KME Copy

AMHERST SYSTEMS INC.

FOVEAL MACHINE VISION FOR ROBOTS USING AGENT BASED GAZE CONTROL

QUARTERLY STATUS REPORT

Contract No. NAS 9-19335

August 10, 1995

Document Control No.: 618-9160001

Report No. 2

Prepared for:

NASA Lyndon B. Johnson Space Center
Houston, TX 77058

Prepared by:

Amherst Systems Inc.
Buffalo, NY 14221-7082
(716) 631-0610

Government Purpose License Rights
(SBIR Program)
Contract No.: NAS 9-19335
Contractor: Amherst Systems Inc.
SBIR RIGHTS NOTICE (JUNE 1987), NASA DEVIATION

These SBIR data are furnished with SBIR rights under Contract No. NAS 9-19335 and subcontract E3439. For a period of not less than 4 years after acceptance of all items to be delivered under this contract, the Government agrees to use these data for Government purposes only, and they shall not be disclosed outside the Government (including disclosure for procurement purposes) during such period without permission of the Contractor, except that, subject to the foregoing use and disclosure prohibitions, such data may be disclosed for use by support Contractors. After the aforesaid period the Government has a royalty-free license to use, and to authorize others to use on its behalf, these data for Government purposes, but is relieved of all disclosure prohibitions and assumes no liability for unauthorized use of these data by third parties. This notice shall be affixed to any reproductions of these data, in whole or in part.

AMHERST SYSTEMS INC.

FOVEAL MACHINE VISION FOR ROBOTS USING AGENT BASED GAZE CONTROL

QUARTERLY STATUS REPORT

August 10, 1995

Prepared for:	NASA Lyndon B. Johnson Space Center Houston, TX	[illegible] Dr. Cesar Bandera Program Manager
Contract No.:	NAS 9-19335	
CDRL Item:	1	[illegible] Daniel G. Sentz Quality Assurance Manager

AMHERST SYSTEMS INC.

The Contractor, Amherst Systems Inc., hereby certifies that, to the best of its knowledge and belief, the technical data delivered herewith under Contract No. NAS 9-19335 is complete, accurate, and complies with all requirements of the contract.

8/28/95
Date

[illegible]
Dr. Cesar Bandera, Program Manager

Section 1
PHASE II OUTLINE

The technical objective of the Phase II effort is to develop an Extra-Vehicular Activity Helper-Retriever (EVAHR) robot that uses active foveal vision and an autonomous behavioral control architecture. This Foveal EVAHR (FEVAHR) shall use Hierarchical Foveal Machine Vision (HFMV) to detect objects, track and pursue deictically and non-deictically referenced objects, and avoid obstacles in a dynamic, housed environment. FEVAHR shall utilize a Grounded Layered Architecture with Integrated Reasoning (GLAIR) to provide gaze control for the vision system, including control of all subsidiary functions (e.g., mobile robot platform kinematics). The Phase II work plan consists of the following tasks:

1. Hardware Design & Integration,
2. HFMV Software Development,
3. GLAIR Software Development, and
4. FEVAHR Integration & Evaluation.

Section 2
WORK PERFORMED DURING PERFORMANCE PERIOD: MAY 1, 1995 - AUGUST 1, 1995

2.1 Hardware Design and Integration

This task consisted of defining the FEVAHR scenario, analyzing vision system resolution requirements for FEVAHR to successfully operate within the defined environment, and developing the FEVAHR Hardware Specification document. The Hardware Specification Document for the FEVAHR system was accepted by NASA Contracting Officers Technical Representative Ken Baker. Internal Part Numbers, Finalized Drawings, Purchase Requests, and Purchase Orders were generated and all FEVAHR components were ordered. Off-the-shelf components, such as C40 motherboards (TDM412) and C40 processor boards, were received, while specialized components, such as the Nomad200, are expected within six weeks. The Zebra Vision Head from TRC has the longest period before the anticipated delivery date in October, 1995.

The delay in completing the Hardware Specification and the delay incurred in the purchasing process has had an adverse impact upon the schedule (Refer to Section 2.5.1). Amherst Systems has taken the following steps to minimize the impact of the hardware procurement delay:

- Amherst Systems has procured an advance copy of the Robot Simulation software, which is being used to develop GLAIR, from Nomadic Technologies. The control software developed with the simulator can be directly ported to the Robot Hardware.

- Amherst Systems has obtained from Transtech Parallel Systems "loaner" hardware in order to facilitate the development of HFMV software. This hardware, used in conjunction with Amherst Systems' IR&D machine vision hardware, allows the development of essential HFMV algorithms.

- Amherst Systems will reallocate engineering staff (i.e., the engineer originally allocated to develop the Zebra control software), so that the financial impact to the contract will be minimized.

In addition, Amherst Systems is attempting to obtain an advance copy of the Zebra Head documentation from TRC. This will facilitate the design and implementation of gaze control algorithms.

2.2 HFMV Development

HFMV development consisted of the implementation and evaluation of FEVAHR image processing algorithms and system support functions. The image processing algorithms include parallel implementations of segmentation, edge detection, line fitting, and corner detection. These algorithms were developed using Amherst Systems' IR&D machine vision system (i.e., Pulnix TM1001 progressive scan camera with a 6.5*mm* lens, a Transtech TDM435 frame grabber with C40 for image rexelization, and a 40MHz TDM411 single C40 board, 3L Parallel C). The IR&D hardware was augmented with two TDM411 boards provided by Transtech. These algorithms will be extended to accommodate FEVAHR RGB image data in the next performance period. System support functions consist of a communication manager for the C40 network (i.e., polygon communication manager) and a communication manager for the control computers; data definition and data flow diagrams for the control computer system are provided in Appendix B. The control computer communication kernel, which will facilitate inter- and intra- computer communication, is currently being implemented.

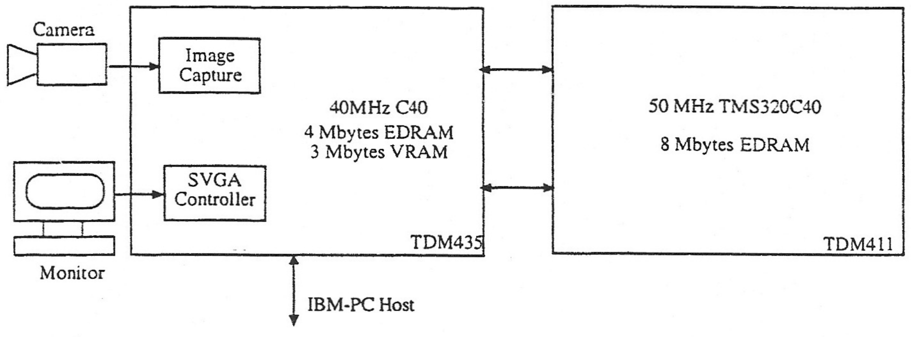


Figure 2.2-1: The Physical Architecture Of The System For Canny's Edge Detection

The polygon communication manager and configuration tools have been tested extensively for simple message communication for various network configurations. To test performance a parallel implementation of Canny's edge detector was evaluated. A four level foveal polygon was used; the size of each layer is 128 x 128 rexels. Two TMS320C40 processors were used in the simulation: one to accommodate the master and the grabber tasks and the other one to accommodate the logical computation nodes. There were two available links between the two processors. The physical architecture of the system is shown in Figure 2.2-1. The logical architecture of the system for the simulation is shown in Figure 2.2-2. The performance of the image processing engine with 1, 4, 8, and 16 logical processors under the support of the development environment was simulated. No hidden communication layer is needed for the single processor implementation. The hidden communication layer is necessary to route the information to the desired destination for multiple processor implementation. The exact route of each piece of the messages going through depends on the dynamic load of the communication channels.

