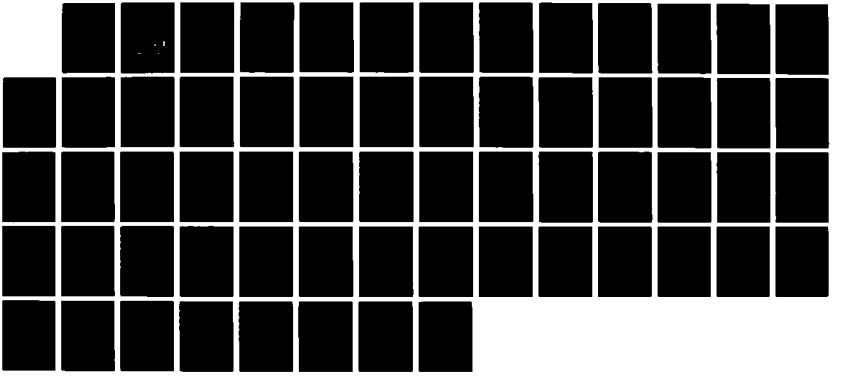
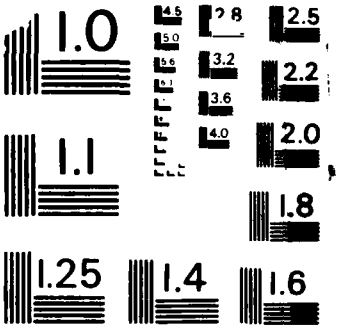
AD-A179 533 DYNAMIC ROBOT PLANNING IN HITAS (HIERARCHICAL INTELLIGENT TARGET ASSESSME..(U) ARMY ARMAMENT RESEARCH DEVELOPMENT AND ENGINEERING CENTER DOV.. 1/1

UNCLASSIFIED R L MEROLLA ET AL. APR 87 ARFSD-SP-87001 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

(12)

AD

AD-E401 672

SPECIAL PUBLICATION ARFSD-SP-87001

DYNAMIC ROBOT PLANNING IN HITAS

RICHARD L. MEROLLA
MARY G. DEVITO
NORMAN P. COLEMAN
KEN LAM

APRIL 1987

US ARMY ARMAMENT MUNITIONS & CHEMICAL COMMAND
ARMAMENT RDE CENTER

U. S. ARMY ARMAMENT RESEARCH, DEVELOPMENT AND ENGINEERING CENTER
FIRE SUPPORT ARMAMENT CENTER
PICATINNY ARSENAL, NEW JERSEY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SPECIAL PUBLICATION ARFSD-SP-87001	2. GOVT ACCESSION NO. ADA179533	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DYNAMIC ROBOT PLANNING IN HITAS		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard L. Merolla, Mary G. DeVito, Norman P. Coleman, Ken Lam		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS ARDEC, FSAC Fire Control Div (SMCAR-FSF-RC) Picatinny Arsenal, NJ 07806-5000		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS ARDEC, IMD STINFO Div (SMCAR-MSI) Picatinny Arsenal, NJ 07806-5000		12. REPORT DATE April 1987 13. NUMBER OF PAGES 61
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Artificial intelligence
Dynamic planning
Cooperation
Dynamic robot planning
Coordination
Expert systems
Spatial and geometric reasoning
Synchronization
HITAS (Hierarchical Intelligent Target Assessment System)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Dynamic robot planning has become an important area of research in the artificial intelligence community. Such research also has many important military applications. Progress in this area offers many economic benefits such as direct application of resulting technologies to the automated factory of the future. HITAS (Hierarchical Intelligent Target Assessment System) investigates issues associated with the intelligent control of a robot with several types of sensor input.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

CONTENTS (cont)

FIGURES

INTRODUCTION

Dynamic robot planning has become an important area of research in the artificial intelligence community. Such research also has many important military applications. Progress in ths area offers many economic benefits such as direct application of resulting technologies to the automated factory of the future.

RESEARCH

This section describes some projects that inspired the ideas behind HITAS. Only major concepts will be described.

STRIPS

In general, a plan is a solution that consists of a sequence of operations that transform an initial state to a goal state. One pioneer project in robot planning is Nilsson's STRIPS project (ref 1). STRIPS consists of several components:

1. The initial world state that contains a list of objects in the workspace and their locations.

2. A set of operators, a list of the action routines that are available to the robot. Each operator description consists of two main parts, effects and preconditions. The effects of the operators are defined by an add list (those facts which become true when an action routine has been completed) and a delete list (those facts that become false when an an action routine has been completed). The preconditions are those facts which must be true of the world state before the action routine can be invoked.

3. The goal state, the state of the world that the user wishes the robot to obtain. The search strategy used by STRIPS is the state-space approach in which each state is represented as a pair. Each pair consists of two elements (current world state and list of goals to be achieved). An example of this is [M2, (G2, G1, G0)] where the goal G2 must be solved before the goal G1 can be solved, etc. The state in the world model where no unsatisfied goals remain is called the terminal state, e.g. (M4, ()).

Creation of a plan is a result of mechanical theorem proving. The search starts by trying to prove, using a resolution based theorem prover, that the final goal G0 is satisfied by the current world state M0. This fact would mean that the initial state is equal to the goal state. Typically, the proof fails. At this point STRIPS needs to find a sequence of operators that will transform the current world state to the goal state. This is done by means-end analysis which extracts the difference between the goal state and the current world state. The difference is defined as the incomplete proof, which consists of clauses from the goal state that remain outstanding when the proof attempt is abandoned. STRIPS then chooses operator that will remove part of the difference between the goal state and the current state.

ABSTRIPS

Although STRIPS was a successful prototype, it was lacking in two main areas. The first deficiency was the search strategy, which was straight depth-first. STRIPS could not distinguish between actions that were critical to a successful plan and those that were unimportant details. The second major deficiency was that the rulebase had no structure.

These deficiencies led to the development of ABSTRIPS that had the ability to recognize the most significant and critical features of the problem and then develop an outline. This allowed the planner to deal with the less important details after the high-level plan was laid out. Another important feature of the ABSTRIPS system was the use of the triangle table, which has two purposes: to monitor the execution of the plan, and to save these plans so they could be used in subsequent planning problems.

ABSTRIPS consisted of a hierarchical level of problem representation. There is an abstract level that omits any details and a ground level that consists of the detailed version. The concept of passing information between levels is very similar to the ideas presented by Albus (ref 2).

The first step taken by ABSTRIPS to solve a problem was to determine the details that will be ignored in the first pass at solution. This is done by the use of the operators' preconditions. The relative importance was indicated by a criticality value that was applied to the hierarchy. For criticality n, all operator preconditions with criticality less than n were ignored. The criticality for each clause in the operator's precondition is determined by the following three rules:

1. If the truth value of a clause cannot be changed by an operator, it was given the highest criticality.

   Ex: TYPE (box1,object). The fact that box1 is an object can never change, no matter what the state of the world.

2. If the precondition for an operator includes a clause that can be achieved once the other preconditions in the same operator are satisfied, then these clauses were given a lower criticality than the first type.

   Ex: INROOM(robot,rx) and INROOM (box1,rx) and NEXTTO(robot,box1) and PLUGGEDIN(x)

   This clause is probably satisfied by the previous preconditions.

3. If the importance of a clause depended on additional preconditions besides those in the previous example, less than the maximal criticality was given to these clauses.

   Ex: PLUGGEDIN(x)

ABSTRIPS started the search by forming a crude plan at the highest level of abstraction and then successively refined it. Like STRIPS, it computed the difference between preconditions and current world conditions and then found a relevant operator to reduce the difference. However, ABSTRIPS did not try to solve the preconditions whose criticality was less than the current crticality. This technique permitted early recognition of deadends, which was a feature not present in STRIPS.

STUART

All planning systems operate by combining actions or subplans so that the total plan satisfies some constraint. Usually ths constraint is a goal or subgoal state. Knowledge-based planners use knowledge of how actions interact with each other or the world to determine which combinations of actions are to be used. The planning technique described by Stuart (ref 3) is the decomposition of high level plans into lower level partial orderings of subplans. The lowest level subplans are then tested for conflicts.

Once the conflicts have been detected, restrictions on sequences of actions are created so that there is conflict avoidance and cooperation. It is Stuart's contention that these restrictions are expensive and that synchronizing primitives can be used to resolve conflicts and produce plans that should be as unrestrictive as possible. The models of plan and action presented by Stuart are appropriate for simple agents engaged in paralled and/or repetitive tasks.

Stuart makes the following definitions:

ACTIONS: Sequences of events that change the state of the world.

EVENTS: Discrete transformations that combine to form actions.

ENVIRONMENT: A world state, plus a set of actions currently being executed.

AGENTS: Acceptor of strings. The formal model of the agent is a nondeterministic finite state automation. It has a set of nodes called agent states and a set of arcs with allowed transitions. Execution of a plan corresponds to some sequence of messages between the environment and an agent. Let A be the set of operations. A sequence of messages will be denoted by a string over the alphabet (begin,end) X A. Any alpha that is an element of the set of operators (begin alpha) corresponds to the agent sending alpha (a plan) to the environment to cause its execution to commence and (end alpha) corresponds to the environment sending to the agent to indicate that the action has been completed. Intuitively, the agent for the simple plan (alpha) is an automation accepting the string [(begin alpha), (end alpha)].

PLANS:

A: Set of operators, (op1, op2,...,opn)

M: Set of memory states.

S: Set of signals.

For any alpha that is an element of A: alpha is a plan for executing a single agent.

For any memory state that is a member of the set M, and any signal that is a member of the set of the set S: (setm), (send s), and (guard m s) are synchronizing primitives.

If p1 and p2 are two plans, then p1:p2 is the plan to execute them in sequence, p1 || p2 is to execute them in parallel, p1 | p2 is to execute one or the other by nondeterministic selection.

Stuart, like Georgeoff, uses correctness conditions imposed on actions and events to ensure that, given a state of the world, a particular robot is allowed to perform a particular action. This is referred to as the interaction problem. Stuart models the world as a set of propositions. Events are constrained to add or delete propositions in the world state. The synchronizing primitives (SEND s), (SET s), and (GUARD s) correspond to arcs in the automation that have side effects on plan execution without changing any messages with the environment.

GEORGEOFF

Georgeoff (refs 4 through 6) describes a process model that is a finite state transition graph that can be used to synchronize plans and allow for concurrency where possible. A formal definition of the process model is: (S,F,C,&,P,ci,cf) where,

S = a set of world states s1,s2,...,sn and at any given instant the world state, which is defined as the conditions that are true of the world at a given moment.

S* = the set of all finite sequences over S.

F = the set of atomic transitions that are the primitive objects that compose actions and events. They are the arcs in the graph.

C = the set of control points that are the nodes and states.

& = the process control function that maps one state to the next. Given a control point and an atomic transition, the process control function returns a new control point.

P = The correctness condition associated with each control point that specifies the set allowable world states at the control point. Correctness conditions define what the state of the world must be before that particular action routine can be executed.

ci,cf are initial control point and final control point, respectively.

A general state of execution is defined by the pair (u,c) where,

S* = U1,U2,...Un U1 = s1,s2,...,sn C = c1,c2,...,cn

The state of execution e1 = (Us1,c1) directly generates the next state of execution e2 = Us1s2,c2), if either of the following facts are true.

Fact One:

&(c1,tr) = c2 and (s1,s2) is an element of tr

This means that the following is true: First, there is a transition from c1 to c2 by way of transition tr. Secondly, there is also a transition from the world state s1 to the world state s2 by way of tr.

Fact Two:

c1 = c2

This fact is true if the following conditions hold. Robot i stays at the same state from time ti through time ti + 1; therefore, robot i has performed no action. However, the world around it has changed during that time frame because of some transition made by another robot. This fact will allow robot i to make a transition to the next state automatically, without having to perform an action routine.

Georgeoff also explains how to check for possible interference between two or more actions. Consider two actions: action(A) and action(B). It is given that action(A) is at control point c1, action(B) is at control point c2, and the current state of the world is such that P(c1) and P(c2) are satisfied. If action(B) continues by executing the atomic transition given by &(C1,tr) = c3, this action would result in the following conditions becoming true:

1. There is a new world state

2. Action(B) is at control point c3

3. Action(A) is still at control point c1

Did the execution of the atomic transition by action (B) result in the possibility of action(A) failing? Georgeoff advises that before action(B) continues, it should first checked to see that if it would continue, at this time, could it result in the possibility of action(A) failing? If this possibility is true, then the atomic transition made by action(B) may cause action(B) to interfere with action(A). For action(A) not to fail the following must be ture: s1 element of P(c1) and P(c2) and (s1,s2) element of tr implies s 2 element of P(c1)

The definition above has the following meaning:

S1 element of P(c1) intersect P(c2) means that s1 is an allowable state at both c1 and c2; therefore, S1 is at a valid state.

(s1,s2) element tr means that there exists a transition from s1 to s2 in the world state; therefore, the next state to be attained is an allowable sequence in the world model.

S2 element of P(c1) means that s2 is an allowable state at c1; therefore, action(A) will not fail in the current world state.

In conclusion, Georgeoff believes that if the process model that he developed is correctly implemented, the system will provide for maximum concurrency while avoiding conflict.

HARMON

The coordination and cooperation of distributed systems requires a sophisticated communication protocol. Harmon (ref 7) proposed such a protocol. Two types of messages are used in this protocol: plans, that control robot action and the state of the world, and reports, that maintain the consistency between the distributed parts of the world model. This type of architecture will allow for the synchronization of multiple robots in a distributed network.

In Harmon's work, the functions of each robot are distributed among three subsystems: sensor, control, and knowledge base. The sensor subsystem analyzes all sensory data and makes this information available to the other subsystems. The control subsystem processes the symbolic sensory data and generates the next plan. The knowledge base consists of domain specific rule sets applicable to the robot's functions.

A plan consists of a plan name, an initiation condition, a trigger condition, and a plan action. The plan is represented as a production rule. The initiation condition indicates what the state of the world must be before the plan can be invoked. The termination condition indicates the situation after which the plan will never again be invoked. The trigger condition indicates what the state of the world must be before the next subplan can begin its execution.

A report message communicates either plan status information or information about some part of the world model. Plan status reports convey information about a plan's execution state to the plan initiator. A plan will generate plan status reports indicating the success or failure of a subplan. Reports containing world model information are responses from report plan status.

The cooperation of these distributed systems is coordinated by a blackboard mechanism that represents the world model for each subsystem. The highest level of the hierarchy in each subsystem can access the blackboard to update its own world model and then incorporate this information into its own decision making process.

ALBUS

Research in hierarchical control for robots in the automated factory of the future is currently being performed at the National Bureau of Standards under the guidance of Albus (refs 2 and 8 through 12). General plans are generated at the highest level of a tree and are further decomposed at each successive level in the hierarchy.

