

AD-A193 181

















INTELLIGENT MOBILE AUTONOMOUS SYSTEM (INAS)(U) DREXEL  
UNIV PHILADELPHIA PA DEPT OF ELECTRICAL AND COMPUTER  
ENGINEERING A M MEYSEL 1987

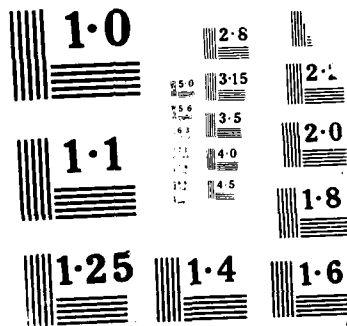
177

UNCLASSIFIED

F/G 12/9

NL



1

DTIC FILE COPY

AD-A193 181

INTELLIGENT MOBILE  
AUTONOMOUS  
SYSTEM  
(IMAS)

DTIC  
ELECTED  
APR 18 1984

S E

Drexel  
University



88

88-3-181-8

*Fields of Science:*    *Control Theory*  
                          *Intelligent Control*  
                          *Artificial Intelligence*  
                          *Computer Architecture*

**INTELLIGENT MOBILE  
AUTONOMOUS  
S Y S T E M  
(IMAS)**

**TECHNICAL REPORT**

**SUBMITTED TO FORT BELVOIR RD&E CENTER  
BY A. M. MEYSTELE**

**DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING  
DREXEL UNIVERSITY  
PHILADELPHIA, PA 19104  
1987**

DISTRIBUTION STATEMENT A: Approved for  
public release: distribution is unlimited.

**DTIC  
ELECTE  
APR 18 1988  
S D  
E**

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

## EXECUTIVE SUMMARY

This report concludes the research initiated in 1983 with the objective to contribute to fundamentals of the new emerging area of autonomous robotics. The goal of this research is to develop a theory of design and functioning of Intelligent Mobile Autonomous Systems (IMAS) to be utilized for solving various problems of combat engineering. It was the initiative of the research team to combine the fundamental research with the experimental analysis of the IMAS testbed. This work has premises different from those determined in the corresponding DARPA research. It started before the DARPA program (at January 1983), and is based upon the scientific principles, that at the time of the beginning of the research, were pioneering in this area.

This research resulted in creation of a theory of nested hierarchical systems and in particular, of the structure of a multiresolutional nonhomogeneous system of representation interacting with a similarly built planning/control system (PLANNER-NAVIGATOR-PILOT-CONTROLLER). The hierarchical nested structure, together with the unique structure of knowledge representation, as well as capabilities of minimum-time dynamic navigation, and the knowledge-based controller, are promising for IMAS, and are expected to be expanded into the domain of multilink manipulators as well. Testing has confirmed the viability of scientific premises, and has clarified the program of subsequent analysis and scientific research. ↙

This report contributes to a new rapidly developing area of *autonomous robotics*. Actual experience of dealing with autonomous robots (or rather, robots with some elements of autonomy) does not exceed just a couple of decades. And yet, the pace of development is so quick that a number of highly ambitious programs is in progress at the Universities, industrial corporations, and multiple research centers. These programs are focusing the effort of the scientists and engineers not only on the problem of autonomous mobility of robots, they are also premonishing the future stages of R&D such as coordination of the relatively autonomous subsystems of the mobile robot, cooperation among several autonomous mobile robots working for the same goal, etc. Thus our results can be implemented in most of the contemporary systems which employ unmanned technology.

Indeed, the rapid advancement of autonomous and semi-autonomous systems is spurred by a direct

practical demand which has been generated in a variety of application areas. Outdoor means of transportation as well as indoor industrial transport devices are limited in their capabilities to perform various tasks, in their efficiency (and/or cost-effectiveness): they require RF link, or underfloor wiring. Most of the flexible manufacturing systems as well as combat engineering robotic systems, depend on the autonomous mobility as the tool for the increase of productivity and viability. Numerous applications based upon the unmanned character of operation as a major property of the job to be done (such as mine neutralizing, nuclear stations maintenance, underwater operations, operations in the space, etc.), have generated a powerful demand for the mobile autonomous systems that can substitute a human operator in a variety of unstructured situations where the motion as well as the job performance cannot be prescribed in advance exhaustively. Of course, the military applications (e.g. in the area of combat engineering) constitute a substantial domain of "vested interest".

However, the most powerful drive toward the accomplishment of definite scientific and engineering goals in the area of Autonomous Mobile Robots, is determined by the structural changes in the science and engineering. Indeed, the blossoming in the combined area of the Artificial Intelligence and Robotics, the results in the area of Neural Networks, advances in the areas of computer vision, hierarchical intelligent control, and knowledge base systems, as well as in the 5-th generations of computers which can be clearly foreseen within this decade, all of these factors are focusing our attention on the Cognitive Science as a part of the electrical and computer engineering. Thus, the Autonomous Mobile Robots can be considered as an example and the first area of application for this nascenting science.

This research is based upon analysis of a complete account of the theoretical and experimental results obtained during the last two decades in the area of Autonomous Mobile Systems (see our prior report "Intelligent Control of Mobile Autonomous Systems: The State-Of-The-Art", Gainesville, FL 1983). An attempt was made to build some available generalizations upon the body of the two decades experience. These two major features of this report: the account of existing experience, and the generalizations upon this experience, were used to generate a number of engineering recommendations which can be utilized at the stages of design, manufacturing, and testing of Autonomous Mobile Robots. Certainly, the skeleton of the abovementioned science is looming behind the engineering tissue of this report: science of cognitive engineering which is to tie together all of the diversified issues related to unmanned autonomous mobility of robots.

Most of the research and design results of the numerous robotic teams in US, France, England, West

Germany, Japan, etc. are reflected in the abovementioned report. A survey of the information about the accomplishments of the USSR and the other East European countries in this area gives another, somewhat different perspective of development in the area of Autonomous Mobile Robots. These two perspectives in combination, form a basis for viewing the alternatives of solutions for the emerging sets of requirements in the application areas.

However, most of the results presented in this report, the overall philosophy of the approach, as well as the software packages recommended for simulation and/or , are produced within the Laboratory of Applied Machine Intelligence and Robotics (LAMIR) at Department of Electrical and Computer Engineering of Drexel University (see our prior reports "**Primer on Autonomous Mobility**", Drexel University, Philadelphia, PA 19104, 1985, and "**IMAS: briefing**", Drexel University, Philadelphia, PA 19104, 1985). Since these results are generated by the University research team under the contract aiming for a practical application, a dual significance of this report can be visualized. It can be used directly by the engineers of R&D groups at the stages of design, manufacturing and testing of autonomous robots. On the other hand, this report may be recommended as a source of additional information for existing graduate courses in robotics, and for the corresponding research groups. The state-of-the-art information on autonomous systems for intelligent operation of robots, seems to be a reasonable supplement to the course (depending on its orientation).

The concise sketch of the theory of control for an autonomous land vehicle (totally on-board computer controlled with no umbilical cord, no RF communication), is presented which has been developed by Drexel University under contract with Belvoir Development and Engineering Center ("Intelligent Mobile Autonomous System"-IMAS, the updated version- IMAS-2). The vehicle base is Sandair Dune-Buggy, however the intention was to develop a universal solution that could be used with any type of mobile base, or a standard vehicle.

IMAS-2 is based on a theoretical concept of Nested Hierarchical Control which employs joint multiresolutional planning-control procedures, algorithm of enhanced nested dynamic programming (combined with A\* search), and the hybrid world representation using "look-up tables" for the "analytical" part, and linguistical clauses for the rule-bases. The software package NEST (for Nested Hierarchical Control) has been developed and tested. Part of this package is a rudimentary learning subsystem which updates the information of the world, and adds new suggested actions to the list of rules (how to deal with traps).

The computer architecture is built around AI computer structure ("Symbolics 3460") with a network of single board computers (BCC-52). The objective was to implement the existing computer equipment for the NEST-architecture, and to investigate the real-time actuator control systems using LISP as a main control language.

IMAS-2 is equipped by a vision system based upon a Fairchild CCD-3000 camera, Data-Cube VG/AF-123 frame-grabber, and the IBM-PC with 256K memory. The input scenes are successive images (256x240 by 8-bit gray level) which are processed and interpreted in real time, submitting information about "go" and "no go" areas as well as of distance to these zones. The objective was to demonstrate the innovative concept of computer vision with no edge-detection which allows for substantial increase in efficiency and enables real time operation of the vehicle.

Testing has demonstrated the behavioral "habits" of autonomous robot under NEST control. Report is illustrated by a video-tape of the IMAS-testbed demonstration (available on request).

The following statements can be considered as statements of accomplishment (in addition to the theoretical results reflected in the corresponding papers and intermediate reports).

1. IMAS\* should contain a nested hierarchical system of knowledge representation. This system must be equipped with the reasoning system for map updating. Both of these systems are a part of IMAS-2.
2. Planning/control is a joint recursive process performed upon system of nested hierarchical knowledge representation. This system is a nonhomogeneous one, and the subsystem of planning and control should be able to deal on-line with a variety of knowledge-types. This recursive process dealing with nonhomogeneous knowledge is a part of IMAS-2 software package.
3. Planning/control process is becoming more and more dynamically involved while descending top-down in its hierarchy. Thus, dynamics is reflected at the stages of

---

\*All further statements are related to all Intelligent Mobile Autonomous Systems (not only to IMAS, and IMAS-2 produced at Drexel University). We believe that the generality of our results suggests their broad use for design of autonomous robots.



planning and navigation which is a novelty for contemporary systems of guidance-navigation-control. Our team is the first in development of dynamic planning with searching in a state-space under a number of criteria to be minimized.

4. Unique results are obtained, and a set of practical algorithms has been developed and tested in the area of motion planning in traversability spaces. Our planning/control system can efficiently compute the minimum cost path in the environment where different regions are traversable however the cost of traversal can vary.

5. A vision system has been developed which does not require any edge detection. Nested Hierarchical information representation is being employed for making the subsequent processes of interpretation more efficient.

6. The processes of piloting has been thoroughly investigated, and a number of knowledge-based PILOT systems has been developed. These systems of vehicle guidance are working on-line, dealing with nonhomogeneous knowledge in real time, and are finding near optimal solutions.

7. One of the most advanced systems of PILOT imitates two types of human decision-making behavior: one which is based on a complete knowledge of the situation, and one that does not require almost any information of the world. The blend of these two PILOT personalities leads to a number of advantages in the IMAS functioning.

8. A fundamental result is obtained applicable for the lowest level execution controller. The sequence of the ""switching commands" generated to achieve minimum-time actuation, is being compensated based upon current information about the stochastic properties of the surrounding environment.

At the same time we would like to list the three major shortcoming of our research all of which reflect noninvolvement in the three important topics of investigation:

A. IMAS operates with a very simplistic system of perception. It has only non-color vision, and sonar sensors for "touch" imitation. One can presume that using color vision, and a variety

of sensing subsystems of various modalities of sense can enhance the capabilities of autonomous system. However, this was not our goal: to produce *the* autonomous mobile robot. On the contrary, we were interested in focusing upon problems of representation, and planning/control. Thus, we were interested in using a *minimal perception*. Development of a vision with no edge detection is a side result of our research. Our minimal perception enabled us to easily organize it as a nested hierarchical system which can work in association with nested hierarchical systems of representation and planning/control. On the other hand, we have received a number of results that suggest that using this organization of perceptual system in the future, can simplify the processes of interpretation.

B. IMAS cannot learn new rules of operation from experience. This means that it does not have any subsystem of *cognitive learning*: learning that allows for making some inductive hypotheses based upon similarities of prior experiences, and for generating new *concepts* implied by the most plausible among these hypotheses. Certainly, it has rudimentary *learning* (see, *MAP learning* for map updating in the system of representation, and *REPORTER learning* the traps in the PILOT subsystem. However only conceptual learning can provide the reliable long term autonomous missions in real environment.

C. Thus, no real autonomy is possible with inadequate perception, and with no conceptual learning. One can expect that IMAS should be used only for limited time and limited function missions with limited autonomy. Then its perceptual deficiencies, as well as conceptual dependency won't affect the results of operation. IMAS should be considered as a strictly *master dependent system*. This means that the system should be made *deeply and quickly communicable* with a human operator. This property can be achieved on the basis of a language of IMAS-master communication which will enable the automated reprogramming the system during the process of communication. Present IMAS does not have this important feature.

The abovementioned shortcomings have determined ours plans for the future research in the area of Intelligent Mobile Autonomous Systems:

1. Development of a system for IMAS-master communication based upon language with flexible structure.
2. Development of a system for conceptual learning.
3. Development of the Nested Hierarchical Multisensor System with structural mechanisms of image interpretation employing conceptual learning, and occasional communication with the master.

The research team included the following graduate students (their theses contributed to the success of the final results, and are mentioned in the list):

1. S. Waldon, MS EE Thesis, "Nested Hierarchical Representation of Uncertain Spatial Information: Problems of Updating" (in progress, due 1987), an excerpt from the thesis see in **Section 2**
2. P. Graglia, MS EE Thesis "Multiprocessor Architecture for Planning in Variable Traversability Space", (in progress, due 1988), an excerpt from the thesis see in **Section 3**
3. D. Gaw MS CS Thesis "Knowledge Management in a Nested Hierarchical Controller for an Autonomous Robot" (in progress, due 1987), an excerpt from the thesis see in **Section 4**
4. R. Bhatt, MS EE Thesis, "Planning and Control of Robot Motion with Hybrid Analytical Linguistical Knowledge Representation", (in progress, due 1988), an excerpt from the thesis see in **Section 5**
5. S. Uzun, PhD EE Thesis, "Nested Hierarchical Architecture for a Computer Vision System in an Intelligent Mobile Autonomous System", (in progress, due 1989), an excerpt from the thesis see in our prior report to Fort Belvoir RD&E Center "**Primer on Autonomous Mobility**", Drexel University, Philadelphia, PA 1985
6. S. Uzun, MS EE Thesis, "Image Structuring for an Intelligent Mobile Autonomous System", Drexel University, Philadelphia, PA 19104, 1986, an excerpt from the thesis see in our prior report to Fort Belvoir RD&E Center "**Primer on Autonomous Mobility**", Drexel University, Philadelphia, PA 1985
7. C. Isik, PhD EE Thesis, "Knowledge Based Motion Control of an Intelligent Mobile Autonomous System", University of Florida, Gainesville, FL 32611, 1985, an excerpt from the thesis see in our prior report to Fort Belvoir RD&E Center "**Primer on Autonomous Mobility**", Drexel University, Philadelphia, PA 1985
8. M. Roberts, MS EE Thesis, "Minimum-time Controller for Robotic Actuators", University of

Florida, Gainesville, FL, 32611 1985, an excerpt from the thesis see in our prior report to Fort Belvoir RD&E Center "**IMAS: briefing**", Drexel University, Philadelphia, PA 1985

9. E. Koch, MS CS Thesis, "Path Planning in Binary Traversability Spaces", Drexel University, Philadelphia, PA, 19104 1985, an excerpt from the thesis see in our prior report to Fort Belvoir RD&E Center "**Primer on Autonomous Mobility**", Drexel University, Philadelphia, PA 1985

10. J. E. McKisson, MS EE Thesis, "Guidance and Navigation for Intelligent Mobile Autonomous Systems", Gainesville, FL 32611, 1983, an excerpt from the thesis see in our prior report to Fort Belvoir RD&E Center "**Intelligent Control of Mobile Autonomous Systems: State of the Art**", University of Florida, Gainesville, FL 32611 1983

Our research team is grateful to Mr. Uri Bar Am who was a manager and supervisor of the research activities during the period 1985-1987. Unusually high productivity of the researchers should be credited to his contribution to the management activities and operations, to his wit, optimism, and energy.

Help and supervision of the Fort Belvoir RD&E Center monitoring officers is highly appreciated.

Dr. B. Eisenstein, Chairman of the ECE Department of Drexel University, has done everything to make our work enjoyable as well as constructive and resourceful. We thank him for help and cooperation.

## Table of contents

<b>Section 1. Planning/Control Architecture for IMAS with Nonhomogeneous Knowledge Representation.....</b>	<b>11</b>
<b>Section 2. Multiresolutional Representation of Spatial Knowledge for Autonomous Robots.....</b>	<b>28</b>
<b>Section 3. Path Planning in the Random Traversability Space.....</b>	<b>47</b>
<b>Section 4. Pilot with Behavioral Duality.....</b>	<b>60</b>
<b>Section 5. Minimum Time Execution Controller for IMAS with Model Based Production System.....</b>	<b>75</b>
<b>Perspectives.....</b>	<b>89</b>

## SECTION 1

### PLANNING/CONTROL ARCHITECTURE FOR INTELLIGENT MOBILE AUTONOMOUS SYSTEM WITH NONHOMOGENEOUS KNOWLEDGE REPRESENTATION

The problem of partial autonomy for IMAS was raised in the Preface (see page 8). The related problem of computer architecture for intelligent controller for robots with partial autonomy is addressed in this section. IMAS is a robot with partial autonomy, and it can be considered a degenerated case of a fully autonomous robot. Since the nonhomogeneous representation system is applied, the process of man-machine communication is shaped by the hybrid character of languages implemented. The problem of master-robot control is formulated, and the conditions are determined for generating a language for the process of communication between master and IMAS within this framework of consideration.

The problem of master-dependent autonomy is a key problem for contemporary efforts in the area of robotic autonomy. No actual autonomous system can be created in a foreseeable future because of the available on-board computer power, impossibility to create human-like knowledge base, and inadequacy of the computer perceptual systems. Only limited autonomy can be considered. One should realize that despite of the upsurge of research in the area on neural networks, no conceptual learning can be expected on-board in the foreseeable future. (As a matter of fact, neural networks so far are being contemplated for learning the patterns under a teacher supervision, i.e. no autonomy is being claimed). Partial (master-dependent) autonomy is meant to compensate for the deficiencies of human perception and/or performance (DHPP) when the CRT is used as a "narrow window" to the world, and no or limited other means of perception are available.

**Problems to be considered.** Three different problems can be visualized in this situation:

- the problem of design of the IMAS-master (man-machine) control system,
- the problem of the human participation in the IMAS performance, in its the decision making processes, in the operations of situation and image recognition.
- the problem of master-robot communication supporting the solutions of two previous problems.

The problem of design of man-machine control system was first addressed in [1]. Based upon the notion of a limited "bandpass required of a man" the paper suggests that the control system should be allowed autonomy in "integration, differentiation, feedforward", and a number of other computation procedures in order to allow a man "to operate as a simple amplifier". The fact that a human operator still has to perform a unique job of information integration and decision making, at this stage was overlooked. However, very soon the human capabilities to be a universal feedback with unlimited "sensor fusion" talents were questioned in [2] especially when the complex task or multiple task situation is involved [3].

Rapid growth of the requirements to productivity and quality factors, brought DHPP mentioned above to the attention of researchers in the area of advanced control systems. The variety of teleoperated and supervised automated systems described in [4-6] concentrate upon enhancement of human capabilities by allocating some properties of autonomy within the machine control system. The state-of-the-art reflected in the literature, implies the sequence of development stages as shown in Figure 1. A need in the Interactive Manual-Automatic Control arises in a natural way due to the DHPP. According to [6], manual-automatic control allows for some motions under the manual control whereas the remaining motions are performed automatically referenced to a variety of sensor data. One

of the system created at JPL, is featuring a menu which can be activated by an operator in a modifications starting with fully manual and ending with fully automated operation.

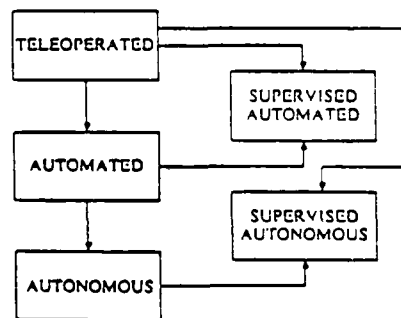
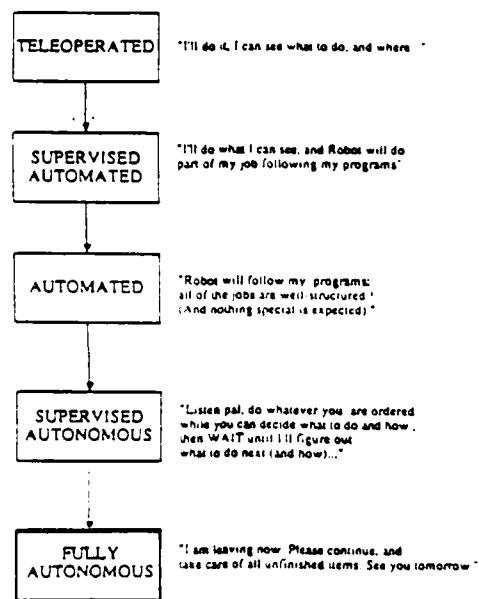


Figure 1. Typical approach to the problem of partial autonomy

Figure 2. Our approach to the problem of partial autonomy

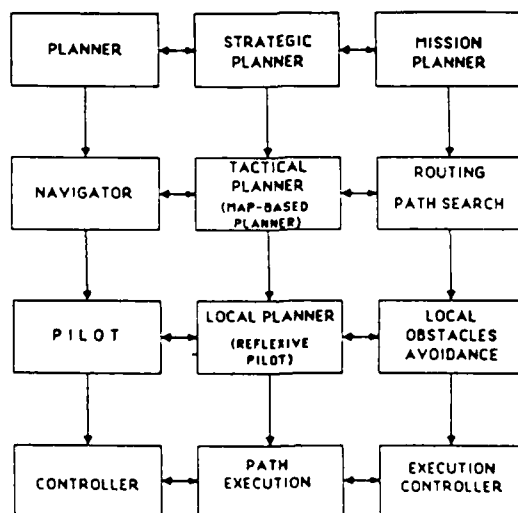


Figure 3. Comparison of planning/control structures from [8-12]

Thus the second problem is being approached: the problem of human participation in the robot decision making. This problem is treated in [7] where the dynamic reallocation of tasks between human and computer system, in the multitask situation is suggested. Apparently the following solution is considered to be preferable: the human performs whichever tasks he chooses while the computer system attempts to perform the remaining tasks [7]. The structure of human decision making activities in the supervisory control situation was analyzed [8,9], and the distribution of the multiplicity of tasks between the human and the robot seems to be predictable.

The idea of combined computer/manual controller was explored in [10] where the results of human operation are statistically generalized and utilized for generation of new routines of motion. We are interested in a simpler form of the computer/human symbiosis in which the human operator tends to communicate to the robot the results of recognition and generalization while does not interfere with the lower level motion routines: they will be changed autonomously if the robot apply the results of human recognition and generalization. The advantages of human involvement in these particular procedures are known for recognition with partially missing data [11], for recognition where the context plays the decisive role [12], for the context permeated processes of task structuring and recognition [13], and for proper organization of previously unknown spatial data [14]. Finally, the problem of master-robot communication is expected to be dependent on the area of human interference. We would like to explore a possibility to confine the channel of supervisory communication to the boundaries of the part of the system vocabulary, in particular to the upper level vocabulary of the hybrid system of representation which is being created and monitored by a master [13, 15-19].

**Approach to the master dependent autonomy.** This type of the partial autonomy can be approached in the following way. Let us imagine that the structure of a system is available with full autonomy of operation. How does this structure look like? What are the components of this structure that can and cannot be achieved by the technological means under consideration? It may happen that some elements of the autonomous operation can never be achieved, or can be achieved in a very remote future. The set of such "unachievable-now" operations can be considered **domain of impossibility of the mission (DIM-area)**. Then it would be reasonable now to make a step back from the ideal image of the autonomous machine, and build a system which rely on human participation however only within the limited part: within the DIM-area.

This approach to the problem of partial autonomy can be illustrated by Figure 2. It shows that the supervised automation can be viewed as a retreat from the demand for a fully automated system. In the same vein, the partially autonomous, or supervised autonomous robot can be considered a trade-off between the demand for a fully automated system (which has never been satisfied), and a demand for a fully autonomous system (which cannot be satisfied, at least currently).

**Focus of concentration** Problem of autonomous robot operation is often erroneously considered a problem of intelligent perception (only). In fact, even having the problems of perception and knowledge organization solved, the problem of motion planning and control, or more generally, the problem of autonomous (as well as partially autonomous) **decision making** remains unresolved. This paper attempts to give a structure of recommended methods of theoretical analysis for partially autonomous planning/control processes of a class of intelligent robots equipped by a control architecture designed for fully autonomous operation. This control architecture has an important distinction which affects supervised operation with partial autonomy: areas of motion planning, and motion control are treated as a part of a unified recursive computational process within IMAS.

Planning/control systems for IMAS were introduced first in [20-23]. IMAS should operate in unknown environment with limited human involvement or with no human involvement at all. IMAS are using human-like procedures of perception, maintain sophisticated information structure capable of



learning and self-organization: **Knowledge Based Control System (KBCS)**. Knowledge is understood as a structured information incorporating numerical data as well as linguistic, or symbolic information, which must be interpreted in a definite context. KBCS employs the **principle of nested hierarchies** which allows for an efficient knowledge organization as well as for the efficient processes of knowledge-based perception and planning/control.

**Control solutions to be considered** The word *control* is being used for decision making activities including *planning, navigation, and guidance*. This inseparable triad is presumed to be applicable at least, to systems for control of mechanical motion. It should provide a mapping of the a)task, b)system, and c)environment, from the domain of knowledge - into the domain of output specifications. An overview of the recent results in the area of IMAS [24], suggests that different control solutions show definite resemblance with human teams: they have a hierarchy of decision-making units even when the system is equipped by a single actuator [25]. In Figure 3, three examples of control structures are compared [26-30]. Each of these structures tends to form an intelligent module for automatic generation of control strategies, policies, and commands.

However, the most remarkable thing about all of these systems is that in all of them *the scope of decision making is becoming smaller, and the resolution of decision making is becoming higher when the level of decision making is becoming lower*. This sequential multiple process of decision making in the same situation at different resolutions, and with different scope of attention, seems to be almost obvious. Thus, the need in proper theoretical justifications escapes from our attention. We would assume that the underlying theory should be applicable to IMAS. The major result pronounced within the theory, is related to the information structure required to run the system. The realization of this information structure is a special, control oriented knowledge structure [30]. Theory of control oriented knowledge organization as a part of the IMAS, is focused upon development of models of KBCS for motion, structures of algorithms, and design of systems for optimum motion of autonomous or semiautonomous systems. Roughly speaking, any IMAS is organized as a *team of human decision-makers* which allows for using many efficient solutions developed for human teams. This problem gets a specific content: information structure should be suited a proper knowledge quantization [20-23].

**Hierarchical structures of attention and resolution.** Hierarchical decomposition of systems is a well known phenomenon, and is used usually as a method of dealing with their complexities. A system is being decomposed in parts (partitioned) when each of these parts contains some active elements of the systems subject to control (actuators). Thus, it is typical to consider multiactuator control systems in a form of "tree-hierarchies" where the hierarchical decentralized techniques are recommended: relative independence of the levels is as important, as coordination of the branches [31-35]. The hierarchy is being preserved by the fact of resolutional as well as attentional *nesting*.

**Decision making processes in a nested hierarchical structure.** Decision-making process which is being done upon the nested set of representations can be characterized by different frequency, accuracy, and combinational power. The frequency of decision-making at the lowest level of representation can be in the interval ( $2 \cdot 10^{-4}$  to  $10^{-3}$  Hz), then at the next higher level ( $10^{-3}$  to  $10^{-1}$  Hz), and finally at the highest level of representation it will be ( $10^{-1}$  to 5Hz).

Certainly, on the level of controller compensation, the required frequency as well as accuracy might be even higher. In the meantime, the number of "objects" of the world represented in all of the levels can be kept the same due to the difference of resolution. We can easily find that at a given frequency of decision-making at a level of consideration, the moving robot will pass during the period of time between two consecutive decision-making processes, the same quantity of resolution units so that the ratio

$\frac{(\text{maximum state space changes per decision})}{(\text{time interval between two consecutive decisions})} = \text{const}$

is constant at all levels. After transformation we receive that

$\frac{(\text{time interval between 2 consecutive decisions})}{(\text{resolution at the level})} = \text{const}$

for all levels of consideration.

Similar consideration brings us to the conclusion that "combinatorial power" is also the same at all levels of consideration; it is just applied to different units of information, units that have different resolution. This hierarchy of representations at different resolution can be easily transformed into a hierarchy of decision making processes at different resolution.

**Decision making processes of planning-control procedures** In the most general form, the controller can be represented as a box with three inputs, and only one output. These inputs can be specified as follows (see Figure 4,a):

- Task: the goal (G) to be achieved, (and conditions to be satisfied including the parametrical constraints, the form of the cost function, its value, or its behavior).
- Description of the "exosystem", or map of the world (M) including numerous items of information to be taken in account during the process of control; M is often incomplete, or even deceptive.
- Current information (I) is the information set delivered by the sensors in the very beginning of the process of control, and continuing to be delivered during the process of IMAS operation.

The processes within the controller are illustrated in Figure 4,b,c,d. Input T has determined where the points  $SP_i$  and G are situated within the input map M. Input I gives a limited zone  $O_{1,i}-O_{2,i}$  (in the vicinity of  $SP_i$ ) in which the information set is more reliable, or even exhaustive. Thus, in the overall planned trajectory  $SP_i-G'_i-G_i$ , a part  $SP_i-G'_i$  can be determined with more reliability than the rest of it. In other words, the overall trajectory  $SP_i-G'_i-G_i$  from SP to G might be changed if the future input I will update M in the way that the computed trajectory will become not the most desirable alternative. However,  $SP_i-G'_i$  part of the plan (bold line) won't be changed: no new information is expected.

Let us concentrate now on the part of the trajectory  $SP_i-G'_i$  which is assumed to be more known than the rest of the overall plan. Label the point  $G'_i$  the "new goal", or the "goal for the lower level of the nested hierarchy". Notice that instead of the planned curve segment  $SP_i-G'_i$  the stripe is known with boundaries  $B_{i+1}$ . Within this stripe, a new planned motion trajectory can be determined at a higher level of resolution pertained to the level (i+1). Again, only part of the trajectory  $SP-G^{i+1}$  can be refined within the sector  $O_{1,i+1}-O_{2,i+1}$ . This in turn, brings us to the more precise part  $SP_{i+1}-G^{i+1}$  which can be subsequently refined at the level (i+2). Eventually "new goals" of the adjacent levels are becoming indistinguishable, and the trajectory  $SP-G_n$  can be utilized for computing the actual control sequence. All previously determined trajectories are plans at different level of refinement.

This simple consideration includes the following components of the joint planning/control process .

1. Finding the optimum plan SP-G based upon the map M, and the task formulation (SP, G, cost-function, constraints).
  - 1.1 Computing the alternatives of SP-G.
  - 1.2 Comparison of the available alternatives.
  - 1.3 Selection of the preferable alternative.
2. Updating the map information M in the vicinity of SP by using the sensor information I.
  - 2.1 Recognition of the set I.
  - 2.2 Comparison between M and I.
  - 2.3 Deciding upon required changes in the cases of:  $M/I \neq \emptyset$ ,  $I/M \neq \emptyset$ ,  $I \equiv M$ .
  - 2.4 Creation of the new map in the vicinity of SP with required deletions and additions.
3. Refined planning the path within the updated zone of the map.
  - 3.1 Determining the subgoal G' (e.g. as a point of intersection of SP-G and the boundaries of updated part of the map).
  - 3.2 Finding the optimum plan SP-G' ( which implies that the whole set of procedures mentioned in p.1, should be repeated).
4. Track the optimum path SP-G'.
5. Upon arrival to the new point of the selected trajectory loop to the position 2.

The set of recursive procedures 1 through 3 is illustrated in Figure 4,b. Two major recursions are presumed within the set of accepted decision making activities. Firstly, position 3.2 generates a recursion which requires consecutive refinement within the zone which allows for this refinement. Secondly, the major loop from 5 to the position 2 presumes that the whole set of procedures after each increment of motion will be repeated. So, the process of path planning is substituted by a consecutive determining of subgoals located closer and closer to the current position. However, we still must determine a) the trajectory SP-G, and b) the zone in which the refined information should be obtained.

**Nested dynamic programming.** Method of Nested Dynamic Programming (NDP) was developed in [6-9]. It is an algorithm of joint planning/control process illustrated above. In NDP optimum control is found by consecutive top-down and bottom-up procedures, based on the following rules.

*Rule 1.* NDP should be performed first at the most generalized level of information system with complete (available) world representation.

*Rule 2.* NDP is being performed consecutively level after level top down. The subspace of the search at each of the consecutive lower levels is constrained by the solution at the preceding upper level recomputed to the resolution of the next lower level.

*Rule 3.* When during the actual motion, due to the new information, the optimum trajectory (determined at a given level) must violate the assigned boundaries, this new information should be submitted to the upper level. This generates a new top-down NDP process.

*Rule 4.* When arrival of the new information is bounded (e.g. by a "limit of vision"), then the recursion of nested process of planning is being done with consecutive process of subgoals creation.