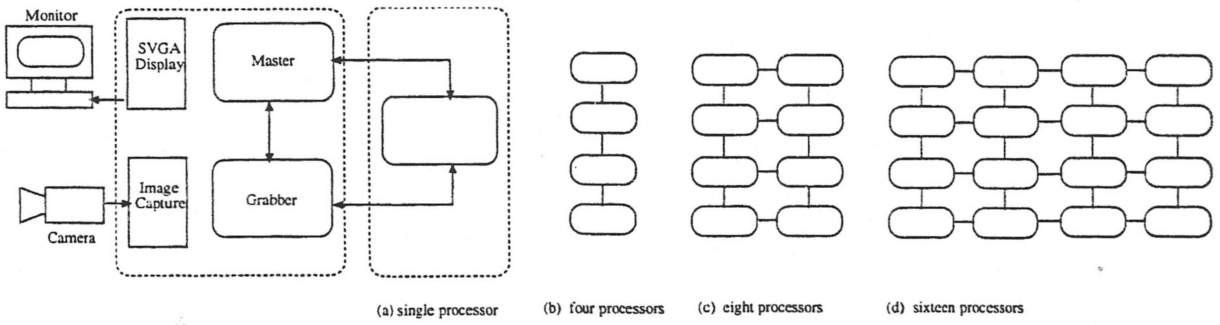


Figure 2.2-2: The Logical Architecture Of The System For Canny's Edge Detection

The execution time for Canny's edge detection under different conditions is listed in Table 2.2-1. Note that the time to distribute the subimages into the computing nodes and the overlap between the subimages to maintain edge continuity are taken into account.

Table 2.2-1: The execution time of Canny's edge detector under different conditions

logical processors	1	4	8	16
execution time in *ms* ($\sigma = 0.5$) processor utilization rate	850 100%	990 86%	1036 82%	1120 76%
execution time in *ms* ($\sigma = 1.0$) processor utilization rate	1018 100%	1120 91%	1180 86%	1250 81%

Suppose the utilization rate is 100% for a single processor implementation. The processor utilization rates for different implementations under the support of our development environment are also listed in Table 2.2-1. Obviously, the ratio of communication to computation of a specific task is also one of the important issues in the overall system performance. Considering that communication between tasks inside memory is generally slower than communication through physical links even when zero wait state SRAM is used, implementing each logical node on a physical processor in the above simulations will lead to at least a linear speed up (better than linear speed up in real applications) compared to the simulation. In fact, a single channel communication between two tasks on a 40 MHz TMS320C40 with EDRAM (15ns access cycle) is approximately 30% slower than communication between two tasks over a physical communication link. And the figure is approximately 10% for a 50 MHz C40 with the same memory configuration. This means that, for a typical σ value of 0.5, 16 TMS320C40/44s connected in a 2D mesh can achieve a frame rate of 15MHz for edge detection using Canny's edge detection scheme for the four layer foveal polygon (each layer is a 128 pixel square image).

2.3 GLAIR Development

A copy of the U.B. Status Report, which details the development of GLAIR, is provided in Appendix A.

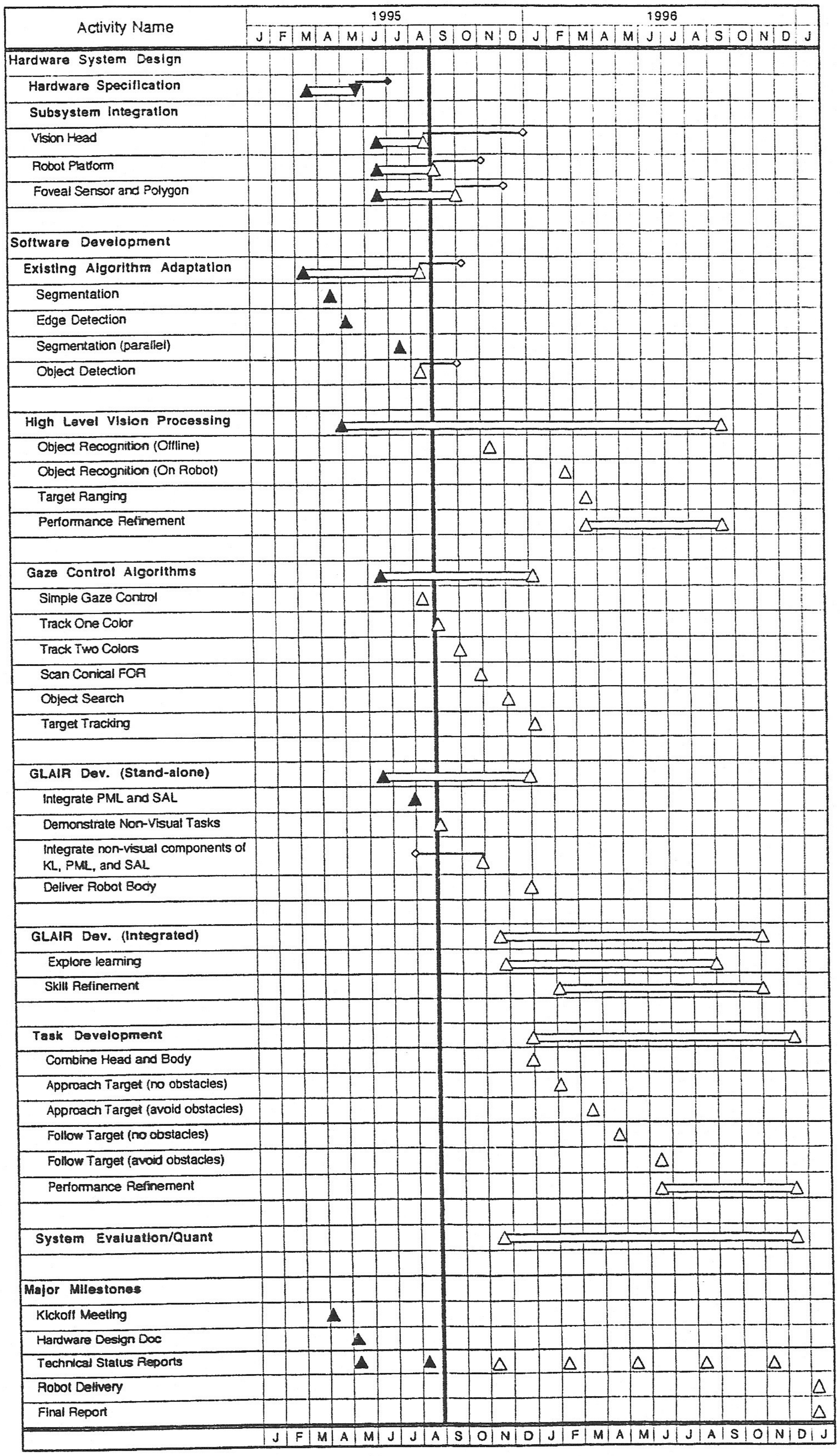
2.4 FEVAHR Integration & Evaluation

No work has been performed under this task.

2.5 Management

Work on the FEVAHR program has been progressing as expected, with the exception of unanticipated delays with the specification/procurement of hardware. HFMV algorithm development is on schedule. GLAIR development is on schedule, and should not be significantly impacted by hardware delays; the Nomad 200 is expected in September 1995. The delay in the delivery of the Zebra head (October 1995) will have the most significant impact upon the proposed schedule (Refer to Section 2.5.1). As previously mentioned, Amherst Systems has taken several steps to minimize the impact of this delay.

2.5.1 Milestone Chart



2.5.3 Financial Status

Amherst Systems' costs without fee totaled $62,377 during the performance period of May 1, 1995 to August 1, 1995. Of that, $20,400 was disbursed to subcontractor SUNY at Buffalo. Delays in hardware procurement during the reported period are reflected in the lower than estimated costs; the original projection for the quarter was $83,791.

Progress Report of Work Performed from February to May, 1995
for
Amherst Systems Inc.
Subcontract No. 150-7176A

Design of GLAIR for the Foveal Robot

Stuart C. Shapiro and Henry Hexamoor
Department of Computer Science
and Center for Cognitive Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, New York 14260

August 1, 1995

1. Overview

We received the Nomad200 simulation software for the FEVAHR robot. We have begun the following activities: (a) we are implementing the FEVAHR room using the simulation, including simulation of the vision system, and (b) we have developed the perceptuo-motor level component of FEVAHR to work with the Nomad200 simulation. The implemented PMA uses sonar data. The PMA does not yet use vision data and it is not yet coupled with the Knowledge Level (KL) component of FEVAHR.