The decomposition of these plans into subplans requires the analysis of three relevant pieces of data (fig. 1): (1) sensory information extracted from the environment, (2) reports generated from the lower-levels of the hierarchy representing the results of previous actions, (3) difference between the current state and the predicted state. If these two states are not equal, then one of two actions can be taken: either the plan expectation must be changed, or a corrective action must be generated that results in the current state being equal to the predicted state.

Communication between the various modules in the hierarchy is through a mailbox. This mailbox is used not only to pass information to the higher and lower levels in the hierarchy, but it is also used as a synchronization mechanism, because when data is written to or read from the mailbox, it carries with it a time tag indicating when the information was presented to the mailbox. This scheme prevents the possibility of old data, which may no longer be valid, from being mistaken as current valid data.

ROSENSCHEIN

The artificial intelligence community has shown a growing interest in distributed system architecture. Distributed systems are processes where a collection of agents cooperate and coordinate their activities to achieve some goal. The advantage of these systems is: increase in efficiency, and increase in capability.

Almost all plan executions require that tasks be performed in a rigid sequence. For example, in the HITAS project, the robot must pick up a previously decided on missile before it can load the cannon. Therefore, for a plan to be successfully completed, the agents must not only perform the actions necessary to complete the plan, but these actions must be executed in accordance wth the execution sequence.

Rosenschein (ref 13) describes two types of distributed systems: planning for multiple agents, and distributed problem solving. Planning for multiple agents is a paradigm where a single agent constructs a plan and then decomposes this plan into subplans and distributes these subplans among various agents. This type of paradigm is implemented in the HITAS project. Distributed problem solving is the process in which a group of agents work together to compose a plan to solve a problem.

Rosenschein presents a framework for multiagent planning in a distributed system that consists of several parts: formalism, communication, and ordering.

Formalism is the manner by which knowledge about the agent's functions and capabilities are represented. Communication primitives are the means by which agents transmit and receive information about the state of the environment. Ordering mechanisms are implemented into the decision making process of the control strategy. This ordering structure contains a list of the actions and states that must have been successfully completed before the given action can be initiated. Agents will only accept a task if the associated preconditions of that task are true. The method used by an agent to accept a task only if the preconditions of that task are satisfied insures multiagent synchronization.

GOALS AND DESIGN OBJECTIVES

Some of the basic research goals and objectives associated with this project are discussed. In general, an attempt will be made to make explicit the paradigms used in conducting the experiment. Many ideas have been incorporated from various areas of artificial intelligence and integrated into the hierarchical arrangement that best suits the domain under study. Each major subproblem is addressed using a tool or model with characteristics which clearly indicate its use for that subproblem. The overall experiment is a variation of the Albus hierarchical intelligent control system.

Research Goals and Achievements

1. Develop a test bed integrating a multisensored robot into an hierarchical arrangement of decision and control modules similar to the Albus hierarchy. The designed software will be highly modular to facilitate the implementation and evaluation of different approaches at each level of the hierarchy or within a module in the hierarchy (a hybrid, multiparadigm approach).

HITAS integrates a multisensored robot consisting of limit switches, a rangefinder, infrared beams, and a force/torque sensor into the overall system framework. Its control strategy uses the Albus report and plan passing technique as a means of communication between the various agents in the hierarchy. The interface between these modules is extremely clean and will facilitate the implementation of more sophisticated hardware and software enhancements.

2. Investigate how a knowledge-based and knowledge-rich approach can extend the performance of current intelligent robotics applications; tailored rule sets are distributed to control the system operation.

The HITAS project integrated domain specific rule sets into each component in the system. For example, there are specific rule sets for the following: the robot, the robot sensors, the vision system, and the various agents. This approach allowed the distribution of subproblems to the various components in the hierarchy and then let these components apply their rule sets to solve the subproblem. It also made it easier to implement a distributed system and permit easy modification for subsequent work.

3. Discover the important components of world modeling, specifically in the area of geometric and spatial modeling and reasoning, and how these components can be incorporated with rules for plan generation.

In the HITAS project, the two most important components of world modeling and reasoning are: (1) gathering the minimal amount of information needed to perform realtime spatial and geometric reasoning, and (2) presenting that information in an organized fashion. The minimal information gathered is in the form of sets of points, which are used internally to manipulate the objects. The information presented to the other modules in the hierarchy were in the form of organized frames. For example, the information passed up the hierarchy to the tank commander module from the situation assessment module was a frame that consisted of two slots: the enemy target's name and the linear distance between the friendly tank and the enemy vehicle. This form of organized frame passing between the various modules in the hierarchy has a two fold advantage: easily implements information hiding a software engineering technique and facilitates later development where the less sophisticated algorithms will be unplugged, (e.g., replace the more sophisticated algorithms such as high level primitives that run on the VICOM vision system). This will allow the system to make military choices based on more realistic battlefield information.

4. Investigate generic planning primitives that use data in a geometric and spatial data base in a structured manner.

The HITAS projects used nine generic planning primitives, which the tank commander and/or loader organized into a dynamic sequence of events that represented the overall plan. These nine primitives can be performed in any sequence. The responsibility of correctly ordering this sequence is shared by the tank commander and loader modules. This plan could be altered as the result of reports generated by the situation assessment module (spatial and geometric reasoner), which indicated a change in the prioritized target list.

5. Explore the additional reasoning mechanisms and modeling techniques necessary to deal with an uncertain and changing environment.

HITAS needed two types of mechanisms to deal with a dynamic environment: to recognize a change in the environment and to replan if the environment change warrants one. The mechanisms used to recognize a change where: the reacti command is in Val and the constant monitoring of the situation assessment area is by a vision system. The mechanism for replanning was a forward chaining control strategy.

6. Investigate a planning technique where a single agent generates the overall plan to be carried out and then assigns the responsibility of different portions of the plan to his agents.

In the HITAS project, the overall plan to be carried out is generated by the tank commander module. It assigns the responsibility of loading the cannon to the agent loader and assigns the responsibility of generating a prioritized list to the agent situation assessment. These three agents coordinate and

cooperate their operations to successfully carry out the function of a tank crew. HITAS proves that this technique is adequate for this domain.

7. Experiment with planning horizons to determine when updates to the models must take place, anu how these updates will affect the planner's actions in a real time dynamic environment.

Through experimentation with the HITAS project, it was realized that there were classes of planning primitives. The classes were distinguishable by the effect that they could have on the high-level world models. For example, the dropping of a missile or the realization, by the robot, that a missile was not present belong to the same primitive class, because their effect on the loader world would be the same: a decrease in the number of missiles available.

8. Evaluate the use of multiple models and representation schemes within these models.

The HITAS project uses several models; each of the agents in the hierarchy used a different model to represent its knowledge. The tank commander module represents its knowledge in the form of frames; the loader module in the form of if-then constructs; and the situation assessment module in the form of x, y points. HITAS establishes the use of multiple models and representation schemes in a robot planner.

9. Carry out a systematic set of experiments which will exercise the entire system and especially the recovery planning and geometric modeling and reasoning components.

A systematic set of experiments was carried out by forcing the system to handle a number of dynamically changing environments. Some of the changing environments tested were: placing unexpected obstacles into the environment, changing the configuration of terrain features, changing the configuration of military vehicles, dropping missiles, changing the prioritized target list, and removing missiles from the missile rack. HITAS responded appropriately for each of these cases.

10. Evauate the hierarchical hybrid approach identifying the class of problems that can be modeled by the prototype and compare the approach to other important systems such as STRIPS, ALBUS, GEORGEOFF, TOUR, MERCATOR, etc.

The hierarchical hybrid design of the HITAS project allowed for: concurrent processing, dynamic planning, increase in efficiency, and increased capability. One result of HITAS is that the proposed hierarchical-hybrid approach has considerable merit. While HITAS is specifically oriented towards solving problems in the military domain, these problems are also widespread in the nonmilitary domain.

Design Objective and Achievements

The HITAS project is the integration of a multisensored robot into a distributed network of computers running a hierarchically organized set of specialized modules some of which are expert systems. The various modules in this hierarchy could use completely different paradigms to sole a subproblem. The definition of interfaces between major system modules is explicit and will not change, but the internal workings of the modules may, allowing for the investigation of alternate approaches to resolving issues at the subproblem level. However, this hybrid approach will remain integrated so that once a particular approach had been evaluated at the subproblem level, its effect on the performance of the entire system could be subsequently determined. Modularity will also facilitate future experiments with more than one robot and various sensor subsystems which are certain to change over time.

The following design goals were achieved in the HITAS project: specialized modules, different paradigms for different subproblems, and a clean interface between the major modules. Specialized modules are designed to perform specific activities and are responsible for performing a distinct task. In the case of HITAS, the specialized modules are: tank commander module that performs the task of a human commander in a tank, loader module that performs the task of a human loader in a tank, and situation assessment module that presents vision data in an organized fashion to the tank commander concerning the situation assessment area.

There are several paradigms used in the HITAS framework. The paradigm used by the tank commander is a forward chaining expert system that uses if-then constructs to instruct and oversee fire commands. The paradigm used by the loader is a DFSA, whose nodes are the nine-robot primitives and whose arcs indicate which robot primitives must have already been performed before a transition can be made to the next state. The paradigm used by the situation assessment modules is also a forward chaining expert system whose world model is integrated with vision information from the situation assessment area. It uses its rules to manipulate its data base, to generate the prioritized target list.

HITAS was specifically designed with a clean interface between the major modules (tank commander, loader, and situation assessment) in the hierarchy. This design has a two-fold advantage: (1) it allowed easy implemention of the software engineering technique called informatoin hiding, and (2) the integration of more sophisticated algorithms into the HITAS hierarchy will only require the unplugging of one module and the integration of the new algorithm.

PROBLEMS

Experimental Domain

The environment chosen to achieve the research goals is a simplified model of a battlefield. The task is to:

(0) Obtain sensor data

(1) Assess a situation

(2) Generate a prioritized list of targets with associated ammunition selection

(3) Replan if necessary

(4) Load the artillery piece using the robot

This task list is subject to interruption at any time by external events such as arrival of new targets, failure to load the ammunition, etc.

The domain consists of two basic regions. The artillery loading area uses a variety of robot mounted sensors while the situation assessment area uses a ceiling mounted vision system (fig.2).

The artillery loading area contains a PUMA 560 robot, three missile types (two of each type), and a cannon. The model representing the artillery loading area contains the location of the artillery ammunition, the location of the robot's gripper, the contents of the robot's gripper, the location of the cannon, and the contents of the cannon.

The situation assessment area consists of simplified terrain features (mountains and hills) and friendly and enemy military vehicles (tanks and resupply vehicles). The model representing the situation assessment area contains the present configuration of objects of the type previously described. A camera, positioned above the target assessment area, returns information concerning the location and type of objects present. This vision system is referred to as the situation assessment vision system. All situation assessment vision scenes are two dimensional silhouette scenes. This system does not attempt to process a nonstatic vision frame.

Modeling Issues

Research has indicated that often one of the most difficult areas in problem solving is representation. In the artillery loading area, the primary form of input is provided by sensors (force/torque, limit switches, infrared beam, and rangefinder). In the situation assessment area, the primary form of input is provided by vision; therefore, the decision making involves the objects and their relationships. This requires geometric or spatial modeling and reasoning.

The objects (terrain features and military vehicles) in this area are modeled by frames. A frame is a knowledge representation and not a vision frame. The slots inside each frame are initialized by the situation assessment vision system, and updates are made during planning horizon processing.

As an example, the military vehicles frame consists of the following slots:

Distance: The distance between the artillery loading area and the military vehicle.

Active/Inactive: This indicates whether or not the military vehicle is operational or not.

Friendly/Enemy: The friend or enemy relationship between the military vehicles and the artillery loading area.

Dangerousness: A list of military vehicles that can destroy it.

Offensive Effectiveness: A list of military vehicles that it can destroy.

Visibility: This indicates whether the military vehicle is concealed by terrain.

Location: The coordinates of the military vehicles in the situation assessment area.

Each terrain feature is also represented by a frame which consists of the following slots:

Name: This slot indicated the terrain type (e.g. mountain or hill).

Corners: This record described the four corner points of the terrain feature.

For future work, additional models and reasoning over these models will be required.

Planning Issues

The planning issues addressed in this experiment involve the investigation of dynamic planning problems, such as recovery planning, associated with a hierarchical arrangement of planing modules acting as agents. This arrangement is similar to the Albus model and to the Hierarchical Intelligent Fire Control System (HIFCS) (ref 14) already implemented at ARDEC. Each planning module in the hierarchy is viewed as a planning agent with planning responsibilities and limited action authority. The higher the module, the greater its responsibility and authority. Each planning module contains rule sets associated with its subdomain. Interfaces between each planning module provide for plan passing down the hierarchy and report passing up the hierarchy (ref 7). This message passing scheme facilities plan interruption as well as plan decomposition since report passing up the hierarchy provides higher level modules with information necessary for replanning in case unexpected environmental changes were detected by the lower levels.

In the event that such a change is detected at one of the lower levels, and it is not within that module's authority to make a decision about what action to take, a report (interrupt) to a higher level module is generated. These reports are continually passed up the hierarchy until the report reaches a module that has the authority to act. Once at a module that could make the decision, both

the report and the latest completed planning horizon are considered before a subsequent plan of action is decided upon. A decision is made with these two pieces of information.

As an example, consider the case where a low level module (an obstacle avoidance module) encounters an obstacle that it determines to be unavoidable because it has continually attempted to navigate the gripper around the obstacle but was unsuccessful after two attempts. Assume that this obstacle avoidance module has neither the authority or responsibility to make a decision about how to deal with this problem. The module will then report to the next higher module in the hierarchy, the loading planner module. Assume that it is not the responsibility of the loading planner module to determine what to do in this situation.

In that case it (loading planner module) passes a report up the hierarchy to another module. This process would continue until the report reaches a module whose responsibility encompasses this situation. Once the report reaches this module, that report and the last planning horizon (subplan) that was successfully executed would be evaluated to determine what subsequent action should be taken.

The HITAS aproach is that the design of a system to address these types of planning problems contains many cooperating agents (each having models) either declarative or procedural or mixed that are part of their areas of responsibility. It is important to note that an environmental change by itself is not enough to generate a new plan. The latest completed planning horizon must also be taken into account. Only then can the module make an informed decision concerning the generation of a recovery plan. Also, interrupt information (reports) may consist not only of negative news such as problems encountered during the execution of a subplan, but may signal the successful completion of a planning horizon.

Constraints Applied to the Battlefield Scenario

The following constraints are applied to the battlefield scenario: (1) allowable set of objects in both the situation assessment area and the loading area, (2) configuration constraints for each domain, and (3) environment assumptions.