Any combinatorial algorithm can be applied as an operator of generating plans or **solution alternatives** in the NDP decision-making process. Consider A\*-algorithm for the search of minimum path trajectory [42,45,48], or any type of conventional or "enhanced" dynamic programming [36,37,44,46]. A number (value) is assigned to each of the combinations generated (preferability, closeness, propensity, cost-effectiveness, etc.) which enable the decision-maker to make his choice under the accepted **strategy of decision making**.

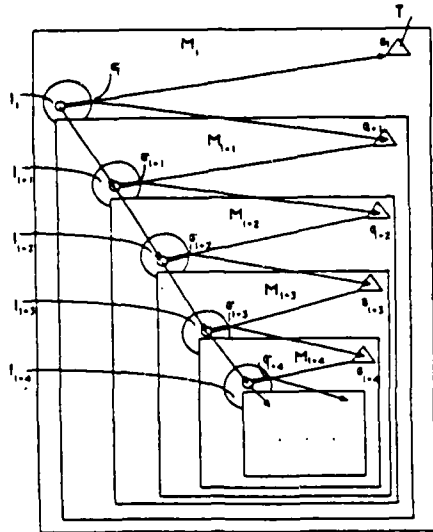


Figure 4. Joint Planning/Control Process

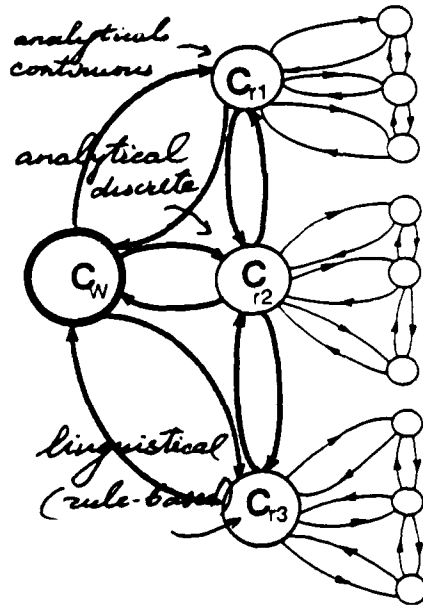
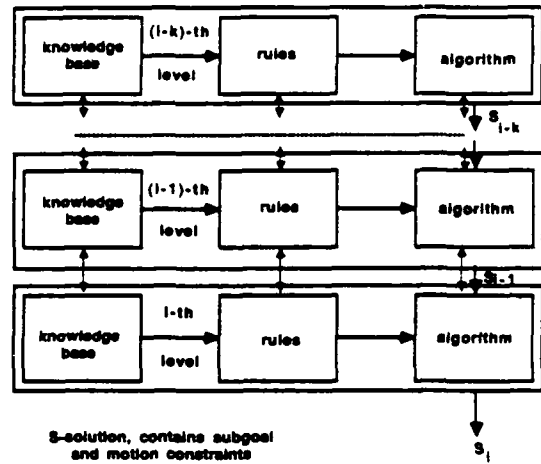


Figure 6. Categories of World Representation



S-solution, contains subgoal and motion constraints

Figure 5. Relations between representation and control

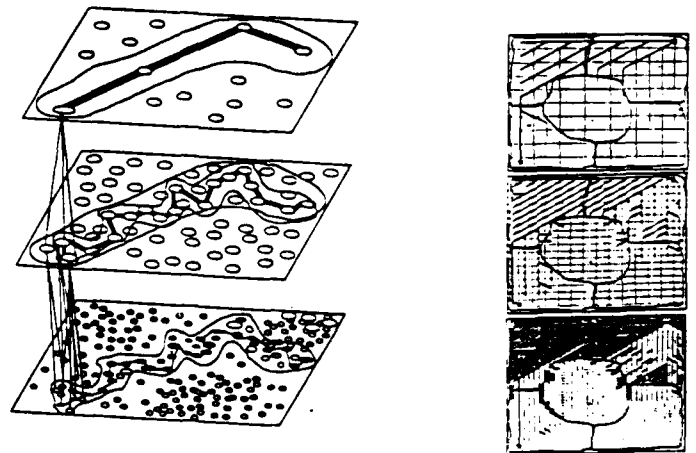


Figure 7. General Solutions Preceding the particular ones

**Partitioning and nesting of representations.** The information set is assumed to be the past measurements and controls for all levels of our hierarchy nested by generalization and focus of attention. It is shown in [20-23] that these phenomena is characterized best by the operators of inclusion using some procedures unusual for the conventional control theory. Decomposition of the categories of representation is done through decomposition of objects and morphisms represented in this category [38]. Decomposition implies dividing objects and relationships in parts, so the components of objects are becoming of interest unlike before the process of decomposition. This, in turn, implies higher resolution of world representation than before the decomposition. If we continue with decomposition, the hierarchy of decompositions will describe the same world with higher and higher level of resolution. Changes in time are represented by sequences of the snapshots in the domain of representation describing sequences of the world states. Thus, change in the time-scale is intrinsically linked with the value of resolution. Certainly, different levels of the hierarchy can be characterized by different time-scales. All these phenomena are illustrated in Figure 5.

This system of nested variables and nested mappings is obtained as a result of natural, and or artificial discretization (tessellation, partitioning) from an initial continuous space of the isomorphical world description. The tessellated space can be considered as a nested state-space for a dynamical system in which the trajectory of motion can be represented for a given moment of time (snapshot of the world). Then, a sequence of snapshots can be obtained. The conventional state-space is a superposition of all snapshots which describe the process.

This tessellation can be done in such a manner that the "points of interest" can symbolize the variables on a grid of the uniform space tessellation at the lowest level, words describing the physical objects at the intermediate level, and words describing the unions of physical objects at the highest level of the representation system. These "points of interest" are placed to the center of the "tile" of tessellation at a level (grain, discrete, pixel, or voxel of the space). These terms can be used intermittently, and each of them characterizes the level of generalization. One cannot discern any another point within the tile, this is a minimum grain at this level of resolution. So, the tile of the tessellation determines **resolution of knowledge** defined as a minimum discrete of information, or minimum word. All of them allow for producing implications (well formed formulas, wffs), or difference equations, which is the same.

Hierarchies created in this way, satisfy the following principle: **at a given level, the results of generalization (classes) serve as primitives for the above level.** Then each level of the hierarchy has its own classes and primitives; thus, it has its own variables (vocabulary), and the algorithms assigned upon this vocabulary can be adjusted to the properties of the objects represented at this level. This leads to the consistency rule: at least two adjacent levels of the hierarchy must be considered simultaneously. We are interested in the pairs of adjacent levels where the upper level is discretized in a "natural" way (based upon "words" of the context), and the lower level is discretized in an "artificial" way (based upon grids, scales, etc.). Decision making in this case depends on the ability of these two levels to communicate efficiently despite of the differences in representation. The role of the **master** is critical for providing real consistency in a context.

**Resolution and accuracy of representation.** *Resolution* is a measure of distinguishability of the vectors in the state space. Accuracy can be understood as the value of the relative error of the quantitative characteristics assigned to the node or the edge of the graph. This evaluation can be based on estimation of probabilities, zones of uncertainty, fuzzy membership and/or in any another way. The value of uncertainty is assigned to the numerical values associated with the properties of the tile of representation which is considered as a minimum unit at a level.

After assigning to the cluster a new class-label (a word to be included to the vocabulary), a new primitive appears with its own properties, the numerical values are assigned to these new properties, and they are characterized by the accuracy depending on the accepted evaluation theory for this particular property (including probabilistic approaches, fuzzy set theory, etc.). This means that accuracy and resolution are formally independent. The rules dealing with information: what must be given in details, retained, or dropped in generalization, cannot be formulated for all possible situations. How long the information will be needed can probably be determined only by a **master**.

So, under human supervision the knowledge in a context can be represented as a space tessellation at a definite resolution, and as a system of nested tessellations with different scale where scale is defined as a ratio between resolution at a level and resolution at the lowest level. Each of the larger tiles at the upper level is a "**generalization**" for the set of smaller tiles at the lower level of the hierarchy. The inclusion predicate has a meaning of "belonging to a class". Then discretization of the state space will contribute simultaneously to a) minimum required interval of consideration, b) hierarchy of classes of **belonging to a meaningful neighborhood**.

We would like to stress the link between the generation of entries for the look-up tables (LUTs) as an approximation in the space of consideration, and the problem of **generalization oriented tessellation** (discretization of the space). At the level of discretization determined by the context, i.e. determined by the words of the semantic network, the results of tessellation are always determined by the vocabulary of the **master**. In most of the cases the "automatic" tessellation of the space creates discrepancies when the correspondence between the two adjacent levels is to be determined. These discrepancies might be eliminated by the interference of the **master**.

**Knowledge Bases: Semantic Networks With Context Oriented Interpretation**  
 Knowledge Bases (KB) are considered to be a collection of rules (wff's) for man-machine as well as for autonomous decision-support systems. Knowledge consists of linguistic discretetes (entities), or **units of knowledge**. Nesting reflects semantic discretization of resolution in representation from human experience. Thus, a nested hierarchy of semantic networks can be formulated. This semantic nesting can be interpreted within a context only under human supervision since the subtleties of the context knowledge must be properly coded. Under master's supervision we receive, two types of nested hierarchies for declarative wff's: existential nesting (nested statement of existence) including nested statements of objects and relations among them, and transitive nesting (nested statement of change). Both of these nested units of knowledge are presented in implicative form (nested clauses). This another form of representation in turn, can be divided in two different kinds of representation: continuous (using differential equations) and discrete (using difference equations and/or finite state machines (see Figure 6). In both cases, the property of nesting holds.

The same world can be represented within the same levels of resolution in a different way. Representation is defined as a structure (e.g. algebraic, or information structure) which is homomorphic to the **world**, to the structure of reality (or a domain of reality). Representation consists of both numerical as well as descriptive information about the objects and systems, and is assumed to be obtained from prior experience, and or derived theoretically (based upon multiplicity of existing and possible tools of logical inference). Knowledge Bases as a model of information system contain many representations **homomorphic** to each other such as systems of difference, or differential, or integral equations, logical formulas, and others (see Figure 6).

**Two major types of representation.** The apparatus of differential and integral calculi (DIC) is applied at the lowest levels of control where the tiles of tessellation have no meaning in the natural

language, which is actually the language of technological application (LTA). Thus, the whole bulk of LTA-knowledge cannot be utilized in the problem solving process because of constraints imposed by the DIC-language which is always less diversified than LTA-language. Solution obtained within DIC-representation, should be translated back to LTA. The following inclusion holds:

$TR_{LTA} \supset TR_{DIC}$  which determines the same situation with representations of the output of the design process

$CS_{LTA} \supset CS_{DIC}$  (TR-task requirements, CS-control specifications). DIC-techniques allows for solving only well formulated problems with limited types of relations represented. So, only those problems are assigned for the upper level which must not be abridged by the process of mapping the set of technical requirements from LTA-representation into DIC-representation. DIC-representation cannot be satisfactory for all of the problems that arise in intelligent robot control.

After decomposition is completed, the lower levels contain too much information for using in the planning/control process of the higher levels. The knowledge of the robot **R** explicated from the **LTA-representation**, is frequently presented in such a way that the resulting **R-model in DIC-representation**, turns out often to be quite rudimentary, or at least, too simplistic. The rest of the knowledge should be taken care of, using master-robot communication. The uncertainties cannot be judged upon using the existing probabilistic models since the information does not exist for building such models. Master can prefer using more flexible apparatus of linguistic variables.

New information about the R-parameters will be obtained during the IMAS operation, thus the discovery of the new IMAS models is expected as a part of planning/control procedure (part of the learning process). The variety of the expected operation modes is taken in account by the master judgment, and the processes of operation under all possible conditions are taken in consideration by him. Even for a simple robot, one cannot easily answer the question what is the "best", and what is the "worst" cases, and is there any hypothetical "average" case which could be accepted as a "design" case. The model of external environment (**EXO**) is becoming a part of **R-model**.

The task is posed imprecisely, the number of functions to be provided by a control system is incomplete, and the situations of the operation are introduced to the control system in an approximate way. In many cases it is determined by the very fact of impossibility to transfer the knowledge of a particular context, to a robot controller. In many cases it is determined by intractability of the problem as it seems to the R system of representation.

**Recognition and Identification.** An example of interpreting the situation with identification, recognition, and learning the new patterns of the entities of the world, is shown in Figure 7. If A is a robot location, and B is the goal, then minimum time path can be sought in one of the following areas: I, II, and III. It can be demonstrated geometrically, that the minimum distance path can be found in the area II. However, since each turn within the "slalom-type" area creates losses of time (speed at the turning point should be reduced to avoid skidding), then the minimum time trajectory seems to be within one of the areas I or III.

Even before we start to compute all and possible trajectories within these areas, they can be analyzed as some entities with common properties. Indeed, area I can be described as an area which has "obstacle 1" on the right of the moving robot all over the motion, area III has "obstacle 2" on the left of the moving robot, and area II has obstacles on both sides. Areas I and III are not different when robot is moving far from the obstacle, but they differ substantially if we try to minimize the length of the trajectory: trajectory in the area I will have five linear segments and four turns while the trajectory III will have only two linear segments and only one turn.

Further analysis shows that the cost-functions have different form for all three areas. Moreover, within one particular area cost-function can be considered monotonic in the sense of [45]. These kind of areas are named "topoways" [46], and they can be considered entities for minimum time problem solving at the higher level of planning/control system. Rules can be formulated in which the **condition** part will describe the relative obstacle location while the **consequent** part will be related to the most general description of the **topoway** recommended for further consideration. These rules have to be recognized by a master, however they can be utilized with no master's direct participation.

**On the equivalence between the difference equation model, and the production rules.**  
Difference equations can be easily transformed into production rules, Indeed, instead of the form

$$x(k+1) = A(k)x(k) + B(k)u(k)$$

where  $k$ -is the number of the stage of recursion,  
 $A(k)$  and  $B(k)$  are the matrices of system parameters,  
 $x$ -is a state, and  
 $u$ -is a control.

Instead of having this information in a form of an equation one can state

**IF { $x(k) \wedge A(k) \wedge B(k) \wedge u(k)$ }-> THEN { $x(k+1)$ }**

which actually describes the **causality** exposed by a particular system: "if present state is given, and the parameters are known, and some definite control is applied, then the state will change as follows". In most of the real systems causalities obtained experimentally and/or from the other models of representation can be **inverted** in a form of rules-prescriptions [41]

**IF { $x(k) \wedge x(k+1) \wedge A(k) \wedge B(k)$ } -> THEN { $u(k)$ }**

which can be interpreted as follows: "if present state is given, and the the following state is required, while the parameters of the system are known, then apply this control".

In some cases even the form of trajectory can be utilized for rules formulation. Indeed, the form

$$x(k) = F(k,0) x(0) + \sum_{l=0}^{k-1} F(k, l+1) B(l) u(l), \text{ and}$$

$$F(k+1,l+1) = A(k) F(k, l+1)$$

implies the following ways of subsequent planning/control activities:

- 1) solving the problem by sequential search and finding the trajectory by on-line computation, submitting to the execution controller, and storing it for the subsequent use in a similar situation,
- 2) retrieving the previously stored solution for the same or similar conditions, obtained from off-line training, or from on-line experience.

**Applying the production rules at different resolution levels.** The above consideration can be expanded if the prior behavior is taken in consideration in order to estimate how the reality differs



from what was expected and provide correctives for the control signal, or "tuning" for the matrices A and B .

Thus, the following sets of information can be considered initially known:

**S**-states :  $\{x(0), x(-1), x(-2), \dots\}$   
**P**-parameters :  $\{A(0), B(0), A(-1), B(-1), \dots\}$   
**G**-goal states :  $\{x(k), x(k-1), \dots\}$ .

The general form of a control-generating rule can be represented as follows

**IF [P^S^G]\* -> THEN DO U**

where **U** -is a string  $\{u(1), u(2), \dots, u(k-1), u(k)\}$ ,  
 \*-means that the solution is Pareto-optimal.

Solution can be found by one of the selected search procedures. There is a set of optimum solutions, the cost difference between them can be found only at higher resolution. NDP is done a the following sequence of procedures:

**SUBSTITUTE THE "OUT-OF-REACH" GOAL BY AN ACHIEVABLE GOAL  
 FIND A CONTROL-GENERATING RULE  
 IF THE CONTROL-GENERATING RULE IS NOT FOUND, APPLY  
 SEARCH  
 SUBMIT THE SOLUTION TO THE NEXT LOWER LEVEL  
 IF THE SOLUTION IS NOT FOUND,  
 REPORT THE PREDICAMENT TO THE NEXT UPPER LEVEL**

This approach is very general and produces the trajectories of motion in all cases: when the general rule exist and is known to the planning/control system, or when it does exist but is unknown. All algorithms of path-search in 2D binary world start with a statement **IF THERE IS NO OBSTACLE BETWEEN THE ROBOT AND THE GOAL, GO DIRECTLY TO THE GOAL, ELSE DO THE SEARCH** [42]. In fact, this rule must be proven by geometrical methods (i.e. at the higher level) or found by a search after discretization and putting the problem to the lower level. This is the way of finding the trivial trajectories A and B in Figure 8,a. After the trajectories A and B are obtained as a result of the search procedure, the corresponding strings of commands can be stored for subsequent using them as solutions in corresponding rules [50].

In the case shown in Figure 8,b we have a less trivial situation. Indeed, the lower level search can lead us in a straightforward manner to the trajectory A. However, when the back-up motion is allowed, at least one additional trajectory should be added: a backing up motion B and then motion forward C. This new opportunity immensely reduces the productivity of search. One would probably prefer to first select one of these opportunities (A or B+C) at the higher level, and then compute the actual trajectory.

The following examples of higher level rules, have been explored in IMAS-2 [42,43,47-50].

**IF ROBOT IS BECOMING CLOSE TO AN OBSTACLE (DISTANCE IS LESS  
 THAN D)  
 (ON THE RIGHT, OR ON THE LEFT)  
 THEN PUT MESSAGES TO PILOT AND CONTROLLER TO 1 PRIORITY**

AND  
 IF DISTANCE IS LESS THAN D\* AND MORE THAN D\*\*  
 THEN REDUCE SPEED TO V\* AND MAKE A TURN (LEFT, RIGHT)  
 AND  
 IF DISTANCE IS LESS THAN D\*\*  
 THEN STOP AND REPLAN

One can see that these rules do not refer to any specific global coordinates of the trajectory to be executed, and the local coordinates (position of the obstacle relative to the robot) are given with low accuracy using linguistic variables as it was done in [41]. Similar rules are formulated for a more subtle maneuver when after the local goal was determined in a particular situation, it turned out to be in an inconvenient location, e.g. the actual location of  $G''_{i+2}$  (see Figure 4) cannot be achieved in a near-minimum time fashion by a simple continuation of the motion ahead.

IF G(X) IS LESS THAN  $R_{MIN}^T$   
 AND G(Y) IS LESS THEN  $2 R_{MIN}^T$   
 AND ORIENTATION IS LESS THEN  $|90^\circ|$

THEN BACK-UP AND TURN IN THE DIRECTION OPPOSITE TO THE GOAL

(G(X) and G(Y) are the "x" and "y" coordinates of the local goal in a local reference frame, i.e. when the coordinate system is attached to the robot,  $R_{MIN}^T$  is the minimum radius of turn).

This rule leads to a nontrivial motion (Figure 8,c) which reproduces an anthropomorphic way of dealing with the problem. No wonder - it was introduced by a human "master". There is no experience in automatic formulation such rules. Using human experience and intuition in formulating rules is beneficial for dealing with predicaments arising during processes of operation.

The overall process of using the rules previously formulated and stored, and the search process for cases when the rule is not found in the list, is illustrated in Figure 9. The process of learning with new rules generation is illustrated in Figure 10. The stage of "generalization" can be performed so far only with "master" direct participation. Indeed, the control strings found as a response to the concrete situations should be stored but the subsequent analysis should be done by a "master" unless the reliable algorithms of generalization are developed.

In the meantime, the other types of learning (primarily, inductive learning) can be exercised with no human involvement. They include memorizing the snapshots of the world, analyzing the evolution of the snapshots, and computation of correctives for the matrices A and B as well as for the string of U.

**Retrieval of rules.** The first experience of operation of IMAS-2 has demonstrated that using the lists of rules is inefficient, and the hierarchical structure increases the productivity of the rules retrieval. The structure of the retrieval system is based upon the theory presented in [51].  $L_x$  is the language of conditions consisting of their vocabulary and the weighted relationships among them.  $L_y$  is the language of consequents also consisting of the vocabulary and the grammar of weighted relationships among the words of this vocabulary. The matching algorithm is based upon the grammar translator consisting of the weighted relationships among the words of vocabularies  $L_x$  and  $L_y$ , correspondingly. The matching process C ends with obtaining several "best" matchings, then evaluation, selection, and decomposition of the proper rule is done ((E,S,D)).

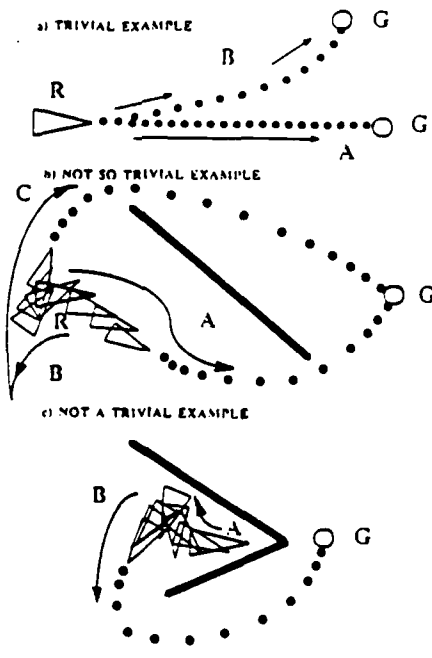


Figure 8. Examples of typical motions

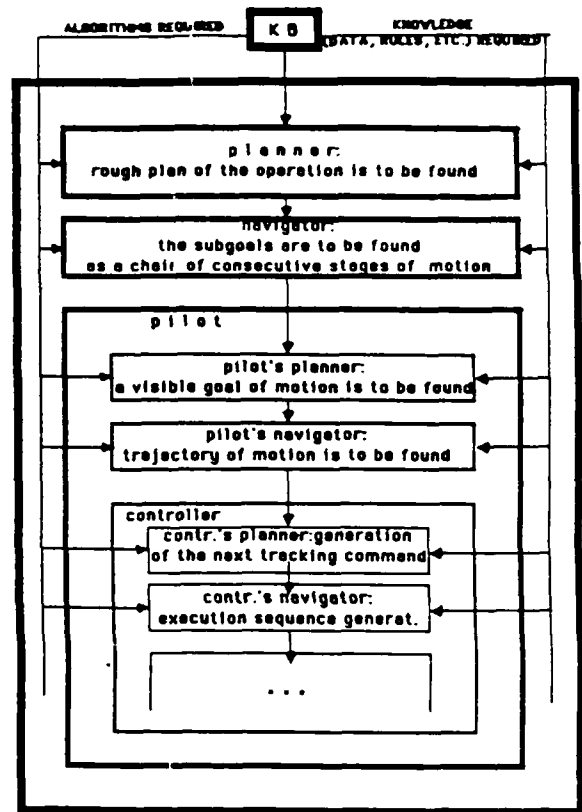


Figure 9. Rules search in NHC

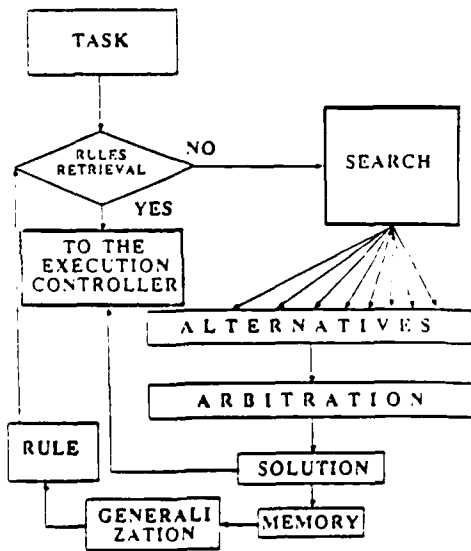


Figure 10. New rules acquisition

### References to Section I

1. H. P. Birmingham, F. V. Taylor, "A Design Philosophy for Man-Machine Control Systems", Proc. of the IRE, Vol. XLII, No.12, December 1954
2. N.H. Mackworth, "Researches on the Measurement of Human Performance", from Medical Research Council Special Report Series 268, London, GB 1950, Published in Selected Papers on Human Factors in the Design and Use of Control Systems, ed. by H. Wallace Sinaiko, Dover, New York, 1961
3. R. M. Hogarth, "Decision Time as a Function of Task Complexity", In Utility, Probability, and Human Decision Making, eds. Wendt and Vlek, D. Reidel, Dordrecht, Holland, 1975
4. Bejczy, A.K., "Smart Sensors for Smart Hands", in Progresses in Astronautics and Aeronautics, Vol. 67, Publ. AIAA, New York, 1979
5. Bejczy, A.K., "Effect of Hand-Based Sensors on Manipulator Control Performance", in Mechanism and Machine Theory, Vol. 12, Pergamon Press, 1977
6. Bejczy, A.K., Control of Remote Manipulators, in Handbook of Industrial Robotics, ed. by S. Y. Nof, Wiley, New York, 1985
7. W.B. Rouse, "Human-Computer Interaction in Multitask Situations", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-7, No.5, May 1977
8. S. Baron, "A Control-Theoretic Approach to Modelling Human Supervisory Control of Dynamic Systems", in Advances in Man-Machine Systems Research, Vol.1, JAI Press Inc.,1984
9. M. Kamil-Tulga, T.B. Sheridan, "Dynamic Decisions and Work Load in Multitask Supervisory Control, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-10, No.5, May 1980
10. A. Freedy, F.C. Hull, F.L. Lucaccini, J. Lyman, "A Computer-Based Learning System for Remote Manipulator Control", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-1, No.4, October 1971
11. J. K. Dixon, "Pattern Recognition with Partly Missing Data", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-9, No. 10, October 1979
12. O. D. Faugeras, "An Optimization Approach for Using Contextual Information in Computer Vision", Proc. of the First Annual Nil Conf. on AI, Stanford, CA 1980
13. J. G. Wohl, E. E. Entin, D. L. Kleinman, K. Pattipati, "Human Decision Processes in Military Command and Control", in Advances in Man-Machine Systems Research, Vol.1, JAI Press Inc.,1984
14. P. W. Thorndyke, S. E. Goldin, "Spatial Learning and Reasoning Skill", in Spatial Orientation: Theory, Research, and Application, eds. H. L. Pick, Jr., L. P. Acredolo, Plenum Press, New York, 1982
15. D. Miller, P. H. Friesen, "Archetypes of Strategy Formulation", Management Science, Vol. 24, No. 9, May 1978
16. J. Rasmussen, "Strategies for State Identification and Diagnosis in Supervisory Control Tasks, and Design of Computer-Based Support Systems", in Advances in Man-Machine Systems Research, Vol.1, JAI Press Inc.,1984

17. N. J. Hooper, G. C. Gustafson, "Automation and Recording of the Image Interpreter's Mensuration Tasks for Man-Made Objects", Proc. of SPIE, Vol. 424, Airborne Reconnaissance VII, San Diego, CA 1983
18. W. Klein, "Deixis and Spatial Orientation in Route Directions", in Spatial Orientation: Theory, Research, and Application, eds. H. L. Pick, Jr., L. P. Acredolo, Plenum Press, New York, 1982
19. J. C. Baird, M. Wagner, "Modelling the Creation of Cognitive Maps", in Spatial Orientation: Theory, Research, and Application, eds. H. L. Pick, Jr., L. P. Acredolo, Plenum Press, New York, 1982
20. A. Meystel, "Planning in a Hierarchical Nested Controller for Autonomous Robots", Proc of the 25-th IEEE Conference on Decision and Control, Athens, Greece, 1986
21. A. Meystel, "Nested Hierarchical Controller for Intelligent Mobile Autonomous System", Proc. of the International Congress on Intelligent Autonomous Systems, Amsterdam, Netherlands, 1986
22. A. Meystel, "Nested Hierarchical Intelligent Module for Automatic Generation of Control Strategies", Proc. of the NATO International Advanced Research Workshop on Languages for Sensor-based Control in Robotics, Ed. by U. Rembold, K. Hormann, Karlsruhe, FRG, 1986
23. A. Meystel, "Hierarchical Algorithm for Suboptimum Trajectory Planning and Control", in Recent Trends in Robotics, eds. M. Jamshidi, J.Y.C. Luh, M. Shahinpoor, North-Holland Elsevier, New York, 1986
24. A. Meystel, "Autonomous Mobile Device: A Step In the Evolution", in Applications in Artificial Intelligence, Ed. by S. Andriole, Petrocelli Books, Princeton, NJ, 1985
25. A. Meystel, "Intelligent Control of a Multiactuator System", in IFAC Information Control Problems in Manufacturing Technology 1982, ed. by D.E. Hardt, Pergamon Press, Oxford, 1983
26. A. Meystel, G. Hoffman, E. Koch, T. Looney, J. McKisson, M. Roberts, "Anthropomorphical Control of Intelligent Mobile System", Proc. of IEEE Southeastcon-82, Destin, Fl, 1982
27. G. Giralt, R. Chatila, M. Vaisset, "An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots", In Robotics Research, Ed. by M. Brady and R. Paul, MIT Press, Cambridge, MA 1984
28. A.M. Parodi, J.J. Nitao, L.S. McTamoney, "An Intelligent System for Autonomous Vehicle", Proc. of IEEE Intl Conf. on Robotics and Automation, San-Francisco, CA 1986
29. D.W. Payton, "An Architecture for Reflexive Autonomous Vehicle Control", Proc. of IEEE Intl Conf. on Robotics and Automation, San-Francisco, CA 1986
30. A. Meystel, "Knowledge-based Controller for Intelligent Mobile Robots", in Artificial Intelligence and Man-Machine Systems, ed. H. Winter, series "Lecture Notes in Control and Information Sciences, 80, Springer-Verlag, Berlin, 1986
31. G.N. Saridis, Self-Organizing Control of Stochastic Systems, Marcel-Dekker, New York, 1977
32. G.N. Saridis, "Toward the Realization of Intelligent Controls", Proceedings of IEEE, 67, August, 1979
33. G.N. Saridis, "Intelligent Robotic Control", IEEE Transactions on Automatic Control, Vol. AC-28, No. 5, 1983

---

*Technical report on Intelligent Mobile Autonomous System (IMAS)*

34. G.N. Saridis, J.H. Graham, "Linguistic Decision Schemata for Intelligent Robots", *Automatica*, Vol. 20, NO 1, 1984
35. G.N. Saridis, "Foundations of the Theory of Intelligent Control", Proc. of the IEEE Workshop on Intelligent Control, Troy, NY, 1985
36. R. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ 1957
37. Y. Bar-Shalom, "Stochastic Dynamic Programming: Caution and Probing", *IEEE Transactions on Automatic Control*, Vol. AC-26, NO. 5, Oct. 1981
38. H. Herrlich, G.E. Strecker, Category Theory, Allyn and Bacon, Inc., Boston, 1973
39. A.N. Kolmogorov, S.V. Fomin, Introductory Real Analysis, Prentice Hall, Englewood Cliffs, NJ 1970
40. M.D. Davis, E.J. Weyuker, Computability, Complexity, and Languages, Academic Press, NY, 1983
41. C. Isik, A. Meystel, "Knowledge-based Pilot for an Intelligent Mobile Autonomous System", Proc. of the First Conference on Artificial Intelligence Applications", Denver, CO, 1984
42. E. Koch, C. Yeh, G. Hillel, A. Meystel, C. Isik, "Simulation of Path Planning For a System With Vision and Map Updating", Proc. of IEEE Int'l Conf. on Robotics and Automation, St. Louis, MO. 1985
43. R. Chavez, A. Meystel "Structure of Intelligence For An Autonomous Vehicle", Proc. of the IEEE Int'l Conf. on Robotics and Automation, Atlanta, GA, 1984,
44. A. Guez, A. Meystel, "Time -Optimal Path Planning and Hierarchical Control via Heuristically Enhanced Dynamic Programming: a Preliminary Analysis", Proc. of the Workshop on Intelligent Control, Troy, NY, 1985
45. P.E. Hart, N.J. Nilsson, B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum-Cost Paths", *IEEE Transactions on Systems, Science, and Cybernetics*, Vol. SSC-4, No. 2, July 1968
46. A. Meystel, A. Guez, G. Hillel, "Minimum Time Path Planning for a Robot", Proc. of the IEEE Conf. on Robotics and Automation, San Francisco, CA, 1986
47. P. Graglia, A. Meystel, "Minimum Time Path Planning in the Traversability Space of an Autonomous Robot" < Proc. of the IEEE Symposium on Intelligent Control, Philadelphia, PA, 1987
48. D. Gaw, A. Meystel, "Minimum-Time Navigation of an Unmanned Mobile Robot in a 2 1/2 World with Obstacles", Proc. of IEEE Conf. on Robotic and Automation, San Francisco. CA, 1986
49. A. Meystel, Primer on Autonomous Mobility, Drexel University, Philadelphia, PA, 1986
50. R. Bhatt, D. Gaw, L. Venetsky, D. Lowing, A. Meystel, "Dynamic Motion Guidance and Trajectory Tracking for an Autonomous Vehicle", Proceedings of the IEEE Intl. Symposium on Intelligent Control, Philadelphia, PA, 1987
51. A. Meystel, "Purposeful Arrangements via Search in the Version Space", Proceedings of the IEEE Intl. Conference on Systems, Man, and Cybernetics, Halifax, Nova Scotia, Canada, 1984

## SECTION 2 MULTIRESOLUTIONAL REPRESENTATION OF SPATIAL KNOWLEDGE FOR AUTONOMOUS ROBOTS

In this section the concept of Multiresolutional Representation of Spatial Knowledge (MROSK) is defined and explored. The sensory process is examined to define two categories of error: resolutional error and perceptual error. This result is used for the formation of a MROSK which structural and maintenance mechanisms are discussed. MROSK is elaborated for a case with binary classification of space but extensions to more complex situations are given. MROSK was developed at Drexel University to support the Nested Hierarchical Controller for autonomous robots [1,2]. The material presented in this section is a condensed version of that given in [3] and represents an extension of an earlier work [4]. It is concerned with the task of developing a system that can function independently in a partially or fully unstructured environment. Environment is considered to be unstructured not only if it is unknown, or contains unknown object, but also if it consists of the objects which are related to each other in a previously not recorded configuration. This in turn requires existence of mechanisms that store and maintain knowledge of the environment. In this section the problem of dealing with a specific subset of this knowledge is addressed which contains two important classes of knowledge: physical knowledge (PK) and semantic knowledge (SK).