We continued implementation of the KL component of FEVAHR with a SNePS network. We used the graphics tool Garnet to simulate the FEVAHR room. The KL component is made to work with the simulated room. In the remainder of this report we present the details of the KL component.

2. Knowledge Level Progress

We have partially implemented FEVAHR's Knowledge Level. This has involved partial implementation of:

- a grammar for understanding natural language input and for natural language generation,
- plans and primitive actions;
- alignment of KL representations of primitive actions and other entities with PML representations, and
- a simulated robot and environment.

2.1 The Garnet Simulation

Since this effort predated the arrival of the Nomad simulation software, we used the Garnet[1] graphical user interface package to create the simulated robot and environment.

According to the specifications, FEVAHR will be in a 17' x 17' room, containing

at least 1 named individual, e.g., ``John,"
at least 1 individual unique by description,
at least 1 indistinguishable class,

all of which will have a minimum dimension of 1'.

Figure 1 shows the objects in the simulated room. In the upper left is FEVAHR, represented by a cyan circle. In the lower-right is a blue square with a "name tag" representing John. In the upper right is a green circle representing "the green robot." In the lower left are three red circles representing "red robots."

The room itself is simulated by a graphics window whose size is a scaled 17' x 17'. In the same scale, John is 1 $^1/_2$' on a side, and each of the 5 robots is 1' in diameter.

John and the green and red robots can be moved by a user dragging them with the mouse, simulating their movement themselves, or their being moved by an agent external to FEVAHR. FEVAHR, however, moves only under program control.

All objects are implemented as Garnet CLOS objects. Their locations within the simulated room are maintained by Garnet as values of certain slots.

[1] Brad A. Myers & Andrew Mickish. "Overview of the Garnet System," Carnegie Mellon University, Oct., 1993.

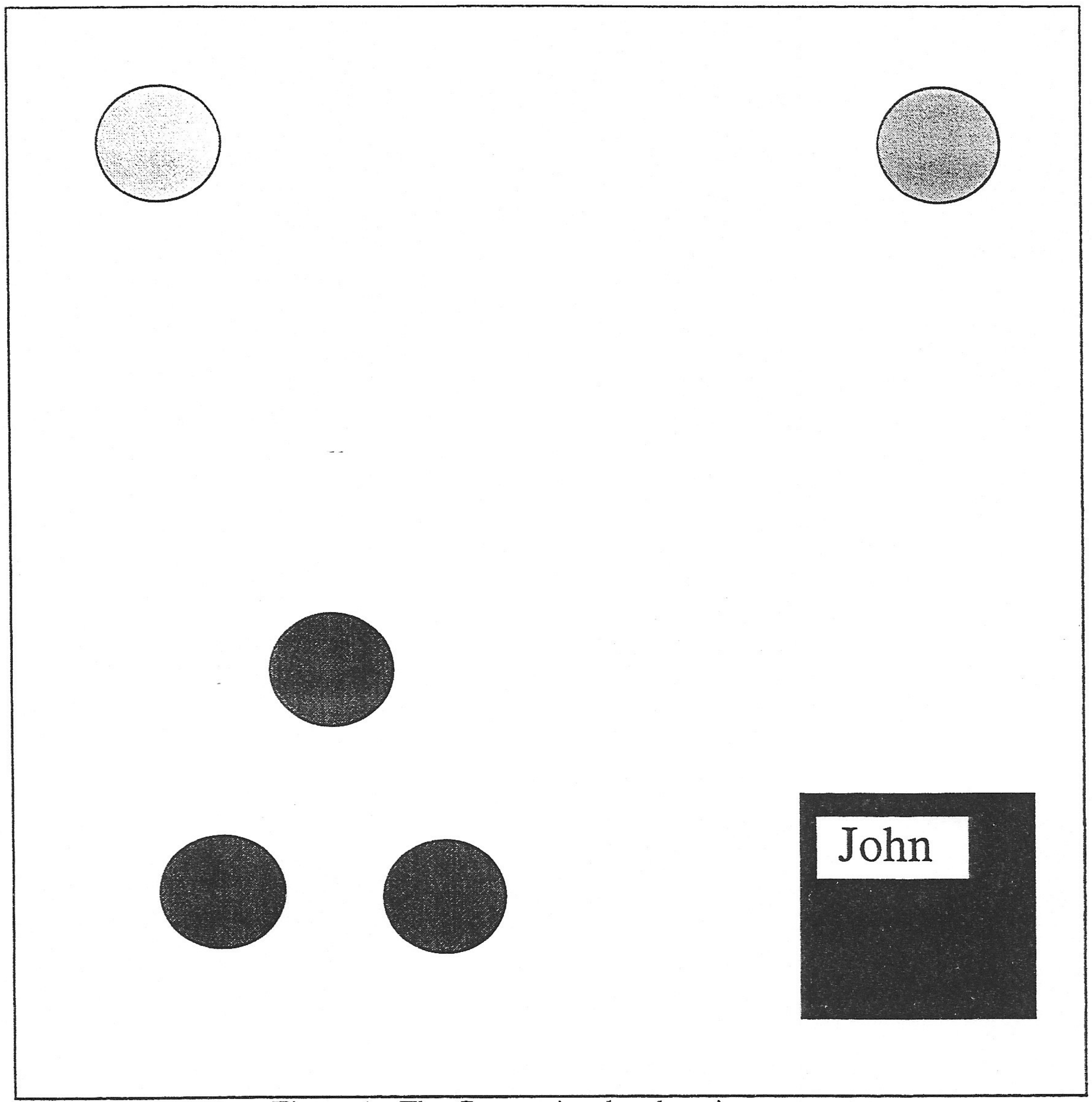


Figure 1: The Garnet simulated environment.

2.2 Alignment of Entities

At the KL, an entity is represented by a SNePS node representing the entity as FEVAHR thinks of it. At the PML and SAL, however, FEVAHR can only have a sensory impression of the entity. Therefore, SNePS nodes are aligned with descriptions, where a description is implemented as a list of the color of the entity and its shape. This description of John is `(#k<OPAL:BLUE-FILL> FEVAHR-WORLD:SQUARE)`. The green robot and the red robots don't have descriptions themselves. Instead the SNePS node representing the category of robots is aligned with the description `(NIL FEVAHR-WORLD:CIRCLE)`, the node representing the color green is aligned with the description `(#k<OPAL:GREEN-FILL> NIL)`, and the node representing the color red is aligned with the description `(#k<OPAL:RED-FILL> NIL)`.

The alignments, themselves, are implemented by a global assoc. list, ***alignments*** which is of the form (· · · (*node* . *description*) · · ·).

2.3 Natural Language Commands

According to the FEVAHR specifications, the minimal command language is

```
<command> ::= Stop | <action> <np>
<action> ::= Go to | Follow
<np>   ::= <npr>
| (that | a) [<adj>] <n>
```

In fact, the currently implemented grammar includes

```
<command> ::= Stop | <action> <np>
<action> ::= Find | Go to | Follow
<np>   ::= <npr>
| (that | this | the | a) <cat>
<cat>   ::= [<adj>] <n>
```

The relevant lexicon is

```
<adj> ::= green | red
<np> ::= robot
<npr> ::= John
```

giving 1 + 3 x (1+ 4 x 3 x 1) = 40 different commands:

1. Stop.

2. Find John.

3. Find a robot.

4. Find the robot.

5. Find that robot.

6. Find this robot.

7. Find a green robot.

8. Find the green robot.

9. Find that green robot.

10. Find this green robot.

11. Find a red robot.

12. Find the red robot.

13. Find that red robot.

14. Find this red robot.

15. Go to John.

16. Go to a robot.

17. Go to the robot.

18. Go to that robot.

19. Go to this robot.

20. Go to a green robot.

21. Go to the green robot.

22. Go to that green robot.

23. Go to this green robot.

24. Go to a red robot.

25. Go to the red robot.

26. Go to that red robot.

27. Go to this red robot.

28. Follow John.

29. Follow a robot.

30. Follow the robot.

31. Follow that robot.

32. Follow this robot.

33. Follow a green robot.

34. Follow the green robot.

35. Follow that green robot.

36. Follow this green robot.

37. Follow a red robot.

38. Follow the red robot.

39. Follow that red robot.

40. Follow this red robot.

The 18 commands using "this" or "that" must be combined with a deictic gesture pointing to an object. In this simulation, we use clicking with the middle mouse button on an object in the simulated environment. The six commands marked with a "*" are semantic anomalies, referring to "the *np*" when there is more than one. In fact, if a user uses one of these commands, FEVAHR responds "`Which one do you mean?`," and the user may then enter one of the deictic *np*s and point to the intended object.