1. The allowable set of objects which can be placed in the situation assessment area are:

- Military vehicles: one friendly jeep, two enemy jeeps, and two enemy resupply vehicles

- Terrain features: mountains and hills

The allowable set of objects which can be placed in the artillery loading area are:

- Three missile types (two of each type)

- One cannon
- One missile rack
- Unanticipated obstacles

2. The configuration constraints on both the situation assessment area and the loading area are:

In the situation assessment area, the terrain configuration must be laid out before the start of the demonstration. Once the demonstration begins, the terrain configuration remains unchanged; however, it can change in subsequent demonstrations. In other words, the HITAS system is robust enough to handle more than one terrain configuration. A second configuration constraint is that no objects in this domain can be placed on top of other objects in the domain.

In the loading area, the configuration constraints are similar to those found in the situation assessment area. The first constraint is that the cannon must be oriented before the start of the demonstration and cannot change. The final constraint applied to this region of the domain is that no objects may be placed on top of other objects.

3. There are several additional simplifying environment constraints.

- Objects in both world models are distinguishable by colors (e.g., tanks can only be colored red or blue and green tanks will not be allowed in the scenario). This is to simplify vision processing since only a simple vision system is currently available.

- Once the missile is in the cannon, it is instantaneously fired at the chosed target.

- The rangefinder sensor will be turned off once the robot arm is within 5 inches of the cannon breech; therefore, no obstacles can be detected or avoided within that space. If a missile is dropped, a recovery plan is generated which instructs the robot to return to the missile racks and pick up another missile rather than trying to locate the dropped missile. However, a vision stub is provided for in the hierarchy, and future work on this project will incorporate a vision system over the artillery loading area. This will allow the robot to pick up the dropped missile once it has been located by the vision subsystem, and then continue with the current load plan.

- The situation assessment area and the loading area are two distinct regions that do not physically overlap.

Additional Constraints

There are several conditions upon which rules for target assessment are based. The following are a few examples:

- Linear distance between the artillery loading (firing) area and each enemy military vehicle

- Absence or presence of concealment by terrain features

- Position and status of surrounding military vehicles

A few of the conditions that may change the planning horizon of the fire command are:

- Addition of one or more military vehicles in the situation assessment area. This will be detected by the situation assessment vision system and may cause replanning if new targets are to be engaged as determined by the tank commander module.

- Movement of an existing military vehicle. This also may cause target reassessment as determined by tank commander module.

- Selected ammunition may fall from the robot's grasp. This situation causes round replanning.

- Requested ammunition is not available. The missile loading planner may choose an alternative round if it is approved by the tank commander module.

- Unexpected obstacle is encountered along a planned robot arm trajectory. This situation may require an abort of the load plan if it is determined by the robot path planner that the obstacle is unavoidable.

HITAS SYSTEM OVERVIEW

The simulated battlefield models (frame based) are initialized by a vision system in the situation assessment area and the robot sensors in the artillery loading area. Once these slots are filled, the situation assessment expert system begins to apply the rules for situation assessment. The result of the application of these rules is a prioritized target list. This information is sent to the tank commander planner where it begins to instruct the loading expert system to perform the appropriate load command.

Situation Assessment Control Flow

The situation vision system continually monitors the situation assessment area and sends this information to the situation assessment specific frame analysis module. This module analyzes the pixel data to determine what is in the situation assessment area. This information is then passed to the situation assessment expert system for evaluation and model update.

If the situation assessment expert system produces, as output, a new prioritized target list, it is immediately reported to the tank commander planner that a target with greater priority has been located and that a new load and fire

command might have to be generated. It then becomes the responsibility of the planner to determine if there is time to abort the current fire command. If there is time, then it immediately notifies the loading expert system that there is a change in the loading plan horizon. Recall that a planning horizon is a subplan. If the fire command cannot be aborted, then it is also the responsibility of the planner to determine how this affects future actions.

Loading Control Flow

The tank commander planner sends the loading expert system a load command to be performed. The following nine loading command primitives are associated with the loading process:

- Move the robot gripper above the missile bay
- Move the robot gripper to the missile bay
- Pick up the selected ammunition from the missile bay
- Move the robot gripper along the trajectory to the loading position
- Move the robot gripper to the load point
- Load the missile into the cannon
- Move the robot gripper back from the cannon
- Move the robot gripper back to the start position
- Put back the missile currently in the robot's gripper if there is a change in the load plan which requires a different missile

The loading expert system is responsible for analyzing the sensory information reported by the lower levels. If there is an unexpected problem in carrying out the loading command primitive then the loading expert system determines if it can correct this problem locally by using the range finder to avoid the obstacle. If the system determines that it cannot recover locally from the problem, then it reports back to the tank commander planner.

The system reports back two facts: that it could not complete the loading command primitive, and why it could not complete the loading command primitive. It then becomes the responsibility of the tank commander planner to generate changes in the loading planning horizon.

HITAS ARCHITECTURE

Hardware and Software Configuration

Both the hardware and software of HITAS are organized to provide levels of parallelism. This means that the PUMA, a vision system, and the Silicon Graphics computer can all be working on different problems at the same time. The PUMA may be loading a shell, while the Silicon Graphics is doing situation assessment on the latest update of the tank commander's situation assessment world model, and at the same time the situation assessment vision system is continuously taking pictures of the situation assessment area checking to see if any relevant changes have taken place. At any time, the tank commander could detect a change in its situation assessment world model that could change the prioritized target list.

Based on the state of the artillary loading area world model it must be determined whether or not changes in the firing order can be accomplished by aborting the present load plan and then reloading with another shell. In the artillery loading area, interrupts may be generated because of unavoidable obstacles or perhaps because the cannon was moved just before loading, or because the missile was dropped. All these contingencies are supported by the hierarchical arrangement of hardware and software described in figures 3 and 4.

Physical Process Distribution

A distinction must be made between the physical and conceptual representations of HITAS. Physically, HITAS runs on three different CPUs: the Silicon Graphics workstation running UNIX, the IBM PC (with frame grabber) running MSDOS, and the PUMA controller running the VAL II monitor. The physical links between these devices are RS232 lines which allow communication to take place (fig. 3). This configuration is less than optimal even though it permits a degree of parallelism. Viewing HITAS from an operating systems perspective, it would be advantageous to have a parallel process for each major function taking place. That is, for every cooperating agent, a dedicated process and CPU should be running linked through a common operating system and a high bandwidth communications bus. Such a configuration would allow testing of the efficacy of any work done at the software level since the hardware more realistically reflects the modeling being attempted (i.e., parallel cooperating processes).

Since the necessary hardware or software to implement such a highly parallel architecture was not available, attempts to simulate these activities to achieve quasi-parallelism were made. A physical layout and description of the hardware and processes that run the HITAS system are shown in figure 5. There are really five independent processes running concurrently during any execution of the HITAS system (fig. 5). The three processes on the Silicon Graphics machine can never run in parallel since there is only one main CPU, the M68010. The HITAS main driver process actually contains all the high level reasoning modules (tank commander, loading planner, and situation assessment modules). The other two processes resident on the Silicon Graphics machine, Puma com (PUMA communications) and PC com (PC communications), constantly monitor their respective RS232 ports for information coming in from either the PUMA controller or the PC. If

either of the processes is not in a monitor mode, then it is sending information out to the PC or the PUMA (fig. 4). The reason for the intermediate file system is to provide a clean interface between these independent processes. More specifically, a physical link between the silicon and its parallel PUMA controller running the missile loading primitives is desired. To service the communications needs of these parallel subprocesses on the Silicon Graphics (communications models), each with equal quantums of CPU time, the RS232 line should be constantly monitored. Reports passed to the Silicon Graphics from the PUMA or the PC are placed in a transfer file.

Conceptual Process Distribution

There are five processes running concurrently during any execution of the HITAS system. Conceptually, however, there are really only three. These three processes are independent agents each in charge of a particular function. The tank commander module carries out the functions associated with the overall control of the friendly tank. The loading planner module models the function of the friendly tank loader. Situation assessment performs a hybrid function somewhere between a tank gunner and a forward observer. Although the loading planner module and the situation assessment modules are physically separated from what they control (i.e., the PUMA 560 and the vision system on the PC, respectively), conceptually these two modules, along with their hardware, are considered two independent processes. The tank loader module is also viewed as an independent process with the vision and loader modules as subordinate processes.

This conceptual viewpoint is important, since all the agents depend on efficient communications as in a real battlefield scenario. The communication paths are designed to simulate the conceptual framework within the constraints of the available hardware.

Generic Architecture

Each of the major software modules described in figure 4 is designed using the same architecture. As previously stated, subproblems are major goals. The rules and models of each module will differ, but the overall design is the same. Each module can be broken down into four major pieces: a planner (an expert system), a set of structures that represent that module's representation of the world it is concerned with, a report acceptor that serves as an interface between the outside world and its own internal language, and a plan producer which translates its language into a general language that is understood by the outside world.

Higher up the HITAS hierarchy models of each module are more sophisticated and represent greater portions of the outside world than the modules below. The higher level modules have more general rule sets with less detail than lower level modules, reflecting a broader level of responsibility. A diagram that can be used as a template for the architecture of each module is shown in figure 6.

Mechanisms for Concurrency

The HITAS projects involve the integration of three machines and five processes. To successfully synchronize and coordinate the activities needed to carry out the simulated functions inside of a tank, an eleborate mechanism for concurrency is integrated into the decision making process of the various agents in the hierarchy. Concurrency is implemented through the use of files. There are two types of files that are manipulated: (1) transfer files that are used as a means of communication between the high-level models in the system, and (2) lock files that are used to synchronize process and prevent them from reading and/or writing to the transfer file if another process has already accessed the transfer file.

The transfer files are checked by HITAS high-level modules on a regular basis to analyze the reports generated by the lower-level modules. Also, these files are used to pass plans down the hierarchy.

Since the processes are functioning independently, a rather elaborate communications protocol had to be designed to handle the readers-writers problem when attempting to access the transfer files that are considered resource that only one process can access at a time. Therefore, when a process gains access to the file, it notifies other processes that may attempt to open the file that it is locked by locking another file called the lock file. If "1" is in a lock file associated with a particular transfer file, then other processes may not access that transfer file.

Also a protocol has been set up to indicate what process wrote to the transfer file last so the any data in the file will not be mistakenly overwriten or read.

For example, if the PUMA sends data over the serial line to be written to the Puma Transfer File so it can be read by HITAS high-level modules, but the HITAS high level modules don't read the data before the next transmission, then the puma com module must be aware that it must not overwrite these data since it is vital that reports reach HITAS high-level modules.

PUMA communication takes place through the use of two files: (1) Puma Transfer File used to send plans to the robot and to receive reports from the robot, (2) Puma Lock File indicates whether another module is reading or writing to the Puma Transfer File and used to solve the readers-writers problem.

Vision communication takes place through the use of two files: (1) PC Transfer File used to send plans to the vision algorithms and to receive reports from the vision algorithms, (2) PC Lock File indicates if another module is reading or writing to the PC Transfer File. This is the file that is used to solve the readers-writers problem. Testing of HITAS has shown that these mechanisms work well.

Dynamic Planner Structure

The dynamic planning aspect of the HITAS project basically takes place in two modules: tank command and loader. These modules have been broken down into a sequence of states, similar in configuration to a DSFA. As the conditions of each state are met, a command is relayed to the lower levels of the hierarchy signaling the robot to make the transition to the next state in the loading process or sequence. Although these modules are associated with a standard sequence of states, each of them can dynamically reconfigure this sequence. This dynamic reconfiguration of states occurs when an unexpected change is detected in the environmental conditions; for example; such a reconfiguration would occur when an obstacle is encountered, a missile is dropped, there is a change in the prioritized target list, etc.

The standard sequence of events that takes place in the tank commander, loader, and PUMA primitives modules is:

Tank Commander's Plan (control system flow)

Perform situation assessment, choose target, generate a load command, monitor agents, resolve unanticipated activities, update world models

Loader Plan (load command)

Position robot above missile bay, position robot above missiles, pick up chosen missile, position robot above missile bay, robot moves along trajectories toward load point, load missile, robot moves back from load position, robot returns to start position, monitor reports from PUMA primitive modules, resolve unanticipated activities

PUMA Primitive Plan (example: pick up chosen missile)

Reacti (1006), stop, speed 50, tool gripper, openi, move ready, break, move (to chosen missile location), twist, reacti (1001), break, reacti (1006), stop break, speed 30, move (chosen missile).grabit, break, speed 50, move (chosen missile).twist break, move (chosen missile).untwist, break, return report

NOTE: During each step in the planning primitive execution, the obstacle avoidance algorithms can automatically be called by placing the statement "Reacti 1006, obstacle" in each of the robot primitives.

New Plan if Obstacle Encountered

Scan right using the rangefinder to see if the area is free of obstacles (if the area is free, shift right), return scan left using the rangefinder to see if the area is free of obstacles (if the area is free, shift left), return scan upwards using the rangefinder to see if the area is free of obstacles (if the area is free, moveup), if no path is free of obstacles, go back to starting point, return report

Change in Missile Type Needed for New Target

Put missile back, update missile count, pick up new missile type, update current target name, call loader planner

No Change in Missile Type Needed for New Target

Update current target name, call loader planner

The manner in which unanticipated activities (tank commander's plan and loader's plan) are resolved requires additional description. If an unexpected event takes place, both the tank commander and the loader have rules to handle these conditions. The type of unanticipated activities handled by the tank command are: (1) a change in the prioritized target list during the loading process, (2) an obstacle cannot be avoided by the loader, and (3) a missile not being present, etc. The type of unanticipated activities handled by the loader are: (1) the appearance of an unexpected obstacle, (2) reporting to the tank commander the round the loader was to pick up from the missile rack is not present, and (3) the missile is dropped, etc.

Exactly which module resolves the unexpected event has previously been decided and is based on the responsibility of each module. For example, if an unexpected obstacle is encountered, it is the responsibility of the loader to instruct the Puma robot to try and avoid the obstacle by using the rangefinder sensor; if the Puma robot cannot avoid the obstacle, then it becomes the responsibility of the tank commander to resolve this conflict. As the example shows, each module tries to locally solve the problem at hand; however, if it cannot solve the problem or if the solving of the problem is beyond that module's responsibility, then the problem must be reported and resolved by a higher-level module.

Tank Commander

The purpose of the tank commander module is to simulate the decision making functions of a human commander in a tank. To simulate these functions, the tank commander calls five procedures to carry out several high-level tasks. The models used are: situation assessment, pick target, loader, update world models, and speech (fig. 7).

The situation assessment module generates the prioritized target list from the vision data collected in the situation assessment area.