PK reflects the underlying assumption that the world of interest to IMAS exists as a set of physical objects, and that the act of sensing (measurement) does not affect the structure of the world. The physical world can be divided into regions which contain objects and occupy specific volumes. Partitioning of these regions is governed by the physical properties of these objects and their location in space. Information necessary for partitioning is obtained from measurements (sensing). The patterns, that emerge are characterized by quantities detected by a sensor (coordinates, color, shape, density, etc.) and are identified with the supposedly existing objects. When the changes are observed, the temporal aspect is included the resulting structure of the world. This includes moving objects, or their deformation. Physical objects as well as the results of sensing are uninterpreted knowledge of the world.

SK emerges in the process of interpretation of PK. Thus, the structures of SK are considered to be representatives of PK structures. SK structures appear as a result of labeling PK structures, including observations, and their combinations. In turn, SK structures may also be created via combining other SK structures. In both the PK and SK, the method of combination of structures to create other structures is governed by rules (grammars). In PK the rules of grammar are contained within the world itself). In SK these rules partially represent the relational information extracted from our observations and partially reflect our mechanisms of dealing with information.

This section is concerned with a portion of SK that contains structures corresponding directly to PK objects with a definite **spatial locations**. These SK define a System of Spatial Representation. Spatial Representation is a limited domain but it is very rich in information, and in this section we will limit further the scope of the world to be considered. Motion is excluded from our system of interpretation. Change is allowable if objects can disappear or change position (instantaneously and with limited frequency) creating a conflict between the state of the world predicted by memory and that created by nature. This definition can include cases of motion. Indeed, even if motion is represented explicitly the possibility for conflict with existing knowledge still exists.

The world is limited further since it is flat and does not contain curvilinear shapes. The only boundaries of objects allowed, are represented as polygons and only in the two dimensions of the horizontal plane.

**Mechanisms of Perception.** The overall process of perception is illustrated in Figure 1. The structure of the world is converted to raw sensory information which is then interpreted by the object identification module to determine the existence of objects. As the overall system becomes more familiar with its environment the learned information can be fed forward to aid with the object identification procedure. This mechanism is considered not to be involved with in the system of SK and is not necessary for the topic at hand.

Attention will be focused on the processes of the Object Identification Module (OBIM) since it is there that the natural structure of a Spatial Representation will reveal itself. It was found that two classes of errors of the Sensory Perception Systems can affect the operation of OBIM: 1.) Resolutional Error and 2.) Perceptual Error (illusions). A explanation for both of these types of error will be given for vision and ultrasonic systems. It should be emphasized that these are characteristics of all sensory systems.

**Formal Description of Perception.** The process of perception can be described using the following terms:

$I_{raw}$  -the set of raw sensory data in the vocabulary of the sensory mechanism known as the sensory parameter space

$I_{proc}$  -the set of sensory data that has been converted into the language (parameter space) of the knowledge structure. e.g. Range, angle and resolution.

O - an object which represents the connected subset(s) of  $I_{raw}$  or  $I_{proc}$  that are not allowable for the robot to travel in.

S is free space; it corresponds to the connected subset(s) of  $I_{raw}$  or  $I_{proc}$  that are allowable for the robot to travel in. The three functions that characterize the overall process are presented as follows.

$$P_{oi}: I_{proc} \rightarrow I_{proc} \quad (1.a)$$

$$P_{oi}: I_{raw} \rightarrow I_{raw} \quad (1.b)$$

Statements (1a) and (1b) define functions that partition either raw sensory data or sensory data in the language of the knowledge structure into subsets that correspond to objects and free space.

$$P_{trans}: I_{raw} \rightarrow I_{proc} \quad (2)$$

In the statement (2) the function of Perceptual Translation ( $P_{trans}$ ) is defined. It is a function which maps the information that is in the language of the sensor ( $I_{raw}$ ) to that of the knowledge structure ( $I_{proc}$ ). This function is order preserving with respect to the partitions that are created by the function given in (1a) and (1b). A feature that is necessary for those systems that perform object identification on  $I_{raw}$  (1a) rather than on  $I_{proc}$  (1b).

The sensory process can be further characterized as an information source. To simplify the description the source will include the process  $P_{trans}$ . This allows it to be modelled as a binary source. The probability of occurrence of each symbol will in reality be a function of the environment. However, the entire world is considered to be sufficiently diversified in the size and number of obstacles that the probability of object and of free space will be equal. Mathematically, Sensory System of an IMAS can be modelled as a Markov Process with finite memory.



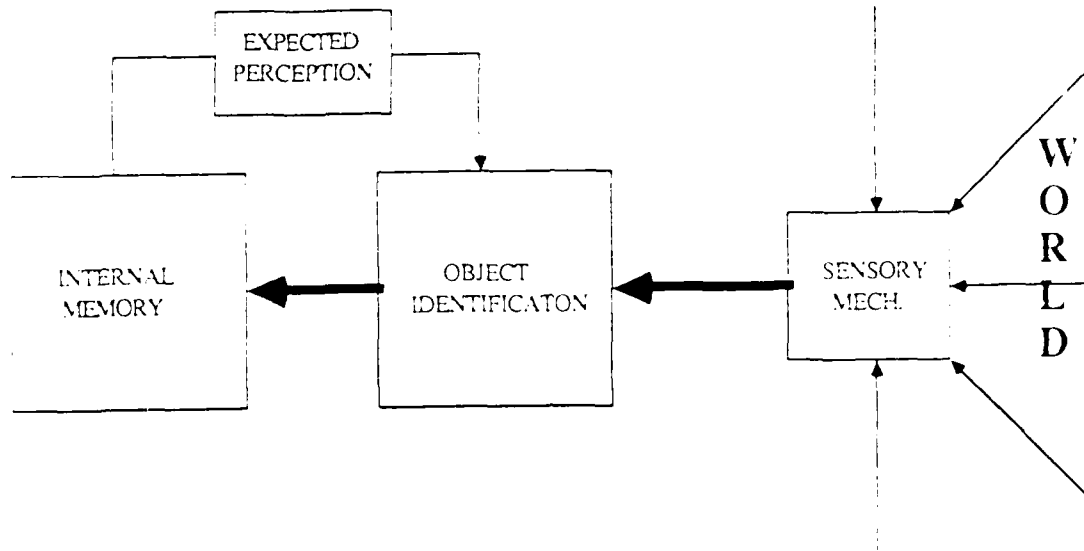


Figure 1. The overall perceptual process

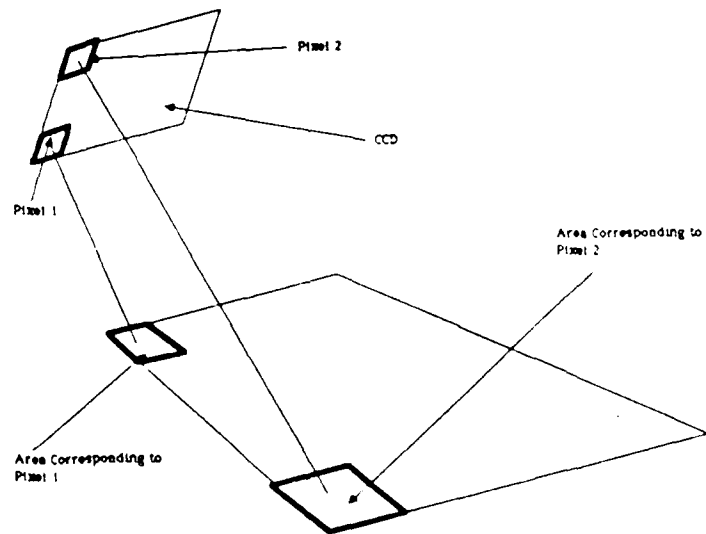


Figure 2. The error of CCD

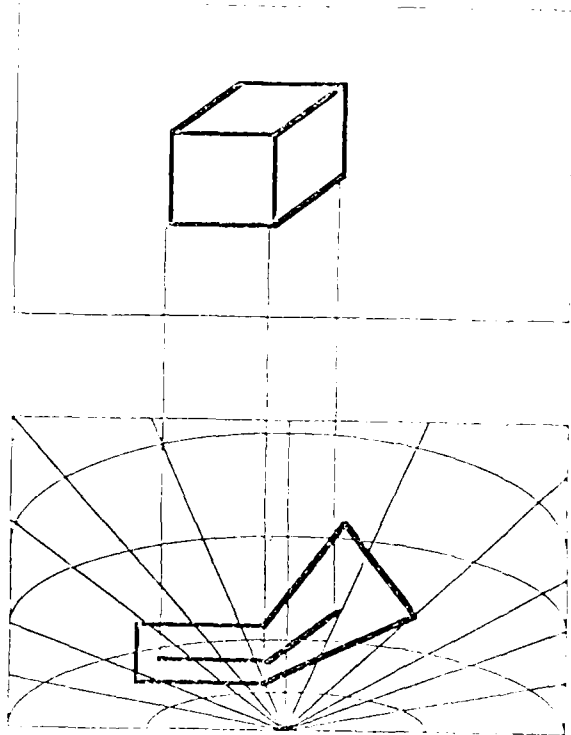


Figure 3. Resolutional distortion

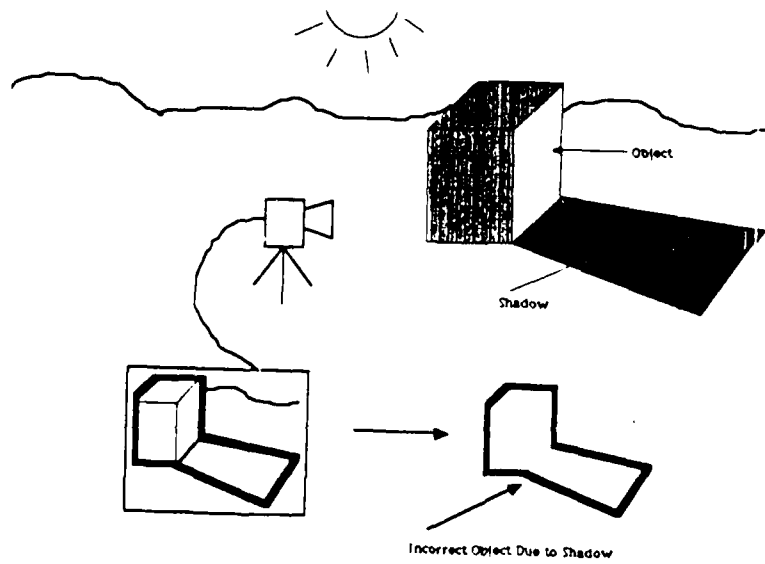


Figure 4. Perceptual error (in a vision system)

The probability of a symbol to represent an object would be a function of the previous symbols occurrences, and could be quantified if a measure of the average size of objects could be determined. However, for a first approximation the probability of object and the freespace to be equal can be calculated as  $H(x) = \sum P_i \log (P_i)$ , and for  $P_1 = P_2 = \dots P_n$ ;  $H(x) = \log (P_i)$ , therefore  $H(x) = \log (.5)$ .

**Resolutional Error and Illusions of Perception.** A complete and detailed description of the perception system discussed in this section can be found in [5]. (This system is accepted here because it fits to the overall concept of multiresolutional nested hierarchical control, and because the test-bed is based upon this system of computer vision). It determines the range to object boundaries using a unique algorithm for "Blob detection" in conjunction with a fixed camera angle and triangulation to determine distances to object boundaries.

In Figure 2 error from a finite CCD grid is shown. It is evident that the resolution of the system is different for pixels at the top of the image plane than for those at the bottom. This is shown for pixel 1. which represents a smaller area than pixel 2. This area corresponds to the accuracy with which the boundary can be determined. The distortion this would produce if modeled explicitly is given in Figure 3. This diagram shows the multi-resolutional nature of the information conveyed by this type of sensor. In Figure 4 the idea of Perceptual Error is displayed. Because of PK factors such as lighting, texture etc. the vision system is subject to errors that represent gross mistakes and can not be characterized by statistical analysis.

**Resolution and Perceptual Error of Ultrasonics.** The resolutional nature of the ultrasonic ranging system is depicted in Figure 5. The major factors that affect the errors in this system are beam width, discretization of scanning, and accuracy of the circuit used to detect the time of reflectance of the ultrasound pulse. Illusions of ultrasonics are, again, determined by a number of environmental factors. A few of the more common ones are demonstrated in Figure 6. In Figure 6a. the pulse is totally reflected and no object is detected. This could be alleviated if the sensitivity of the detection mechanism is increased but this will lead to another error shown in Figure 6b. The distance to the object is wrong since the range detected will be  $(x+y+z)/2$  rather than  $x$ .

**Nested Hierarchical Representation.** It is clear that all mechanisms that provide interface to the real world produce information that is multi-resolutional in nature. A representation that makes use of these phenomena was developed. In Figure 7 a Nested Hierarchical Representation (NHR) of an object boundary is given. The representation is said to be nested because all knowledge at level  $i+1$  is contained within the bounds of the resolution of level  $i$  (in a generalized form).

This structure must be easily maintained and updated because it should contain as much information as possible. Thus, it is necessary to introduce efficient and adequate Algorithms of Generalization (AG). Indeed, the contribution to any given level of the representation is limited by the processes of natural generalization that exist in the sensor (e.g. incorporated in the mechanisms of preprocessing). Also the amount of information given to any level for a single "snapshot" will decrease as the number of represented levels increases. The process depicted in Figure 8 shows how the process of AG helps fill in the upper levels (levels of low resolution) by generalizing higher resolutional information. AG should incorporate rules of inclusion from [1,2].

**Formal Description of Nested Hierarchy.** In this section a formal description of the NHW representation described will be given. The following meaning of symbols is accepted.

W - World ("Universe") of Discourse

R - Resolution defined as minimum distinguishable size of a feature

$K_w$  - Knowledge of the world

$K_w^i$  - Knowledge of the world at a specific (i-th) level of Resolution

O - Objects of the world

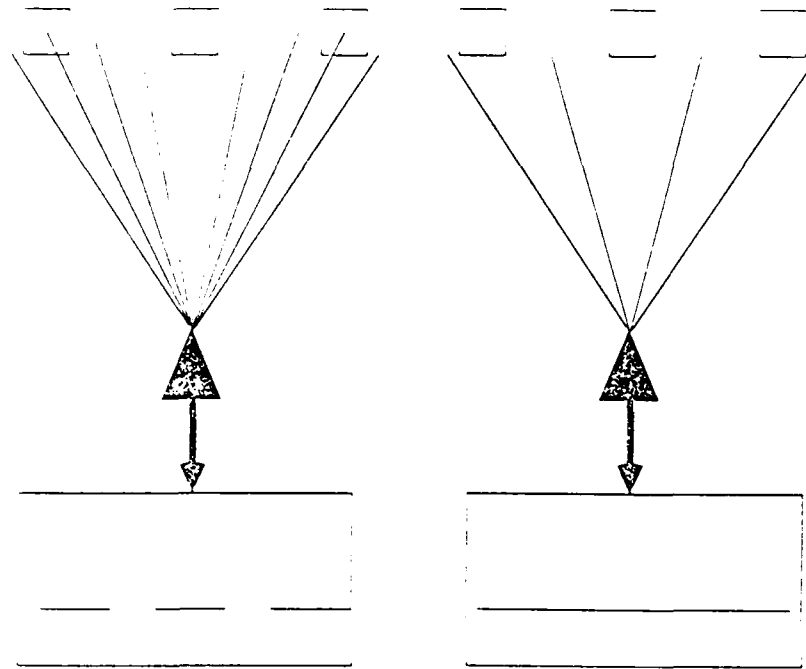


Figure 5. Design of Ultrasonics for Natural Generalization

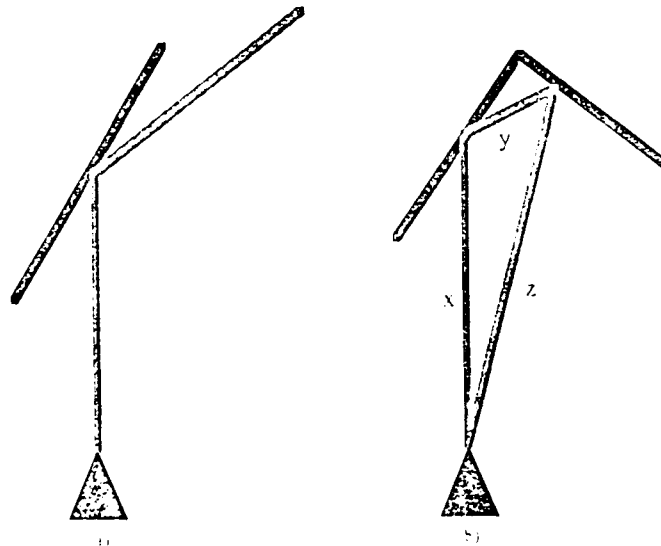


Figure 6. Illusions in Ultrasonic Sensors

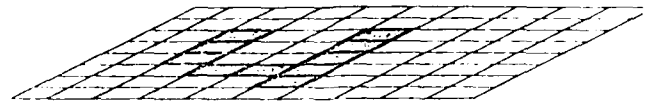
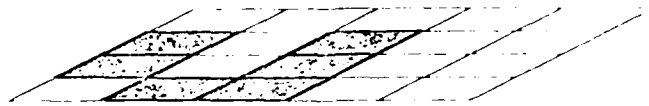


Figure 7. Nested Hierarchy of a Boundary

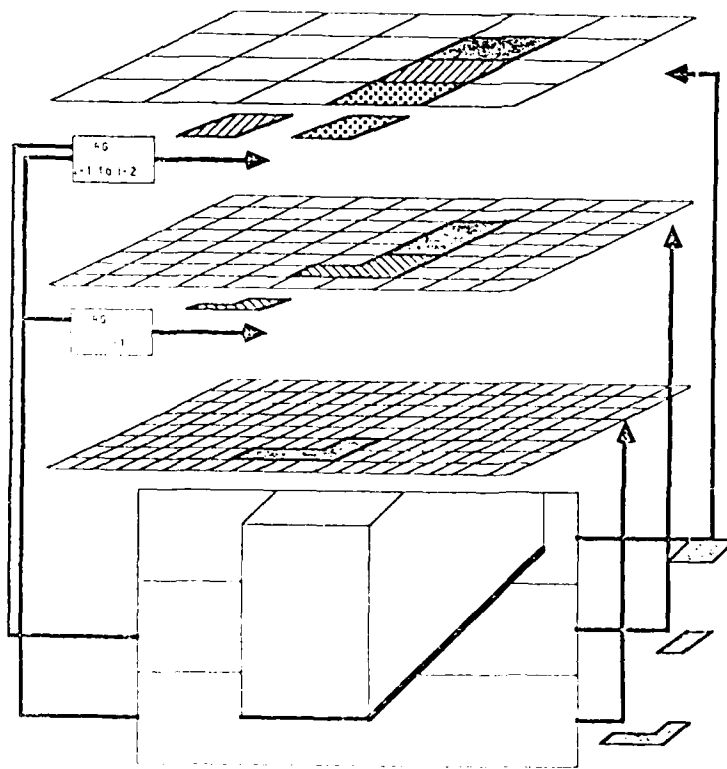


Figure 8. Process of Building a Nested Hierarchy

- $O^i$  - Objects of the world at a given level of resolution
- $S_f$  - Free Space of the World
- $S_f^i$  - Free Space of the World at a given level of resolution
- $S_u$  - Unknown Space
- P - Perception of the World
- $P^i$  - Perception of the World at a given level of Resolution
- $O_p$  - Objects of Perception
- $O_p^i$  - Objects of Perception at a given level of Resolution

In addition to the above notations, two classes of structuring relations will be defined.

- 1) Relations that determine Vertical Structure with respect to Resolution [1,2]. Within this class there are two relations
  - a) Generalization - G
  - b) Focus of Attention - Fa
- 2) Relations that determine Horizontal Structure for a given Resolution. In this class falls the mechanisms of Updating [3,4].

Algorithm of Generalization is a Recursive computational process that maps one level of Knowledge or Perception into a higher level. For every element of level "i" a corresponding element at level "i+1" is determined. Properties of this mapping are discussed in [1,2].

$$G : K_w^i \rightarrow K_w^{i+1} \quad (3)$$

$$G : P^i \rightarrow P^{i+1} \quad (4)$$

**Generalization.** By examining the process pictured in Figure 8 it is evident that there are two sources that provide knowledge to any given level of the hierarchy: the sensors, and Algorithm of Generalization (AG). It is this second process that plays a key role in maintaining consistency in the knowledge structure. To enable this it is necessary that AG be isomorphic to the actual mechanisms of PK that are used to generalize in the parameter space of the sensors, and are applied in the system of preprocessing. Since it would be cumbersome to duplicate algorithms utilized for the sensor parameter space, it was sought to find a language (alternate parameter space) and a transformation to that space that would enable a single mechanism to work for all the different types of sensors. This language is exactly that of the knowledge structure and the transformation is given by expression (2).

The overall process of AG as it functions in knowledge structure, parameter space is given in Figure 9. First the allowable size of a feature is reduced, see Figure 10 (reduction in resolution) then, second (as a result of this reduction) the objects that existed at level "i" will be transformed into another set of objects at level "i + 1". In order to determine the set of objects that exist at the level "i + 1", a procedure of object identification must be performed (see Figure 11). This is similar to the procedures of clustering but because of a significant number of heuristics that exist for this domain it is sometimes difficult to see the similarity.

**Artifacts of Generalization.** A number of interesting observations can be made that appear as a result of the mechanism of generalization and its application to a representation based on obstacle boundaries. It is evident that at some level of the hierarchy an object will be reduced (or abbreviated) to a single point, or a line. In Figure 12a the process of transition is illustrated from level "i" where the object has spacial dimension in all of the directions of its cartesian space (i.e. for two dimensions it has

length and width for three dimensions length, width and height.) to level "i + 1" where it has no spacial dimension and exists as a single point. A similar transition is shown in figure 12b here, the object loses

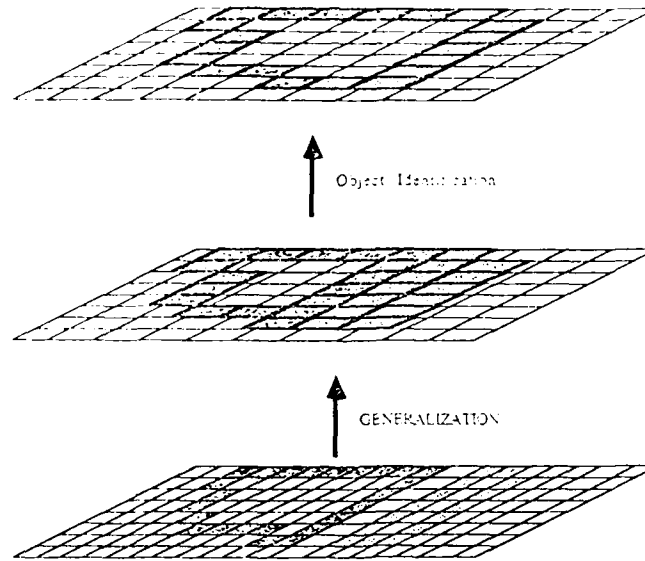


Figure 9. The Entire Process of Generalization

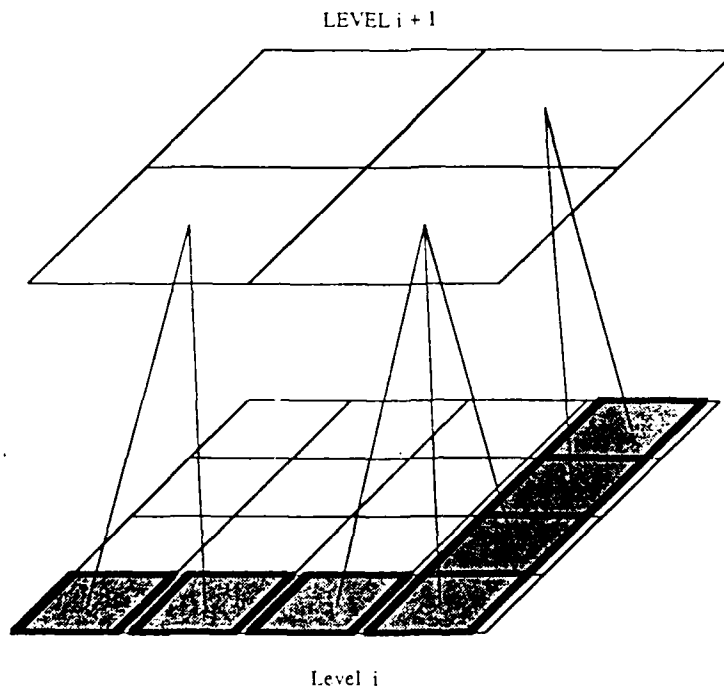


Figure 10. Generalization in a Nested Hierarchy

only one of its dimensions but at some level ( $i + n$ ) will eventually lose the second dimension as well.

These concepts are important for a number of reasons. One of the major reasons for identifying the correct transition level is to protect the insignificant objects from representation at the levels of very low resolution. This in turn poses restrictions on the correct change in resolution from one level to the next ("rate of resolution change"). To examine this problem, and to provide a more formal theoretical basis for the concept of Generalization, its information-theoretic aspects will be considered in the next subsection.

**Information Theoretic Description of Generalization** . In Section 2.3 the sensory system was shown to be characterized as a binary source delivering the two symbols, object and no-object, where the probability of occurrence of each symbol is (.5). In this section the effects of applying AG upon this information will be examined from an information theory side . The flow of information through the knowledge structure is shown in Figure 8. The amount of information at any particular level is given by:

$$I(x) = -\sum p_i \log(p_i) \quad (5)$$

where:  $p_i$  represents the probability of occurrence of the information in each grid cell.

However, since the probability of occurrence of each symbol is .5 the total information for N grid cells will be:

$$I(x) = -.5 N \text{ Log } (.5) \quad (6)$$

During the process of Generalization a grid cell at the level " $i + 1$ " is determined to contain an object boundary if any of the tiles nested with in it at level  $i$  contain an object boundary (see Figure 10). By referring to Figure 13 it can be seen that this causes a loss in information about free space. The maximum information loss that may result can be calculated using (5). In the case portrayed in Figure 13,a there is maximum information transferred between levels since there is no information about free space to be lost. The amount of correct information represented at level " $i + 1$ " is:

$$I_{i+1a}(x) = -.5 (x*y) \log (.5) \quad (7)$$

where  $x*y$  = the number of nested cells at level " $i$ "

In the case demonstrated in Figure 13,b the amount of information lost is maximum since it level " $i$ " contains the maximum amount of information about free space. The amount of correct information represented is given by:

$$I_{i+1b}(x) = -.5 \log (.5) \quad (8)$$

Therefore in the worst case the amount of information lost during generalization will be the difference between (7) and (8) given below:

$$I_{\text{loss}}(x) = -.5 (x*y - 1) \log (.5) \quad (9)$$

**Updating.** The representation scheme developed to this point would be rather ineffective since no mechanism(s) was introduced yet to modify it so as to reflect and incorporate the changing nature of the world. This mechanism is shown in Figure 14 for a single level of the hierarchy. In the first step of the procedure the Expected Perception of knowledge-to-appear is already formed in memory. Then



this is compared with the actual view to determine possible matches between objects in memory and objects currently in view. These candidates for possible matches, are then grouped into categories according to the amount of overlap and the number of overlapping objects (see Figure 15).

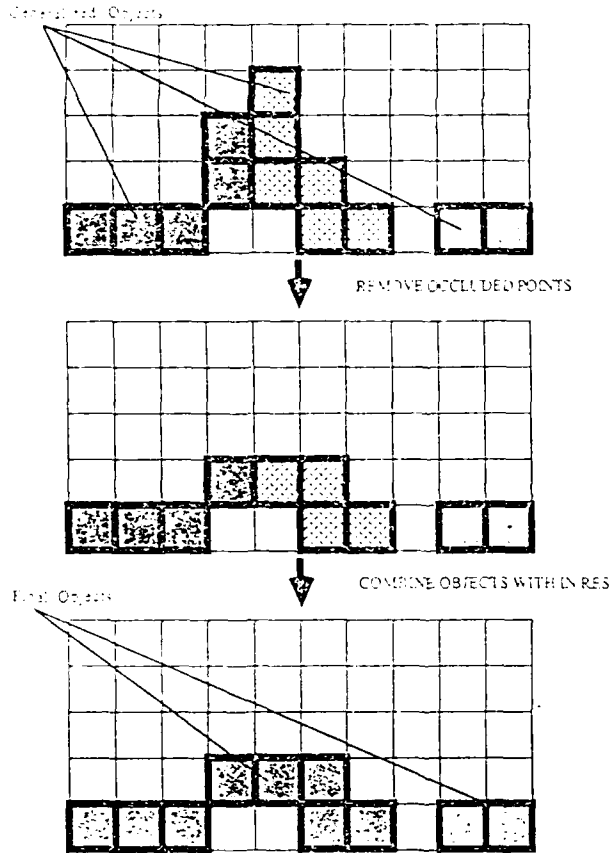


FIGURE 11. OBJECT IDENTIFICATION

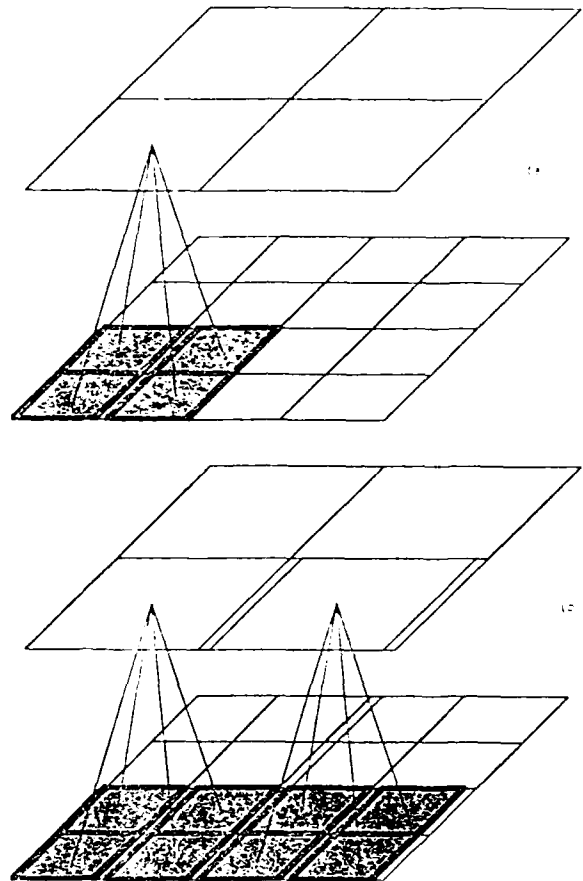


Figure 11. Object Identification

Figure 12. Artifacts of Generalization

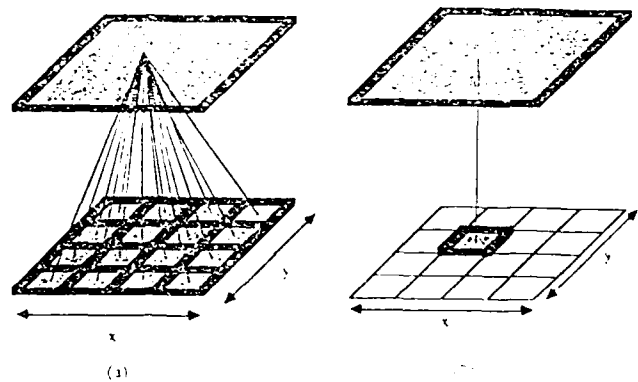


Figure 13. Loss of Information During Generalization

For each candidate for a possible match, a statistical decision is made to determine if there is actually a correspondence between new and old knowledge. If there is, then the old and new information are combined statistically and heuristically to produce an updated view of the object. If no correspondence is found then a decision is made whether to delete the existing object or to discount the new information as an object of illusion.