Focusing on an object is simulated by storing the Garnet representation of the object in the global variable ***STM***, simulating iconic short term memory. The effects of the *Find* commands are to place an appropriate object in ***STM***. The *Go to* commands combine a *Find* with moving the simulated FEVAHR near the simulated object in the simulated environment. After a *Follow* command, the simulated FEVAHR goes to the appropriate object, and then stays near it if the user uses the mouse to move the simulated object. The *Stop* command cancels the *Follow* command, if necessary, and replaces whatever is in ***STM*** with **NIL**.

2.4 Primitive Actions

The primitive actions are *find*, *finddeictic*, *gotofocussed*, *staywithfocussed*, and *stop*. SNeRE, the SNePS rational engine, maintains the association between SNePS nodes representing primitive actions and actual Lisp functions that effect them. These Lisp functions represent the actions at the PM Level. The effects of the primitive actions are as follow:

(find *object-node*) Finds the description associated with the SNePS *object-node*, then finds the Garnet object satisfying that description, and stores that object into ***STM***.

(finddeictic *category-node*) Retrieves the Garnet object that the user points to or has just pointed to with the middle mouse button, and stores it into ***STM***. The actual *category-node* is ignored at this time, so if, for example, the user says "that red robot" and points to John or to the green robot, the object pointed to will be accepted with no complaint. This may be corrected at a later time.

(gotofocussed) Uses Garnet routines to move the simulated FEVAHR to a point near the Garnet object that is stored in ***STM***.

(staywithfocussed) Uses Garnet routines to move the simulated FEVAHR to a point near the Garnet object that is stored in ***STM***, and then uses the Garnet constraint mechanism to assure that the simulated FEVAHR stays near that other object even if the user moves it with the mouse.

(stop) Cancels the constraint on the location of the simulated FEVAHR, if necessary, and replaces the value of ***STM*** with **NIL**.

Both **gotofocussed** and **staywithfocussed** use the function **(near *figure ground*)**, which returns a point that is "near" the ground object, based on the sizes of both objects, and is between the ground object and the center of the room. For example, Figure 2 shows FEVAHR near John when they are in the lower right quarter of the room. shows FEVAHR near John when they are in the lower right quarter of the room.

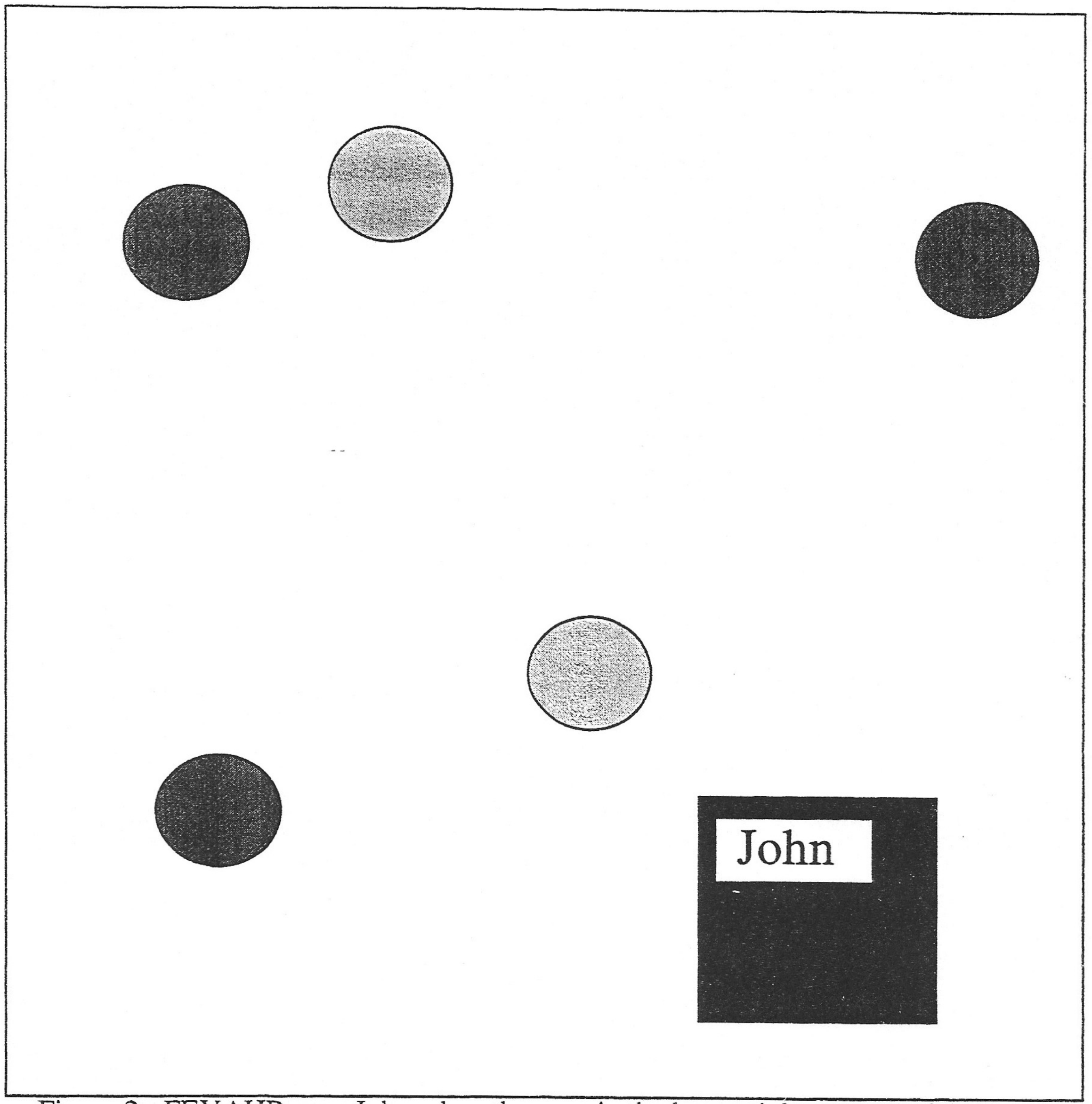


Figure 2: FEVAHR near John when they are in the lower right quarter of the room.

2.5 Complex Acts

The natural language commands *Find* and *Stop* are implemented by the primitive actions *find* and *stop*, respectively.

The natural language command *Go to np* is implemented by the complex act (**go** ***obj***), where ***obj*** is the node representing the entity denoted by *np*. The complex act (**go** ***obj***) is performed by first doing (**find** ***obj***), and then doing (**gotofocussed**).

The natural language command *Follow np* is implemented by the complex act (**follow** ***obj***), where ***obj*** is the node representing the entity denoted by *np*. The complex act (**follow** ***obj***) is performed by first doing (**go** ***obj***), and then doing (**staywithfocussed**).

If an *np* is *this* or *that* followed by some *cat*, the parser calls (**finddeictic** ***category-node***), where ***category-node*** is the node that represents the category denoted by *cat*. The parser then returns (**recognize *STM***) as the node that represents the denotion of the entire *np*.

The function (**recognize** ***object***) simulates vision by returning the SNePS node that represents the entity simulated by the Garnet object *object.* If that is the object simulating John or the green robot, the same node is returned that would have been if the *np* had been *John* or *the green robot* in the first place. However, if the Garnet object is one of those simulating a red robot, a new node is created and returned, and a belief is entered into the KL that the entity represented by this node is a red robot. This is the same action that would have been performed by the parser if the original *np* had been *a red robot*.