Pick target is responsible for taking as input the prioritized target list, generated by situation assessment module, and producing as output the most dangerous enemy vehicle that can be fired upon.

The loader module uses the Puma robot to carry out the fire command generated by the tank commander module.

At the completion of a successful fire command the world models of the tank commander, loader, and situation assessment modules must be updated. This function is carried out by the procedure update world models.

The speech module reports the decision making process being performed by the tank commander module and its various agents. The generic architecture for the tank commander is shown in figure 8.

Loader

The purpose of the loader module is to carry out the functions of a human loader in a tank. To perform this service, the loader calls upon five modules to assist in carying out these functions. The modules are: Put Puma, Get Puma, Get Next State, Analyze Results, and Abortable (fig. 9).

The module Put Puma places the next robot state (subroutine that the robot should perform next) into Puma Transfer File. This information is then sent to the Puma controller, which directly controls the robot.

The module Get Puma reads the results associated with the last robot action from the Puma Transfer File, and sends a report to the loader.

The action that the robot sould perform next is determined by the module Get Next State. The module implements a DFSA through the use of if-then constructs, to determine this state.

Analyze Results accepts a report from the module Get Puma and analyzes the results of the last action performed by the robot. It determines if the last robot action was a success or failure. If the last action was a failure, Analyze Results must determine why the action could not be successfully completed.

The function of the module Abortable is to determine if the present fire command can still be aborted.

The module Abortdable is used to indicate whether or not there is still time to change the present fire command. This decision is based not only on the present state of the loading process but also on the new enemy vehicle type that determines the type of missile needed. This is important, because if the missile type that is presently in the robot's gripper is the same missile type that is needed for the new enemy vehicle target, then the fire command can be aborted up until the state at which the cannon is fired. In this situation the only change required in the loading process is the cannon angle. However, in the case where the missile types are different, the state at which the present fire command can be aborted occurs much earlier in the loading process because the missile presently in the robot's gripper must be placed back on the rack and a newly chosen missile must be picked up by the robot. Sometimes in this situation the time required to put the missile back on the rack and then have the robot pick up another missile is greater then the time required to complete the current fire command. The generic achitecture for the loader module is shown in figure 10.

Robot Communication

The function of the robot communication module is to synchronize and coordinate the communication between the Puma controller and the Silicon Graphics machine. This module consists of three procedures: Get Puma, Put Puma, and Puma Com; and two files: Puma Lock File and Puma Transfer File (fig. 11). As previously mentioned, Get Puma and Put Puma get and put information concerning robot action to the Puma Transfer File. These modules are the communication link between the Puma Com module and the loader module. The Puma Com module also gets and puts information concerning robot action to the Puma Transfer File. This module is the communication link between Get Puma, Put Puma, and the Puma controller. The Puma Lock File is used to synchronize the reads and writes to the files. It prevents modules from writing over information that has not been read, and it also prevents modules from reading old information that may no longer be valid.

Vision Communication

The function of the vision communication module is to synchronize and coordinate the communication between the IBM-PC and the Silicon Graphics machine. This module consists of two procedures: Get Vis Data and PC Communication and two files: PC Lock File and PC Transfer File (fig. 12).. Get Vis Data gets and puts information concerning vision activity to the PC Transfer File and is the communication link between the PC communication module and the situation assessment module. The PC communication module also gets and puts information concerning vision activity to the PC Transfer File and is the communication link between Get Vis Data and the IBM-PC. The PC Lock File is used to synchronize the reads and writes to the files. It prevents modules from writing over information that has not been read, and it also prevents modules from reading old information that may no longer be valid.

Sample Run

Three examples of a battlefield script are provided and the major events are described disregarding many of the low level details.

Although the examples will only demonstrate one change in the initial planning horizon, HITAS is robust enough to allow for several planning horizon changes during a single fire command.

Example One demonstrates the actions that take place in a scenario if there are no changes in the initial planning horizon. The loader expert system is given a fire command from the tank commander planner after choosing a target from the pirotized target list that was generated by the situation assessment expert system. The loader expert system calls its agents to orient the cannon and carry out the loading process. The agents instruct the robot to: pick up the selected ammunition, move along a planned trajectory towards the cannon, and then load the cannon. Since there are no changes to the initial planning horizon in this example, this sequence of actions will complete the fire command.

Example Two demonstrates the sequence of events that take place when there is a change in the situation assessment area which in turn causes a change both in the situation assessment expert system planning horizon and the loader expert system planning horizon. The loader expert system is given a fire command from the tank commander planner that was generated by the situation assessment expert system. The loader expert system calls upon its agents to orient the cannon and carry out the loading process. In this example, the robot picks up the selected ammunition and begins to move along the planned trajectory towards the cannon. As the robot moves, the situation assessment expert system notifies the tank commander planner that a new prioritized list has been generated as the result of a configuration change in the situation assessment area.

The tank commander planner decides that there is sufficient time to abort the current fire command. Therefore, a new target is chosen, and the loader expert system is instructed to change the loading plan horizon. For this example, assume that the selection of new ammunition is required (not all changes in the situation assessment expert system planning horizon will require new ammunition selection). The robot is then passed a loading command primitive, from the loader expert system, instructing the robot to place the ammunition currently in its gripper back on the rack and then pick up the newly selected ammunition. After the robot has completed these actions, it can proceed to move along the planned trajectory towards the cannon. It then loads the cannon, thereby completing the fire command.

Example three demonstrates the sequence of events that take place when there is a change in the loading area which in turn causes a change in the loader expert system planning horizon. The loader expert system is given a fire command from the tank commander that was generated by the situation assessment expert system. The loader expert system calls its agents to orient the cannon and carry out the loading process. In this example, the robot picks up the selected ammunition and begins to move along the planned trajectory as the loader expert system receives reports from the lower level sensory modules that an unexpected obstacle has been encountered.

This information results in an immediate change in the loading plan horizon. The obstacle avoidance algorithms are invoked and the rangefinder is used to avoid the obstacle. Once the obstacle has been avoided, the loading process can continue; the balance of the planning horizon is still valid and is carried out by the robot, thereby completing the fire command.

Comparison of Methods

HITAS versus ALBUS

HITAS closely resembles some of the work done by Albus at National Bureau of Standards. HITAS uses the concept of report passing up the hierarchy and plan passing down the hierarchy. However, the Albus hierarchy does not propose an intelligent distributed system. HITAS successfully integrates a multiple agent distributed system into its framework and uses a collection of

intelligent agents (tank commander, loader, and situation assessment) that cooperate and coordinate their activities to achieve a goal. This architecture gives HITAS both improved capability and flexibility when compared to nondistributed systems.

HITAS versus STRIPS

STRIPS was one of the first intelligent robotic application systems. Although STRIPS was a successful prototype, its major downfall was that it used a raw depth-first control strategy. Once the system was given a nontrivial problem, it became bogged down analyzing useless information and states, because it did not look ahead into the plan to realize that those operations could not be performed. This resulted in a combinatorial explosion problem, because there was no structure imposed on rule selection. ABSTRIPS was a slight improvement but also succumbed to the combinatorial explosion problem.

HITAS, on the other hand, decomposes planning into well defined functional levels and implements a simple forward chaining control strategy. Global searches are not necessary and even within the functional levels enough domain knowledge is incorporated to provide for good run times and avoid the combinatorial explosion problem. This structure allowed HITAS to handle more complex problems. The relationship between problem difficulty and the amount of CPU needed by HITAS to solve the problem does not grow exponentially. This will be investigated in future work.

HITAS versus GEORGEOFF

Georgeoff has presented several papers that discuss his theory of multiagent and/or dynamic environment. Many of the ideas given by Georgeoff have a sound mathematical basis, and his algorithms are based on set theory. Georgeoff has developed a device called a process model (describes actions open to agents), which if correctly implemented, would allow for maximum concurrency while at the same time avoiding states of interference.

As yet, Georgeoff, has not implemented his theories into a complete system. His progress is being monitored and many of his ideas may be implemented in the future development of the HITAS project.

The synchronization of HITAS is not at all like that proposed by Georgeoff. Instead HITAS uses a simpler interrupt-driven scheme from operating systems theory.

HITAS versus HARMON

HITAS uses the same type of message protocol as Harmon (plans and reports). This message-passing technique allowed for a clean synchronized interface between the various agents in the hierarchy.

The technique implemented by Harmon to represent a plan is similar to the technique used by HITAS. Harmon represents plans by production rules. The

condition indicates what the state of the world must be before the plan can be invoked. The termination condition indicates the situation after which the plan will never again be invoked. The trigger condition indicates what the state of the world must be before the next subplan can begin execution. HITAS represents a plan as a DFSA. The arcs represent the robot actions that must have been successfully completed before a transition can be made to the next state. The manner in which this graph is traversed is based on the knowledge of both the tank commander and the loader agents.

HITAS versus ROSENSCHEIN

There are two paradigms for distributed artificial intelligent systems: planning for multiple agents, and distributed problem solving. Rosenschein and HITAS both implement the first distributed system paradigm.

Rosenschein presents a framework for multiagent planning in a distributed system that consists of several parts: formalism, communication, and ordering.

Formalism is the manner by which knowledge about the agent's functions and capabilities are represented. HITAS represents knowledge about agents in several forms: if-then constructs, graphs, and frames. These three different types of formalism made it possible to represent different types of knowledge into a form that best suited the representation of that knowledge.

Communication primitives are the means by which agents transmit and receive information about the state of the environment. HITAS uses files (lock file and transfer file) as its communication primitive. With the use of these files, concurrent agents were successfully synchronized.

Ordering mechanisms are implemented into the decision making process of the control strategy. This ordering structure contains a list of the actions/states that must be successfully completed before the given action can be initiated. HITAS's ordering mechanism is represented as a DFSA whose arcs represent the robot actions before there can be a transition to the next node.

Dynamic Planning Limitations of HITAS

Currently, HITAS has two major dynamic planning limitations:

1. Lack of a realtime environment. This is due to several facts: vision data collection is performed on an IBM/PC; algorithms for the loader and situation assessment agents each have portions of their algorithms running on two different machines; utilization of files to perform synchronization between the agents; and lack of interrupts.

2. Lack of sophisticated military rules. Decisions in HITAS are based on a limited set of military knowledge. Although the knowledge of how to handle unexpected events is integrated into the overall structure, the system lacks the knowledge to make what military personnel would consider an informed

intelligent decision. The integration of this knowledge would allow for a more realistic battlefield simulation.

CONCLUSIONS

Many valuable discoveries were made during the course of the HITAS experiment. These discoveries should be further explored in future dynamic planning and spatial and geometric reasoning research at Picatinny Arsenal and at East Stroudsburg University.

HITAS is a multiagent distributed system that functions in a dynamic environment. An important step in designing this system architecture is to distinguish the responsibility of the various agents.

There must be a clean interface between these agents. For example, the information provided by the spatial and geometric reasoner must be presented in a form that could be used by the tank commander module. This was facilitated, in the HITAS project, by passing organized plans down the hierarchy and passing organized reports up the hierarchy.

The agents must be synchronized. HITAS provides for synchronization through the use of files that restrict more than one process from acessing information at the same time. This provided for data integrity.

Also, a distributed system that resides on several machines, as does the HITAS project, requires that a communication protocol be provided. HITAS provides a simple protocol where processes on the Silicon Graphics machine constantly poll the PUMA controller and the IBM-PC. This polling is used to determine if there is any information that these machines need from (on to send to) the Silicon Graphics machine.

Another important result from HITAS that is important in the development of a dynamic planning system is to provide for mechanisms that recognize important changes in the environment and have the ability to replan based on these environmental changes. HITAS was designed with these two types of dynamic planning mechanisms integrated into the system framework.

Through the use of a distributed knowledge base approach, many combinatorial explosion problems associated with earlier systems were avoided while simulating parts of the dynamic, complex decision making process of tank crew members.

The multiparadigm approach used by HITAS proved to be extremely beneficial. It allowed implementation of a paradigm that was especially well suited to the type of subproblem trying to be solved.

HITAS provides a firm base upon which to build more sophisticated systems.

If the hierarchical approach is determined to be effective in the simplified 2-D battle scene, the prototype could easily be generalized to address a larger

set of more realistic physical domain problems. Also, by adding more modules to a level of the hierarchy, the test bed could be extended to simulate situation assessment for more realistic battlefield scenarios.

For instance, the spatial reasoning taking place with the present proposal used a simplified two dimensional model. The module and the model associated with two dimensional reasoning could be replaced with one that performs three dimensional spatial reasoning by including heights of objects in the assessment area instead of just planar relationships between objects.

The HITAS approach covers a broad class of problems whose solutions required the modeling of intelligent response for integrated sensory subsystems. The constant monitoring of these sensors in a dynamic environment has profound effects on the planner's decisions. The synchronization of many separate processes is required to monitor the environment so that changes in the environment could be detected and responded to in real time. Future work can address real-time distributed, intelligent systems. Such a hierarchical, distributed architecture as found in HITAS is also capable of subsequent generalization.

RECOMMENDATIONS

One major step in extending HITAS's dynamic planning architecture is to extend the system to allow for multiple robots. A second robot could be used to perform several tasks in the simulated tank. For example: reload the missile rack, open and/or close the cannon hatch, unload the fired missile, etc. This would require augmentation of the generic module architecture so that the world model of each robot would contain a list of its valid states and functional characteristics. The control strategy at the highest level of the hierarchy would have to generate commands to each robot and monitor the execution of each of these commands not only to ensure that they carry out these plans, but also to ensure that they avoid states of execution that should be mutually exclusive (those states which if performed in parallel may cause the robots to interfere or collide with each other).

Two leading researchers in this area are Georgeoff and Stuart. Stuart proposes the use of synchronizing primitives to resolve possible conflicts similar to those used in operating systems protocol to solve the readers/writers problem. Georgeoff suggests the use of a process model, which is based on set theory, to avoid mutually exclusive states of execution. A process model is a finite state transition graph that allows for maximum concurrency while avoiding possible robot conflict. HITAS can be extended to experiment with both approaches.

Currently, HITAS does not run in real time due to several facts: vision data collection is performed on an IBM PC; algorithms for the loader and situation assessment agents each have portions of their algorithms running on two different machines; use of files to perform synchronization between the agents; and lack of interrupts. The vision algorithms on the IBM PC will be replaced with a real time vision system from the PAR Corporation. The algorithms for

the loader and situation assessment agents will each reside on their own machines. The use of files to perform synchronization between the various agents will be replaced with operating system primitives like signals and monitors. This will greatly enhance the speed of plan down the hierarchy and report passing up the hierarchy. Finally, to achieve a real time environment, the system will need to integrate the operating system of interrupts, similar to those found on PUMA controller.

