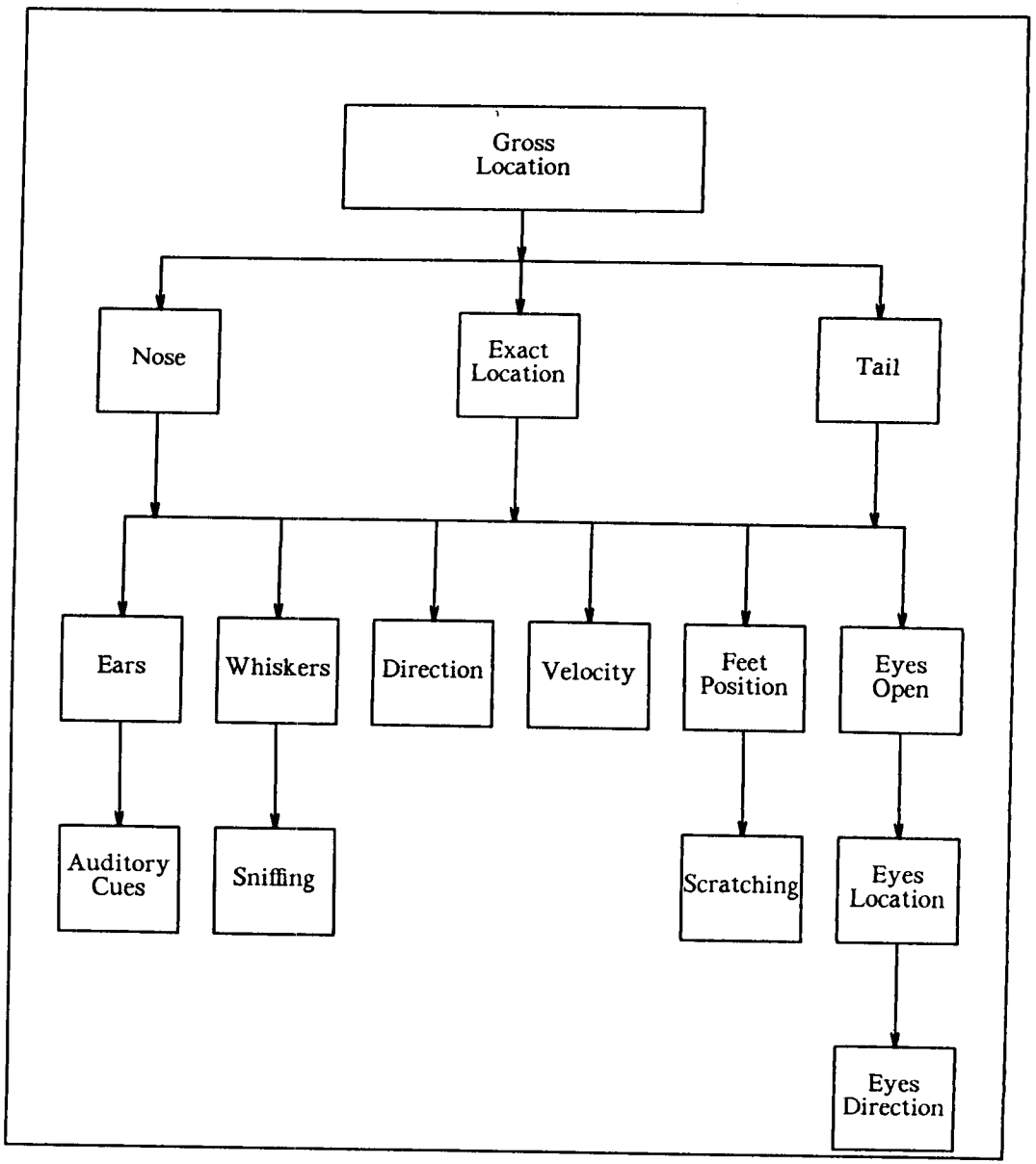

Figure 2. INFORMATION DEPENDENCIES

DISTRIBUTION LIST

Copy No.	
1 - 3	National Aeronautics and Space Administration Applied Engineering Division Engineering Directorate Goddard Space Flight Center Greenbelt, MD 20771 Attention: Mr. Raymond G. Hartenstein Code 735
4 - 5*	NASA Scientific and Technical Information Facility P.O. Box 8757 Baltimore/Washington International Airport Baltimore, MD 21240
6 - 7	W. N. Martin, CS
8	R. P. Cook, CS
9 - 10	E. H. Pancake, Clark Hall
11	SEAS Publications Files

* 1 reproducible copy

JO#8876:jlb

UNIVERSITY OF VIRGINIA
School of Engineering and Applied Science

The University of Virginia's School of Engineering and Applied Science has an undergraduate enrollment of approximately 1,500 students with a graduate enrollment of approximately 560. There are 150 faculty members, a majority of whom conduct research in addition to teaching.

Research is a vital part of the educational program and interests parallel academic specialties. These range from the classical engineering disciplines of Chemical, Civil, Electrical, and Mechanical and Aerospace to newer, more specialized fields of Biomedical Engineering, Systems Engineering, Materials Science, Nuclear Engineering and Engineering Physics, Applied Mathematics and Computer Science. Within these disciplines there are well equipped laboratories for conducting highly specialized research. All departments offer the doctorate; Biomedical and Materials Science grant only graduate degrees. In addition, courses in the humanities are offered within the School.

The University of Virginia (which includes approximately 2,000 faculty and a total of full-time student enrollment of about 16,400), also offers professional degrees under the schools of Architecture, Law, Medicine, Nursing, Commerce, Business Administration, and Education. In addition, the College of Arts and Sciences houses departments of Mathematics, Physics, Chemistry and others relevant to the engineering research program. The School of Engineering and Applied Science is an integral part of this University community which provides opportunities for interdisciplinary work in pursuit of the basic goals of education, research, and public service.