**Determining New and Old Knowledge Correspondence.** The problem of classifying new knowledge can be formulated in the terms of "detection theory". The first step is to characterize the signal mathematically so that it can be evaluated. In Figure 16 each object that is identified in the new sensor view can be thought of as a signal  $R_{\text{new}}(a)$ : the distance to a point on the object at a given angle  $a$ . A similar signal can be constructed for the corresponding object in memory  $R_{\text{old}}(a)$ . The representation can be extended to three dimensions by the addition of a third variable, namely, a second angle to vary  $R(a)$  vertically. Using this representation it is then possible to formulate the hypotheses that will be decided upon by the box labeled "Classify" in Figure 14. There are a total of three hypotheses:

$$H_1: R_{\text{old}}(a) + N(a) \quad \text{object in memory} \quad (10)$$

$$R_{\text{sen}}(a) = H_2: R_{\text{new}}(a) + N(a) \quad \text{new object} \quad (11)$$

$$H_3: R_{\text{ill}}(a) + N(a) \quad \text{object of illusion} \quad (12)$$

The first hypothesis ( $H_1$ ) represents the decision that the object that is currently being sensed is the same as the Expected Perception of the object already in memory plus some additive noise ( $N(a)$ ). The second hypothesis ( $H_2$ ) represents the decision that the object currently being sensed is not the same as the object that is in memory is actually a new object that is now in the same place as the old object. This could also be the same object but slightly moved or rotated, however, to the robot this is the a different object (see Section 1-2 for a better explanation). The third hypothesis ( $H_3$ ) represents the situation where the newly sensed object is really not an object but is a product of Sensory Illusion.

In the formulation of this as a *signal detection and estimation* problem a number of issues regarding knowledge of the statistics of the three classes of signals arise. The first major assumption that is made is that the noise that corrupts the three signals in (10), (11) and (12) is the same, and that a good estimate of its statistical quantities is known. The second assumption is that the signal corresponding to the object in memory ( $R_{\text{old}}(a)$ ) is a deterministic one, and can be used as a template. The largest error will be caused by the lack of information about the signal that makes up the deterministic portion of the input. The only true knowledge that we have, is that it is deterministic and that it is either the same as the signal in memory or it is not.

Numerous schemes exist for detection of known signals such as likelihood ratio test or crosscorrelation each with their limitations. For the purpose of demonstration the formulation and drawbacks of two popular methods will be given. The formulation using a cross-correlation method of detection will be presented first :

$$I_{R_1 R_2} = \sum R_1(a) R_2(a + b) \quad (13)$$

The discrete convolution given in (13) represents the cross-correlation of the object in memory with the object that is currently being sensed. It will produce a delta function at "b" if the two signals are similar and a flat curve if the signals are not similar. One of the major drawbacks of such a process is

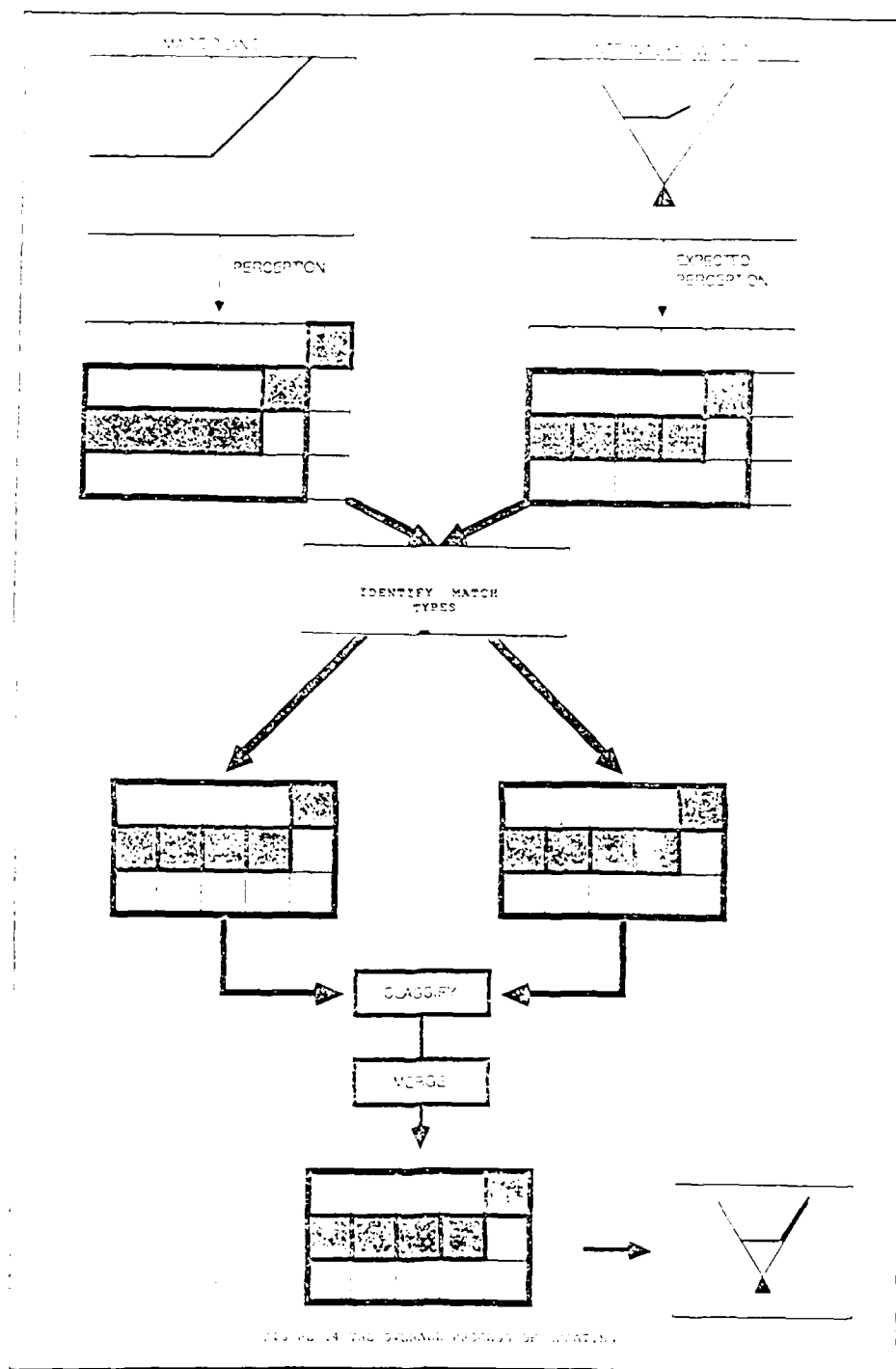


Figure 14. The Overall Process of Updating

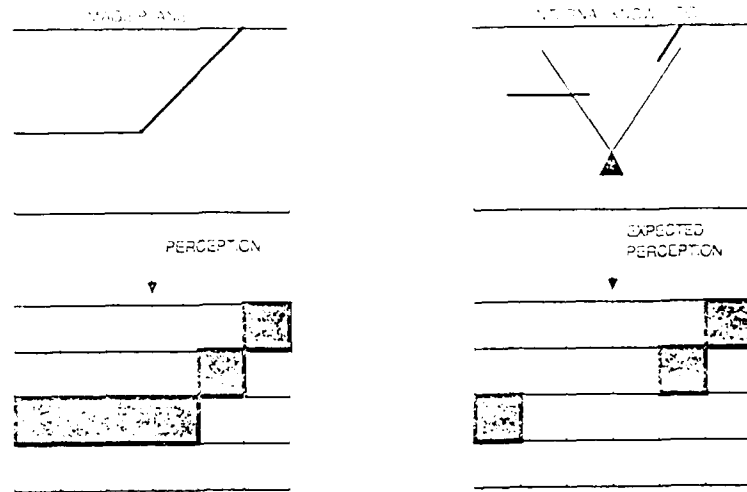


Figure 15. A case : The New Knowledge is Distributed Among Two or More Objects Contained in Memory (Which Provides a Unifying Role)

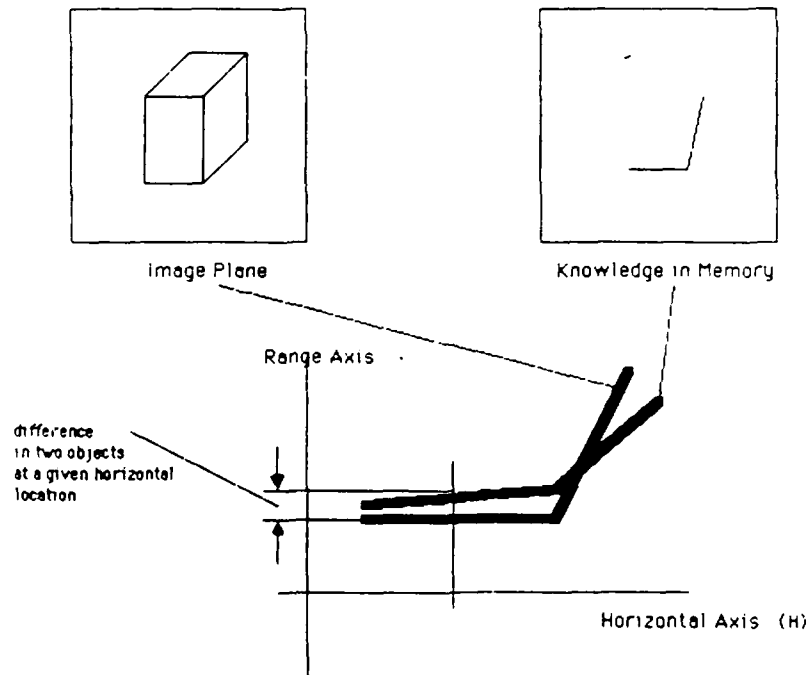


Figure 16. Each Object is Delivered with a New Sensor View

that for signals that are of very low frequency in nature, such as the distance information of a flat wall perpendicular to the field of view. This would have severe repercussions in a system of this kind since these type of inputs will be the rule rather than the exception.

The mean squared error (MSE) between the two signals is another measure of similarity measure that can be used to determine the correspondence between two signals. This can be formulated as follows:

$$\text{MSE} = (\sum |R_1 - R_2|^2)/N \quad (14)$$

This is essentially a measure of the variance of the two signals. If the value falls within the limit of error of a given level of resolution then the two signals can be judged to be the same.

After a decision has been made to determine which hypothesis to accept either (1) or both (2) and (3) further processing is required to find the best estimate of the new signal (if 1. is decided upon) or to differentiate between hypotheses (2) and (3).

**Incorporating New Information.** The process of incorporating new information if hypothesis (1) is accepted requires that the new information be combined with the existing information to produce the best estimate of the object. In a conventional approach based upon small Gaussian errors and a special form of the cost-functional, the process of incorporating new information can be formulated in terms of a Kalman Filter. This leads to a very complex computational structure which cannot be justified especially when the assumptions of the theory do not fit to the particular case of application. A much simpler approach is adopted here which involves the well known estimator of the mean:

$$R_{\text{updated}}(a_i) = (R_1(a) + R_2(a))/2 \quad (15)$$

**Deleting Old Information.** If the result of the decision process yields that either hypotheses (2) or (3) should be accepted, the problem of deciding between reality and illusion emerges. To deal with this a measure of Perceptual Certainty for the known object and the new information about this object, must be devised. This measure should be based on the 4 following criteria.

1. The number of times the object in memory was confirmed by hypothesis (1) or unconfirmed by (2) and (3).
2. The amount of the objects in memory that can be seen in the current view.
3. The overall Illusion/Reality Ratio for a given World and/or for a given Robotic Intelligence.

Since no sound theory for the development of such a scheme is presently known the following heuristics is devised based upon intuition. For each object a Perceptual Certainty Factor (PCF) is created employing the initial Illusion/Reality Ratio of the sensor. Each time a portion of the object is confirmed by hypothesis (1) the PCF is increased by an amount proportional to the length of the object that was in view. The PCF is decreased every time the hypothesis (3) is chosen by an amount proportional to the length of the object in view. If the PCF of an object falls below the PCF of the newly sensed data the new information is taken as fully correct, and the prior object representation is deleted from memory.

**Results and Discussion.** A crude implementation of the above theory was developed on a Symbolics 3640 for IMAS-2. Presently, a simulated sensor is used to produce three levels of

multi-resolutional data. This information is delivered to the knowledge organization module which incorporates the new information and confirms or deletes the existing knowledge. A typical representation of an object is built up over successive views from different distances and multiple angles. The three levels comprising the existing knowledge constructed in this fashion, are depicted in Figures 17a-19a. The current view of the world is given in Figures 17b-19b. The updated knowledge of the object after the procedures described in this section have been applied, is given in Figures 17c-19c.

Algorithm of Generalization was implemented using a polygonization routine along with the above mentioned method of determining the new objects and verifying the old ones. Using the polygonizer obtains the necessary reduction in resolution.

The current work is being performed to incorporate a vision system and sonar ranging device to replace the simulated sensor and to make the progression into real environments. Although the examples given in this section have been developed for binary polygonal object representation, this is not a necessary requirement. If the procedures of generalization and updating are implemented on a grid base, then any combination of shape primitives may be used to describe the object. All that is required is 1) a procedure to map the set of shape primitives onto the grid, and 2) a reverse procedure to map from the grid into the set of shape primitives.

If the extension to three dimensions is required, then another dimension must be added to each level of the hierarchy. In doing so the grid tiles will become three-dimensional cells (volumes and not areas). The problems encountered will be computational complexity and memory required. It is also possible to extend the representation to include information from texture and color. This can be accomplished by assigning a description of the color or texture that is contained in each grid cell that is enclosed by one of the boundaries. A few problems now occur that will require future investigation.

The mechanism of AG may become increasingly complex to furtherly develop since very little is known about processes that create effects such as a surface with rough texture appearing smooth at a distance or a multi-colored pattern that appears as a single color in low light levels. A careful examination of the Natural Generalization that occurs in the Sensor Parameter Space will be necessary if this is to be accomplished.

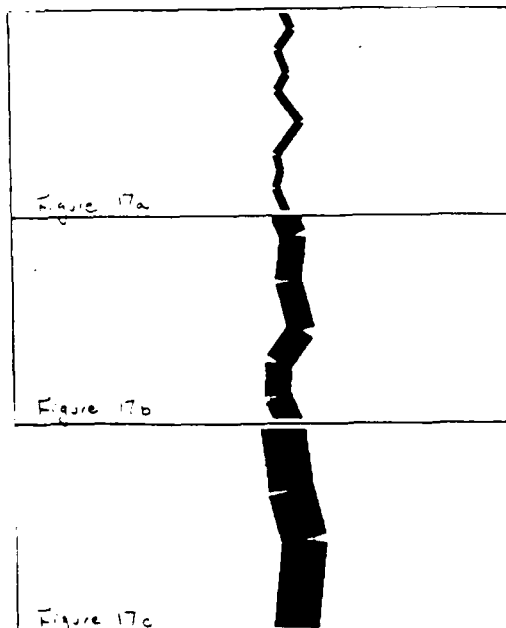


Figure 17. Simulation results I

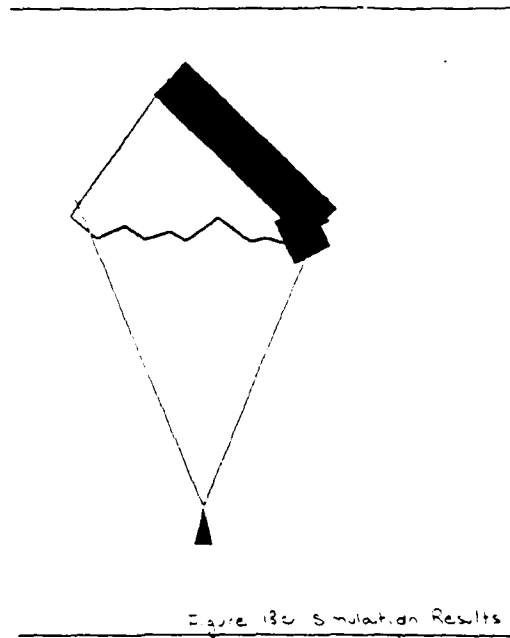
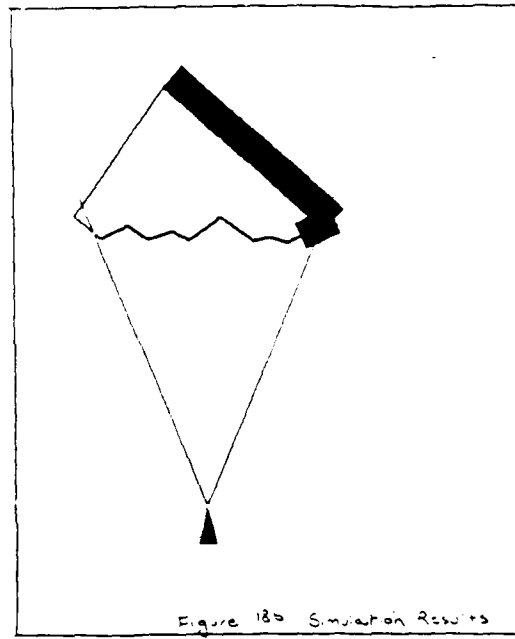
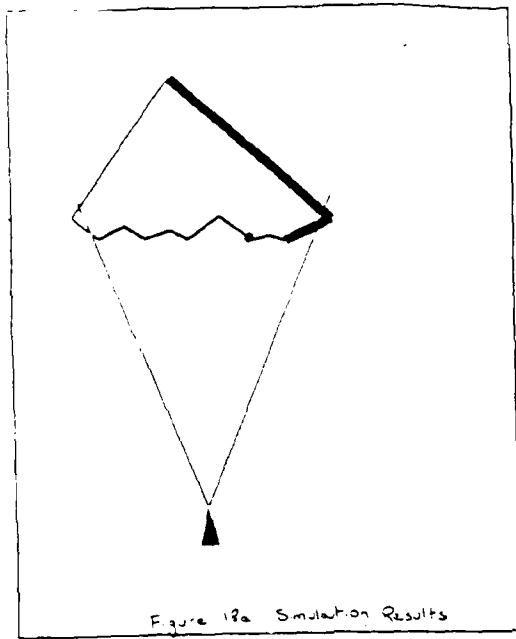


Figure 18. Simulation results II

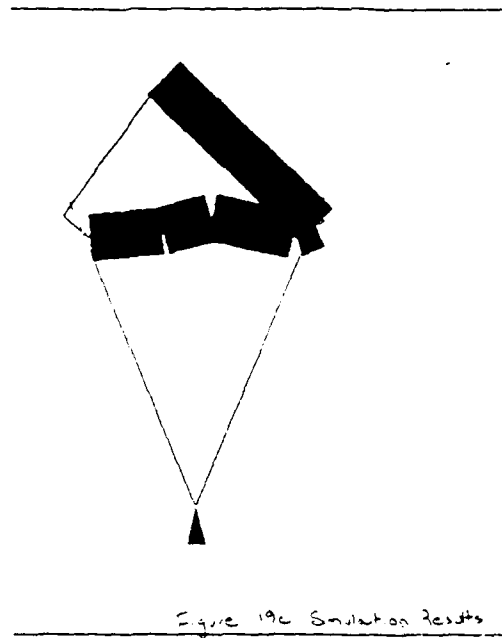
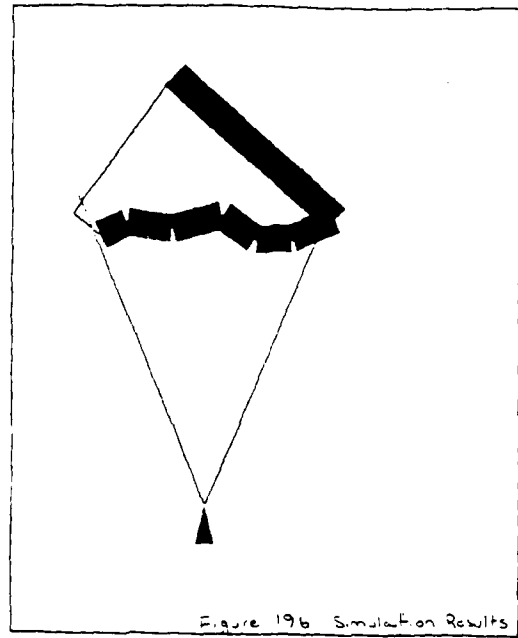
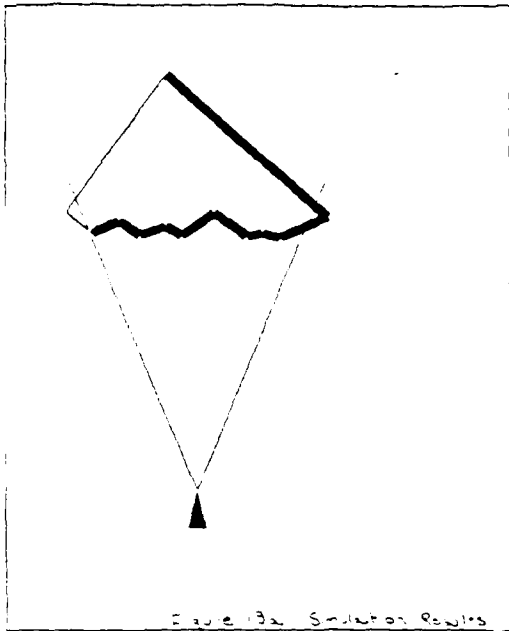


Figure 19. Simulation results III



### References to Section 2

1. A. Meystel, "Theoretical Foundations of Planning and Navigation", *International Journal of Intelligent Systems*, v. 2, No.2, Wiley, 1987
2. A. Meystel, "Planning in a Hierarchical Nested Controller for Autonomous Robots", Proceedings of the 25-th CDC, Athens, Greece, 1986
3. S. Waldon, Multiresolutional Representation of Spatial Knowledge for Autonomous Robots, MS Thesis, Drexel University, Philadelphia, PA 1987
4. S. Waldon, D. Gaw, A. Meystel, "Updating and Organizing World Knowledge for an Autonomous Control System", Proceedings of the IEEE Intl Symposium on Intelligent Control, Philadelphia, PA 1987
5. S. Uzun, A. Meystel, "Computer System For Structuring Images With No Edge Detection", Proceedings of the IEEE Intl Symposium on Intelligent Control, Philadelphia, PA 1987

### SECTION 3 PATH PLANNING IN THE RANDOM TRAVERSABILITY SPACE

Traversability space is introduced and developed as the representation in the path planning system of IMAS. A second (generalized) level of traversability space is introduced to reduce computational complexity and make the problem of control tractable. This lower resolution generalized level of representation is used to guide search in the original traversability space. Recursive repetition of this procedure is presumed to create a resolutional hierarchy of representations applicable for planning, and increasing its efficiency. This is achieved by successive reduction of the search envelopes. The results are analyzed with respect to the value of relative error vs search complexity. Comparisons are made between the envelope of search and heuristics of search.

**Planning in the System for Autonomous Control of IMAS.** In this section, the principles and the algorithms of planning the path in the traversability space are demonstrated in the context of control of an autonomous robot moving in the random field. IMAS is pursuing a goal while modifying its behavior as new information is acquired about its environment. (Theoretical background is given in [1,2]). This system is partitioned into six subsystems: sensor, perception, knowledge base, planning/control, actuation, and the "plant" or robot per se. This system is intended to develop plans and control sequences based upon knowledge stored in the memory and/or acquired during the robot operation. The path planning part of this subsystem combines the perceptual map and the prior knowledge about the system, which does not come from perception, to form a complete state description of the system and searches for a plan of actions. A detailed description of a system which combines information of the environment with its system model is presented in [3].

The goal-oriented procedure of path planning is then applied which generates a sequence of states which best satisfies the goal and is considered to be **the planned path**. These state descriptions are then submitted to the lower levels of control system. Finally, the actuation system performs a mapping of the description of the next state into inputs for the system actuators.

**Representation of the World for Planning in a Mobile Robot: Traversability Spaces.** All information acquired about the environment must be mapped into a form which can be used to control the motion of an autonomous robot. Each point of the space where the motion is to be performed (we will name it **terrain**), thus, corresponds to a location (value of **coordinate**), and a value which determines how that particular location in space will impede the motion of the vehicle (value of **traversability**). The minimal area which can be characterized by both coordinate and traversability is a discrete zone limited by the accuracy of the information available (e.g. distinguishable). The space into which all locations are mapped with their coordinates and traversabilities at a definite level of resolution, is the resolutional **traversability space**. Traversability space is developed in [7].

In fact, there are other factors which affect the motion of the robot besides the external environment. Along with the traversability factors, the physical parameters and the dynamic properties of the vehicle also affect the motion, and thus, the planned path. With all this additional information the complete **state of the robot can be described** at any instance of time. We will concentrate only on the upper levels of **planning in a multiresolutional nested hierarchical controller**, i.e. will consider the motion on a large scale. **When the vehicle's trajectory is considered on a large scale**, the dynamic properties such as inertia of the motion, or nonlinearities, such as turning radius and maximum acceleration, become insignificant. Under this assumption, the traversability space is defined in this section.

**Formal Definition of Traversability Space.** When the accuracy of the The traversability space of a mobile autonomous vehicle is defined as follows:

$$T_L = K(dS/dt)_L \quad (1)$$

where,  $T_L$  is the traversability at location L

$(dS/dt)_L$  is the maximum attainable velocity through location L  
 K is some positive constant determined by the properties of the robot and the environment.

If the location is fairly large with traversability defined in such a way, the time of traversing within a region R having a uniform traversability  $T_R$  for a distance  $D_R$  will be,

$$t_R = D_R / T_R \quad (2)$$

The environment is now mapped into a representation by which paths may be compared according to the time taken for the vehicle to follow a particular path. For example, a path P may be divided into N segments which lie in N traversability regions. The time taken to traverse path P is,

$$t_P = D_1/T_1 + D_2/T_2 + \dots + D_N/T_N \quad (3)$$

Clearly, the "precise" value of traversability can be computed if the size of the segment is equal to the minimal discrete of the space at a given resolution.

**Time Optimal Path Planning Using a Search in the Traversability Space.** Paths in the space can now be distinguished from one another by the cost of the path, (which for IMAS is time). We would like to distinguish the best path, between an initial point and the goal, from all the possible paths. This suggests that a search in the traversability space can be used to find the best path.

At this point we would like to determine the law of discretization of the space to transform it in a suitable form to conduct a search. The simplest tessellation is assumed based upon the scales on each of the dimension axes discretized using the minimal distinguishable value of the particular dimension. Thus our space is partitioned in the cells each of which is a multidimensional cube. To make the 2-dimensional terrain amenable to search, a tessellation of a grid is imposed. Each cell in the grid has **eight neighboring cells**. If the distance from the center of cell A to the center of its neighbor, cell B, is  $D_{AB}$ , then the cost incurred in going from the center of A to the center of B is,

$$C_{AB} = (D_{AB}/2T_A) + (D_{AB}/2T_B) \quad (4)$$

Since this cost is a constant (independent of both the path up to A and the path beyond B), a dynamic programming type of search may be used. Further, the recursive search employed in the dynamic programming may be made more efficient by including a heuristic cost of the euclidean distance to the goal ( $A^*$  with it total cost  $F=G+H$ , where G is the cost of achieving the next node, and H is a heuristic estimate of a possible distance to the goal). Euclidean distance is an admissible heuristic for H in the traversability space. This can be shown if we consider two nodes in the traversability space, n and n', and the goal. A triangle is formed by the three points and the triangle inequality can be written,

$$H(n') \leq C(n',n) + H(n) \quad (5)$$

where  $C(n',n)$  is the cost of going from n' to n. The smallest value  $C(n',n)$  is the euclidean distance from n' to n and for this value the inequality holds. So, for any other  $C(n',n)$  the inequality will also hold proving that the heuristic is monotonic which implies that it is also admissible [6]. The version of search algorithm  $A^*$  applicable to the traversability space is introduced as follows.

(1) Beginning at the initial node, the **eight neighboring successors** are generated and put into list OPEN. Generating a node consists of:

- (a) Assigning the cost (4) of arriving at the neighboring node

$$G_{i,i+1} = (D/2T_i) + (D/2T_{i+1})$$

where D is a width of the grid cell.

- (b) Assigning the heuristic cost to the goal

$$H\text{-cost} = \sqrt{(X_{\text{node}} - X_{\text{goal}})^2 + (Y_{\text{node}} - Y_{\text{goal}})^2} \quad (6)$$

- (c) Assigning the list of predecessors

$$\text{Predecessor-list} = (X_{\text{init}}, Y_{\text{init}})$$

(2) The node in list OPEN with the lowest G + H cost (or F-cost) is selected to be expanded. If this node is the goal node, the search for the optimal path has been successful. If list OPEN is empty, the goal can't be found and the search halts.

(3) All other occurrences of the node selected to be expanded, are removed from list OPEN. When a node is selected to be expanded, it is marked so that it will no longer be marked as a successor.

(4) The node being expanded generates the eight neighboring nodes as successors with the exception of those which are marked as having been expanded. The G-cost is the total cost of arrival to a given node "i" from the predecessor node "i-1", which represented recursively is,

$$G_i = D/2T_{i+1} + D/2T_i + G_{i-1} \quad (7)$$

or as a summation:

$$G_i = (\sum D)/T_1 + (\sum D/T_2 + \dots + (\sum D)/T_n) = D \sum (1/T_i), i=1, \dots, n. \quad (8)$$

where  $T_i$  is the traversability of region  $i$ .

The predecessor list is obtained by appending the location of the previous node to the predecessor list of the previous node.

(5) These successor nodes are then put in list OPEN and the process continues as described above starting with the selection of the best node on list OPEN.

(6) Loop to the stage (2).

**Permissible Computational Complexity For Real Time Control.** In the previous section it was shown that a A\* search operating on a square tessellation of the 2D traversability space of the terrain, yeields a time optimal path through the space. To integrate this planned path into the closed loop controller described in [1], measurements and constraints must be imposed on the results of planning. Given the nature of the path planning process described, path trajectories will be generated periodically, and **not** continuously. Thus, the time consuming search is expected to be less detrimental to the overall control system.

Nevertheless, under condition of limited computational resources we are interested in making the process of planning as less time consuming as possible, in order to use the computer for other levels of the planning/control hierarchy. Meanwhile, the computation time is proportional to the number of nodes expanded and this may be quite large depending on the planning area considered. In practice, search on the original level of traversability space is not fast enough to be considered for actual control application.

Alternatives to limit the computation time include (1) increasing the planning cell (tile) size, (2) defining a subgoal with a restricted area of search, (3) introducing a new level of resolution to guide the search in the lower level.

Increasing the tile size significantly will reduce computational complexity, but introduces an equally significant **discretization error** in the path found. Defining an arbitrary subgoal permits the system to function in **real time**, however, the system behavior while being locally optimal might be globally far from optimal. We will try to unify both of these options by defining manageable subgoals while maintaining a **global perspective** of behavior. This can be achieved by defining a coarser level of resolution and using its results at the lower level.

**Generalization and Creation of a New Level of Traversability Space. Types of the Traversability Spaces.** To reduce the computational complexity of search in the traversability space, a new level of traversability space is introduced. This type of control structure is presented in [5]. This new level should **adequately** represent the original information in a less detailed way. The conditions of adequacy include inequalities of inclusion thoroughly satisfied for all of the information items on hand. In this way, a path found at this new level will always contain the optimum path.

Thus, the coarse results of planning obtained at the artificial upper level of resolution, can be refined at the original level with substantial computational savings. Before the criteria for defining this new level are considered, the types of traversability spaces on which this criteria is applied will be discussed. Since we are dealing only with a two-dimensional traversability space in this section, we will use the term **map** instead of using the term **traversability space**. This will satisfy our common topographical intuitions.

**Uniform Regions Map.** The first type of map to be considered is the map of uniform regions. This map consists of regions of uniform traversability. This is a typical representation for large paved areas, a football field, a lake, etc.. Producing a generalized level for a map of uniform regions is straightforward since the value of traversability in upper cell will be identical to the traversabilities in the lower cells. The only consideration to be made is in dealing with upper cells which fall on the boundary between two or more regions. This is the discretization error introduced by the new level. The traversability of the border cells may be calculated from the weighted average of the lower cells or to be conservative it may assume the worst traversability of the lower cells.

**Non-uniform Regions Map.** The next type of map to be considered is the map of regions with non-uniform traversability. This map is not segmented into regions and each cell value is considered independently of its neighbors. For a real terrain, it is expected that there will be regularities within the map and that coarse regions do exist.