HITAS lacks sophisticated military rules. In the future, additional military rules will come from two sources: military personnel and military reports.

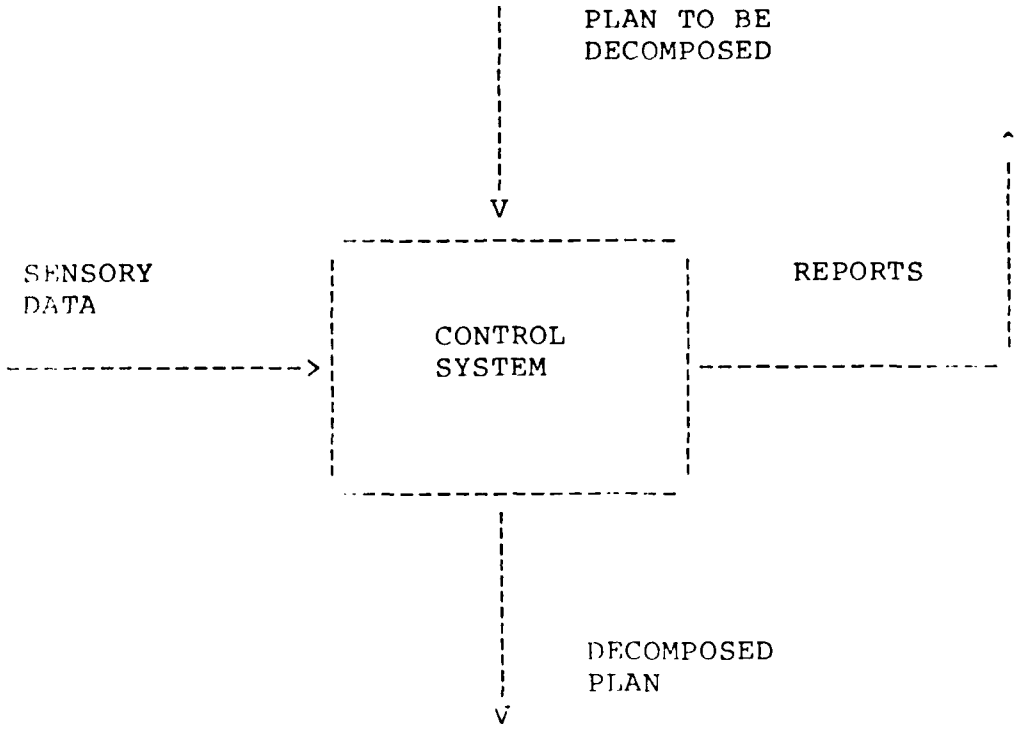
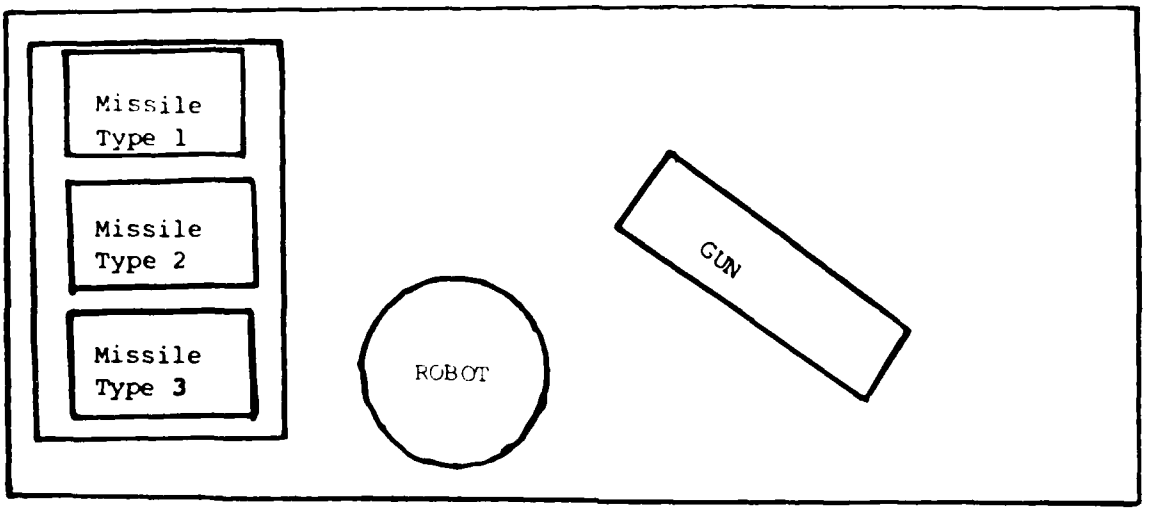


Figure 1. Control system inputs and outputs

Artillery loading area



Situation assessment area

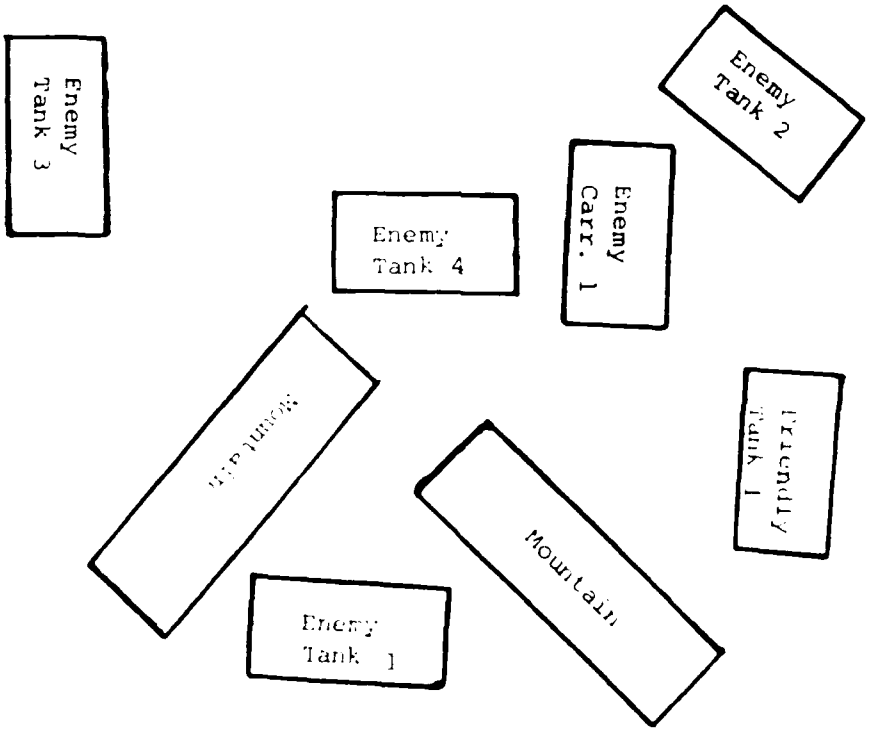


Figure 2. Two basic regions

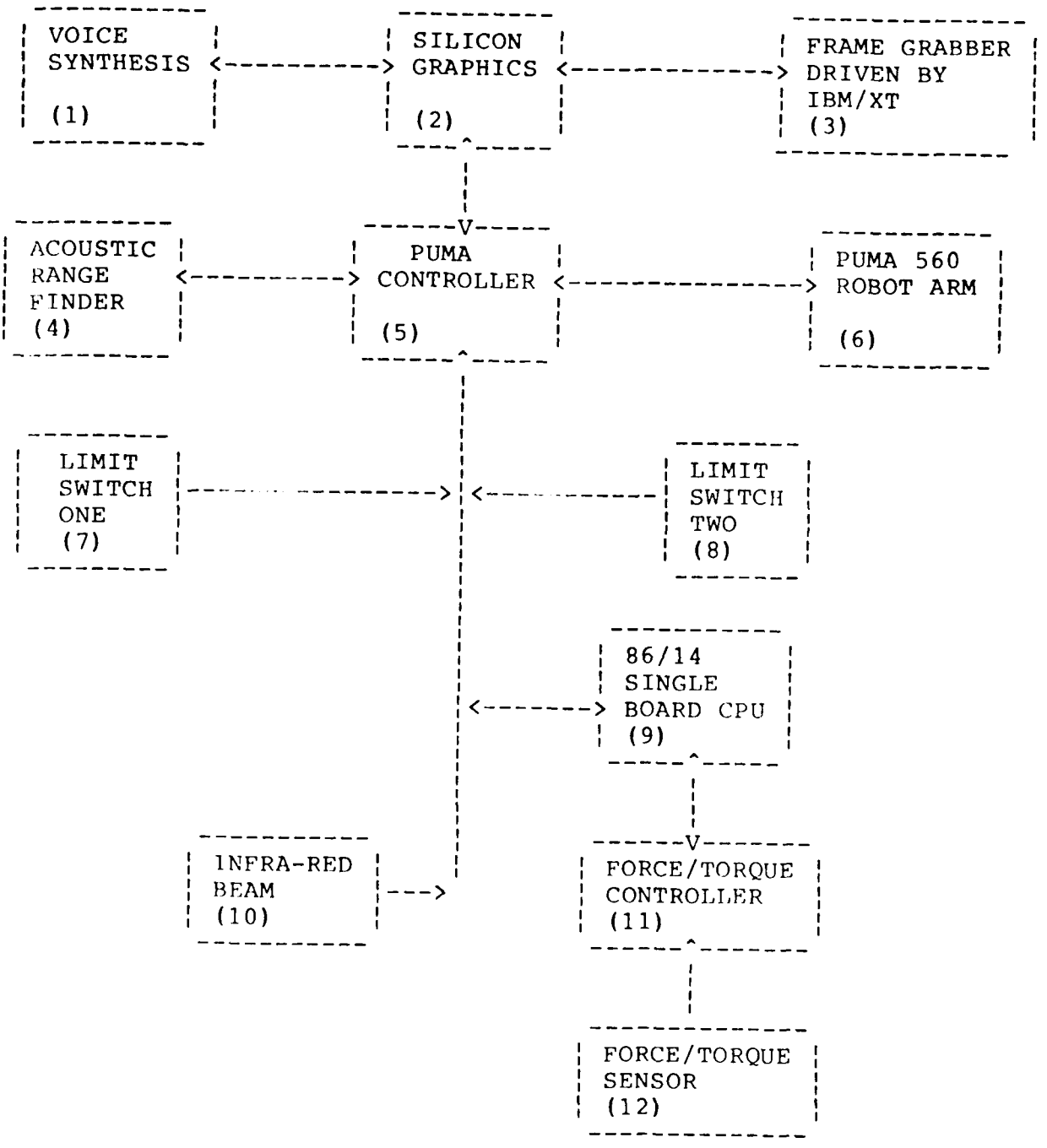


Figure 3. Hardware configuration diagram

(1) VOICE SYNTHESIS: This is a speech synthesis system that will be used to report decisions and actions taken by the overall system.

(2) SILICON GRAPHICS: This is the main computer system. All high level plans will be generated by the software running on this computer. All low level reports will be passed to the software running on this system.

(3) FRAME GRABBER: This is essentially a single-board processor designed to fit into an IBM XT or AT backplane slot. Its only function is to take a picture using a camera and then digitize the image. Once the vision frame has been digitized, the host micro (IBM XT or AT) can process the frame with application software written by our team. The frame grabber comes with 64 K of memory on the board itself. This memory is accessed by means of a memory mapped input/output scheme, that is, memory locations on the XT's mother board are mapped to memory locations on the frame grabber board. With this scheme, programs written on the XT need only manipulate designated areas of the XT's memory. The existence of a frame grabber with its own memory is completely transparent to the application software except for the functions to make a picture and to digitize it. This is quite low level vision processing, but vision processing algorithms are not being investigated.

(4) ACOUSTIC RANGEFINDER: This is a lab-developed sensor which uses a polaroid ultrasonic wafer to transmit and receive sound waves. The driver board for this sensor measures the elapsed time between sound wave generation and the reception of its reflection. The time required for a sound wave to return is a function of the distance traveled by the wave. Therefore, the distance of any obstacles that may be in the path of the rangefinder can be determined. All of the robot movement algorithms have been developed so that the x, y direction movement of the robot's gripper is performed with the rangefinder pointing in the direction of travel. All obstacle avoidance algorithms associated with the rangefinder can be turned on and off so that utilization of information from this sensor is sensitive to its location. This is needed when a missile is about to be picked up or the cannon loaded. The range information can be used since obstacle avoidance routines would interfere with the function being attempted.

(5) PUMA CONTROLLER: This processor is a specialized computer designed to control the PUMA 500 series of articulated arm robots. Its programming language is VAL II.

(6) PUMA 560 ROBOT ARM: This is a six degree of freedom UNIMATE robot.

(7) LIMIT SWITCH ONE: This is a switch located on the bottom outside of the robot gripper designed to indicate when the gripper is within a certain.

(8) LIMIT SWITCH TWO: Same function as 7 above except it is positioned on the opposite side of the end of the robot's gripper.

(Figure 4, Cont)

(9) SINGLE BOARD CPU: This is essentially a separate CPU used to run software to process information returned by the force/torque controller. The processor used for this purpose is an 8086 plugged into an 86/14 back-plane slot. The reason for using this configuration is that the PUMA controller host language does not support many of the highly mathematical functions required to process the force/torque information. Once this information has been suitably processed, it can be passed to the PUMA controller for higher level decision processing.

(10) INFRARED BEAM: A sensor that evaluates the status of an invisible light beam. The light is generated by two diodes positioned between the fingers of the robot's gripper. If the light beam is not broken a signal is generated indicating to the PUMA controller that the gripper is empty.

(11) FORCE/TORQUE CONTROLLER: This is a specialized piece of hardware designed by LORD Corporation to report force along the x, y, and z axes and torques about the x, y, and z axes. These six parameters are continually passed to and processed by the force/torque algorithms running on the 86/14 board.

(12) FORCE/TORQUE SENSOR: A sensor positioned above the robot's gripper that takes readings of forces and torques along and about the x, y, and z axes of the robot's gripper. This information is continually passed to the force/torque controller.