APPENDIX B: FEVAHR TOP LEVEL DESIGN (Preliminary)

Appendix B-1: Control Computer Data Definition

Constant Definitions:

```
NUM_SONAR = 16
```

Enumerated Type Definitions:

```
typedef enum {
        HFMVRecognition,
        PMLTracking,
        } ProcessAddrType;

typedef enum {
        ScanConicalFOR,
        ScanFOR,
        Saccade
        } ScanType;

typedef enum{
        BITCommMsg,
        VisionUpdateMsg,
        VisionUpdateRequestMsg,
        VisionSystemShutdownMsg,
        VisionSystemResetMsg,
        ImageStabilityUpdateMsg,
        ImageStabilityRequestMsg,
        SonarUpdateMsg,
        SonarRequestMsg,
        HeadTruthUpdateMsg,
        HeadTruthRequestMsg,
        HeadInitMsg,
        BodyTruthUpdateMsg,
        BodyTruthRequestMsg,
        StopBodyMsg,
        BodyMotorSpeedMsg,
        TurretPositionCommandMsg,
        DriveBodyToTargetMsg,
        FindObjectMsg,
        ScanFORMsg,
        PrimeVisionSystemMsg,
        HeadPositionCommandMsg,
        DetectionReportMsg,
        AdjustBodySpeedGainMsg,
        AdjustObstacleAversionGainMsg,
        AdjustHeadToTargetGainMsg,
        CollisionRiskUpdateMsg,
        TargetTrackAmbiguityUpdateMsg,
        PointerTrackAmbiguityUpdateMsg,
        } MessageType;
```

Message Definition

FEVAHR Grammer - ASCII text string containing the restricted robot control command set:
"follow," "stop," "that," "this," "red," "green," "blue," "ball," and "box"

Behaviors
- To be defined by U.B.

BodyMotionCommands
- StopBodyMsg
- MotorSpeedMsg
- TurretPositionCommand

BodyCommands
- BodyTruthRequestMsg
- SonarRequestMsg
- DriveBodyToTargetMsg
- StopBodyMsg

BodySensations
- BodyTruthUpdateMsg
- SonarUpdateMsg

HeadPositionCommands
- PositionHeadMsg
- HeadMovementMsg

HeadCommands
- HeadTruthRequestMsg
- HeadPositionCommand

HeadSensations
- HeadTruthUpdateMsg

VisionCommands
- FindObjectMsg
- VisionUpdateRequestMsg
- ImageStabilityRequestMsg

HFMVCommands
- PrimeVisionSystemMsg
- ScanFORMsg
- ScanConicalFORMsg
- SaccadeMsg

VisionSensations
- VisionUpdateMsg
- ImageStabilityUpdateMsg

Message Structure Definitions:

```
typedef struct {
        Int32               time;
        ProcessAddrType     sourceAddr;
        ProcessAddrType     sestinationAddr;
        MessageType         msgType;
        Int32               msgSize;
} HeaderType;

typedef struct {
        HeaderType    header;
        Int32         timeOfRequest;
        Int32         shape;
        Int32         color;
        Int32         size;
        Int32         velocity;
        Int32         confidence;
}VisionUpdateMessageType;

typedef struct {
        HeaderType    header;
}VisionUpdateRequestMessageType;

typedef struct {
        HeaderType    header;
        Int32         imageStability;
}ImageStabilityUpdateMessageType;

typedef struct {
        HeaderType    header;
}ImageStabilityRequestMessageType;

typedef struct {
        HeaderType    header;
        Int32         sonarDetections[NUM_SONAR]
}SonarUpdateMessageType;

typedef struct {
        HeaderType    header;
}SonarRequestMessageType;

typedef struct {
        HeaderType    header;
        Int32         panAngle;
        Int32         tiltAngle;
        Int32         vergeAngle;
}HeadTruthUpdateMessageType;

typedef struct {
        HeaderType    header;
}HeadTruthRequestMessageType;
```

```
typedef struct {
        HeaderType      header;
        PointType       bodyPosition;
        Int32           integratedSteeringAngle;
        Int32           integratedTurretAngle;
        Int32           translationalVelocity;
        Int32           steeringVelocity;
        Int32           turretVelocity;
        Int32           mototStatusByte;
}BodyTruthUpdateMessageType;

typedef struct {
        HeaderType      header;
}BodyTruthRequestMessageType;

typedef struct {
        HeaderType      header;
}StopBodyMessageType;

typedef struct {
        HeaderType      header;
        Int32           speed;
}BodyMotorSpeedMessageType;

typedef struct {
        HeaderType      header;
        Int32           offsetAngle;
}TurretPositionCommandMessageType;

typedef struct {
        HeaderType      header;
}DriveBodyToTargetMessageType;

typedef struct {
        HeaderType      header;
        Int32           color;
        Int32           shape;
        Int32           timeOfRequest
}PrimeVisionSystemMessageType;

typedef struct {
        HeaderType      header;
        Int32           pan;
        Int32           tilt;
        Int32           verge;
}HeadPositionCommandMessageType;

typedef struct {
        HeaderType                          header;
        PrimeVisionSystemMessageType        primeVision;
        HeadPositionCommandMessageType      headPosition;
        ScanType                            scanMethod;
}ScanMessageType;

typedef struct {
        HeaderType      header;
}FindObjectMessageType;
```

```
typedef struct {
        HeaderType      header;
        Int32           basePan;
        Int32           baseTilt;
}ScanConicalFORMessageType;

typedef struct {
        HeaderType      header;
        Int32           basePan;
        Int32           baseTilt;
        Int32           panRange;
        Int32           tileRange;
}ScanFORMessageType;

typedef struct {
        HeaderType      header;
        Int32           timeOfRequest;
        Int32           shape;
        Int32           color;
        Int32           size;
        VelVectorType   velocity;
        PointType       position
        Int32           confidence;
}DetectionReportMessageType;

typedef struct {
        HeaderType      header;
        Int32           timeOfRequest;
} TrackDataMessageType;
```

PsuedoCode for HFMV Process

```
while (runningFlag)
        msgPtr = GetMsg();
        switch( msgPtr->MsgType)
          VisionResetMsg:
                Init();
                SendMsg(HeadResetMsgPtr);
          VisionShutDownMsg:
                runningFlag = false;
                SendMsg(HeadResetMsgPtr);
          VisionUpdateRequestMsg:
                VisionMsgPtr = TargetTracking(MsgPtr);
                SendMsg(VisionMsgPtr);
          FindObjectMsg:
                currentTimeOfRequest = systemClock;
                SendPrimeMsg(msgPtr, currentTimeOfRequest);
                if (msgPtr->scanType == scanConicalFOR);
                {
                    SendPrimeMsg(pointerDescPtr, currentTimeOfRequest);
                    scanDataPtr = CalculateVector(currentTimeOfRequest);
                    SendPrimeMsg(msgPtr, currentTimeOfRequest);
                    SendPositionHeadMsg(scanDataPtr);
                    SendHeadMotionMsg(scanDataPtr);
                }
                else if (msgPtr->scanType == scanFOR)
                {
                    SendPrimeMsg(msgPtr, currentTimeOfRequest);
                    SendPositionHeadMsg(msgPtr);
                    SendHeadMotionMsg(msgPtr);
                }
                else /* simple saccade */
                {
                    SendPrimeMsg(msgPtr, currentTimeOfRequest);
                    SendPositionHeadMsg(msgPtr);
                }
          DectionReportMsg:
                if (msgPtr->shape == pointer)
                {
                   UpdateTrack(msgPtr)
                }
                else
                {
                   if (msgPtr->timeOfRequest == currentTimeOfRequest)
                   {
                        UpdateTrack(msgPtr);
                   }
                   else
                   {
                        RemoveTrack(msgPtr->timeOfRequest);
                   }
                }
          CommBITMsg:

        endSwitch
        UpdateTrackPosition()
endWhile
```