**Binary Traversability Map.** Another important type of map is the map containing only obstacles and free space. This is very common especially for man-made environments. The traversability space representation does not exclude this type of map as each of these represents the extremes of the traversability value.

**Random Traversability Map.** A random map (the most general case) consists of cells which are assigned a **random** traversability value. This fits the description of the non-uniform map, but it presents the **worst case** for determining an appropriate generalized level. So, a generalization function that performs **adequately** for the random map will be insured to perform at least as well, (and probably much better), on **any** non-uniform map that attempts to reflect reality.

**Generalization Criteria.** The criteria for generalization will now be discussed. Let the cell on the new level consist of 2x2 lower cells, having traversabilities  $T_{11}$ ,  $T_{12}$ ,  $T_{21}$ , and  $T_{22}$ . Let the size of the lower cells be unity.

The average time to traverse the larger cell will be the average time of diagonal, horizontal, and vertical traversal. If both horizontal paths and both vertical paths are considered, the average time is (Figure 1),

$$t_{avg} = (\sqrt{2}+1)/4 ( 1/T_{11} + 1/T_{12} + 1/T_{21} + 1/T_{22} ) \quad (9)$$

If the average distance,  $(\sqrt{2}+1)$ , is divided by  $t_{avg}$ , the generalized traversability becomes,

$$T_{gen} = \{ 1/4 ( 1/T_{11} + 1/T_{12} + 1/T_{21} + 1/T_{22} ) \}^{-1} \quad (10)$$

This generalization function works well for random maps and likewise it is appropriate for non-uniform maps and maps of uniform traversability. This generalization function is not, however, appropriate for a map of obstacle and free space. Generalization in this case should focus on critical elements of the lower level. Thus, if half of the large cell is obstacle, then the large cell should be considered an obstacle and should not be averaged or weighted less.

**Complexity of Two Level Control in the Traversability Space.** In the previous section, a generalization operator was introduced to make a new level of resolution. The complexity of path planning in a map with this new level is now considered.

Consider a random map  $R_1(i,j)$  and let  $R_2(i,j)$  be the generalized random map. Let  $e_1$  and  $e_2$  be the planning tile sizes of the two levels. An optimal path is found on level 2,  $O_2$ , and is to be refined on level 1. First  $O_2$  will be segmented into several areas of search for level 1. When the robot approaches an area defined to have a width of  $e_2 f$  and a length of  $D$ , the search begins for an optimal path. Each of these areas will be searched independently and successively and the total area covered on level 1 will be,

$$A_1 = n\pi (e_2 f)^2 + e_2 fD$$

if there are  $n$  subareas.

The number of nodes expanded on level 1 is,

$$N_1 = A_1 / e_1$$

The number of nodes expanded on level 2 is,

$$N_2 = A_2 / e_2$$

where  $A_2$  is the area covered by  $A^*$  search.

The total number of nodes expanded in the refined search is,

$$N_{12} = N_1 + N_2$$

If this is compared with search without the generalized level, a ratio of computational complexity is determined.

The complete search at level 1 is,

$$N_{1tot} = A_2 / e_1$$

The ratio of computational complexity is,

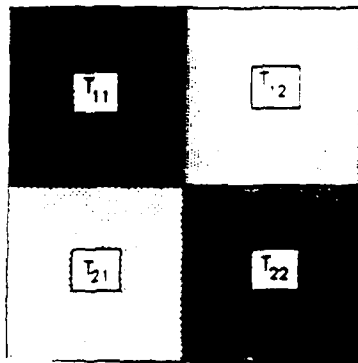
$$R = N_{12} / N_{1tot}$$

This simplifies to,

$$R = e_1 / e_2 + (ne_2 f(\pi e_2 f + D)) / A_2 \quad (11)$$

**Experiments with Two Level Search and Analysis.** Table 1 shows the number of nodes expanded for a single level search and this is compared with the two level search for four random maps. The path taken from the upper level is given to the lower level and made wider so that search is conducted in the stripe. Three widths are given for each map and the results are compared.

The first line under each map (A, B, C, and D) in Table 1 shows the number of nodes expanded in the generalized map. The optimal solution, that is, the lowest cost, is found by allowing unbounded search at the lower level (this is the second line under each new map). If we take the path



$$t_{\text{Horizontal}} = 1/2 [(1/T_{11} + 1/T_{12}) + (1/T_{21} + 1/T_{22})]$$

$$t_{\text{Vertical}} = 1/2 [(1/T_{11} + 1/T_{21}) + (1/T_{12} + 1/T_{22})]$$

$$t_{\text{Diagonal-1}} = \sqrt{2}/T_{11} + \sqrt{2}/T_{22}$$

$$t_{\text{Diagonal-2}} = \sqrt{2}/T_{12} + \sqrt{2}/T_{21}$$

$$t_{\text{avg}} = 1/4 (t_{\text{Diagonal-1}} + t_{\text{Diagonal-2}} + t_{\text{Vertical}} + t_{\text{Horizontal}})$$

$$= (\sqrt{2} + 1)/4 [1/T_{11} + 1/T_{12} + 1/T_{21} + 1/T_{22}]$$

$$T_{\text{gen}} = D_{\text{avg}} / t_{\text{avg}} = (\sqrt{2} + 1) / t_{\text{avg}}$$

$$T_{\text{gen}} = 1/4 (1/T_{11} + 1/T_{12} + 1/T_{21} + 1/T_{22})^{-1}$$

Figure 1. Determining the Generalization Function

Map	Level	Width	Nodes	Total	Cost	%Cost	%Complexity
			Expanded	Expanded	Increase	Decrease	
A	1	--	1381	--	898	--	--
	2	--	353	--	1028	--	--
	1	3	769	1122	916	2.0	18.8
	1	2	401	754	920	2.4	45.4
	1	1	169	522	956	6.5	62.2
B	1	--	1014	--	856	--	--
	2	--	311	--	986	--	--
	1	3	465	776	856	0.0	23.5
	1	2	335	646	863	0.8	36.3
	1	1	163	474	863	0.8	53.3
C	1	--	1119	--	892	--	--
	2	--	291	--	964	--	--
	1	3	413	704	892	0.0	37.1
	1	2	303	594	892	0.0	46.9
	1	1	163	454	902	1.1	59.4
D	1	--	1234	--	869	--	--
	2	--	385	--	1075	--	--
	1	3	533	918	869	0.0	25.6
	1	2	369	754	869	0.0	38.9
	1	1	163	548	880	1.3	55.6

Table 1 Computational Complexity vs Accuracy for Two Level Search with Maps A, B, C, and D

found in the search on level 2 and make it wider by three times the upper level tile size on each side of the path, we obtain the results found in the third line in each map. Note that the number of nodes expanded for this refined search includes the number of nodes expanded in the upper level search. The amount of cushion around the path is then reduced to two times the upper level tile size and then to one times the upper level tile size. The percent difference in cost from the optimal is shown in the last column.

Clearly, it is expected that there will be a tradeoff between finding the optimal path by unbounded search and doing a two level search. From the results shown in these four random maps it is surprising how little accuracy is actually lost. The strong relationship between the information on the two level gives a confidence that the optimal path will be found within an envelope surrounding the upper level path.

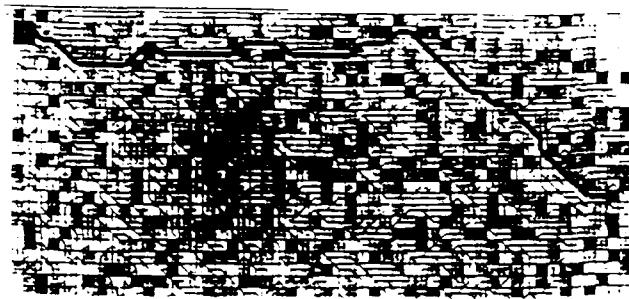
In Figures 2 through 5 the maps of Table 1 are shown with (a) being the level 2 search, (b) the unbounded level 1 search, (c), (d), and (e) being the bounded search with successively smaller envelopes of search. The envelop acts as a heuristic which directs the search. Figure 6 shows map A using best first search instead of A\*. Note how the similar A\* and best first are in the cases of bounded search. Figure 7 shows map A with unbounded search; first with best first and next with A\*. Although the euclidean distance heuristic reduces search drastically it is still less effective than a reduction of search based on the evaluation of generalized properties in the space.

**Conclusions.** 1. The traversability space representation is a uniform representation for the control of an autonomous vehicle accomodating varied terrain and obstacle/ free space environments  
 2. Generalization of properties is straightforward in the traversability space. The generalized map is strongly related to the original map and serves as an excellent means of guiding the search at the original level.  
 3. Because of the uniformity of the traversability space representation, the same algorithm is used on both levels of the system. This is a nice feature when a system is being implemented since no new algorithms are needed to deal with generalized information.

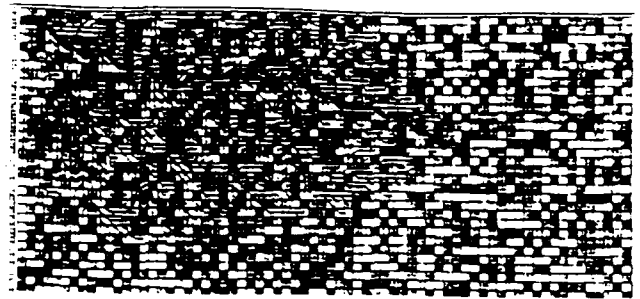
### References to Section 3

1. A. Meystel, "Nested Hierarchical System for Automatic Generation of Control Strategies", Proceedings of NATO Advanced Symposium on Languages for Robotics, Karlsruhe, FRG, 1986.
2. A. Meystel, "Planning in a Hierarchical Nested Controller for Autonomous Robots", Proceedings of the 25th CDC, Athens, Greece, 1986.
3. R. Bhatt, D. Gaw, A. Meystel, "A Real-Time Guidance System for an Autonomous Vehicle", Proceedings of Robotics and Automation Conference, Raleigh, NC, 1987.
4. S. Waldon, A. Meystel, "Updating and Organizing World Knowledge for an Autonomous Control System", Proceedings of IEEE Symposium on Intelligent Control, Philadelphia, PA, 1987.
5. R. Chavez, A. Meystel, "Structure of Intelligence for an Autonomous Vehicle", Proceedings of the IEEE International Conference in Robotics, Atlanta, GA, 1984.
6. R. Dechter, J. Pearl, "Generalized Best-First Search Strategies and the Optimality of A\*", *Journal of the Association for Computing Machinery*, Vol. 32, No. 3, July 1985, pp. 505-536.
7. P. Graglia, A. Meystel, "Planning Minimum Time Trajectory in the Traversability Space of a Robot", Proceedings of IEEE Symposium on Intelligent Control, Philadelphia, PA, 1987.

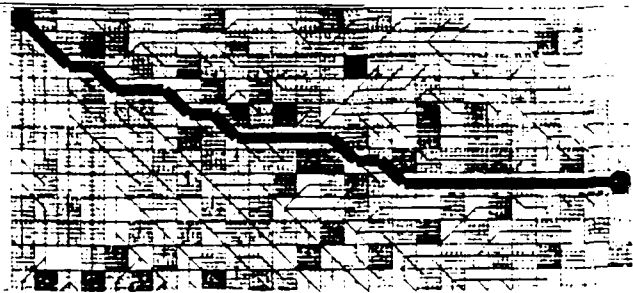




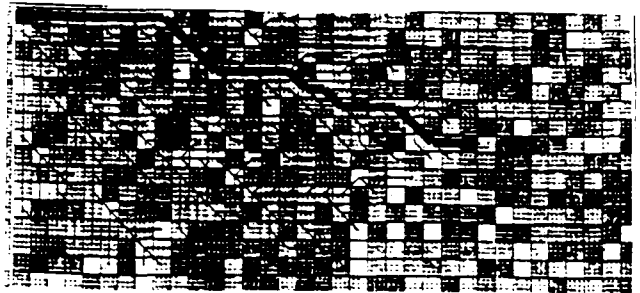
a)



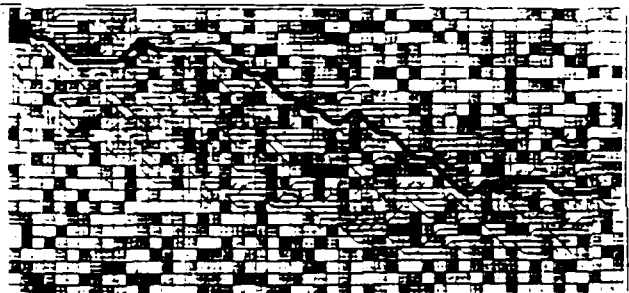
a)



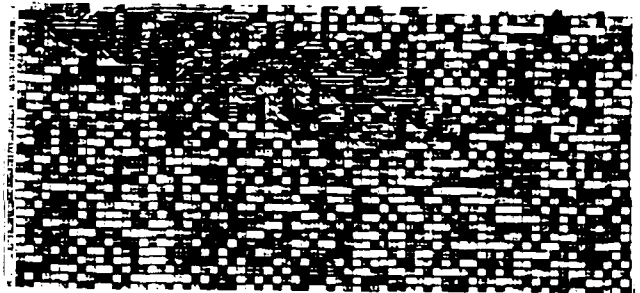
a)



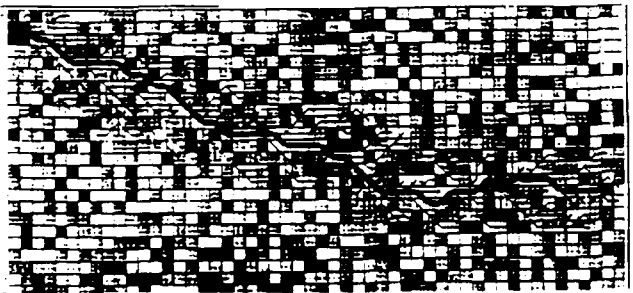
a)



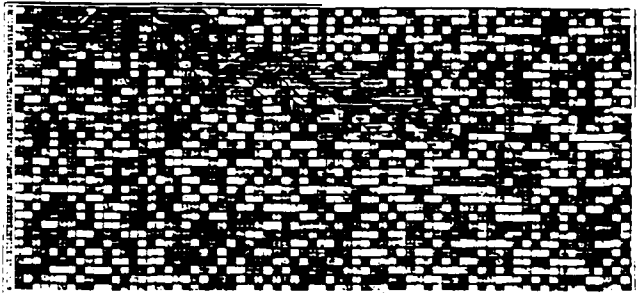
a)



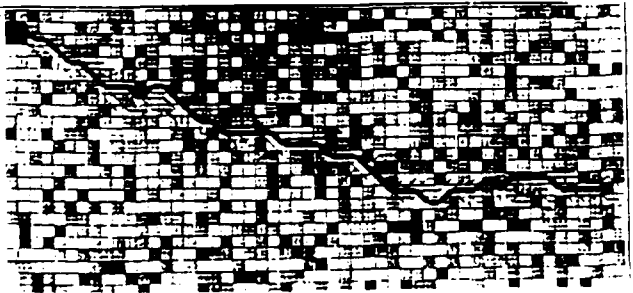
a)



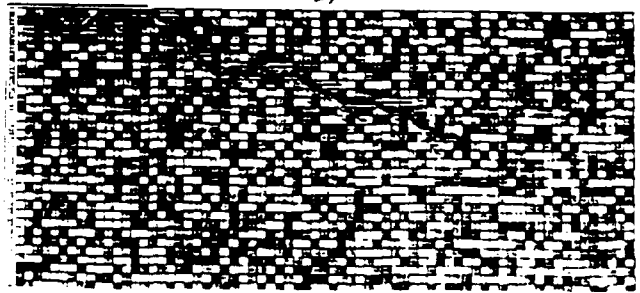
a)



a)



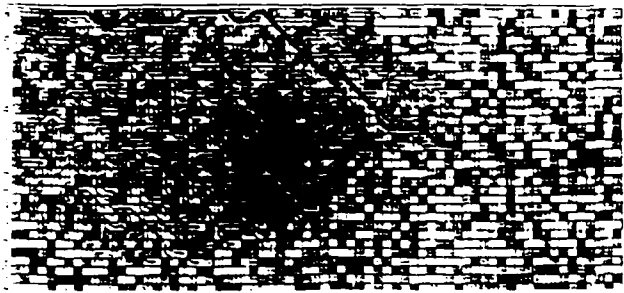
a)



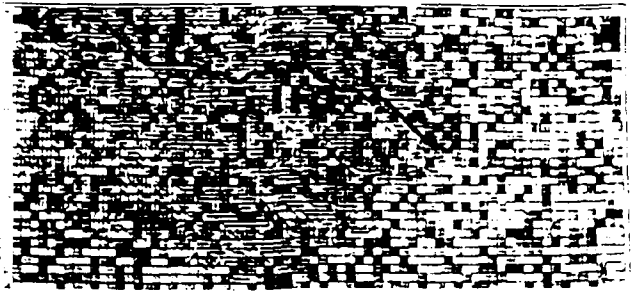
a)

Figure 2. Experiment on Map A (NEST)

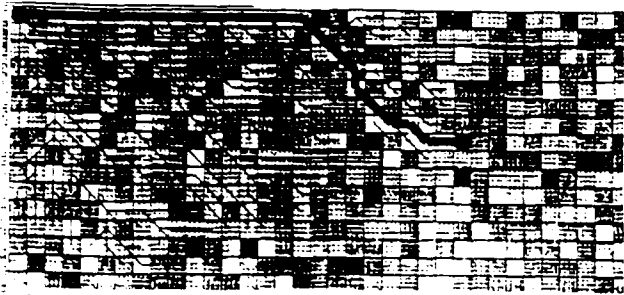
Figure 3. Experiment on Map B (NEST)



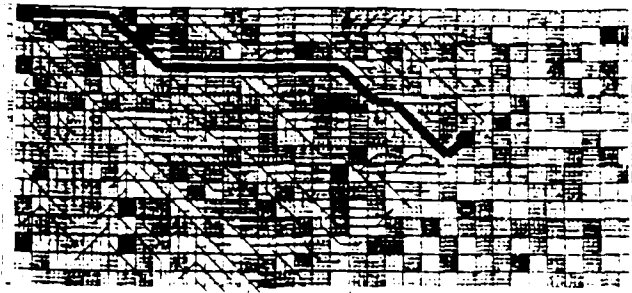
a)



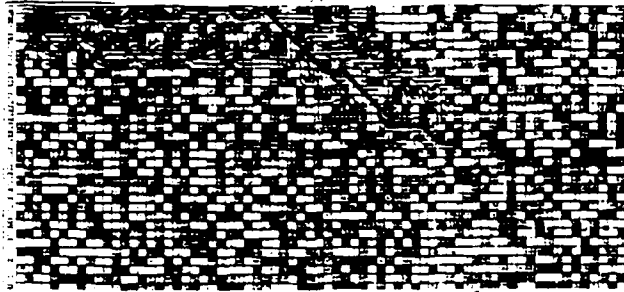
a)



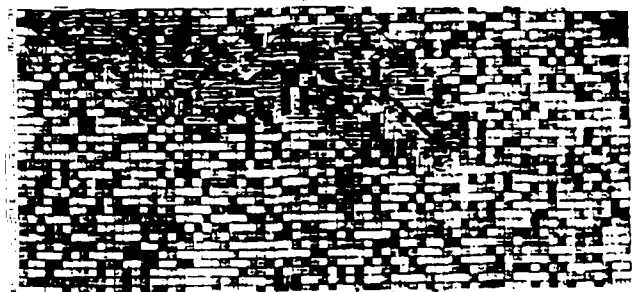
a)



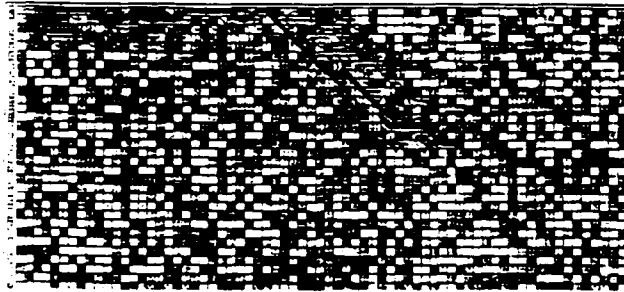
a)



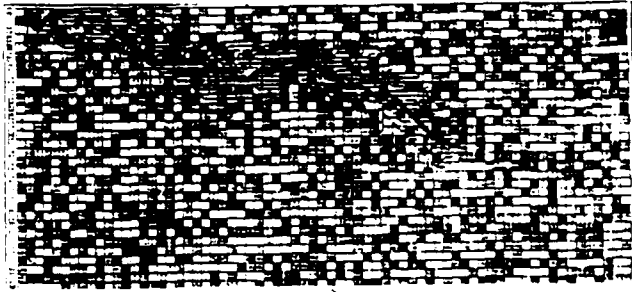
a)



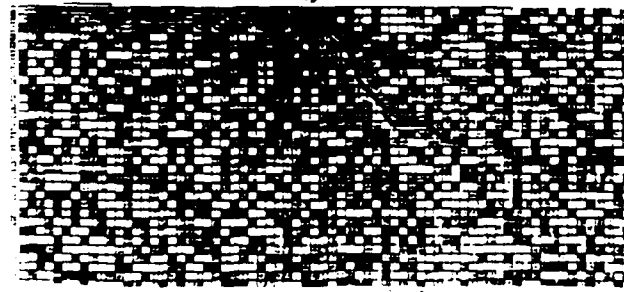
a)



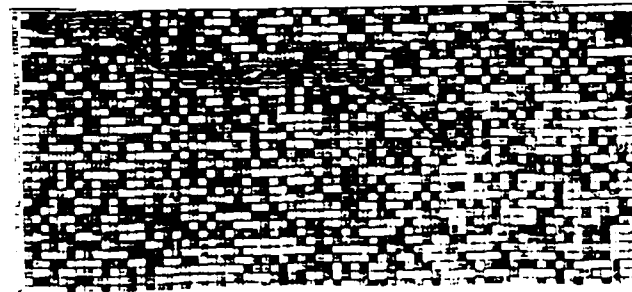
a)



a)



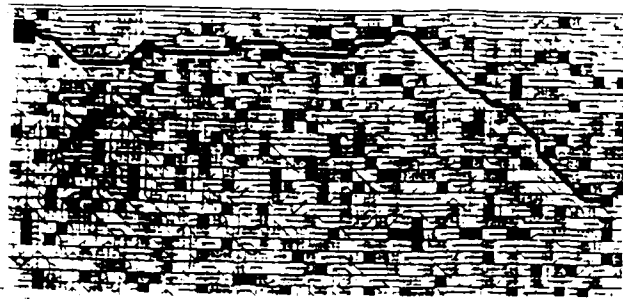
a)



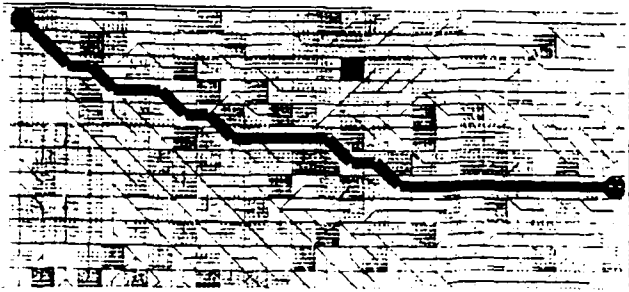
a)

Figure 4. Experiment on Map C (NEST)

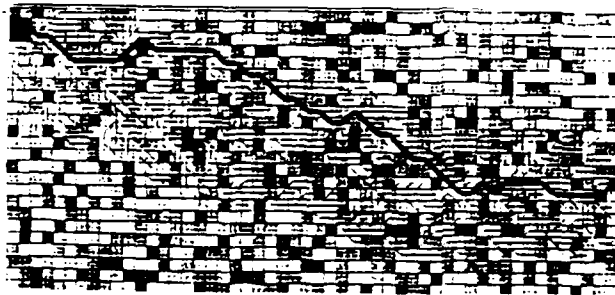
Figure 5. Experiment on Map D (NEST)



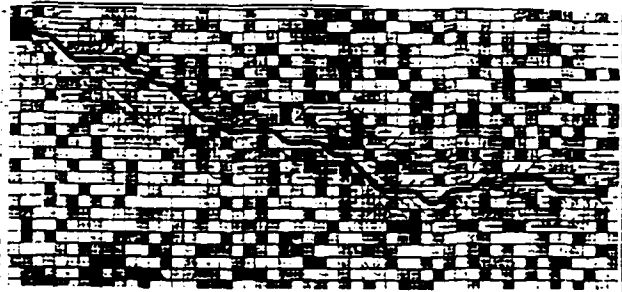
a)



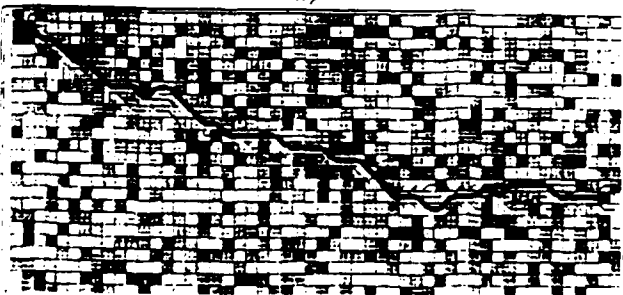
b)



c)

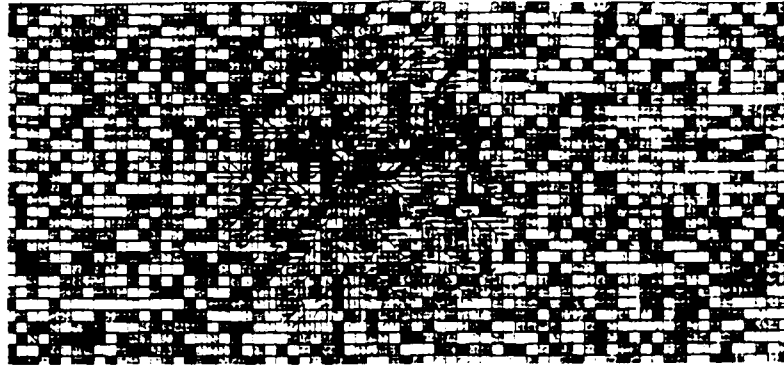


d)

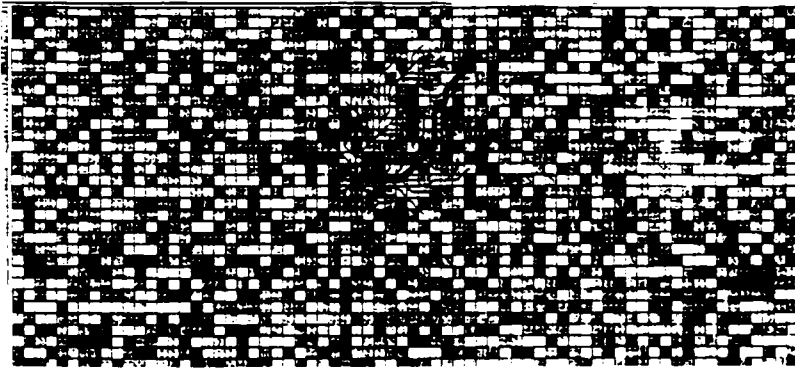


e)

Figure 6. Experiment on Map A using exhaustive search (Dijkstra)



a)



b)

Figure 7. Comparison of the enhancement-front for Dijkstra (a) and A\* (b)

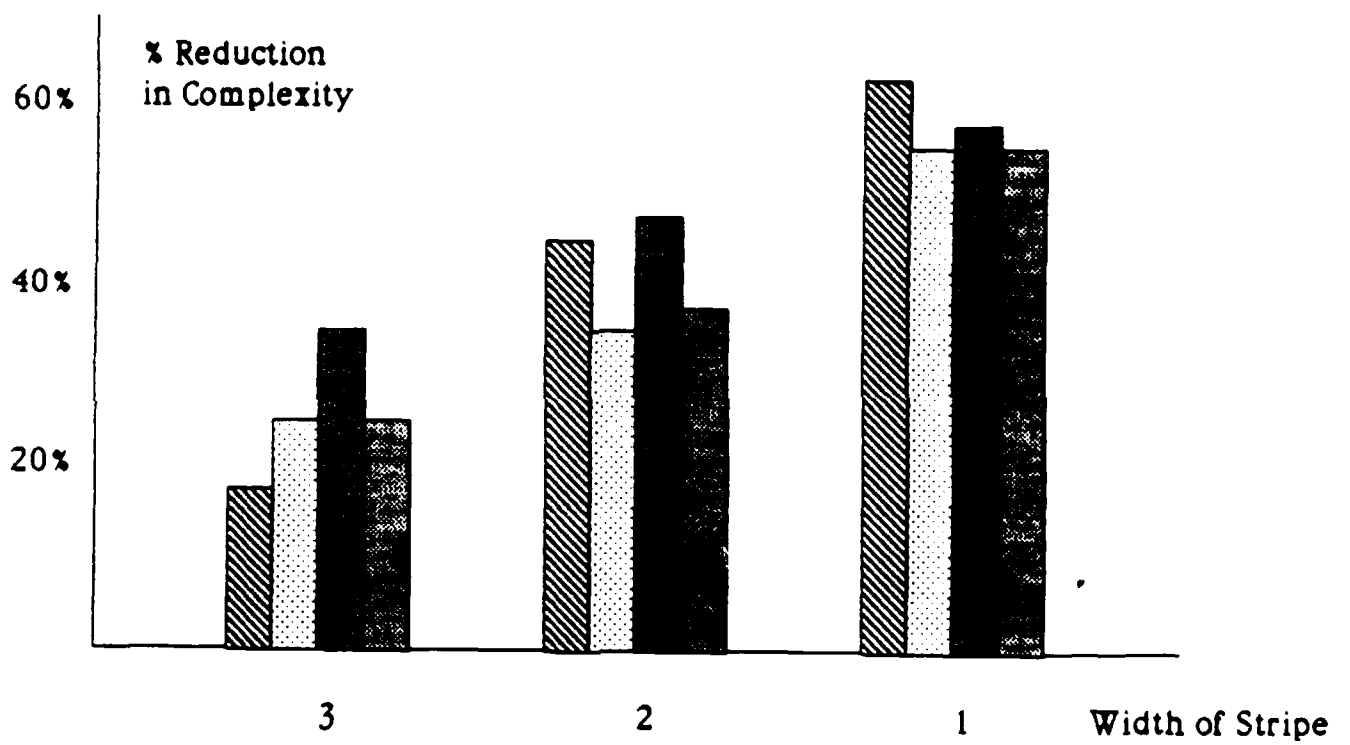


Figure 8. Computational complexity as a function of the "width of stripe"

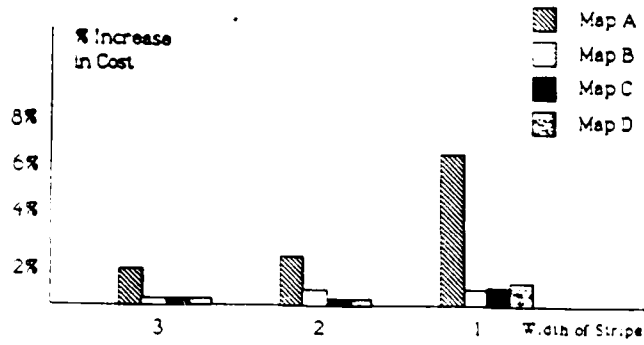


Figure 9. Increase in cost as a function of the width of stripe"

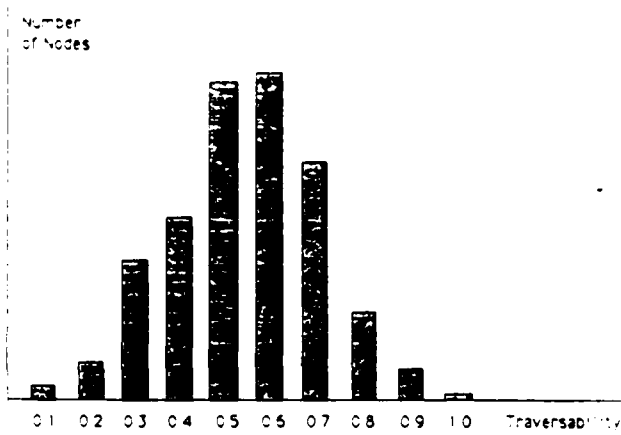


Figure 10. Histogram of Map A after Generalization

## SECTION 4 PILOT WITH BEHAVIORAL DUALITY

This section describes a system of guidance for an intelligent mobile autonomous system (IMAS) based upon an algorithm of "pilot decision making" which incorporates different strategies of operation. Depending on the set of circumstances including the level of "informedness", initial data, concrete environment, and so on, the "personality" of PILOT is being selected between two alternatives: 1) a diligent and thorough strategist which tends to explore all available trajectories off-line and be prepared to follow one of them precisely, and 2) a hasty decision maker inclined to make a choice of solution in a rather reckless manner base upon short term alternatives not regarding long term consequences. Simulation shows that these two personalities support each other in a beneficial way. This section is based on the concept of Nested Hierarchical Controller [1,2]. Intelligent Mobile Autonomous System (IMAS) is considered to be controlled by a Planner-Navigator-Pilot type of an intelligent controller [3]. Only Pilot's level is analyzed. The concept of behavioral duality is a development of our prior papers exploring this level of a multiresolutional controller [4-6]. Pilot level with behavioral duality was thoroughly simulated in the Laboratory of Applied Machine Intelligence and Robotics (LAMIR) of Drexel University, and applied in the IMAS-2 (Drexel-Dune-Buggy) which has been developed under the contract with US Army.

**Types of Decision Making Behavior in the System** . Even with the few existing different approaches to motion control, it seems clear that each algorithm has its strengths and weaknesses. It does not appear that a method can be devised which is totally appropriate for all combinations of environment, situation, and physical system [7]. An obvious remedy then is to design a multiplicity of special purpose algorithms each with clearly defined "operating limits", and to use a meta-control, or dispatcher to choose among these available algorithms depending on the circumstances. Each of these algorithms results in a definite **behavioral "portrait"** of the Intelligent Mobile Autonomous System (IMAS).

A multiplicity of the algorithms with different strategies of operation, and thus, with different behavioral portraits should result in a **behavioral plurality**. We have taken this approach, and use two algorithms, one which is best for known or slowly changing environments, and the other which is more appropriate for unknown or highly dynamic worlds. Our motivation was to analyze a particular case of **behavioral duality**.

A "vocabulary of situations" is introduced. This is a simplified predicate calculus which is used to classify a given world description so that the familiar set might be recognized, and the appropriate control can be found. This language must support "fuzzy" descriptions, since we would quickly be swamped, computationally, if we tried to describe every detail of each situation uniquely. Also these fuzzy descriptions should differentiate between aspects of the situation *which most affect control decisions*, and obscure differences which do not.

**Known world.** In the case of *a priori* known environments, the problem of motion control reduces to that of **planning a complete trajectory**, and then tracking it, with possible compensation for error. Since the world is given, the complete trajectory can be found *before motion begins* (ultimately, off-line) and thus more time consuming methods may be employed.

For most of the applications, typical situation is based on a partially known world. Although the robot may have a complete map of his surroundings (e.g. factory), it will inevitably be required to deal with unforeseen (moving or temporary) obstacles. Thus, compromises are required, and a trajectory should be planned *which is likely to be achievable*. This reduces the computation required for each trajectory, while ensuring that the path found will not become invalid too soon. The "Precise

Preplanning" algorithm is doing this job.

Search is recognized as the most general approach to the trajectory planning problem in known worlds. Often various heuristics are used to reduce computational complexity. Our approach has been to use generalized results of search done *off-line* (with no constraints on the time of search) to create the **generalized rule base** used on-line. The "generalization" of the search results then can be seen as a pruning *after the fact*, where each of the pruned regions are transformed into a description of the (class of ) situation common to the pruned set, and a trajectory representative of the pruned set.

**Unknown World.** The other extreme is a completely unknown world. Undoubtedly, any robot architecture is based on certain assumptions about its intended environment, and in that sense the world is never *completely* unknown. However, we are interested in the case when no information is given to the robot, other than is a part of "built-in" assumptions. In this case, planning complete trajectories, at any stage of the motion, is not reasonable since they will quickly become invalid.

There are far fewer existing "solutions" to the problem of motion planning in unknown worlds.. A main reason for this is that there is no clear method for evaluation of the performance of any such algorithm. Whatever parameter the decision mechanism is attempting to "optimize", must be evaluated in terms of the information the system has at a given moment, since the robot's knowledge is constantly changing.

**Precise Preplanning algorithm ( PP ). General Structure and Concepts.** In a completely known world, the problem of planning a dynamically feasible ( minimum time ) trajectory in the presence of obstacles can be formulated as an optimal control problem, with the obstacles acting as constraints on position. One way to approximate a solution to this problem is by using search in a state space consisting of the state variables of the vehicle. Positions of obstacles can then be superimposed on this space making those positions "illegal" states. Using a model of the vehicle, a *successor generating function* can be defined, which will generate the next possible states, given any one state. Time of motion from one state to the next can be used as the cost of each arc in a graph generated by recursively applying the successor generator. A search algorithm can then be applied to find a minimum time trajectory. Details of an implementation of this trajectory-finding search procedure can be found in [6].

Although the method mentioned above does indeed yield a minimum-time solution to the control problem, it is too computationally expensive to be used *on-line*. The following observations led us to a solution to transform results of search performed *off-line* into a rule form, usable *on-line*. Firstly, in the "similar" situations, optimum trajectories are "similar". Secondly, only a limited number of "characteristics" of the situation determine the resulting trajectory. Thus, a language of situations to put in correspondence a rule base of trajectory classes, and the base of associated situation descriptions. Then the rule base can be defined as a set of rules in a form:

$$(\text{situation description}) \implies (\text{trajectory})$$

where the "trajectory" is an actual trajectory instance meant to be representative of the entire set of trajectories within the bounds of acceptable error.

More formally we can state this as follows. Let the "planning universe" (U) be a discrete space represented at some resolution  $E_u$ . Then any trajectory planned on this space will have accuracy or resolution ( $E_t$ ) of at most  $E_u$ . We will use  $D_i$  to denote the i-th situation description, from the set of all situation descriptions which are possible given the vocabulary V. A situation description is



intended to be an n-tuple, with each component having a range of discrete values which it may assume. The set of all "real world" situations ( at resolution  $E_u$ ) described by a given  $D_i$  will be expressed as  $S(D_i)$ .

Notice that this implies that there is a many-to-one mapping between *situations* and *situation descriptions*. This is because we generally assume that our situation descriptions are at a lower resolution than the "representation" from which they are derived. In some cases, this "representation" may be the world itself. In this case, the "resolution of the world" means the resolution at which we can measure or sense the world. Actually, the resolution at which one should represent situation descriptions will depend on the accuracy of motion that is ultimately required. This relationship between the various discretizations and required accuracy of motion will be discussed later.

Using the notation given above, we can describe a *PP* motion control rule as follows.

Let  $T(D_i)$  = set of minimum cost ( eg. time ) trajectories, one for each element of  $S(D_i)$ .

$T_i$  = a "representative" trajectory.  
 $T_i \in T(D_i) \mid \text{Error}(T_i, T_k) < \text{epsilon}$   
 for all  $k \mid T_k \in T(D_i)$

where "Error ( $T_i, T_k$ )" is some measure of difference between  $T_i$  and  $T_k$  in terms of the cost functional being used.  
 ( eg. time of motion ).

Notice that  $\text{epsilon} < \text{Error}(T_j, T_k)$  where  $T_j, T_k \in T(D_i)$   
 and  $T_j$  is highest cost trajectory, and  $T_k$  is lowest cost.

Then a rule can be represented as:

$$D_i \rightarrow T_i$$

which means " If the situation can be described by a set  $D_i$ , then use trajectory  $T_i$ ".

**A Rule Language.** The rule language must have several important features.

- (a) It must be able to distinguish between situations that would require solution trajectories differing by more than boundaries of allowable error.
- (b) It must be concise and easy to "compute". In other words there must exist simple, fast procedures for instantiating the "words" or "sentences" of the language from some more primitive.
- (c) The language will be easier to use if the "sentences" are of a uniform structure ( analogous to model-based rules ).
- (d) The language should be such that different "accuracies of description" are possible. There should be some fuzzy, but quantitative nature to the descriptions. This will make it easier to augment the language to provide more accurate descriptions, and thus more accurate control.

Keeping the above considerations in mind, along with knowledge of the behavior of our system gained from experience, we have developed a language with which to implement the control rules. Later we will describe the major components of this language, and will give more details of implementation. Earlier, one of the possible ways was explored to derive the "word classes" directly

from the variables that would appear in an analytical, differential equations formulation of the problem [4]. What remains then, is to determine the range of discrete values that each of these words (variables) may assume. One problem is hidden in the characterization of obstacle positions. Even in the analytical formulation of the problem, there is more than one way to represent obstacle positions. Thus in the "translation" to a linguistical form, we have some flexibility in how we will represent the

obstacle positions. In creating the part of the rule language which describes obstacle positions, we have used language criterion (b) above, as well as knowledge of the kinematics of the type of vehicle we are considering, to sufficiently characterize obstacle configurations. Following are descriptions of the word classes used to form the *PP* rules.

- **Goal Angle** (Goal-ang)

*Goal Angle* describes the direction to the goal relative to the current position of the robot. All measurements such as *Goal Angle*, are done in a polar or rectangular coordinate frame, with the origin being the current robot position. Figure 1 shows the values of *Goal Angle* used. Notice that these values are not distributed uniformly over the 360° range. This is because we found that goal angles within certain adjacent regions resulted in the same control recommendations, regardless of the values of the other precondition variables. Thus, those adjacent regions (angles) could be consolidated for the sake of criterion (b).

- **Goal Range** (goal-range)

*Goal Range* describes the distance to the *subgoal* from the current robot position. Recall that the subgoal is determined by selecting a position in an immediately surrounding passageway that allows achievement of the NAVIGATOR's currently planned subgoal, with minimum cost.

- **Phantom Points** (phantoms)

This can be thought of as a discrete "artificial potential field" *surrounding the robot*. They may also be seen as "virtual proximity sensors". They are used to help classify obstacle configurations in constrained situations. Since the underlying representation of obstacles is a local grid surrounding the robot, this structure is fast and easy to use. For each Phantom Point, we only need to reference the obstacle array to determine if it is "stimulated". Alternatively, the *Phantom Points* could be implemented as *actual* proximity sensors, thus bypassing the need for an underlying representation of obstacle positions. Figure 1 shows the configuration of phantom points we have used.

- **Feasibility of Trajectory** (traj-feasible?)

This "word" is a binary variable which is true if the "region swept" by the trajectory of the right side of the rule is obstacle-free. The "region swept" is determined by the width of the vehicle plus some safety margin. In all cases, this must be true for the given rule to fire.

- **Current State** (curr-state)

Represents the state of the vehicle (system) itself via pertinent state variables, such as steering position, velocity, acceleration. In the *PP* algorithm, this is likely to determine the first few commands of a given trajectory. In some cases, this can be a "dominating variable" in that for a given situation, the trajectories found will be quite different based on the value of curr-state.

With these "words" defined, we can now give the actual form of a *PP* rule.

If [ (goal-ang = a) & (goal-range = r) & (phantoms = p)  
& (traj-feasible? (T<sub>i</sub>)) ] Then (Trajectory <-- T<sub>i</sub>)

**Implementation.** The *PP* algorithm operates according with 6 with some modifications. The first step is to transform the NAVIGATOR's currently planned subgoal to a locally reachable "passageway". This is done using simple criteria of minimum deviation from path, and safe width of passageway. Next the precondition variables are instantiated. Rule matching is facilitated by having the rules organized in a loose tree form. Common parts of preconditions are extracted and are incorporated into a "dummy" parent rule, which determines which subordinate set of rules should be

tested. This organization makes matching fast, but at the expense of more difficult addition of rules. Clearly, an automatic "classification.tree" could be formed from the full set of rules.

Most of the above rule vocabulary components depend on an underlying representation. We found a simple binary grid to work satisfactorily. Obstacle boundaries are projected onto the grid, and thus each cell is marked as either "free-space" or "obstacle". This grid is the basis of the PILOT's "context map". Although the sensors provide less than a 360° view, the PILOT Context Map is 360° so that certain trap situations can be avoided. The mapping of obstacle boundaries onto the grid can be facilitated by using built-in graphics functions for drawing lines, which many systems have.

*Goal Angle* and *Goal range* are computed using integer division and a simple table lookup. A minimum division is defined, and then a table is defined which maps each division to a linguistic variable. An alternative would be to use the integer division only. However, it would then be necessary to have substantial redundancy in the rule base.

*Phantom Points* are easily implemented with the grid representation of obstacles. A "ray" of the phantom point is activated ( or set to true ) if any point along the ray corresponds to a cell tagged as "obstacle" in the underlying grid

*Feasibility of Trajectory* is tested using the phantom points structure. For a given trajectory  $T_i$ , the robot position is "conceptually" moved to each position of the trajectory. For each position, the phantom points are checked against the obstacle-grid. A trajectory is collision-free if there are no phantom points activated during such an iteration.

*Current State* is normalized into a linguistic value by simple integer division of the full range of values that are used to measure the state.

**Path Monitor.** After PP produces a sequence of commands, the execution controller issues "actuation positions". In our simulation system we call this mechanism the Path Monitor. Its functions include, firstly, checking the consistency of the next command with the current position estimate. It will then make small adjustments if necessary. Secondly, the Path Monitor compares the latest local sensor information with the currently planned trajectory. If there is no conflict, the next command is issued. The following are criteria that cause the Path Monitor to exit and send a "replan" message to the PILOT.

- (1) A new plan has been given by the NAVIGATOR.
- (2) The trajectory is blocked by the latest sensor information.
- (3) The currently planned trajectory has been completely executed.
- (4) The current vehicle position is significantly off course from the current trajectory.

**Instantaneous Decision Algorithm (ID ). General Structure and Concepts.** In situations where robot's information about the world is changing rapidly, the trajectory generated by the *PP* algorithm will be abandoned quickly. This can be compensated for, to some extent, by adding a mechanism to "predict" how the world ( robot's map of the world ) may change. (The effect of the **predictive decision-maker** personality will be explored in a separate paper). In general, however, it seems that in such environments it is computationally wasteful to look for an entire trajectory when only one or two commands will be used before replanning.

The objective is to develop a control module which quickly, (and rather superficially) classifies the situation, and gives a single command. Operating at a relatively high frequency, this decision procedure is expected to produce reasonable paths for environments that are rapidly changing. ( Actually it may not be the *environment* which is changing, but the robot's *knowledge* of the environment.)

*ID* algorithm is producing a single command per decision cycle, versus the trajectory of the *PP* algorithm. Since *PP* is based on search in the vehicle's state space, vehicle position information is inherent in each of the commands of the trajectory found. The association between state, command, and new-state-resulting-from-command, is explicit in the generation of successors in the search algorithm, and is thus part of each trajectory in the rule base. On the contrary, in the *ID* algorithm, this link is implicit only. It is built into the rule tables in a more indirect way. This results in the decision-making (DM) procedure operate more like a traditional control system. Accuracy of control can be effected by varying the frequency of DM. A second distinction of *ID* from *PP*, is that the "rule" tables are separated into tables which are dependent (implicitly) on the particular vehicle model being considered (as mentioned above), and tables which are not. The obvious advantage of this is

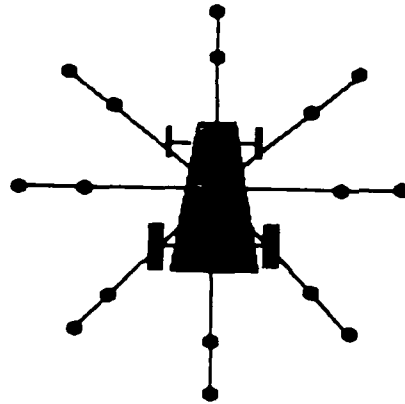


Figure 1. Configuration of the phantom points

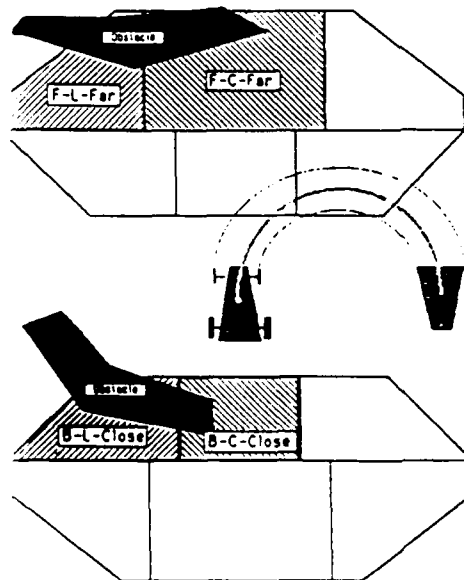


Figure 2. Geometrical representation of the front and rear DAR

adaptability of the algorithm to other mobile robotic systems.

A fundamental premise that the *ID* algorithm rests on, is that for DM in *immediate time-space vicinity* of the present state, we are able to decouple, and consider separately, different components of a situation description, and how they will affect the decision. Consider the following general form of the problem at hand.

Let  $W_i$  = Description of world known to robot,  $C$  = a control to the system, then we are interested in solving:

$$F(C, W_i) = W_k$$

and inverting it to obtain:

$$G(W_i, W_k) = C$$

which in a more linguistic form can be stated as:

**IF (current state is  $W_i$ , AND desired state is  $W_k$ )  
THEN (give control =  $C$ ).**

Now obviously the function  $G$  is likely to be a complex function of the many variables necessary to "describe" the state of the world. ( By state of the world we mean description of the system being controlled, as well as of all aspects of the "external" environment which affect operation of the system e.g. obstacles ). If we consider this multivariable function  $G$  at any single point, however, we can describe that point as a *linear combination* of the variables of  $G$  (in the vicinity of the point) . By distributing, over time, these point values of  $G$ , we can in effect approximate the actual function  $G$ .

The accuracy of this approximation will be dependent on (a) the increment of time at which we are determining a value of  $G$ , (b) the particular topology of  $G$  itself, and (c) the discretization of the variables of  $G$  ( since we are working in a discrete world ). Within definite limits, we have some control over the increment of time ( DM frequency ) at which we approximate  $G$ , as well as over the discretization of the variables, (tradeoff with computational complexity).

An example of exploiting some empirical knowledge of the behavior of  $G$  to increase accuracy can be seen in the variables describing state of the system being controlled. In the *ID* algorithm, the state is described by Steering position, and velocity. It is fairly obvious that regardless of the situation, a command cannot be prescribed without considering position of the steering *and* direction of velocity, *together*. For a given "desired" direction of motion of the robot, the command that should be executed is quite different. Approximating the influence of steering position, and velocity with a linear combination will clearly result in substantial error. In *ID* , all combinations of dominating, interacting variables are explicitly presented.

**Linguistic Variables.** There are several issues concerning linguistic variables which are important for both the *PP* and *ID* algorithms. Earlier, we discussed some criteria for choosing the particular "word classes" or linguistic variables to adequately control the system. Similar methods will be used to first define a set of linguistic variables for the *ID* algorithm. Also the proper discretization, is to be determined of the set of values assigned to each of the linguistic variables.

#### *Choosing the Variables*

Some of the linguistic variables used in *ID* are common to the *PP* algorithm. Goal angle, Goal range, Phantom points, and Current state are all understood in *ID* as they were for *PP*. In addition to these, *ID* uses a construct called "Dynamic Avoidance Regions" ( DARs ). This is simply another

variable to characterize obstacle positions. Figure 2 shows the geometric representation of the front and rear DARs associated with the underlying grid representation of local obstacles.

It is important to remember that part of the "linguistic variables" described above, are composite variables. Goal angle, Goal range, and Current state can be viewed as simple variables, which can take on any one value from a set of discrete or linguistic values. DARs and Phantom points, however, are actually *classes* of variables. Since more than one of the "phantom rays" may be active at once, to consider "phantom-points" to be a single variable, we would have to specify as its value set the entire set of combinations of the component "values". For example "left and left-front" would be a legitimate value for phantom-points. It may be easier though, to consider each of the components of Phantom-points ( and DARs ) to be individual *variables* which can assume only boolean values. This is then a set of variables belonging to the class "Phantom points" ( or DARs ).

We should note now that we have 3 different ways that the configurations of obstacles are being represented. DARs, and Phantom Points are two classes of variables which explicitly, though fuzzily, represent positions of obstacles. Also, by translating the NAVIGATOR's subgoal, to a local subgoal, which is done by both *PP* and *ID*, we are implicitly representing the configuration of obstacles.

#### *Discretization of Variables*

It was stated in [4] note that there are two considerations in determining the discretization of a linguistic variable.

(A) Errors from observation: The information given by a linguistic variable can be at most as precise as its input. Therefore, the range of errors involved in assigning a linguistic value to an observation should be smaller than the support set of the linguistic value.

(B) Goal Angle Position error due to angular resolution will vary with the range, and will be maximum at the maximum range.

(C) Compromise between accuracy of motion and computational complexity. Regardless of the algorithm used, it is reasonable to assume that computational complexity will be positively correlated with size of vocabulary. The more words ( values of variables ) the more rules, and thus the higher the time for rule matching. On the other hand, a finer discretization of the variables is needed to provide accurate motion control. Therefore, we seek a compromise that provides a vocabulary size just large enough to produce the required accuracy of control.

Discretization for some of the linguistic variables used in *ID* is done according to [4]. If we consider error in position to be the scale against which we judge the "accuracy" of our control, we can arrive at minimal discretizations for the variables.

**The *ID* Algorithm.** This subsection will describe the Instantaneous Decision (*ID*) algorithm used to determine control at the lowest level of the IMAS control hierarchy. Previously we discussed the classes of words or variables which *ID* uses to classify a situation in order to prescribe an action. We will describe the algorithm itself.

#### General *ID* algorithm:

1. Normalize Precondition Variables.
2. Rank Desired Direction vectors.
3. Translate ranking of directions into motion command.

Step 1 could also be called "assigning observed values to linguistic variables". We use the word "normalize" since the values assigned to the linguistic variables are actually derived from an intermediate representation, and not directly from "observed" values. For the simple variables such as

*Goal angle* and *Goal range*, normalization is simply an integer division of the currently represented goal position relative to the robot. The table lookup is necessary since the linguistic variables of *Goal angle*, for example, do not represent uniform divisions of the range of the variable.

Values of the *Phantom points* variable are instantiated using the underlying grid representation of local obstacles. Since this grid is continually redefined at each decision cycle, with the robot at the origin, each *Phantom-ray* can have an associated obstacle-grid coordinate set. These coordinates can then be easily referenced to see if a particular *Phantom-ray* is activated. It was shown earlier, that *Phantom points* refers to a class of variables, each taking on a boolean value. Thus normalizing the *Phantom points* requires setting each of these variables to true or false.

The DAR's are handled similarly. Each variable of the DAR class is defined by the set of obstacle-grid coordinates to which it corresponds. Thus to determine whether a particular DAR word is "activated" (contains an obstacle) we need only to check each obstacle-grid coordinate belonging to the set of coordinates defining that word. "Front" and "rear" DARs are treated as two separate classes of words in the current algorithm. Both "front" and "rear" DARs could be handled as a single class however this should not give any computational advantage.

Step 2 of the algorithm alludes to the idea of "Desired Direction" vectors. These are simply a result of a discretization of the space of possible resultant directions (and magnitudes) of motion. Figure 3 shows the particular linguistic directions we have used in this implementation. Notice that this is a uniform discretization of the possible directions, and no knowledge of the kinematic characteristics is taken in account.

The ranking of desired directions, based on the current situation is achieved using a special type of table lookup. Figure 4 shows an example of the table used for this purpose. This each row of this table corresponds to a particular word (or variable value). The numbers in the row effectively impose an ordering of the possible directions of motion as a measure of "compatibility" with each direction. Any situation is described by one word from each class. The rows corresponding to the words describing the current situation are extracted from the table, and are "combined" using some combination operator, in this case addition. For the "special" classes, DARs and Phantom points, the multiple words for a single description are precombined.

The row corresponding to a DAR or Phantom point word can be viewed as an assignment of magnitudes to each direction of the "vector field" defined by the set of desired directions. The set of DAR or Phantom point words instantiated for a particular situation are superimposed to result in a single magnitude for each of the desired directions using a *min* function as follows:

Let  $d_i$  for  $i = 1$  to  $n$  be the set of desired directions represented by

the table.

$W_a$  be the set of words of class  $W$  that are true for the given situation.

$v_k(w)$  = value of word  $w$  for direction  $k$ .

then the ranking of the  $d_i$  due to the word class  $W$  is defined as:

For  $i = 1$  to  $n$

$$d_i = \text{MIN} ( \text{all } v_i(w) \text{ such that } w \text{ is in } W_a )$$

Once a single row is obtained for each word class, a final ranking of the directions is found by column-wise addition of the rows selected for the current situation. This results in a ranking of the

possible directions of motion, which is the input for the next step of the algorithm.

Step 3 incorporates knowledge about the kinematic and dynamic characteristics of the particular vehicle being controlled. The entire ranking of directions is given to this step instead of a single "best" direction, since that best direction may not be directly achievable. Having the entire ranking of directions enables this step to compromise between the direction the robot *should* go, and the direction

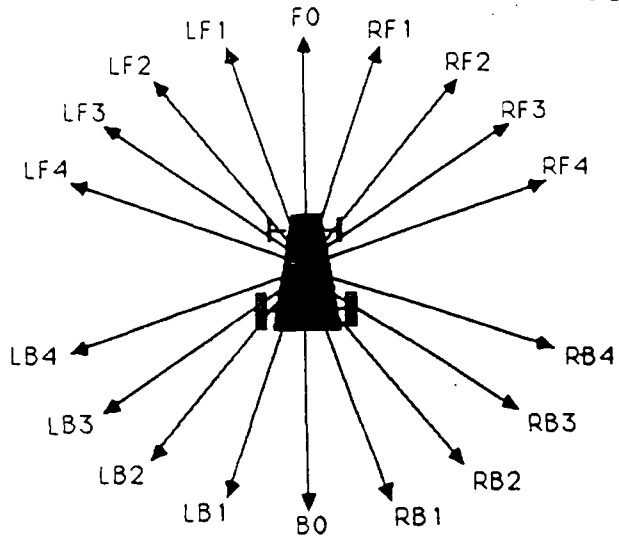


Figure 3. Linguistic directions

Robot	LF1	LF2	LF3	LF4	F0	RF1	RF2	RF3	RF4	B0	RB1	RB2	RB3	RB4	LB1	LB2	LB3	LB4	
00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
09	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4. Example of a table-look-up: input → direction



## " COMMAND TABLE - THRUSTS "

Vel-steer	B0	LB1	LB2	LB3	LB4	LF4	LF3	LF2	LF1	F0	RF1	RF2	RF3	RF4	RB4	RB3	RB2	RB1
F-R1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
F-R2	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
F-R3	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
F-R4	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
F-R5	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
F-0	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
F-L5	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
F-L4	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
F-L3	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
F-L2	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
F-L1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
O-R1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
O-R2	-1	-1	-1	-1	-1	-1	1	1	1	1	1	-1	1	1	-1	-1	-1	-1
O-R3	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
O-R4	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
O-R5	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
O-0	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
O-L5	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
O-L4	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
O-L3	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
O-L2	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
O-L1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
B-R1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
B-R2	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
B-R3	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
B-R4	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
B-R5	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
B-0	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
B-L5	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
B-L4	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
B-L3	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
B-L2	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
B-L1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1

Figure 5. Example of a table-look-up: direction →command

it *can* go.

The transformation of direction into command is done using a simpler form of table lookup. Figure 5 shows the structure of this table. In the simple case, the "best" direction, and the normalized current state are indices into the table that contains the command to achieve that direction from a given state. Since our robot is a steered vehicle, it is biased toward moving forward which creates a number of "ambiguous" cases. Directions that are near  $90^\circ$  from the heading of the robot, for example, can be achieved either by a hard turn and forward motion, or by backing up and then moving forward, in a "three point turn" fashion. For these cases, both commands can be stored in the cell of the matrix, along with the "temporary" direction associated with each of them. Then the command with the highest value of direction ranking will be used.

**Discussion of Simulated Behavior.** Simulated behavior of IMAS is shown in Figure 6 for operation with PP-Pilot, in Figure 7 for operation with ID-Pilot, and in Figure 8 for a Hybrid Pilot (using both PP and ID). One can see that PP-Pilot provides an acceptable trajectory with one serious deficiency: it contains several zones in which no precomputed rule could be found. In this case, IMAS start losing time for a maneuvering eventually bringing it in a point which can be associated with one of the stored rules. Otherwise, even more time should be spent to search for a new rule (as was performed in [6]).

On the contrary, the deficiency of ID-Pilot can be found in its definite lack of long-term optimality with no "wasteful thinking". The compromise is obviously achieved when both of the "personalities" are used within one system. Here the best solution are employed for the cases when these solutions exist. And there is no zones of confusion since the ID-Pilot takes over as soon it becomes clear that that no rule exist for a particular situation.

Full scope of this system opportunities can be determined after the system will be equipped by a mechanism of learning.

### References

1. A. Meystel, "Theoretical Foundations of Planning and Navigation", *International Journal of Intelligent Systems*, vol.2, No.2, 1987
2. A. Meystel, "Planning in a Hierarchical Nested Control System", Proceedings of SPIE, Vol. 727, *Mobile Robots*, ed. by W. Wolfe, Cambridge MA 1986
3. A. Meystel, "Intelligent Control of a Multiactuator System", *IFAC: Information Control Problems in Manufacturing Technology 1982*, ed. by D.E. Hardt, Pergamon Press, Oxford, 1983
4. C. Isik, A. Meystel, "Knowledge-based Pilot for an Intelligent Mobile Autonomous System", Proceedings of the First IEEE Conference on Artificial Intelligence Applications, Denver, CO, 1984.
5. R. Bhatt, D. Gaw, A. Meystel, "A Real Time Guidance System for an Autonomous Vehicle", Proceedings of IEEE Int'l Conference on Robotics and Automation, Vol. 3, Raleigh, NC 1987
6. R. Bhatt, et al, "A Real Time Pilot for an Autonomous Robot", Proceedings of IEEE Int'l. Conference on Intelligent Control, Philadelphia, PA, 1987.