Figure 3. (cont)

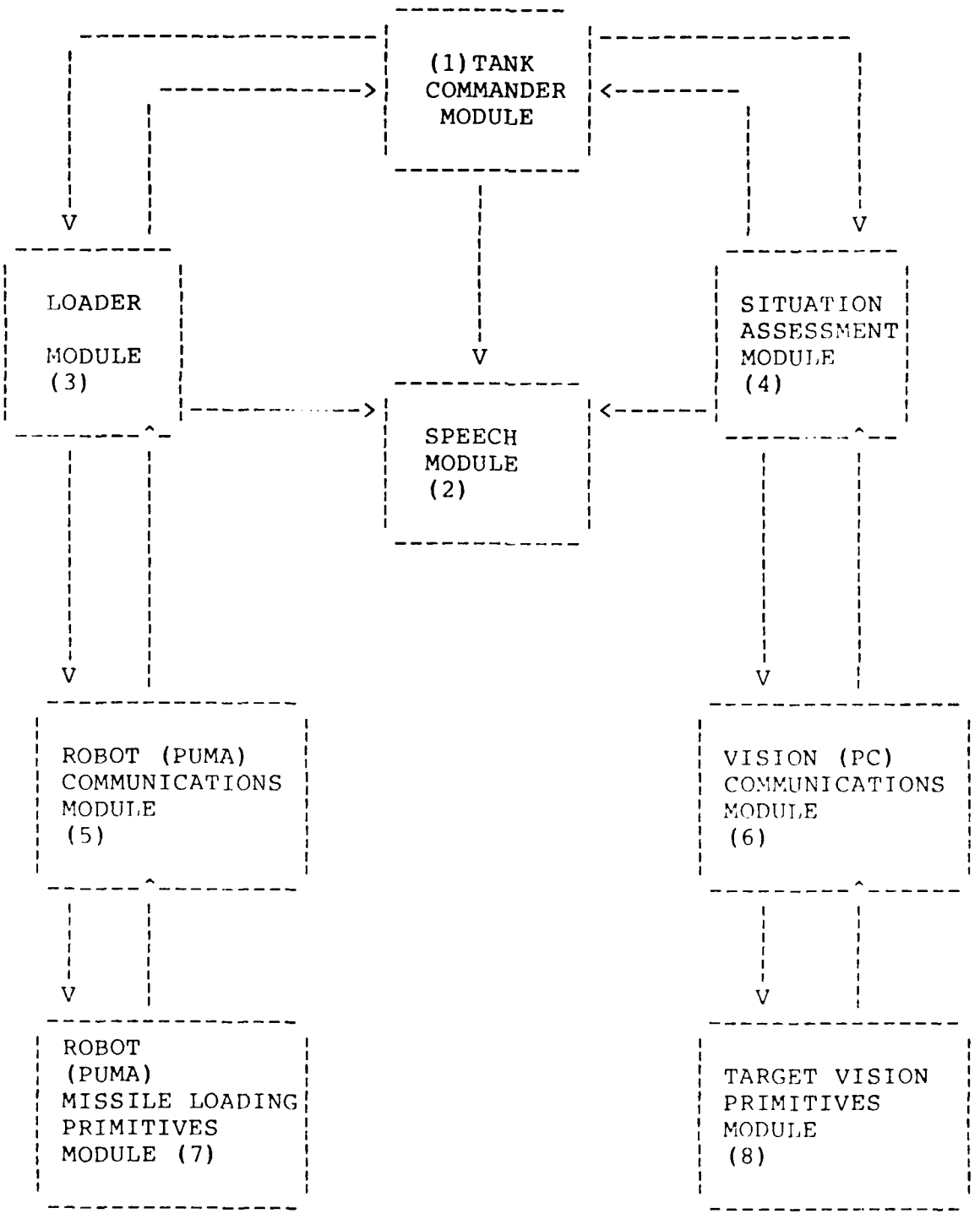


Figure 4. Software configuration diagram

(1) TANK COMMANDER MODULE: This module simulates the functions of a human commander in a tank. It is responsible for the coordination and synchronization of all activities at the highest level of the hierarchy. It receives reports from the situation assessment expert system (situation assessment module) concerning the prioritized target list. Once the tank commander has this report it generates a fire command to the missile loading expert system (loading module). This report is based on both the situation assessment area and loading area world models. A report is generated up the hierarchy to the tank commander once it has been determined that the fire command is successfully completed or aborted. The tank commander then informs the situation assessment expert system on the success or failure of the fire command so that the situation assessment world model can be updated.

(2) SPEECH MODULE: This module is called from the other modules in the hierarchy to report on the status of the battlefield simulation. An example of these reports are: actions of the robot, reasoning in the planning or spatial algorithms, problems that are encountered, etc.

(3) LOADER MODULE: This module receives a fire command plan from the tank commander. It then generates plans down the hierarchy to its agents. These plans are associated with the decomposition of the loading process. The agents are also responsible for generating reports up the hierarchy to the missile loading expert system (loading planner), indicating the status of the various loading planner horizons. Finally, the missile loading expert system generates a report to the tank commander signaling the success or failure of the fire command. The loading planner is also responsible for returning the status of the loading process to the tank commander when he requests it.

(4) SITUATION ASSESSMENT MODULE: This module receives a plan from the tank commander to perform situation assessment. In order for this module to carry out the command, it must call upon its agents and generate plans to each of them, thereby decomposing the situation assessment process. The agents of the situation assessment expert system are responsible for generating reports up the hierarchy to the system (situation assessment). This report will contain the status information concerning the situation assessment process. Once the system has received these reports, it is ready to generate its report on situation assessment (prioritized target list) to the tank commander.

(5) ROBOT (PUMA) COMMUNICATION MODULE: This module is responsible for the synchronization and coordination of the communication links between the PUMA controller and the Silicone Graphics machine.

(6) VISION (PC) COMMUNICATION MODULE: This module is responsible for the synchronization and coordination of the communication links between the IBM-PC and the Silicone Graphics machine.

Figure 4. (cont)

(7) ROBOT (PUMA) MISSILE LOADING PRIMITIVES MODULE: This module receives a plan from the missile loading expert system (loading) to perform a step in the loading process. It is the responsibility of this module and its agents to generate and carry out the sequence of trajectories and end effector manipulations necessary to complete the loading process. This module then generates a report up the hierarchy to the missile loading expert indicating the success or failure of each step in the loading process. At this level, each loading process step is a separate program on the VAL II controller, and can be viewed as a planning horizon for this level. Each planning horizon is a separate step in the overall loading plan and is modeled as a graph (DFSA). As each step is completed, a report is generated from the VAL II controller to the loader module on the Silicone Graphics machine, so that it may update its model as to the success or failure of each planning horizon. The world model is also used to determine if a plan can be aborted in the event that the prioritized target list has changed.

(8) TARGET VISION PRIMITIVES MODULE: This module receives a plan from the situation assessment expert system to take a picture of the situation assessment area. This module and its agents are responsible for extracting pixel data from the area. Once the pixel data are extracted, it must organize the vision data, which will consist of the location of military vehicles, into a report that can be presented to the system. The organization of the vision data requires geometric or spatial modeling and reasoning.

Figure 4. (cont)

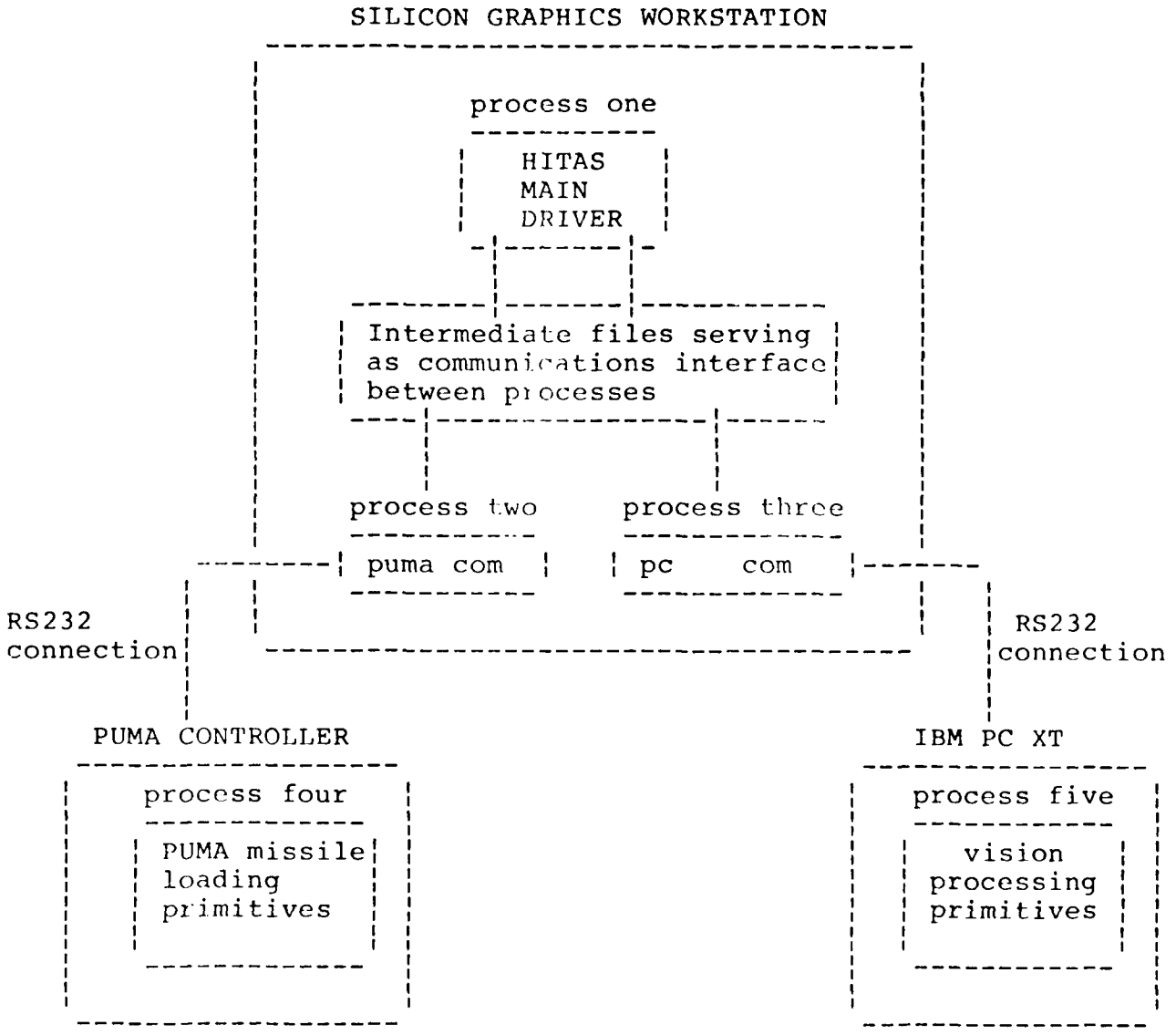
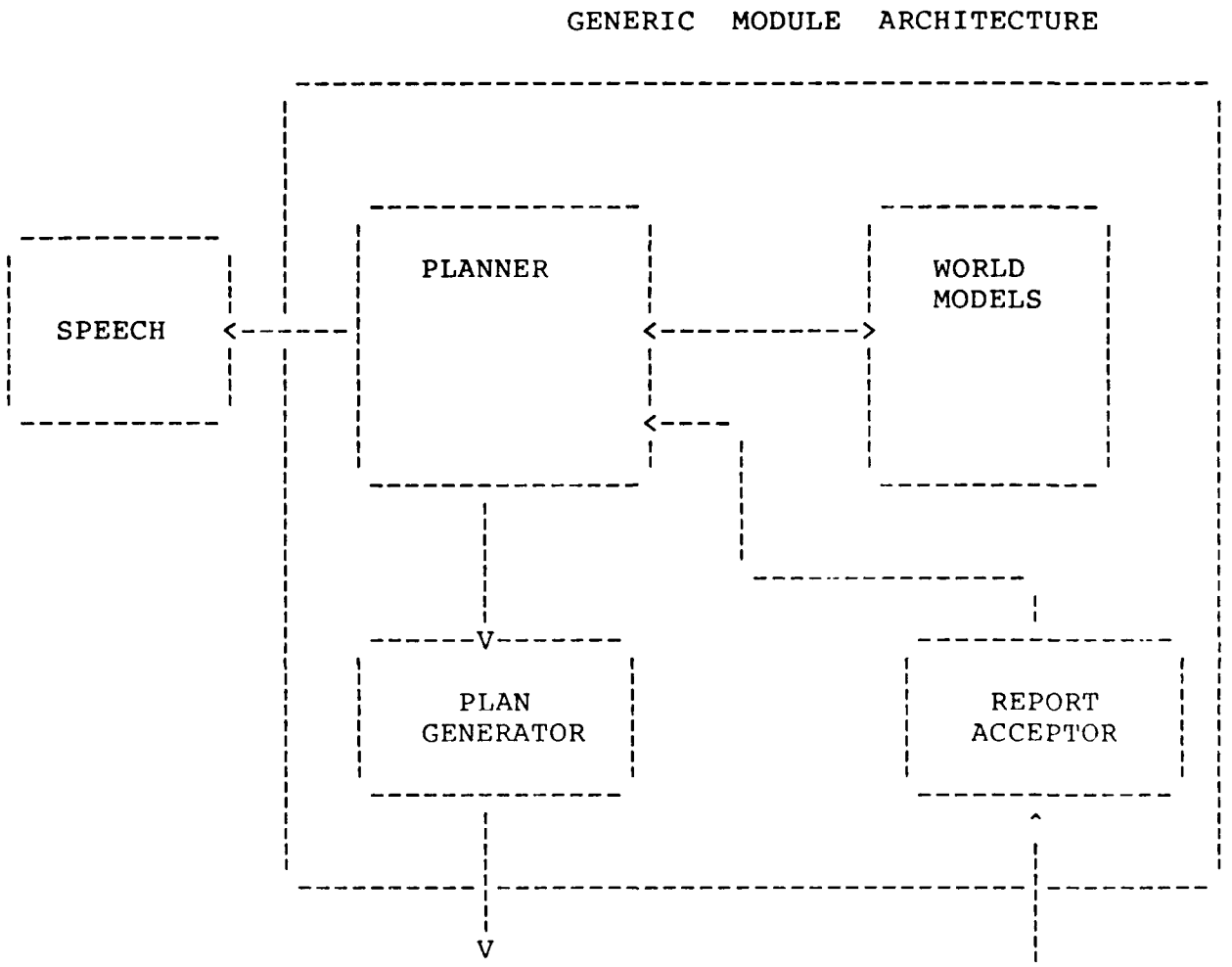


Figure 5. Physical architecture



Note: This architecture is used in modules 1, e, and 4 of the software configuration diagram.