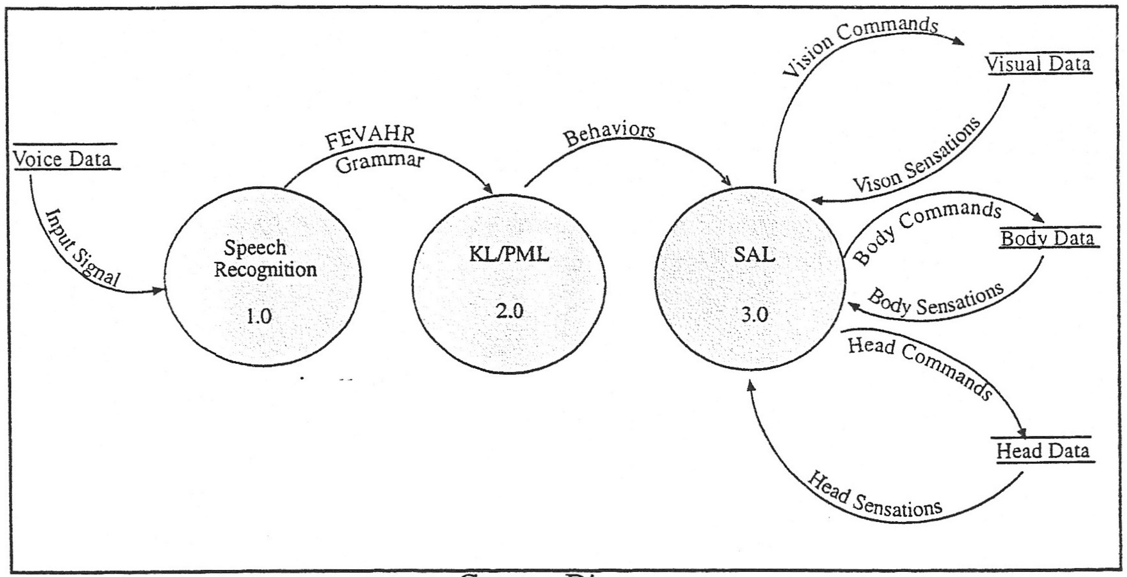
Communication Kernel Psuedo Code

```
<Comminucation Process Monitor>
create and initialize sockets
fork 1 process to wait for data on each socket
while(1)
{
    call 'wait' system call
    when it returns, a child has died...
    if(socket invalid)
    {
        recreate and initialize the socket
    }
    restart the child process to monitor that socket
}
<Socket Monitor>
while(1)
{
  wait for incoming data (via blocking 'accept' system call);
  when it arrives:
  fork off another child which will...
    look up destination socket in lookup table based on message type
    send message out through that socket
  ...while it's parent watches for the next chunk of data
}
```

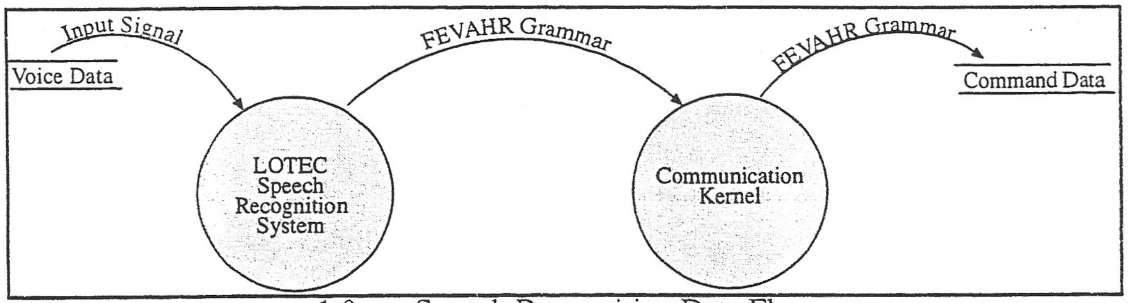
```
<Process Hierarchy>
                    communication
                      process
                      monitor
                     /      \
                    /        \
                 socket     socket
                 monitor     monitor
                /  |         |  \
               /   |         |   \
          message  message  message  message
          handler  handler  handler  handler
```

- Communication process monitor oversees socket monitor processes
- Socket monitor processes wait for data to come into socket and spawn message handlers
- Message handlers examine message and distribute it appropriately