7. A. Meystel, "Planning/Control Architectures for Master Dependent Autonomous Systems with Nonhomogeneous Knowledge Representation", Proceedings of IEEE Intl. Conference on Intelligent Control, Philadelphia, PA, 1987

Figure 6. Simulated behavior of PP-pilot

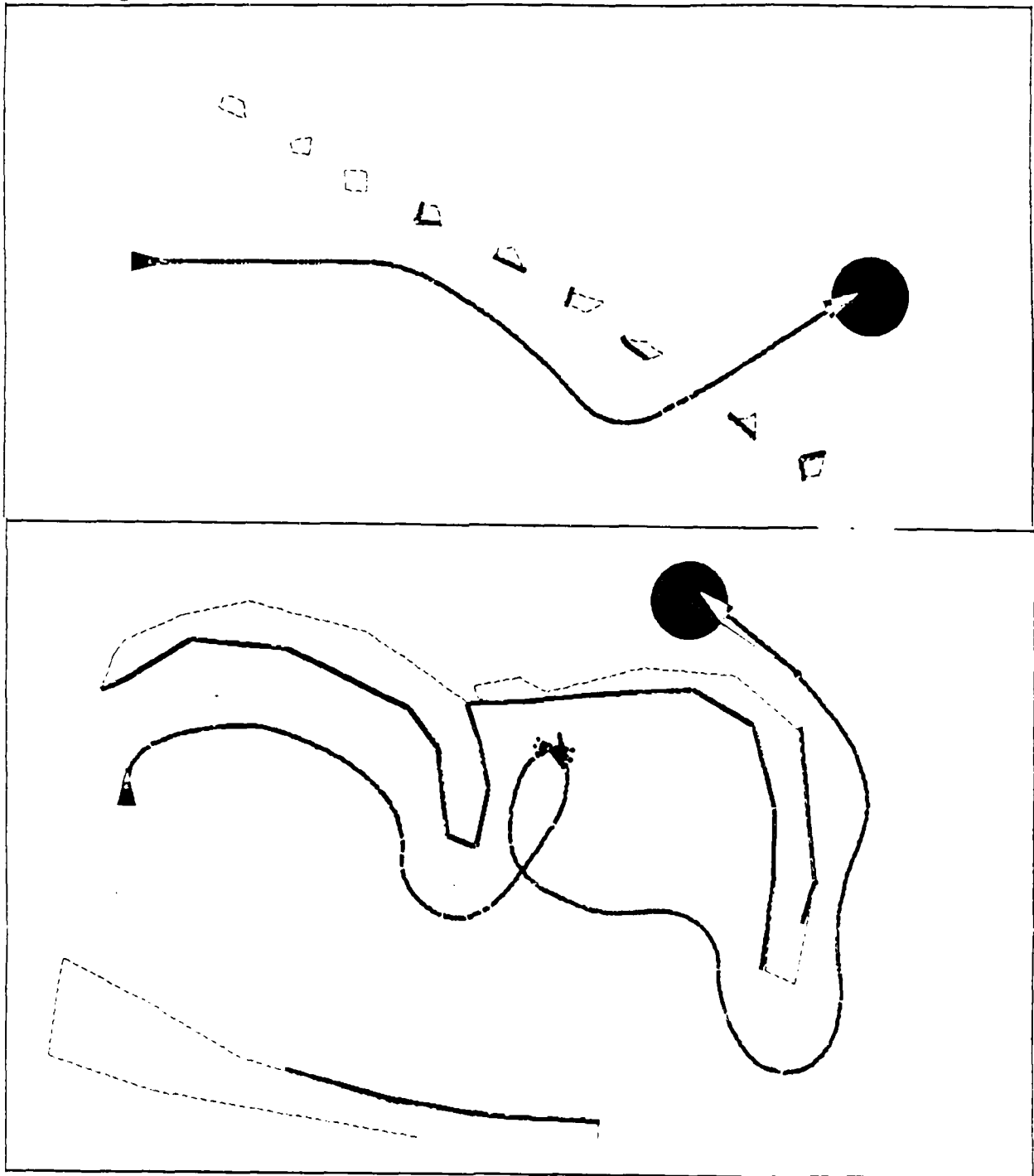


Figure 7. Simulated behavior of ID-pilot

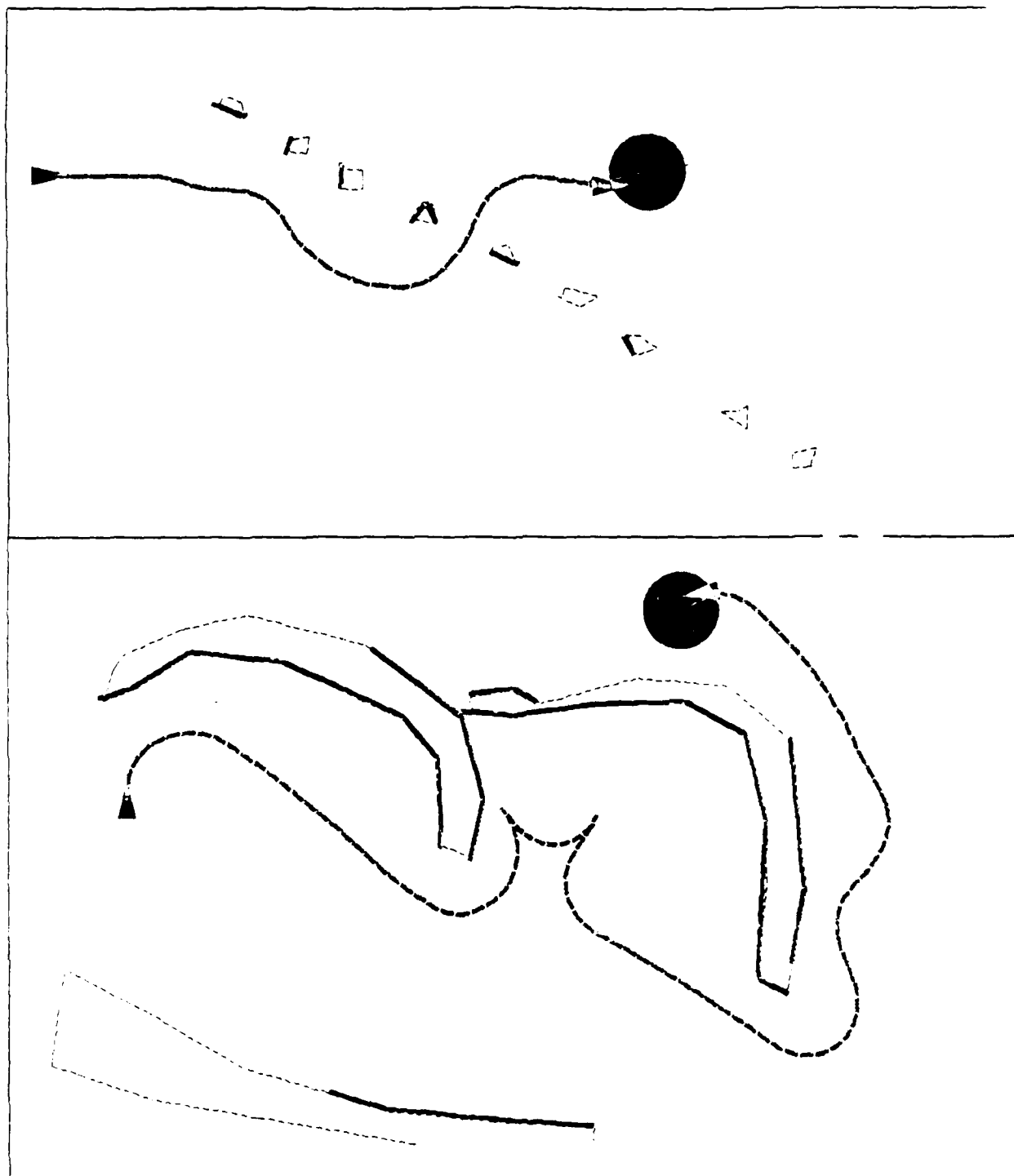
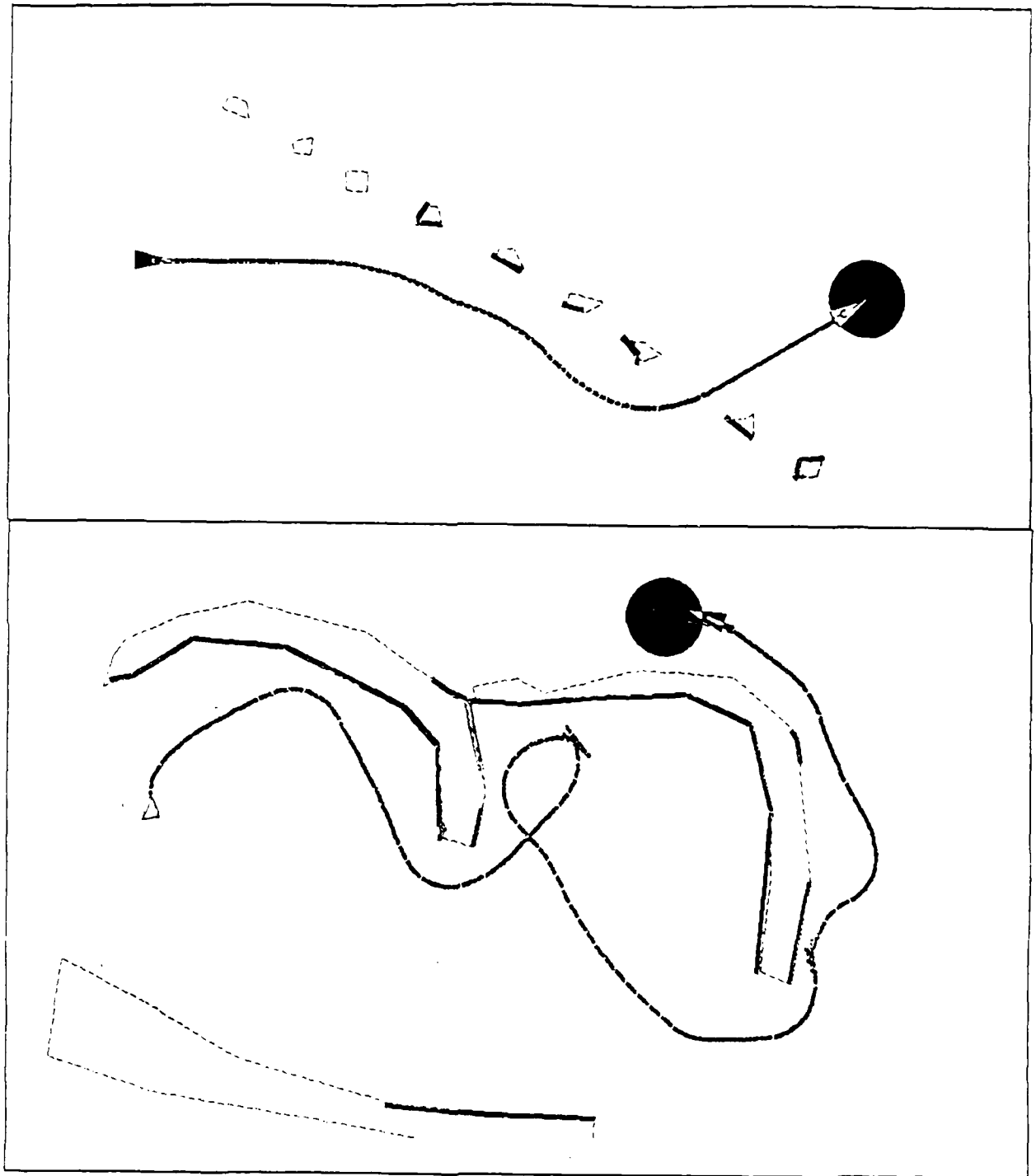


Figure 8. Simulated behavior of (PP+ID)-pilot



## SECTION 5 MINIMUM TIME EXECUTION CONTROLLER FOR IMAS WITH MODEL BASED PRODUCTION SYSTEM

Intelligent Mobile Autonomous System (IMAS) is considered to be a nonlinear stochastic system which is supposed to operate under conditions of constantly changing goal assignment as well as varying components of the mathematical model of the system and constraints. A method is proposed of minimum time control of mechanical motion for an IMAS in which control assignment can be changed during the command execution. Based upon enhanced Bellman's optimality principle for stochastic systems, the string of commands should be recomputed as soon as the new goal of motion is assigned. The problem of computational efficiency is solved by structuring the motion trajectory in such way that new solution is obtained via a simple combining the trajectory from a limited number of standard components. This allows for using a model based production system for synthesizing the control sequences on-line.

**Assumptions Concerned With System Dynamics.** The basic goal of IMAS controller is to find a minimum time command sequence to have the system move a required predetermined distance taking into account the assignment, and the initial conditions to the system. As stated above, the motion of the system should be executed in minimum time and hence the control that would be required would have to keep the system with constraint variables at their maximum limit. Since this is a stochastic system, and since the assigned goal of the motion can be reassigned before the motion is completed, the enhanced Bellman optimality principle is accepted [1] as was suggested in [2]. This principle states for stochastic problems: whatever the present information and the past decisions, the remaining decisions must constitute an optimal policy with regard to the current information set [1].

Execution control system of IMAS is equipped by multiple actuators. Each actuation system within the overall execution controller is considered to be independent of each other and hence can be modelled and controlled individually. This assumption is attributed to the fact that even if actuators are coupled together, one could find a system of representation wherein they could be decoupled or could apply one of many existing methods of decoupling [3].

Each of the independent actuator systems can be modelled by a differential equation of a third order with each of derivatives constrained. These constraints put a maximum and minimum value on the state space variables. Input of the system is the rate of change of force, or time derivative of force (*jerk* input). Since as stated above, the goal is to obtain minimum time control the jerk input is always selected at a maximum level. Hence the input alternatives are maximum positive jerk, maximum negative jerk, or no jerk. After jerk is applied, a change in the current value of force will appear depending upon the direction jerk is applied and the interval of time it was applied. The change in force input causes a direct change in acceleration of the system. This in turn caused a change in velocity and distance moved by the system. Each of these intermediate variables have constraints. As explained earlier jerk input has a maximum value. Next the force and the velocity have also maximum positive and maximum negative values. Finally, the distance traversed is the output variable in the system of execution. Hence for the third order system equations should be written for the state space variables which are devised as follows.

$J(t)$  = Jerk input (rate of change of force).

$F(t)$  = Force input.

Resistance(t) = Force of resistance.

$V(t)$  = Velocity.

$D(t)$  = Distance traversed.

As discussed earlier, minimum time motion control is provided by applying maximum possible input control ("bang-bang" control) and by keeping all variables of the system at their maximum values at all

times. Hence the system of equations for the state-space variables is as follows.

$$F(t) = \int J(t) dt \quad (1)$$

$$V(t) = \int (F(t) + \text{Resistance}(t))/\text{mass} dt \quad (2)$$

$$D(t) = \int V(t) dt \quad (3)$$

Even though the equations (1-3) hold only for linear deterministic motion they can be used for any other type of actuator system. For example, they could be used when we are controlling angular motion, using torque, rate of change of torque, etc.

The system we are dealing with can be represented as a third order dynamic system controlled by "jerks" (first derivative of acceleration) [1,2]. Theoretical diagrams for this type of control are shown in Figure 1. Simulated time trajectories of all of the system controls and state variables if there were no constraints on any of the variables, are shown in Figure 2. Each of the state space variables  $F(t)$  and  $V(t)$  have constraints imposed on them. The constraint will affect the motion as described previously. First of all, the force which is shown to rise between period 0 and  $t_1$  tends to reach a value which is greater than Force-max which is physically not possible. Secondly, the velocity tends to reach a maximum value at  $t_2$  which again can become greater than a permissible limit, and hence the total force applied between 0 and  $t_2$  should to be reduced. Thus, finally we have two constraints imposed on the system equations:

$$\begin{aligned} F(t) &< \text{Force-max} \\ V(t) &< \text{Velocity-max} \end{aligned}$$

We will observe the change in control procedure by application of each of these constraints step wise. First the case where maximum velocity reached by the system will be constrained. This is shown in Figure 3. As it is observed from the graph, the jerk input has to be modified so as to remain within the constraint imposed of velocity. Then, in addition to this constrained system of velocity we will impose the constraint of maximum force both in positive and negative direction. This is shown in Figure 4.

One can see that in order to solve the minimum time control requirements, the value of the time periods and type of jerk input during each period are to be determined. Hence as can be seen from this diagram, all that is required to be changed for different system configurations and various initial conditions, is the timing for application of jerks ("switching times"). The sequence in which these jerks will be applied remains the same. Therefore, each of these periods of jerk application might be classified as different control regions. These control regions can grow or shrink depending on the system initial conditions and variables.

**Assumptions concerned with propulsion force development.** In (1) the input force is considered as a summation of the rate of change of force since the jerk input remains constant over the complete range of growth of the input force.

This assumption might not be valid in a wide range of cases. For example in cases where in braking force is achieved by introduction of excess resistance force (e.g. in propulsion drives with drum brakes) the value of jerk is not constant over the complete range of input force values. So as to compensate for such nonlinearities one can add a component of frictional force which is dependent on current magnitude of force and jerk being applied.





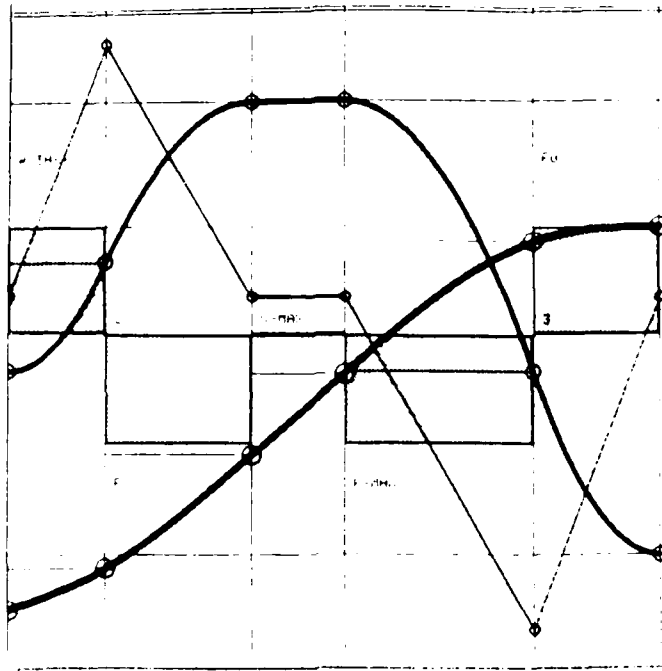


Figure 3. Simulated motion trajectories with velocity constraints

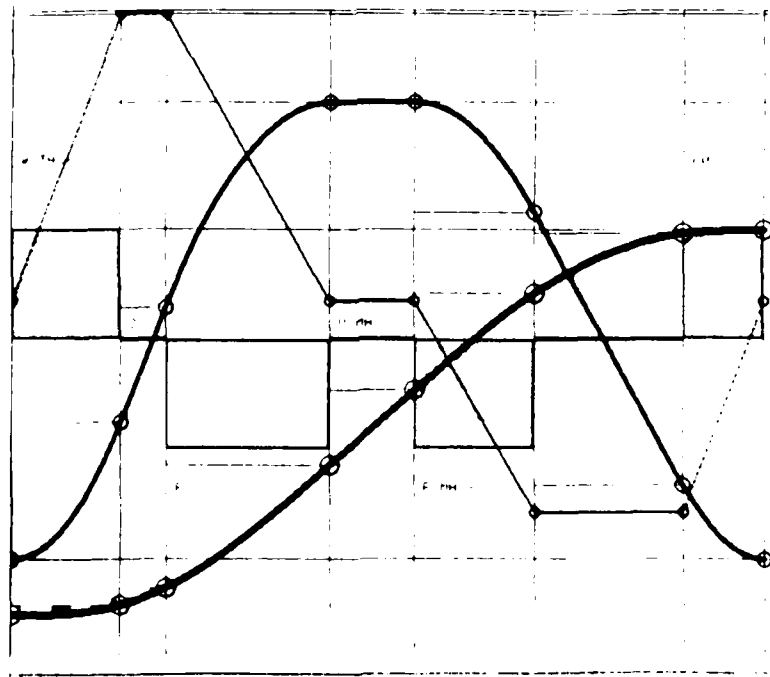


Figure 4. Simulated motion trajectories with velocity and acceleration constraints

One important set of active factors is omitted from the system of equations (1-3), namely, the set of delays which practically exist in most of the actuation systems. This obvious deficiency of the analytical model as well as the set of continuous nonlinearities neglected in the model of motion. Thus, in the beginning we know that our equations contain substantial source of errors. We will assume that all of these sources of errors are included (fictitiously) in the force of resistance.

**Assumptions concerned with force of resistance.** Resistance force can be visualized as a nonlinearity of input force mechanism. This nonlinearity can be attributed to many different facts. The force generator may have its internal nonlinearity and is not able to give out constants output force which has its own stochastic effect for the model of system. Delays in force generation are mentioned above. Three different simple cases of resistance force will be considered here.

1. Constant resistance force (slope).
2. Linear resistance force (velocity).
3. Dry-friction resistance force.

These three different forms of representation of resistance force do not cover the whole variety of existing mechanism of generation of resistance forces, but do in a sense cover the major aspects of immediate practical applications. For example a constant resistance force can be used for modelling motion against a constant gravitational force when the object is going up the slope and hence is storing potential energy. Linear resistance force model is applicable when viscous resistance is encountered and hence the friction force is proportional to velocity. Dry-friction resistance is a combination of the first two types. In dry-friction, the resistance force is constant but when input force is less than resistance force, resistance force becomes equal to input force and hence no motion can be achieved. This means that the resistance for is not an active one as was in Case 1.

These different models of resistance force are made more realistic by adding one more component to the three different cases of resistance force. This is a stochastic component which has different expectations, variances, distribution laws, etc. In this paper, we explore two types of distribution law of the resistance force: uniform and Gaussian distribution at different mean values and different variances.

**Analytical Expressions of the Motion Equations: Determining Standard Components of the Control Cycle.** Each level of the system as mentioned before is constrained and hence the system has for minimum time motion eight different forms of regions within which behavior of the system is to controlled. It is easy to verify that under a broad variety of control conditions the cycles to be performed consist of the same standard components. These eight region are as listed as follows (see Figure 1).

#### STANDARD REGIONS.

- (0) Propulsion force increases from initial value to the value of resistance force.
- (1) Increase force from the level of equality with resistance force to the maximum positive value.
- (2) Propulsion force is kept at constant maximum value.
- (3) Force decreases from maximum positive value to the level of resistance force.
- (4) Force decreases to zero value.
- (5) Force continue to decrease to the maximum negative value (if the resistance force does not act in opposite direction which is valid for the case of positive value of velocity).

- (6) Force is kept constant at negative maximum.
- (7) Force increases from maximum negative value to the level of resistance force.

As one can observe the regions mentioned here define the sequence of switching times (or the time sequence of control commands) for the input actuation so as to obtain a desired response of minimum time motion to the goal state. Each of these regions have a very peculiar property of similarity in the sense of the variables controlled in each time zone. Also it can be seen from the form in which the solution is shown that these regions are very much interdependent as the final goal of the system is to finally get a specific output form of the system.

Another major point to be made here is also about the system of constraints which again impose a certain amount of interrelation between the regions. The actual effect of each region will be considered next. These interrelations will be discussed in the next section. An analytical descriptive production system can be constructed to generate control sequences under all possible initial conditions, and taking in account all possible interrelations between the regions.

As can be observed from the description of the eight regions under consideration, there are only three basic type of input to the system. These basic inputs to the system combined together can give different forms of motion under different initial conditions. The set of inputs together with a tree of alternatives of possible motion models under various initial conditions and applicable constraints constitutes the production system for synthesizing the control sequence. These three different types of inputs are as follows.

- (a) Positive jerk application.
- (b) No jerk application.
- (c) Negative jerk application.

*Group (a). Application of positive jerk.*

Force is increased from initial value to force of resistance by application of positive jerk. During this region the velocity has a growing acceleration value. The distance covered is in turn the integration of the growing velocity with increasing acceleration. The system equations during this period is as follows.

$$\text{jerk-input} = f\text{-dot-max+} \quad (4)$$

$$f\text{-input} = f\text{-dot-max+} * \text{time} \quad (5)$$

$$f\text{-friction} = f\text{-dot-max+} * \text{time} \quad (6)$$

$$\text{velocity} = \int (f\text{-input} - f\text{-friction})/\text{mass} dt = 0 \quad (7)$$

$$\text{distance} = \int \text{velocity} dt \quad (8)$$

*Group (b). No jerk application.*

Force is kept at initial value and held constant during this period. There is no jerk applied during this region. The velocity grows linearly during this period as a result of constant acceleration which is held constant. Also if the initial value of total force applied is zero, the velocity remains constant and hence the distance will grow linearly.

The system of motion equations during this period is presented as follows.

$$\text{jerk-input} = 0 \quad (9)$$

$$\text{f-input} = \text{force-initial} \quad (10)$$

$$\text{f-friction} = \text{const-friction-force} \quad (11)$$

$$\text{velocity} = \int (\text{f-input} - \text{f-friction})/\text{mass} \, dt \quad (12)$$

$$\text{distance} = \int \text{velocity} \, dt \quad (13)$$

*Group (c). Application of negative jerk.*

This is the region where in the total input force is reduced from initial value. This is the same as acceleration reduced from initial value. This causes the system to reduce the rate of growth of velocity. The system of motion equations during this period is presented as follows.

$$\text{jerk-input} = \text{force-dot-max-} \quad (14)$$

$$\text{f-input} = \text{force-dot-max-} * \text{time} \quad (15)$$

$$\text{f-friction} = \text{const-friction-force} \quad (16)$$

$$\text{velocity} = \int (\text{f-input} - \text{f-friction})/\text{mass} \, dt \quad (17)$$

$$\text{distance} = \int \text{velocity} \, dt \quad (18)$$

**Minimum Time Solution of System Equations: Production System For Combining the Control Cycle From Standard Components.** Given the system of requirements for a particular distance to be traversed by the system one has to find the timing for each region. Each region has predetermined type of jerk input. First of all we will list the vocabulary of the system constraints and their relative effects. The system constraints is described as follows.

**Jerk-max+** = Maximum possible positive jerk of system.

**Jerk-max-** = Maximum possible negative jerk of system.

**Force-max+** = Maximum possible force generated by system.

**Force-max-** = Maximum possible braking generated by system.

**Force-resistance** = Resistance force (friction force).

**Velocity-max** = Maximum possible velocity of system.

**Mass** = Mass of the system.

The set of initial conditions is listed below which also changes the behavior of IMAS execution controller. These initial conditions are imposed upon the intermediate variables of the system which have constraints to be taken in account as shown above.

Force-init = Initial value of the force at time zero.

Velocity-init = Initial velocity at time zero.

Distance-required = Distance that is to be moved by system.

These regions appear to be in sequence. Nevertheless, the timing of each one cannot be carried out in a sequence because of the interdependence between the regions. So we will show here the order in which each switching time is calculated and hence will determine the sequence of switching times and the values of the constrained variables during each time period.

A timing region is described by the following variables which define the state of the system at every point during this time period.

Region-left = Pointer to the region left of current one.

Region-right = Pointer to the region right of current one.

Jerk-input = Type of jerk input to be given (Jerk-max+,  
Jerk-max-, 0)

Time = The time at which this region starts operation.

Force = Force value at the start of this region.

Velocity = Velocity value at the start of this region.

Distance = Distance traversed by system at the start of this region.

Delta-time = Time period for which this region remains active.

Delta-force = Change of force over this region.

Delta-velocity = Change of velocity over this region.

Delta-distance = Distance travelled during this region.

It was mentioned above that there are eight different regions. These regions have different type of inputs and constraints. The equation for computing the state variables of each region are shown below.

(0) Increase force from initial value to force of resistance.

Jerk Input = 1

Time = 0

Force = Initial value of force.

Velocity = Initial velocity.

Distance = 0

Delta-time = (Force-resistance + Force)/Jerk-max+

Delta-force = (Force-resistance + Force)

Delta-velocity = 0

$$\text{Delta-distance} = 0$$

- (1) Increase force from initial value to maximum positive.

$$\begin{aligned} \text{Jerk Input} &= 1 \\ \text{Delta-time} &= (\text{Force-max+} - \text{Force-resistance}) / \text{Jerk-max+} \\ \text{Delta-force} &= (\text{Force-max+} - \text{Force-resistance}) \\ \text{Delta-velocity} &= (\text{Jerk-max+} * \text{Delta-time}^2) / (2 * \text{Mass}) \\ \text{Delta-distance} &= (\text{Jerk-max+} * \text{Delta-time}^3) / (6 * \text{Mass}) \end{aligned}$$

- (2) Keep force constant maximum.

$$\begin{aligned} \text{Jerk Input} &= 0 \\ \text{Delta-velocity} &= (\text{Vel-max} - \text{Delta-velocity-reg1} \\ &\quad \text{Delta-velocity-reg3}) \\ \text{Delta-time} &= (\text{Delta-velocity} * \text{Mass}) / (\text{Force-max+} + \\ &\quad \text{Force-resistance}) \\ \text{Delta-force} &= 0 \\ \text{Delta-distance} &= \text{See notes.} \end{aligned}$$

- (3) Decrease force from maximum positive to force of resistance.

$$\begin{aligned} \text{Jerk Input} &= -1 \\ \text{Delta-time} &= (\text{Force-max+} + \text{Force-resistance}) / \text{Jerk-max-} \\ \text{Delta-force} &= (\text{Force-max+} + \text{Force-resistance}) \\ \text{Delta-velocity} &= (\text{Jerk-max-} * \text{Delta-time}^2) / (2 * \text{Mass}) \\ \text{Delta-distance} &= (\text{Jerk-max-} * \text{Delta-time}^3) / (6 * \text{Mass}) \end{aligned}$$

- (4) Keep force constant at force of resistance.  
 (5) Decrease force to maximum negative.  
 (6) Keep maximum negative force constant.  
 (7) Increase force from maximum negative to force of resistance.

This methodology enhances the results of [4] which presume deterministic force of resistance equal at the stages of acceleration and deceleration.

**Simulation of real system** . Once the design of the system is done next step is to simulate and study the effects of sampling and addition of different types of noise to the system. As mentioned before the type of noise we will be adding is resistance force (e.g. friction). This resistance force can have different forms of randomness as described above. In the simulated system we have the state variables of the system which are as follows.

$$\begin{aligned} \text{Time} &= \text{Time at the instance of simulations.} \\ \text{Force}(t) &= \text{Magnitude of force at time } t \\ \text{Velocity}(t) &= \text{Velocity at time } t \\ \text{Distance}(t) &= \text{Distance traversed by system at time } t \\ \text{Resistance}(t) &= \text{Resistance force executed by system at time } t \end{aligned}$$

Depending upon the type of region that the system is in the jerk input is given. This jerk is given for a finite time period and hence change in force is computed using this change in force. Changes in velocity and distance are computed and a new state of the system is achieved. The equations for doing this computations are as follows.

$$\text{Force}(t + \Delta t) \doteq \text{Force}(t) + (\text{Jerk}(t + \Delta t) * \Delta t) \quad (19)$$

$$\begin{aligned} \text{Velocity}(t + \Delta t) = & \text{Velocity}(t) + \\ & ((\text{Force}(t + \Delta t) + \\ & \text{Resistance}(t + \Delta t)) * \Delta t) / \text{Mass} \end{aligned} \quad (20)$$

$$\text{Distance}(t + \Delta t) = \text{Distance}(t) + (\text{Velocity}(t + \Delta t) * \Delta t) \quad (21)$$

Hence using these equations one finds the state of the simulated system at each instance of the system. This simulation assumes that the increase of time  $\Delta t$  is small and hence the assumption that the system behaves linearly during this time. The results of simulation for different combinations of input conditions and parameters of the system, are shown in Figure 5.

**Introducing a Resistance Force into the Model.** As it can be seen from the previous section that resistance force has to be computed for each time period. This resistance force is the type of noise we will be adding to the system. We have two different models for simulating the noise in resistance force in time with (1) Gaussian Distribution, and with (2) Uniform Distribution. The expectation, variance, and the frequency of introducing the new random value can be varied. Simulated cycle of operation is shown in Figure 6. The stochastic component of resistance is simulated in a form shown in Figure 7.1.

These characteristics determine the uncertainty which is typical for the real resistance force, and incorporate a number of other random factors and model deficiencies mentioned above. The resistance force has an average value which is constant, or is being changed with a known deterministic law. The maximum dispersion amplitude is to be varied as a percentage to the average value. The value of the assigned variance of the resistance force is kept constant for a predetermined time period. This time period might remain constant or vary over the complete range of the system simulation. These parameters can be dependent on the type of region the system is in. Hence we have the resistance force varying from (Resistance-force - Amplitude) to (Resistance-force + Amplitude). This range can have different distributions. The simulation has two different ones as shown in Figures 7.2 and 7.3.

As can be observed from the equations for velocity and distance the noise added to the system will cause corresponding component of noise in the velocity and distance parameters. Hence we will have the error of motion cumulate and cause error in the final state of the system. This error can be in Velocity and Distance or both. So using this technique if one has a model of the noise of resistance force an accurate error estimation can be done.

**Conclusions.** As one can see from the simulation results the mechanism of production system can be successfully utilized as a basis for the execution controller structure which serves for the IMAS Pilot described in [5]. Different laws of distribution and other characteristics of the stochastic components lead to the quantitative results easy to identify and take in account.

This suggests a model of the feedback applicable with the production system controller. After the stochastic component is identified, the set of corrections (tabulated prior to system operation) should be introduced, and a new string of the switching times is to be submitted to the execution controller.

### References

1. Y. Bar-Shalom, "Stochastic Dynamic Programming: Cautions and Probing", IEEE Transactions on Automatic Control, Vol. AC-26, No. 5, Oct. 1981
2. A. Meystel, "Nested Hierarchical Controller for Intelligent Mobile Autonomous System", Intelligent Autonomous Systems, Proc. of the Int'l Conference, Amsterdam, Netherlands, 1986

3. A. Meystel, A. Guez, "Elements of the Theory of Design", Proc. of the 4-th Int'l Symposium on Large Engineering Systems, Calgary, Alberta, Canada, June 1982

4. A. Guez, "Minimum Time Control of Multilink Manipulators", PhD Thesis, University of Florida, Gainesville, FL 1983

5. R. Bhatt, et al, "A Real Time Pilot for an Autonomous Robot", Proc. of the IEEE Int'l Symposium on Intelligent Control, Philadelphia, PA 1987

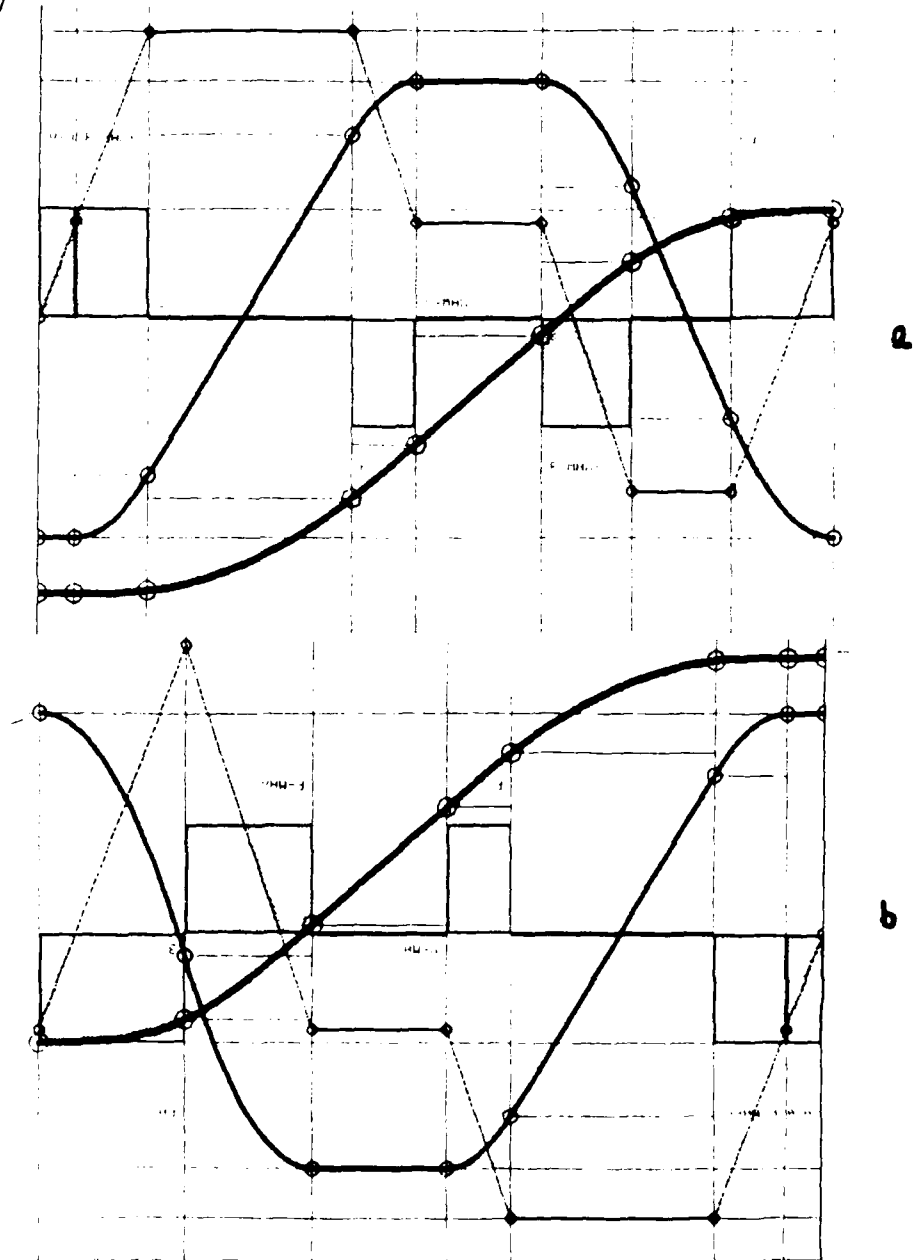


Figure 5. Simulated cycles



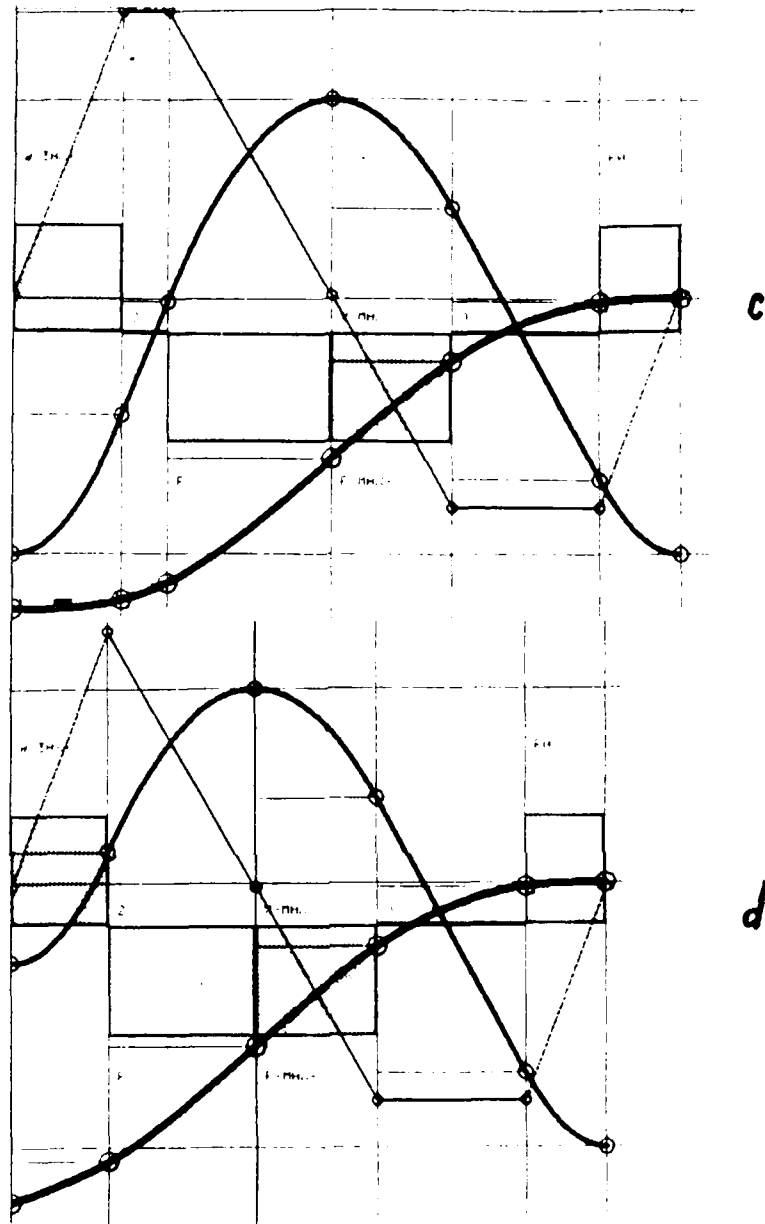


Figure 5. Simulated cycles (cont) (b)

e

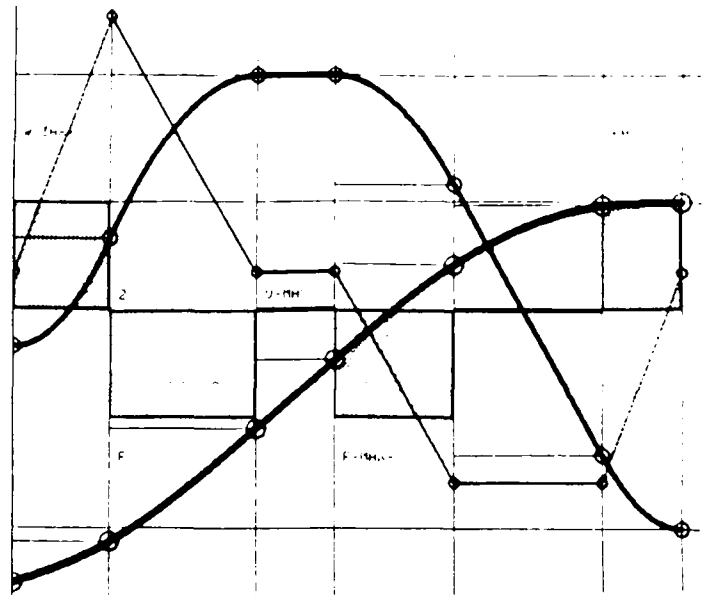


Figure 5. Simulated cycles (cont)

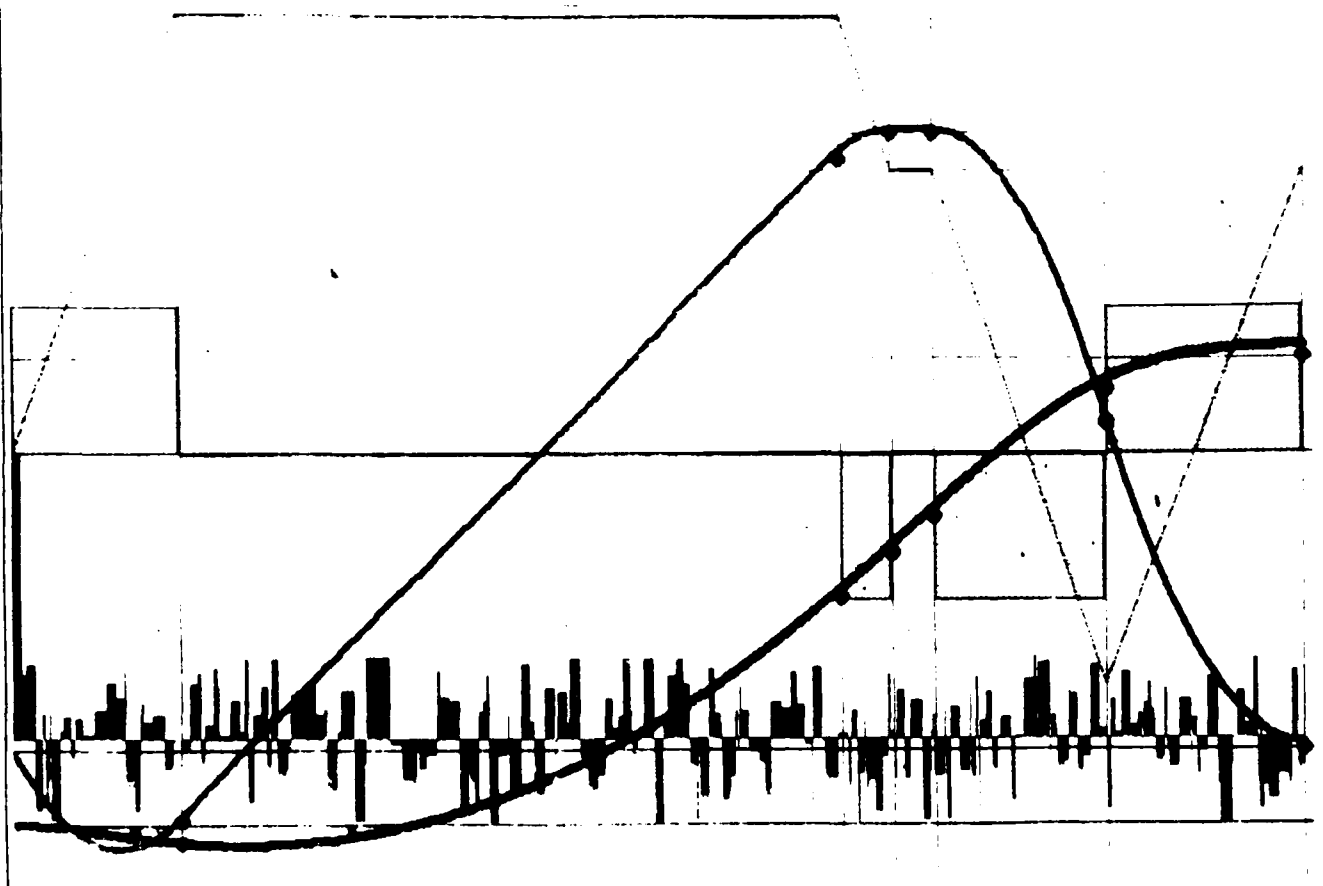


Figure 6. Simulated cycle with stochastic resistance force

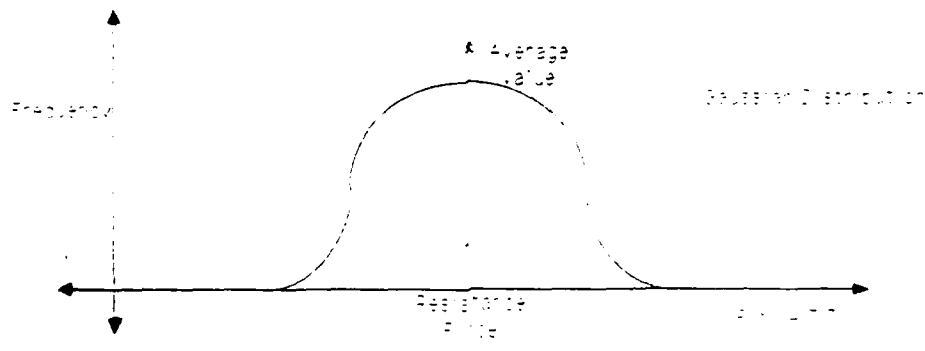
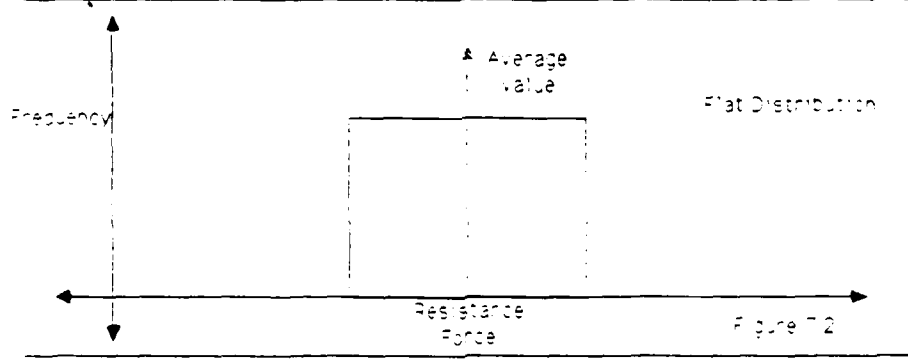
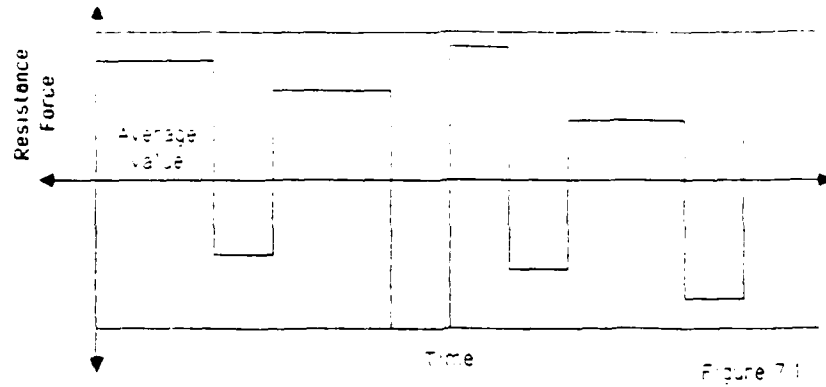


Figure 7. Resistance force characteristics

## PERSPECTIVES

**I. In the area of IMAS planning/control.** The following comments should be taken into consideration during the further research on IMAS.

**1.1 Feature of Supervised Autonomous Planning.** Recent accomplishments of the research and development in the area of autonomous robot control, include some advances in development of the theory of programming for different types of motion: tracking as well as positioning. In both of these cases, the world description is supposed to be given, and the programmer is expected to plan the motion down to the levels for which the analytical routines of the motion, and the chunks of corresponding programs are known, i.e. levels of elementary task recognition). At these levels, a language exists for programming the operation.

Not only this is a time consuming and expensive way of solving the problem of programming the goal-oriented motion, but also this is one of the major obstacles for the growth of flexibility and productivity of multirobot environments (e.g. battlefield, combat engineering situations, and even computer-integrated manufacturing systems) where many of the robots should be applied simultaneously under a limited human supervision. Regretfully, at the present time, minor changes in environment or in specifications, lead to the need in complete replanning, i.e. require constant human involvement in the procedure.

On the other hand, since the world is presumed to be completely known in the existing control solutions, there is a great concern in keeping the world "inventory" unchanged. This is often difficult and sometimes even impossible. Thus, the problem of "automated obstacle avoidance" is formulated which enables the preprogrammed robot to deviate from the preassigned trajectory if the unexpected change of the scene occurs. In most of the real cases, it requires equipping the robot by vision and touch capabilities. However, even in these cases many of existing solutions are based upon inefficient planning-control recommendations.

Our research shows that there is no clear distinction between **planning** and **control** the decision making processing can be made uniform at all levels of resolution. This allows for using AI and Control techniques simultaneously within the recursive decision making structure, and arrive to algorithms, techniques, and theories which are substantially more productive than those presently developed for autonomous robotics by AI and Control Engineering separately. On the other hand, the **nested multiresolutional structure of representation (NMSR)** turned out to be similar in all subsystems of the Autonomous Intelligence. Our prior experience with IMAS shows that NMSR is an appropriate basis for development of original, effectively converging algorithms of classification, pattern recognition, and image interpretation, as well as self-refinement of knowledge structures.

It can be **foreseen** that for a robot with vision and touch, the stage of human-performed planning operations can be substituted by the stage of **supervised autonomous planning**. Certainly, this is just planning in a limited world, with limited capabilities for conjecture and variations of the robot decisions. But it becomes a powerful step ahead in broadening the capabilities of IMAS type systems in unmanned situations. The robot behaves like it has been preprogrammed for a broad variety of situations, or like an **omniprogrammed robot**. The human operator should not worry of reprogramming the system: it does it autonomously.

**1.2 Cognitive Omniprogrammed Robot.** It was shown in the literature, that such level of limited autonomy can be achieved only via joint set of software and architectural developments including implementation of the **nested hierarchical structure** for the joint set of subsystems of **perception, knowledge management, and planning/control**. The principles of **nesting by generalization, and nesting by focusing of attention** should be applied for the world representation, and the branching factor as well as number of layers of the control hierarchy should be intended to minimize the "epsilon-entropy" of the overall system.

These **cognitive omniprogrammed robots (COR)** can be developed on the basis of the recent advances in the theory of autonomous and cognitive control. One cannot expect that the poor cognitive power of the existing computer architectures combined with the poor perceptive capabilities of the existing sensing systems will provide too much of a real autonomy. But it can undoubtedly provide for the capability of reprogramming the robot autonomously, under limited human supervision. In fact, the only thing, that should be given to a **COR** by a human operator, is the **final goal of operation within the ideal world description**. The adjustment of the ideal world description to the reality, as well as the planning and programming of the motion is done by a robot control system with no human involvement.

**1.3 Multiprocessing Omniprogrammed Machine.** The concept of the omniprogrammed robot leads directly to the concept of multiprocessing omniprogrammed machine\*. Clearly the methodology of **omniprogramming**, or autonomous self-programming (based upon nested algorithms of decision-making within a nested hierarchical structure of world representation) should not be oriented only to a set of problems linked with a search of motion trajectory, etc. If the **version space** include coordinates other than coordinates of the physical motion, a multiplicity of other manufacturing problems could be assigned to the system. Assuming the goal given as a final product representation, the version space can be build in such a way as to include capabilities of proper (optimum) selection of the sequence of operation, or individualized design of a synthetic operation, or a process. (For any *state space* a *version space* can be constructed which represents the multiplicity of results of the combinatorial search for an optimum solution. One can understand the *version space* as the multiplicity of *imaginary realities*).

**1.4 Cognitive Guidance-Navigation-Control System (GNC).** The behavior of the simulated intelligent module shows that the system has rudimentary properties of what could be named a **robotic cognition**. In IMAS explored earlier at Drexel University we dealt with only rudimentary vision, and no **conceptual learning** was available. IMAS utilized knowledge, and was not creating any new knowledge. And yet the system was dealing with unknown world, and although the combinations of circumstances IMAS encountered were not prescribed, nevertheless IMAS was able to cope with this. Certainly, this can be considered an initial stage of cognitive behavior, and GNC system with such properties can be named **cognitive GNC**.

These results were obtained at the stage when the overall coverage of the problem by a thorough scientific analysis was far from being complete. At this period of time the study of Knowledge Based Planning/Control System was generally completed, and the structure of Knowledge Representation System was clarified. The subsequent stage of the Drexel effort helped to clarify most of the prior results since the theory, as well as the desirable engineering solutions are concerned with the structure

---

\*This word (machine) should be understood as a combination of computer system, mechanical structure, and a system of sensors (something similar to IMAS).

of control-oriented knowledge bases (which at the present time should be considered as essentially incomplete and unclear).

**II. In the area of Intelligent Perception.** The theoretical and simulation analysis of the nested hierarchical rules-driven multiresolutional NMSR, shows that the nature, the organization, and the protocols of interfaces with the subsystem-user (in this case the hierarchy of Cognitive GNC-system) and the subsystem-source of information (in this case the hierarchy of sensors), determine the algorithm of operation as a recursive multiresolutional stochastic optimum controller.

**2.1 The generalized model of multisensor perception.** Such a model must be developed in coordination with the concept of a representation which could be considered optimal for the accepted model of perception as well as for the model of planning-controlsubsystem. We are convinced that this model should employ a principle of **multiresolutional nested hierarchy**. This will enable us to incorporate the pyramidal structure of advanced vision devices as well as quad-tree type of hierarchical coding system which has proven to be efficient in a number of cases.

**2.2 Knowledge Base for Weakly Supervised Consistent Interpretation.** When the world is as rich as a **real world**, when the behavior is as close to reality as a **moving vehicle** behavior, finally when the feedback is as adequate to the desired IMAS as a **vision feedback**, then the Cartographer knowledge base can be explored and analyzed from the view of its capability to serve with the above mentioned three factors. So, the Cartographer was developed using the **real world operation of a moving vehicle with vision feedback** as a source of information. Certainly, its perception system was rather inadequate, and the vocabulary was poor.

However, now there is no doubts that after enriching the world with the help of more sophisticated system of sensors, the principles of our Cartographer will remain adequate to the planning/control procedure formulated in the beginning of this book. It is important to explore the opportunity of dealing simultaneously with a **combination of sensors of different modality** as a source of enriching the world representation, as well as providing the interpretation system by a variety of built-in means of ensuring the **consistency of interpretation**.

**2.3 Bicameral Representation System.** Thus the **intermediate testbed** testing of which preceded the testing of the IMAS, has finally determined its successful operation in a real operational environment. The software structure for the intermediate testbed reflected the existing means of simulation. A number of enhancements and improvements is planned for the future. One of the major issues which should be explored at the next stage of research, is an issue of using the duplicated intelligent module (IM): parallel intelaced operation of two IMs based upon pictorial and linguistical systems of world representation which will in a sense constitute the right and the left "halves" of the IMAS' "brain".

**2.4 Temporal Perceptual Analysis.** A new property should be given to the subsystem of perception at **all of its levels**: the capability to record and observe series of the "snapshots" of the world, and **and the capability to infer perceptual conclusions from these series**. This mechanism which actually is the subsystem of **world change analysis**, becomes an active part of a planning procedure when two chains of snapshots are considered simultaneously: the sequence of desired ones, and the sequence of actually observed ones. Then the difference between the two corresponding snapshots from these two series would imply the action required to eliminate the dillERENCE.

Then the Wandering Standpoint principle of search in Systems with Limited World Knowledge, can be applied in a broader sense: not just to perceive a physical 3D space, but rather to perceive a **behavioral space** in which the desired and actual operation of IMAS are to be described. Then we

can talk of perceived situation, expected situation, desired situation, apply mission criteria, and legislate the situation based upon a self-awareness of IMAS. Then all of the views: **perceived, expected, desired, and settled**, as well as the mapping among them will be considered in a **temporal sense**.

**2.4 Joint GNC/Representation System.** After the expected structure of the future Cartographer is proven to be operational an effort might be expected to explore a possibility to exercise learning algorithm upon this structure thus transforming the existing planning/control system, or intelligent module (IM) in a first actually operating IMAS with a **cognitive GNC-system**. We expect that IM similar to the module implemented in the prior IMAS, can be a powerful control tool for any type of robot, not necessarily autonomous robot. Instead of autonomy, IMAS intelligent module can be used as a means of self-programming in existing generation of industrial robots.

Numerous controllers for robotic and automation systems for IMAS of next generation will be based on the principles of Intelligent Control: their knowledge representation will be nonhomogeneous (analytical/linguistic). The main technique of storing the required knowledge of the system and its environment, will be tabulating it in a form of production rules. Some of these rules will be derived from the known physical and engineering laws, some of them will be obtained from the experts, some of them will be obtained from experimental data. The two latter will be the main sources of knowledge.

**III. In the area of IMAS knowledge organization for learning.** Problems emerge at the stage of getting knowledge from experts and from unmanned experiments. These two sources of knowledge constitute the main predicament when their use in autonomous robotics program is contemplated since presently neither the experts are familiar with the experiences to be coded in the controller's knowledge base in the future, nor could the experimental data be anticipated to the degree of preparing a format for data recording.

**3.1 Dealing with unexpected knowledge.** The existing experience in autonomous robotics can shed some light on the domain of expected phenomena to be observed and the character of data which can be expected at the input of the sensors. However, we would consider much more important the domain of unexplored situations, and would choose to be prepared to those where unexperienced controller could endanger the mission. Presently we are not prepared for dealing with unexpected knowledge.

**3.2 Generation of rules for unpredicted domain.** IMAS should be prepared for the operation in an unpredicted (and may be unpredictable) domain. The existing intelligent controllers are well prepared to operate based upon the sets of rules using the existing human expertise, and the existing anticipation of information flows to those we are dealing with in the IMAS. In fact, we are not in a position to say so unequivocally; knowledge based intelligent controllers are yet in the phase of basic research, and there is no consistent theory of knowledge based control. Some of the existing successful systems are crafted artfully but are not supported by a proven theoretical basis, and can serve rather as a **confirmation** of the fact that the direction of the research is promising.

**3.3 Analysis of human decision-maker operating in the supervision mode.** In the meantime, substantial information flow can be utilized in the knowledge based controllers which can be extracted from the human operator during his decision making activities. This includes not only obvious variables such as "pulse frequency", "blood pressure", etc. but also a number of more sophisticated variables (and probably, more relevant either) such as EEG, and electrical and magnetic potentials of the field in the immediate vicinity of the human skull. In other words, we would assume that "what they feel can be supplementary to what they say".

**3.4 Development of a calculus of semantic networks.** This sensor information flow together with verbal descriptions provided by a witness and/or participant, can serve as a basis for consistent coding of human knowledge. Most of the human experiences coded in the knowledge based controllers reflect our inability to state a consistent analytical theory or form a model of the system of operation when our experience and insight help us to control this system during the regular man-machine interaction. Either our experiences are difficult to verbalize, or the mathematical would-be model is expected to be unbearably cumbersome; it is hard to say. But the semantic network constructed from a multiplicity of verbalized human experiences is a good source of knowledge for the intelligent controller.

**3.5 Dealing with knowledge not associated with human experiences.** This knowledge should be instrumental for solving a number of potentially important problems. Let us investigate the domain of knowledge which we do not expect to correlate with human experiences. The planning/control subsystem would not be able to incorporate any knowledge of a system if this knowledge were not digested initially by a human expert within his personal particular representation language presumed to be adequate for communicating his human expertise, or within his personal nonverbalized response mechanism. At least, it can be considered a good source of knowledge for the development of the conventional mathematical models, and it can run the system efficiently. For example, we can believe that a human expert can suggest a good enough rule base for moving a multilink manipulator arm here on the surface of Earth; we know that humans are using their "multilink manipulators" long enough to feel what is the best way of doing this. (They may not be able to communicate this knowledge clearly which is a problem of knowledge engineering and intelligent control).

We would not be as optimistic given a situation in which a human expert has to communicate some knowledge which is inconsistent with his human experience and hence is not represented in his human intuitions. So we would not believe in a specific human intuition of performing any mechanical motion in the multiplicity of situations with no gravity. Human perceptual clues are becoming different, our motor-sensor loops should work differently under these circumstances. Another example, is intuition of driving a vehicle, landing and docking it with and without gravity. One should not expect human intuitions to be the best beacon to follow in the aforementioned cases. One can expect a domain of "robot intuitions" to be the legitimate domain of scientific interest. Trained to perform efficiently by the rules suggested by the human experts on Earth, the autonomous, (or semiautonomous) robot may find some of his new experiences contradicting the suggested rule-base, or requiring supplementary rules to be properly interpreted and utilized.

**3.6 Autonomous robot concept generation.** This is not the same as just a computer concept generation since IMAS is equipped with a system of sensors and can arrange for and exercise a system of experiments. In order to be capable of doing this, robot should have an ability to form concepts. We propose a learning mechanism for a robot based upon a special algorithm of generation of stable information units based upon converging multiresolutional decision making algorithm (conceptron). Conceptron differs from the well known perceptron procedure in a number of features: it does not require a teacher for learning, the mechanism of information unit (concept) generation is based upon a definite dynamic procedure, it requires simultaneous operation of the same information represented at several different resolutions, etc.

**3.7 Development of a decision-making/learning computer architecture.** This alternative of solving the problem of knowledge engineering in an unpredictable domain, is using the voluminous results in learning systems of last two decades. As we know, all of them can be characterized by one substantial shortcoming: the structure of the concept to be learned is known in advance whereas in our



case learning should be done with no conceptual system known, and no "teacher" is supposed to be involved in the **problem** of learning. We believe that such a mechanism does exist in complex animals, and that this **mechanism** can be simulated and practically reproduced using definite ideas of **decision-making processes** applicable to the advanced computer architectures.

These novel ideas of decision making processes ascend to the interactive information transformation within the three joint multiresolutional computer structures: perception (sensor-fusion), knowledge-base, and planning/control. It has been shown that new algorithms of recursive self-organized processes on the hypergraphs of representation (no matter pictorial or linguistic ones) do converge, and provide reasonable stability even with small amount of information. A new architecture (**conceptron**) is being developed at Drexel University which combines the features mentioned above.

IV. Theoretical basis for advancements under consideration. This cluster of works has a major objective: to stimulate development of autonomous robots for IMAS exploration, in particular, for IMAS operation in various types of environment independently of human interference, or partially teleoperated. LAMIR at Drexel University has already IMAS experience. Several results will be utilized in the further research as a direct asset based upon our prior activities:

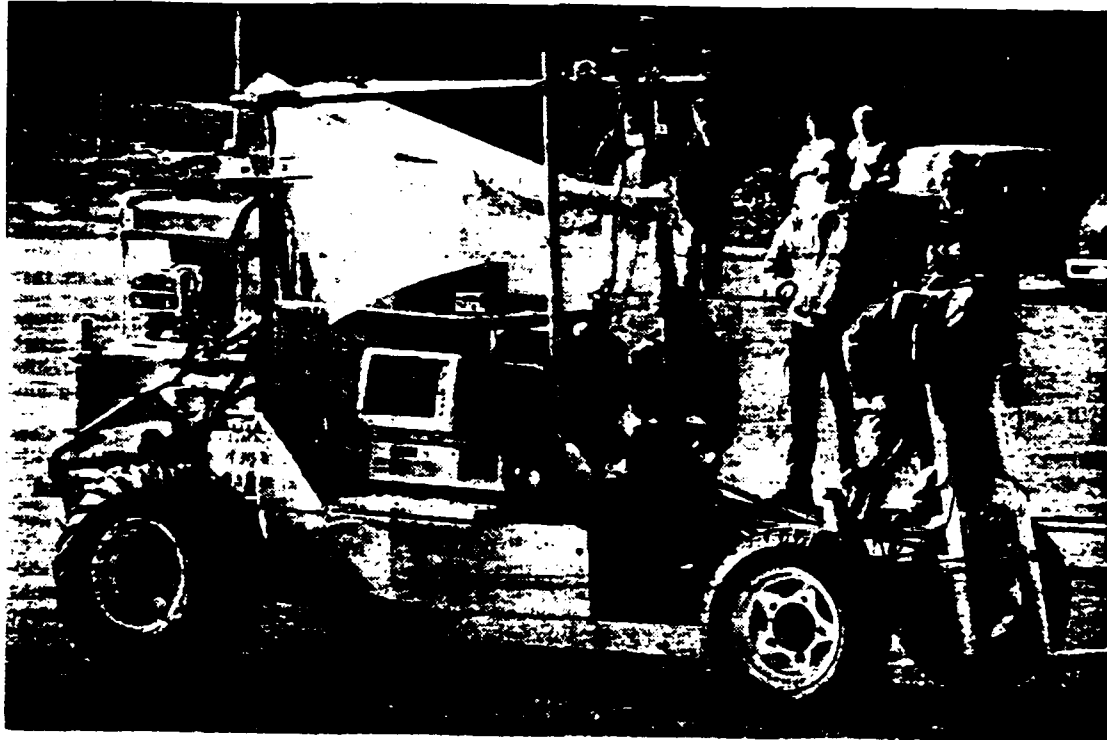
- Planning and control processes are considered to be parts of a joint recursive decision making process (**planning/control**) which starts at a low resolution, broad scope of attention, rough discretization, and linguistical representation (as planning), and ends at a high resolution, narrow scope of attention, fine discretization, and analytical representation (as control). This recursion presumes a number of intermediate stages of planning/control processing with intermediate values of resolution, attention, discretization, and nonhomogeneous representation.

- The Intelligent Guidance/Navigation/Control (GNC) system of the IMAS of next generation will have a **hierarchical architecture of planning/control loops** with information flows connecting horizontally its vertical subsystems: the perceptual system, the system of knowledge organization, and the decision making system; representation in the horizontal loop depend on the resolution level.

- All of the above mentioned subsystems are designed as **resolutional nested hierarchies** with joint pictorial-linguistical representation and with parameters minimizing the probabilistic entropy of the knowledge flow, as well as the  $\mathcal{E}$ -entropy of the nested representation structure.

- Fuzzy linguistical controllers as a part of the GNC-system turned out to be the best choice at most of the hierarchical levels since they allow for using not well structured human expertise which is decisive for autonomous systems operation.

- **Intermediate testbed (IMAS-2)**. It is essential to create a theory and practical computer structures of **knowledge bases** which might be utilized within autonomous robots not as a source of premeditated alternative decisions but as a functional part of the overall hierarchical system of intelligent control guiding the system of autonomous supervised robot. This is why in our prior research IMAS was simulated as a physical model rather than a decision-making agent. This is why an intermediate testbed was created (IMAS-2) which enabled us to observe the first results of IMAS actual activities, and simultaneously give us an opportunity to fill in the designed structure of Cartographer by actual knowledge of the world (see Figure A-1).



The Philadelphia Inquirer / MICHAEL MALLY

Figure A-1. IMAS-2 during the outdoor testing

Methodologies of design and manufacturing can be also enhanced by using our experience of developing IMAS as a basis for a broad diversity of possible applications. Undoubtedly, for all spectrum of future intelligent robots IMAS experience will be instrumental, and IMAS solutions can be used as patterns. In its further research, LAMIR in particular, will incorporate many of engineering principles first explored within IMAS at Drexel University.

---

*Technical Report on Intelligent Mobile Autonomous System*