Figure 6. Template diagram for internal architectures

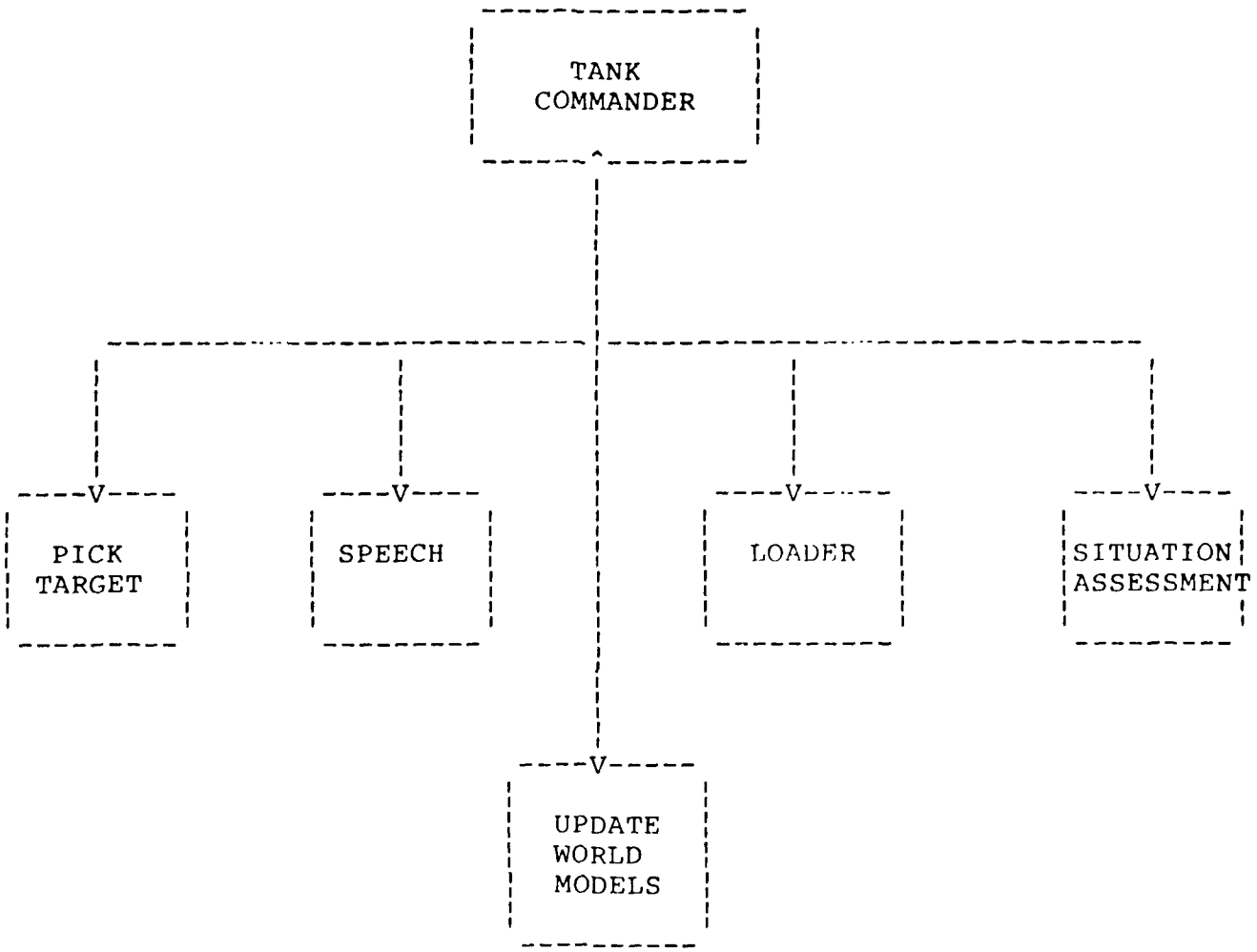
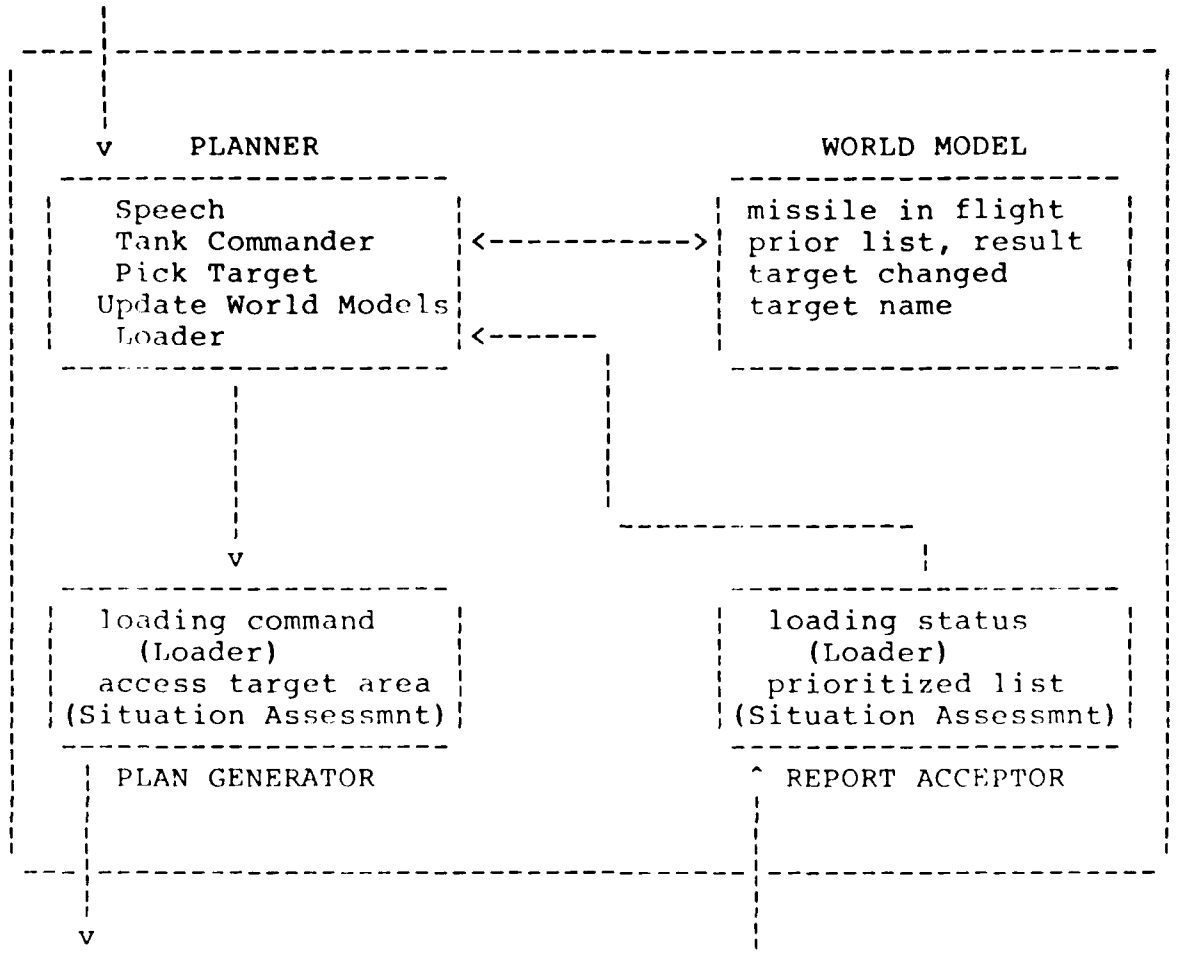


Figure 7. Tank commander module breakdown



missile in flight - the name of the missile currently being word for this load command

prior list - the prioritized target list

result - the result of the last robot action

target name - the name of the enemy vehicle that may be fired upon

target changed - indicates if the prioritized target list has changed

Figure 8. Tank commander architecture

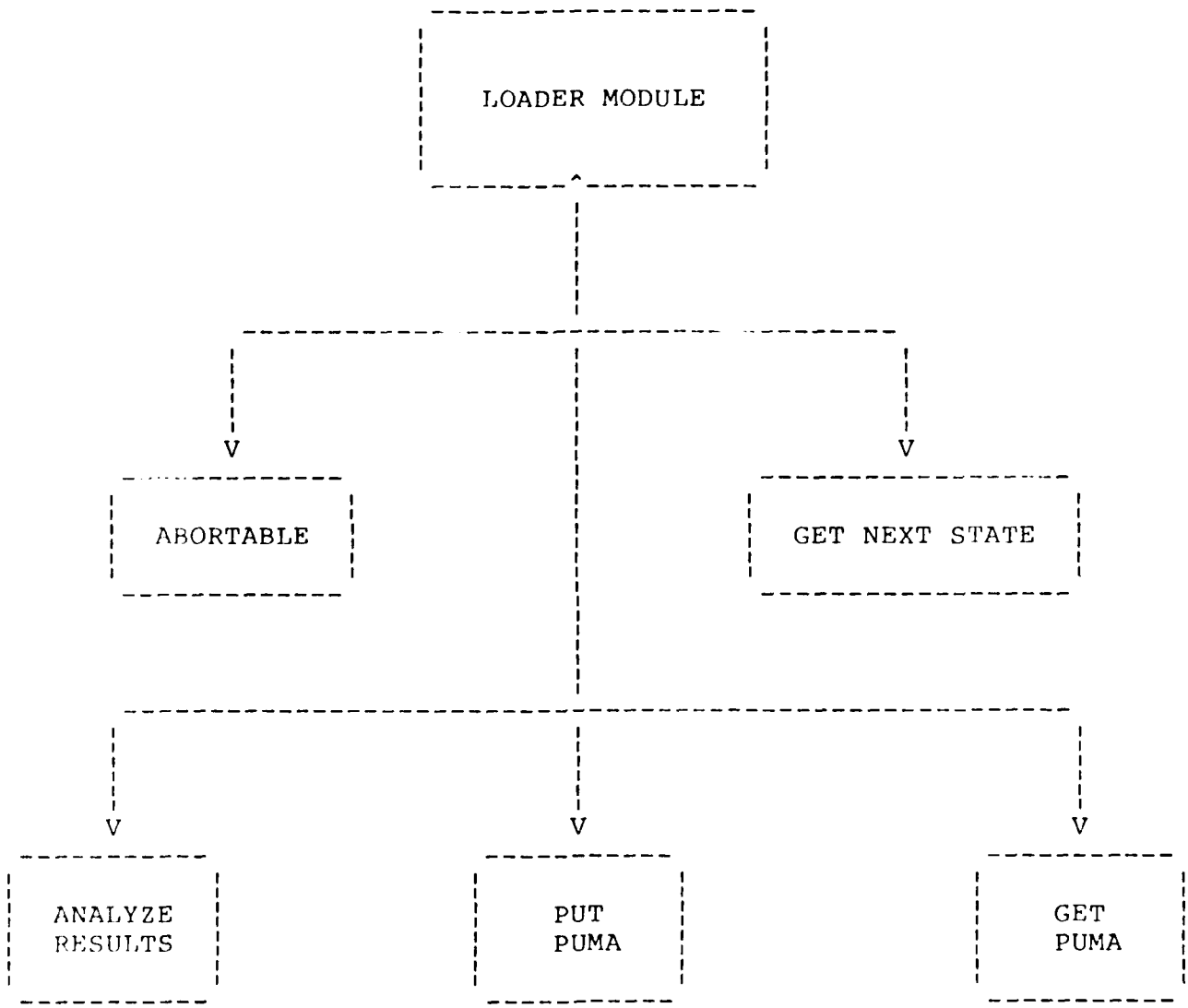


Figure 9. Loader module breakdown

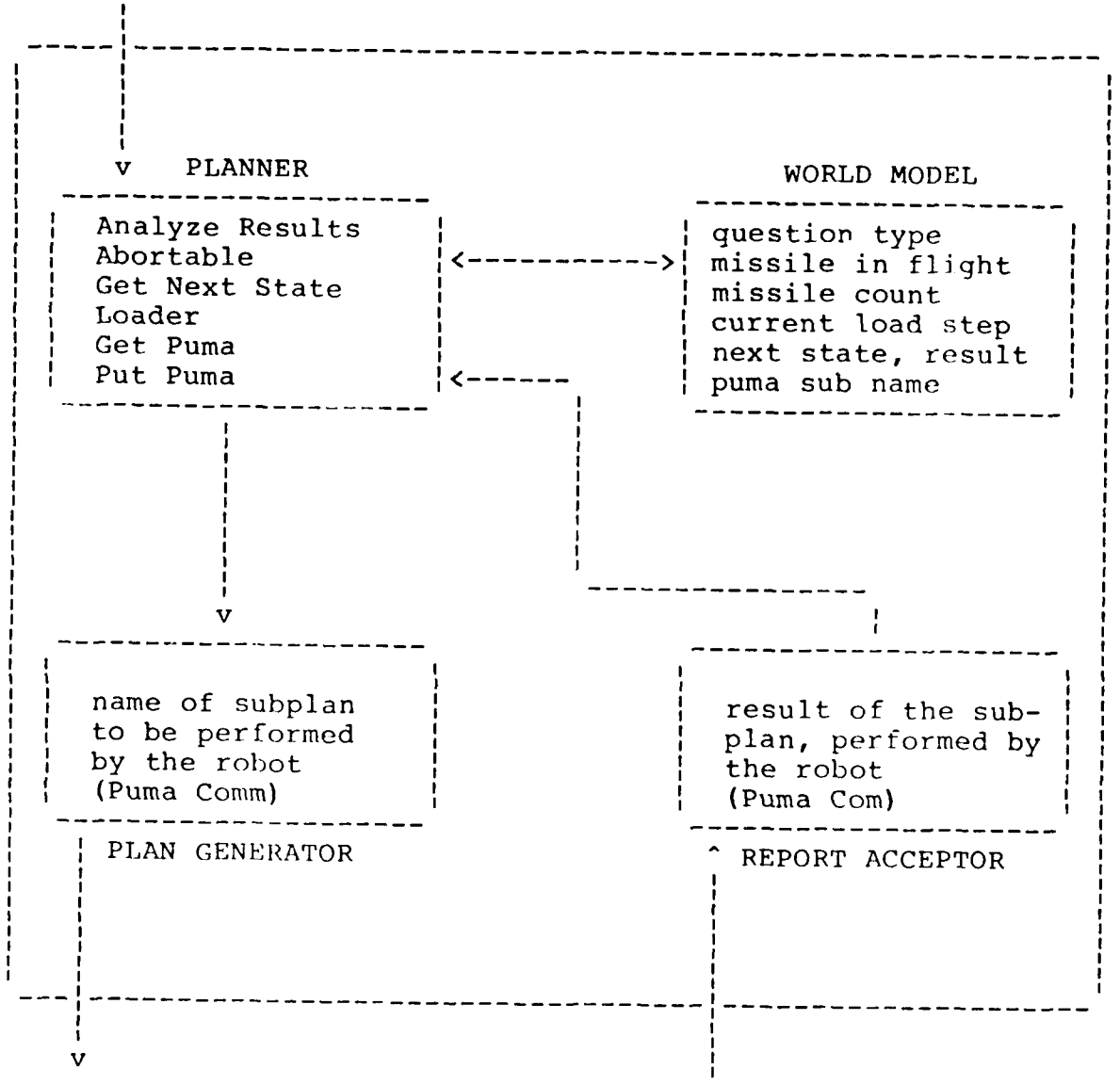


Figure 10. Loader module architecture

missile in flight - missile that is currently being used for this load command

result - the result of the last robot action

question type - the type of question being asked by the tank commander

puma sub name - the name of the subroutine primitve that the robot is to perform

missile count - a structure that keeps track of the remaining missiles

current load step - the current load step (state) that is being performed by the robot

next state - the next state in the loading process

Figure 10. (cont)

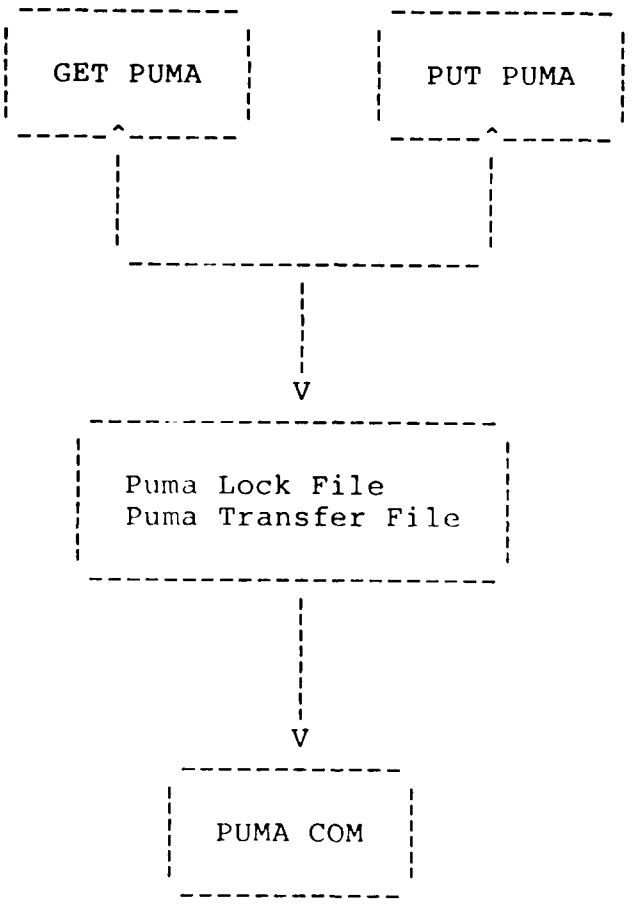


Figure 11. Robot (PUMA) communication breakdown

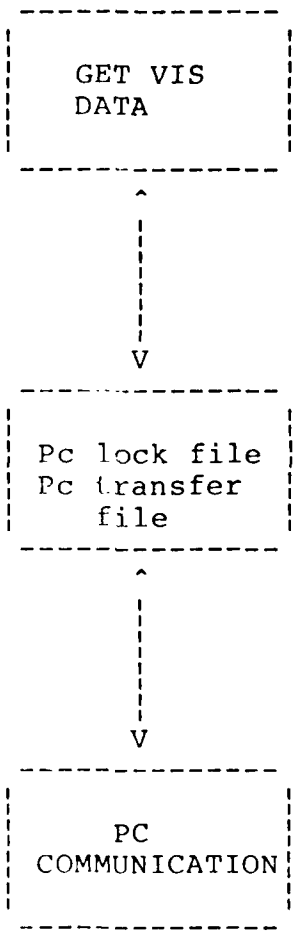


Figure 12. Vision (PC) communication modules

REFERENCES

1. Nilsson, N., "Principles of Artificial Intelligence," Tioga Publishing Company, Palo Alto, California, 1980.

2. Albus, J. S. and Kent, E., "Servoed World Models as Interfaces Between Robot Control System and Sensory Data," National Bureau of Standards, Washington, D.C. 20234, 1983.

3. Stuart, C., "An Implementation of a Multi-Agent Plan Synchronizer," Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85), UCLA, Los Angeles, California.

4. Georgeoff, M., "Communication and Interaction in Multi-Agent Planning," Proceedings National Conference on AI (AAAI), Washington, D.C., 1983.

5. Georgeoff, M., "A Theory of Action for Multi-Planning," Proceedings of the Fourth National Conference on AI, Austin, Texas, 1984.

6. Georgeoff, M., "A Procedural Logic Synchronizer," Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85), UCLA, Los Angeles, California, August 1985.

7. Harmon, S., "Coordination of Intelligent Subsystems in Complex Robots," First Conference on A.I. Applications, AAAI, Denver, Colorado, Dec 1984.

8. Albus, J.S., "An Architecture for Real-Time Sensory-Interactive Control of Robots in A Manufacturing Facility," National Bureau of Standards, Washington, D.C. 20234, 1981.

9. Albus, J. S., "Concepts for a Real-Time Sensory-Interactive Control System Architecture," Industrial Systems Division, National Bureau of Standards, Washington, D.C., 1981.

10. Albus, J. S., "RCS: The NBS Real-Time Control System," National Bureau of Standards, Washington, D.C. 20234, 1983.

11. Albus, J. S., "Theory and Practice of Hierarchical Control," National Bureau of Standards, Washington, D.C. 20234, 1983.

12. Albus, J. S. "Hierarchical Control for Robots in an Automated Factory," National Bureau of Standards, Washington, D.C. 20234, 1984.

13. Rosenschein, J. S., "Synchronization of Multi-Agent Plan," Computer Science Department, Stanford University, Stanford, California 94305.

14. Milligan, S. and Shafer, R., "HIFCS: Hierarchical Intelligent Fire Control System," Army Technical Report 890-DDI-45, 1985.

BIBLIOGRAPHY

1. Aronoff, S. and Jones, G.F., "From Data to Image to Action," IEEE Sprectrum, December 1985, pp. 45-52.

2. Ambler, A., "Inferring the Position of Bodies from Specified Relationships," Artifical Intelligence 6, 1975.

3. Barr, A. and Feigenbaum, E. A., The Handbook of Artificial Intelligence, Vol. I, William Kaufman Inc., 1981, pp. 128-139.

4. Brooks, A., "Symbolic Reasoning Among 3-D Models and 2-D Images,"Artificial Intelligence 17, 1981, pp 349-385.

5. Brooks, A., MODEL-BASED COMPUTER VISION, UMI Research Press, Ann Arbor, Michigan, 1985.

6. Corner, D.; Ambler, R.; Popplestone, R., "Reasoning About the Spatial Relationships Derived from the RAPT Program for Describing Assembly by a Robot," International Joint Conference on Artificial Intelligence, 1983.

7. Davis, E., "The MERCATOR Representation of Spatial Knowledge," Eighth International Joint Conference on Artificial Intelligence, 1983.

8. Doyle, J., "Expert Systems and the 'Myth' of Symbolic Reasoning," IEEE Transaction on Software Engineering, Vol. 11, November 1985.

9. Fikes, R. E. and Nilsson, N. J., "STRIPS: A New Approach to the Application of the Theorem Proving to Problem Solving," Artificial Intelligence 2, 1971, pp. 189-208.

10. Fikes, R. E.; Hart, P. E.; Nilsson, N. J., "Learning and Executing Generalized Plans," Artificial Intelligence 3, 1972.

11. Fikes, R. and Nilsson, N., "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, Vol 2, four numbers, 1971.

12. Forbus, K., "A Study of Qualitative and Geometric Knowledge in Reasoning About Motion," Technical Report AI-TR-615, AI Lab, M.I.T., 1981.

13. Kuipers, B., "Modeling Spatial Knowledge," Eighth International Joint Conference on Artificial Intelligence, 1983.

14. Lieberman, L. and Wesley, M., "AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assemby," IBM J. Res. Develop. 21, 321-333, 1977.

15. Malik, J. and Binford, O., "Reasoning in Time and Space," Fifth International Joint Conference on Artificial Intelligence, 1977.

16. Sobek, R, F, "A Robot Planning Structure Using Production Rules," Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI) UCLA, Los Angles, California, August 1985.

17. Thompson, A. M., "Introduction to Robot Vision," Robotic Age Inc., Vol 1, No. 1, Summer 1979.

18. Wesley, A.; Lozano-Perez, T.; Lieberman, L.; Lavin, M.; Grossman, D., "A Geometric Modeling System for Automated Mechanical Assembly," IBM J. Res. Develop. 24, 64-74, 1980.

19. Wilf, J. M., "Chain-Code," Robotics Age, Vol. 3, No. 2, April 1981.

20. Yin, B., "A Framework for Handling Vision Data in an Object Level Programming Language: RAPT," International Joint Conference on Artificial Intelligence, 1983.

GLOSSARY

Acoustic Rangefinder

A sensor which sends out sound waves to determine if an object is within 9 inches of the gripper, by computing the amount of time that it takes for the sound wave to bounce off the object and return to the sensor. This sensor will be used to detect missiles, avoid obstacles, and locate missiles on the rack.

Albus Hierarchy

A control system whose basic structure is a direct graph. The top module of the hierarchy is the overall plan, which is decomposed into subplans performed by the lower levels.

ARDEC

Army Armament Research, Development and Engineering Center, Picatinny Arsenal, N.J.

Artificial Intelligence

The subfield of computer science concerned with understanding the nature of intelligent action and constructing computer systems capable of such actions. It embodies the dual motives of furthering basic scientific understanding and making computers more sophisticated in the service of humanity.

Artillery Loading Area

That region of the workspace which contains six missiles, one cannon and one robot.

Artillery Loading Vision System

The vision system which is responsible for extracting pixel data from the artillery loading area.

Deterministic Finite State Automation

A labelled digraph with a designated start node and one (or more) final nodes.

Distributed AI System

A collection of intelligent agents cooperating to solve a problem.

Event

The occurrence of one of the steps in the overall plan.

Expert System

A computer system that achieves high levels of performance in task areas that, for human beings, require years of special education and training.

Fire Command

A command given by the fire control center planner to the missile loading expert system, which consists of the chosen target, the ammunition selected, and the cannon orientation.

Fire Control Center Planner (Fire Control Center)

The planner that is responsible for coordinating all activities at the highest level of the hierarchy.

Force/Torque Sensor

A sensor which is used to determine both the amount of force applied along an axis and the amount of force about an axis. These sensory data will be used to guide the missile into the cannon.

Frame

A knowledge representation scheme that associated one or more features with an object in terms of various slots and particular slot values; not a vision frame.

Geometric and Spatial Modeling

A modeling technique that uses the object's location and other features in the workspace to construct a world model.

Geometric and Spatial Reasoning

A reasoning technique that uses the geometric and spatial model to reason about the environment.

HITAS

This stands for Hierarchical Intelligent Target Assessment System the name given to the prototype system.

HITFCS

Hierarchical Intelligent Fire Control System, an ongoing project at ARDEC.

Infrared Beam

A sensor which shoots an infrared beam from one end of the gripper to another. If the beam is broken, the robot has an object in its grasp.

Knowledge Based

Systems that use facts and heuristics given by experts, usually as rules.

Limit Switches

A sensor which uses circuit contact points to indicate if the gripper touches an object with the gripper in a perpendicular orientation.

Plan

A sequence of events which when carried out perform a function.

Planner

The person or module that is responsible for determining the sequence of actions necessary to complete a task.

Planning Horizon

Plans are decomposed into a sequence of components called planning horizons; plan = plannng horizon(1) + planning horizon(2) +...+ planning horizon(n), where planning horizon(i) is a subplan.

Puma 560

A robot manufactured by Unimation which is used to simulate the actions of a human loader.

Readers-Writers Problem

The operating system problem where more then one process tries to access a resource.

Recovery Planning

Plans which are generated to update the current planning sequence because there was an unexpected event which would have caused the current plan to fail.

Scenario

A physical or conceptual environment which simulates the desired workspace under-study.

Silicon Graphics

A Unix-based machine which is manufactured by Silicon Graphics Computer System.

Situation Assessment

The event of scanning the target assessment area and using geometric and spatial reasoning to generate a prioritized target list.

Situation Assessment Vision System

The vision system that is above the situation assessment area.

Slot

A feature description of an object in a frame.

Stub

A procedure which simulates the functions of a module that has not yet been coded into the system.

Target Assessment Area

The region of the workspace which contains terrain features and military vehicles. This is the area upon which situation assessment is performed.

Target Assessment Vision System

The vision system which is responsible for extracting pixel data from the target assessment area.

Test Bed

The environment in which the research goals will be tested.

Vision Frame

The raw data that is received from the vision board.

Vision System

The system which extracts pixel information about the environment and then passes this information up the hierarchy.

DISTRIBUTION LIST

Commander
Armament Research, Development
and Engineering Center
U.S. Army Armament, Munitions
and Chemical Command
ATTN: SMCAR-MSI (5)
SMCAR-FS, H. Garver
T. Davidson
SMCAR-FSA, R. Wrenn
SMCAR-FSA-C, N. P. Coleman
R. L. Merolla
M. DeVito
K. Lam
E. Carroll
J. Lester
J. Lehman
L. Ambrosini
SMCAR-FSA-E, F. Saxe
SMCAR-TDC, H. Grunder
SMCAR-CC, H. Opat
Picatinny Arsenal, NJ 07806-5000

Commander
U.S. Army Armament, Munitions
and Chemical Command
ATTN: AMSMC-GCL (D)
Picatinny Arsenal, NJ 07806-5000

Administrator
Defense Technical Information Center
ATTN: Accessions Division (12)
Cameron Station
Alexandria, VA 22304-6145

Director
U.S. Army Materiel Systems
Analysis Activity
ATTN: AMXSY-MP
Aberdeen Proving Ground, MD 21005-5066

Commander
Chemical Research, Development
and Engineering Center
U.S. Army Armament, Munitions
and Chemical Command
ATTN: SMCCR-MSI
Aberdeen Proving Ground, MD 21010-5423

Commander
Chemical Research, Development
and Engineering Center
U.S. Army Armament, Munitions
and Chemical Command
ATTN: SMCCR-RSP-A
Aberdeen Proving Ground, MD 21005-5066

Director
Ballistic Research Laboratory
ATTN: AMXBR-OD-ST
Aberdeen Proving Ground, MD 21005-5066

Chief
Benet Weapons Laboratory, CCAC
Armament Research, Development
and Engineering Center
U.S. Army Armament, Munitions
and Chemical Command
ATTN: SMCAR-CCB-TL
Watervliet, NY 12189-5000

Commander
U.S. Army Armament, Munitions
and Chemical Command
ATTN: SMCAR-ESP-L
Rock Island, IL 61299-6000

Director
U.S. Army TRADOC Systems
Analysis Activity
ATTN: ATAA-SL
White Sands Missile Range, NM 88002

Commander
U.S. Army Armament, Munitions
and Chemical Command
ATTN: AMCPM-AL, Bldg 455
Picatinny Arsenal, NJ 07806-5000

END

5-87

DTIC