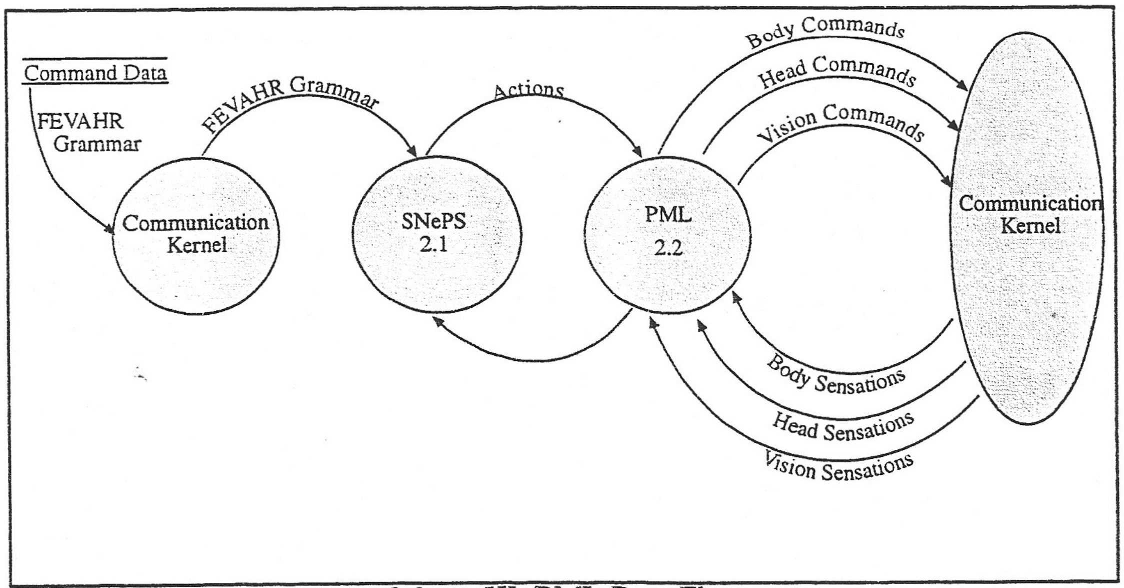
APPENDIX B-2
CONTROL COMPUTER DATA FLOW



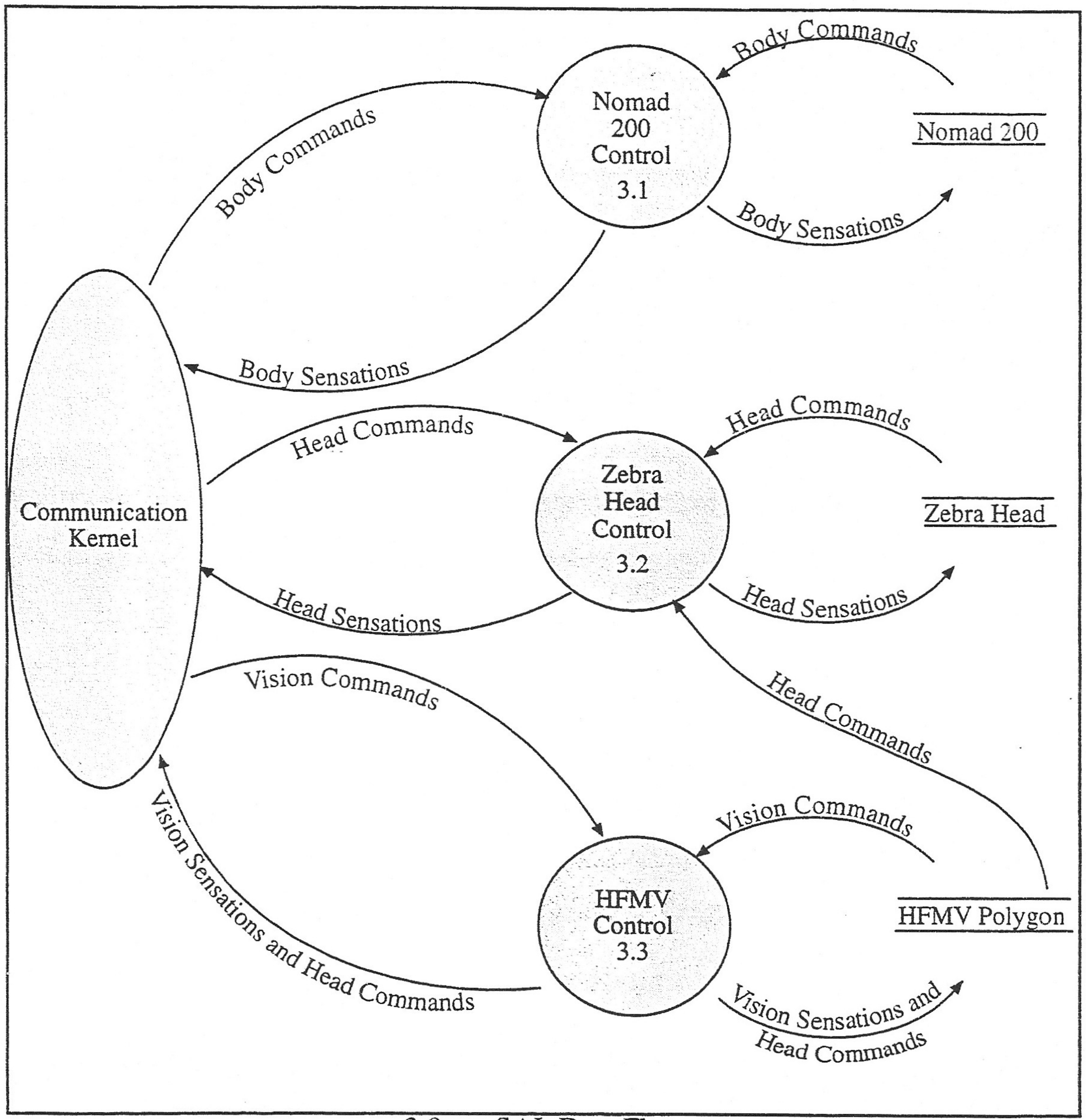
Context Diagram



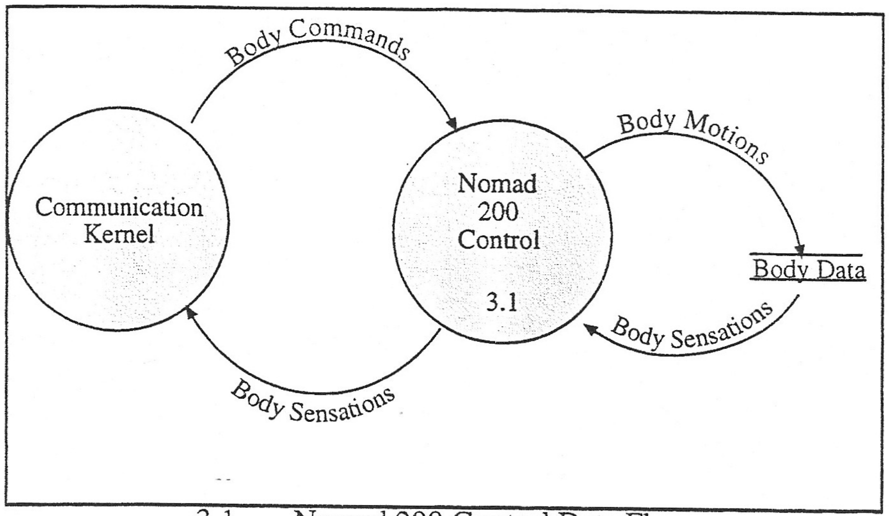
1.0 Speech Recognition Data Flow



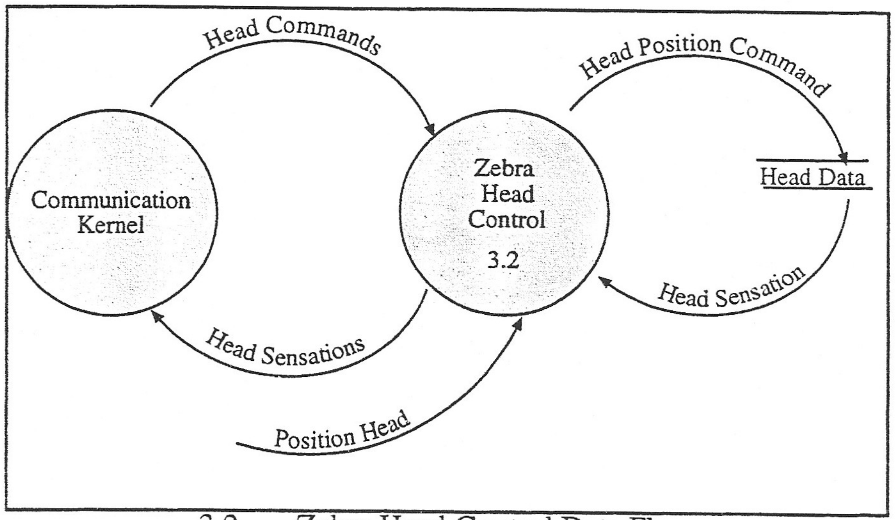
2.0 KL/PML Data Flow



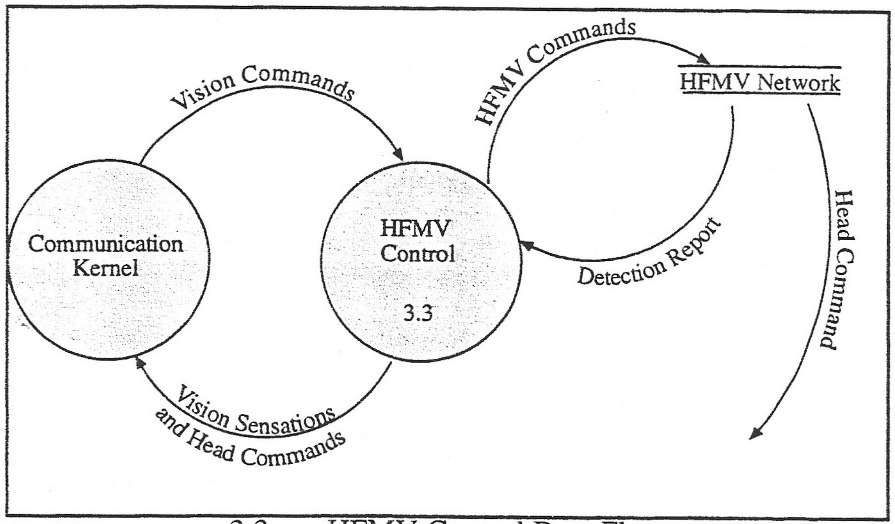
3.0 SAL Data Flow



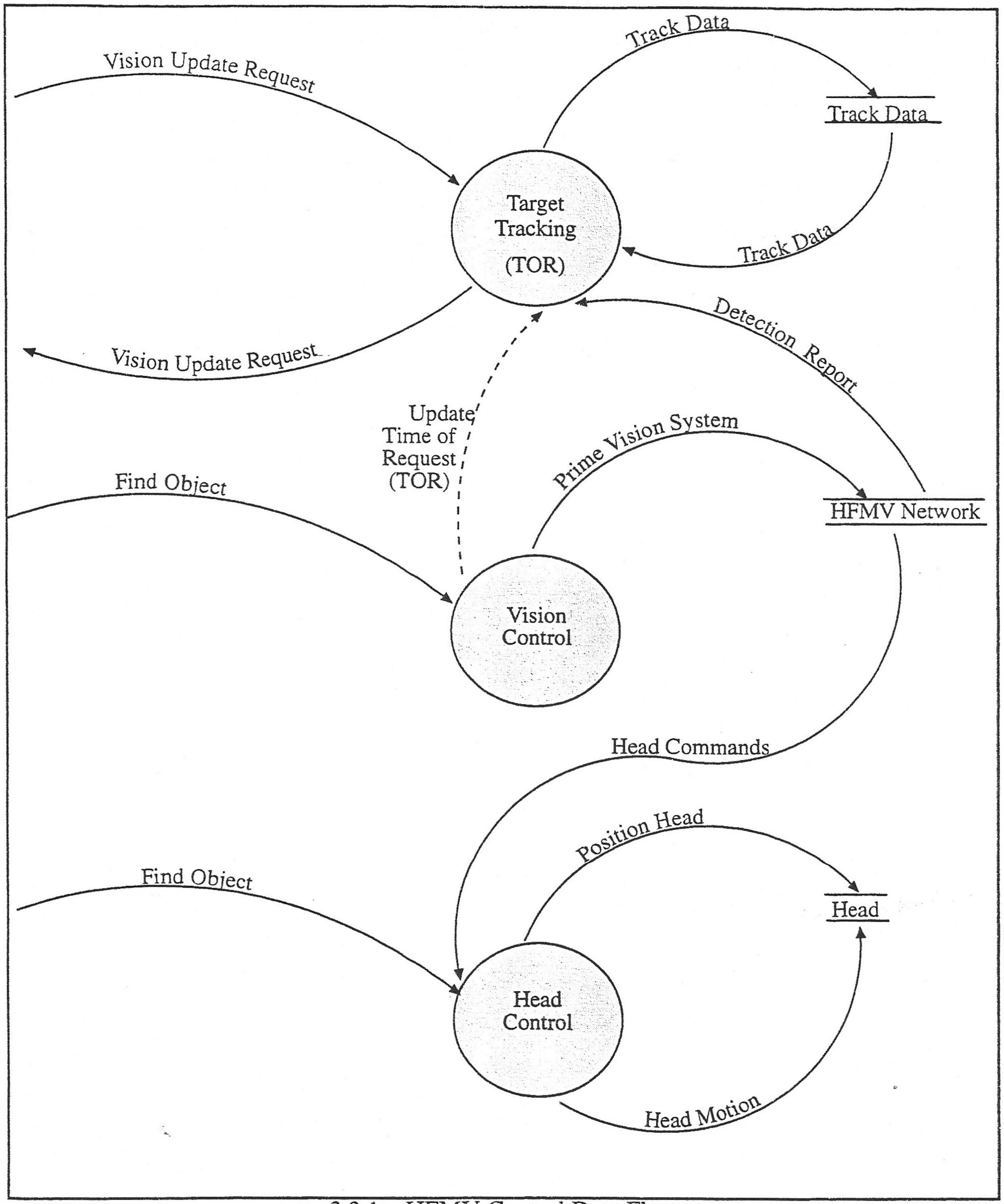
3.1 Nomad 200 Control Data Flow



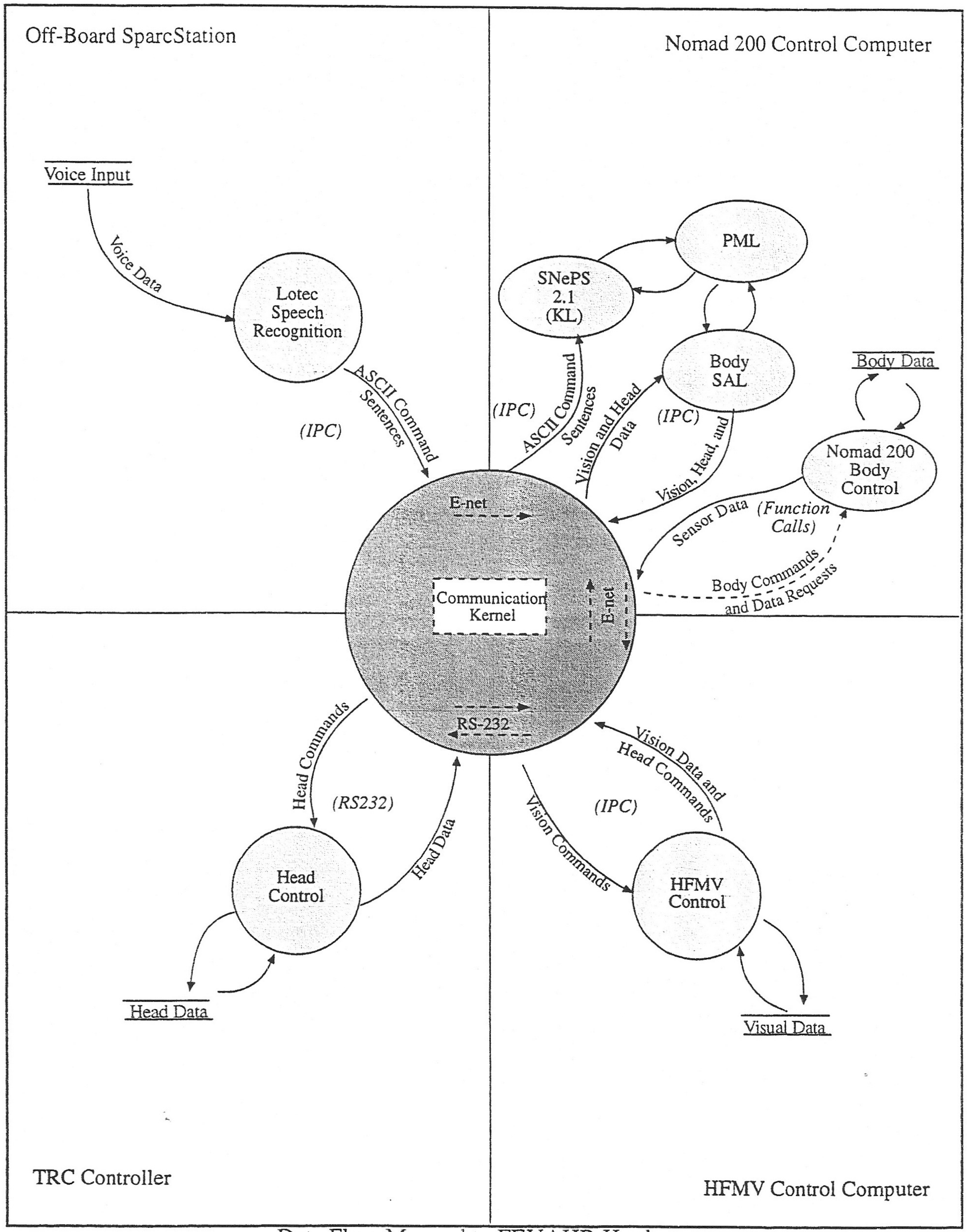
3.2 Zebra Head Control Data Flow



3.3 HFMV Control Data Flow



3.3.1 HFMV Control Data Flow



Data Flow Mapped to FEVAHR Hardware

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE August 10, 1995	3. REPORT TYPE AND DATES COVERED Quarterly Technical Progress Report

4. TITLE AND SUBTITLE

Foveal Machine Vision for Robots Using Agent Based Gaze Control

5. FUNDING NUMBERS

Contract No.:
NAS 9-19335

6. AUTHOR(S)

Andrew J. Izatt

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Amherst Systems Inc.
30 Wilson Road
Buffalo, NY 14221

8. PERFORMING ORGANIZATION REPORT NUMBER

618-9160001

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, TX 77058

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

1

11. SUPPLEMENTAL NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

12a. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The technical objective of the Phase II effort is to develop an Extra -Vehicular Activity Helper-Retriever (EVAHR) robot that uses active foveal vision and an autonomous behavioral control architecture. This Foveal EVAHR (FEVAHR) shall use Hierarchical Foveal Machine Vision (HFMV) to detect objects, track and pursue deictically and non-deictically referenced objects, and avoid obstacles in a dynamic, housed environment. FEVAHR shall utilize a Grounded Layered Architecture with Integrated Reasoning (GLAIR) to provide gaze control for the vision system, including control of all subsidiary functions (e.g., mobile robot platform kinematics). The Phase II work plan consists of the following tasks:

1) Hardware Design & Integration,

2) HFMV Software Development,

3) GLAIR Software Development,

4) FEVAHR Integration & Evaluation.

14. SUBJECT TERMS

Robotics, Active Vision, Hierarchical Processing, Foveal Vision, Multiresolution

15. NUMBER OF PAGES

25

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited