A Petri-Net Coordination Model for an Intelligent Mobile Robot

Fei Yue Wang, *Member, IEEE*, Konstantinos J. Kyriakopoulos, *Member, IEEE*, Athanasios Tsolkas, *Member, IEEE*, and George N. Saridis, *Fellow, IEEE*

Abstract— **A Petri net model of the coordination level of an intelligent mobile robot system (IMRS) is presented. The purpose of this model is to specify the integration of the individual efforts on path planning, supervisory motion control, and vision system that are necessary for the autonomous operation of the mobile robot in a structured dynamic environment. This is achieved by analytically modeling the various units of the system as Petri net transducers and, explicitly representing the task precedence and information dependence among them. The model can also be used to simulate the task processing and evaluate the efficiency of operations and the responsibility of decisions in the coordination level of the intelligent mobile robot system. Some simulations results of the task processing and learning are presented in the paper.**

I. Introduction

DURING THE PAST DECADE, considerable research effort has been focused on various problems related to mobile robots, such as control [5], path planning [9], [20], navigation [8], [26], obstacle avoidance [2], vision [16] and architectures [3], [17], [25]. However, little effort has been reported on the integration of the above areas and especially on the formal specification of the entire architecture of the mobile robot systems. As a result, the task priorities and information dependence among the principal parts of a system are not clear and therefore the problems of synchronization and delay analysis among them cannot be well addressed.

A two robotic arm platform with primary purpose assembly operations is currently under development in the NASA-Center of Intelligent Robotic Systems for Space Exploration (CIRSSE) at Rensselaer Polytechnic Institute. This test bed is going to be assisted by mobile robots, fetching assembly parts and/or providing additional visual information at the assembly regions using cameras that are mounted on them. Such an environment will contain a number of moving entities (arms, mobile robots, parts, etc.). Therefore the need for an autonomous intelligent mobile robot system (IMRS) that can efficiently and safely perform the assisting tasks is eminent. The IMRS should be therefore provided with a sensing system and a local decision making unit that will enable it to plan its motion and avoid obstacles.

Manuscript received January 27, 1990; revised June 18, 1990, February 8, 1991, and March 23, 1991. This work has been supported in part by the NASA Center for Intelligent Robotic Systems for Space Exploration (CIRSSE), and in part by a fellowship of Alexander Onasis Foundation.

The authors are with the NASA Center for Intelligent Robotic Systems for Space Exploration Rensselaer Polytechnic Institute Troy, NY 12180-3590.

IEEE Log Number 9100404.

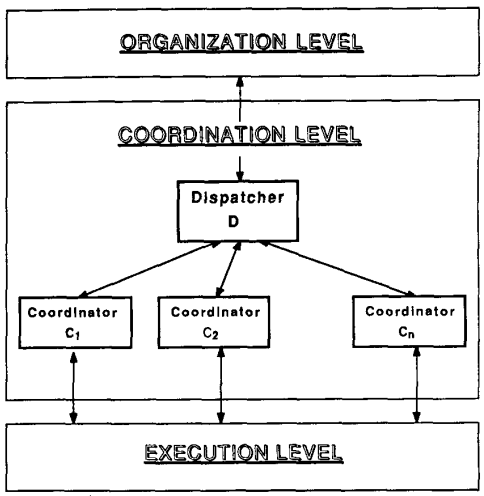


Fig. 1. The structure of intelligent machines.

Such capabilities require the integration of the sensing activities with the decision making processes related to path planning, supervisory motion control, navigation, etc. The framework of IMRS, based on the theory of *hierarchical intelligent control* [21], [23], [24], consists of the three major levels, the organization, coordination and execution shown in Fig. 1.

The *organization level* (the task organizer) generates higher level motion tasks for IMRS to accomplish the required task.

The *coordination level* serves as an interface between the organization and execution level. This level consists of one dispatcher and several coordinators. The dispatcher receives the task plans from the organizer, decomposes the tasks of a plan into coordinator-oriented control actions with qualitative requirements and then dispatches them to the corresponding coordinators. The coordinators translate the control commands further into the operation instructions and load them down to the appropriate execution devices in the execution level for real-time execution.

The *execution level* executes the instructions coming from the coordination level and reports the results to it.

In CIRSSE, several projects have been conducted on the individual areas related to mobile robots, and that are integrated in this work.

1) The *Extended VGraph Algorithm* [4] has been suggested for path planning.

2) The *Hough Transform* has been implemented [27] for visual detection and kinematic estimation of moving objects,
3) The *Supervisory Control Strategy for Navigation of Mobile Robots in Dynamic Environments* [12] has been developed to integrate path planning with vision and tracking control, and
4) The *Coordination Theory for Intelligent Machines* [30] has been established to integrate the above individual efforts efficiently to form the final IMRS.

To formalize the integration of the IMRS, one has first to select a proper model as the basic module. Finite states machines (FSM's) have already been used by Brooks [3] to describe the basic modules of the mobile robot system that he proposed. However, since the connection of several FSM's is no longer a FSM, the communication specification between modules can not be achieved using the FSM framework. To accommodate this, Brooks [3] has recently endowed his FSM's with registers and timers. Since Petri nets were originally introduced to describe the communications of finite state machines, we believe that Petri nets are more suitable to specify mobile robot systems. The primary purpose of this paper is to develop a Petri net model for coordination of IMRS based on the coordination theory of intelligent machines [31].

The coordination level of IMRS is composed of one dispatcher and three coordinators: a path planning coordinator, an obstacle avoidance and tracking control coordinator, and a vision system coordinator. The dispatcher and the coordinators have been modeled as Petri net transducers and the coordination structure constructed by those transducers has been used to represent the task dependence in the coordination level. The task processing and learning in the coordination level of IMRS has been simulated using this model. The main contribution of this work is to show that,

1) Petri nets can be used as basic modules of a mobile robot system.
2) The communication or connection of modules can be efficiently specified within the Petri net framework.
3) A control and communication mechanism for task coordination of a mobile robot system can be established based on a Petri net model.

Section II introduces the Petri net model for coordination of intelligent machines and some coordination process properties. A layout of the architecture of the coordination level of IMRS and a description of its units, along with their Petri net transducer models is presented in Section III. The system integration and task execution procedure based the Petri net model are also specified in this section. The task simulation is performed and the results are presented in Sections IV and V. Finally, Section VI concludes the paper.

II. Coordination Theory of Intelligent Machines

The coordination level of intelligent machines is composed of one dispatcher and a number of coordinators (Fig. 1). The dispatcher receives the task plans (i.e., the sequences of tasks) from the Organizer, translates the plans into coordinator-oriented control actions and then dispatches them to the corresponding coordinators. A coordinator, just after it gets the control commands from the dispatcher, translates them into the operation instructions and sends them down to the appropriate execution devices of the execution level for real-time execution. The process of task translation is continued until the job issued by the Organizer is completed. A coordination theory for such operation has been developed [30] based on a formal model of the coordination level of Intelligent Machines that is called *coordination structure.*

A. The Coordination Structure (CS)

The basic construction module for coordination structures is Petri net (PN). A Petri net $N = (P, T, I, O)$ consists of a finite set of places P, a finite set of transitions T, an input function I, and an output function O. The set of places describes the states of the system, and the set of transitions defines events that can change the states of the system. The input function specifies the preconditions for each event to occur and the output function gives the effects of the occurrence of each event. A place may contain a nonnegative integer number of tokens. The state of a Petri net is represented by its markings, i.e., the distribution of tokens among its places. Petri net has been proved to be an excellent tool for system modeling, especially when concurrency and conflict are involved [18].

However, the ordinary Petri net model is incapable of describing language translation. Therefore, in order to use Petri net to implement the linguistic decision schemata [22] for describing the task translation processes in the dispatcher and the coordinators, the Petri net transducer (PNT) is introduced [28]. A PNT= $(N, \Sigma, \Delta, \sigma, \mu, F)$ is a language translator that translates a given input task plan into an output task plan. The Petri net $N = (P, T, I, O)$ of PNT is the controller of translation. μ is the initial marking of N, i.e., the initial state of PNT. Σ is the input alphabet represented input tasks, and Δ is the output alphabet represented the output tasks. σ, the *translation mapping* from $T \times (\Sigma \cup \{\lambda\})$ to finite sets of Δ^* (where λ is the empty string and Δ^* is the set of all finite length strings over Δ), specifies for a given input task the processing transitions in N as well as the output subtask plans that may be used for that task. Finally, F is a set of final markings indicating the termination of the task translation. Two PNT's can be combined together to perform the task translation according to the rule of synchronous composition.

A coordination structure (CS) is then constructed by integrating the Petri net transducer models of the dispatcher and coordinators of the coordination level using a set of connection points, formally,

$$CS = (D, C, F, R_D, S_D, R_C, S_C)$$

where

1) the dispatcher $D = (N_d, \Sigma_o, \Delta_o, \sigma_d, \mu_d, F_d)$ is a PNT with a Petri net $N_d = (P_d, T_d, I_d, O_d)$;
2) the coordinators $C = \{C_1, C_2, \ldots, C_n\}$, $n \geq 1$. Each coordinator C_i is a PNT $C_i = (N_c^i, \Sigma_c^i, \Delta_c^i, \sigma_c^i, \mu_c^i, F_c^i)$ with Petri net $N_c^i = (P_c^i, T_c^i, I_c^i, O_c^i)$;
3) the connection points $F = \bigcup_{i=1}^{n} \{f_I^i, f_{SI}^i, f_O^i, f_{SO}^i\}$. f_I^i, f_{SI}^i, f_O^i and f_{SO}^i are the *input point, input semaphore, output point,* and *output semaphores* of C_i, respectively.

4) the dispatcher *receiving mapping* R_D and *sending mapping* S_D: mappings from T_d to subsets of F that satisfy the following connection constraints:
   a) $(t, f_I^i) \in S_D \Leftrightarrow (t, f_{SI}^i) \in R_D, (t, f_O^i) \in R_D \Leftrightarrow (t, f_{SO}^i) \in S_D$;
   b) $(t, f_I^i) \notin R_D, (t, f_{SO}^i) \notin R_D, (t, f_O^i) \notin S_D, (t, f_{SI}^i) \notin S_D$;
   c) for any firing sequence s of transitions in N_d,
   $$0 \leq \#\{t \mid t \text{ in } s, (t, f_I^i) \in S_D\} - \#\{t' \mid t' \text{ in } s, (t', f_O^i) \in R_D\} \leq n_i + 1$$
   where $n_i \geq 1$;
   d) for any f_I^i and f_O^i, there exist t and $t' \in T_d$ such that $(t, f_I^i) \in S_D$ and $(t', f_O^i) \in R_D$.
5) The coordinator receiving mapping R_C and sending mapping S_C: mappings from $T_c = \bigcup_{i=1}^n T_c^i$ to subsets of F that satisfy the following connection constraints:
   a) $(t, f_O^i) \in S_C \Leftrightarrow (t, f_{SO}^i) \in R_C \quad \& \quad (t, f_{SI}^i) \in S_C$;
   b) $(t, f_O^i) \notin R_C, (t, f_{SI}^i) \notin R_C, (t, f_I^i) \notin S_C, (t, f_{SO}^i) \notin S_C$.

The receiving mappings R_D and R_C specify the way in which the dispatcher and coordinators receive information from the connection points F. The sending mappings S_D and S_C specify the way in which the dispatcher and coordinators send information to F. Therefore, they define the configuration of the connection between the dispatcher and the coordinators of the coordination structure. The connection constraints guarantee that each coordinator C_i is bidirectionally connected with the dispatcher D. D can issue tasks to C_i only when C_i is available and C_i can report the execution result to D only when the communication facility is ready. The number n_i represents the task buffer capacity of C_i.

Various connection patterns can be designed by using different receiving and sending mappings. A simple connection configuration has been selected for the coordination level of IMRS by imposing the following additional conditions: 1) C_i only accesses its own connection points; 2) there is only one initially enabled transition, t_s^i, in C_i with $(t_s^i, f_I^i) \in R_C$; 3) there is only one transition, t_f^i, in C_i with $(t_f^i, f_O^i) \in S_C$; 4) only t_f^i has its output place as the input place of t_s^i. A CS with these properties is called a *simple coordination structure*.

A coordination structure CS is operated by applying the standard execution rule of Petri nets to a Petri net derived from CS. This Petri net, called the Petri net underlying CS, specifies the precedence relationships of the task activities in the dispatcher and coordinators and therefore defines the information structure of the entire coordination level. The formal specification of the underlying Petri net is the following:

$$N = (P, T, I, O) \text{ with } P = P_d \bigcup P_c \bigcup F, \quad T = T_d \bigcup T_c$$

$$I(t) = \begin{cases} I_d(t) \bigcup \{f \mid (t, f) \in R_D\} & \text{if } t \in T_d \\ I_c^i(t) \bigcup \{f \mid (t, f) \in R_C\} & \text{if } t \in T_c^i \end{cases}$$

$$O(t) = \begin{cases} O_d(t) \bigcup \{f \mid (t, f) \in S_D\} & \text{if } t \in T_d \\ O_c^i(t) \bigcup \{f \mid (t, f) \in S_C\} & \text{if } t \in T_c^i \end{cases}$$

and the initial marking of N is

$$\mu(p) = \begin{cases} \mu_d(p) \text{ or } \mu_c^i(p) & \text{for } p \in P_d \text{ or } P \in P_c^i \\ n_i & \text{for } p = f_{SI}^i \text{ or } f_{SO}^i \\ 0 & \text{otherwise.} \end{cases}$$

The underlying Petri net N also provides a way to use concepts and analysis methods of Petri net theory to study the process properties of the coordination level, such as liveness, boundedness, reversibility, consistency, repetitiveness, etc. The following two theorems give the results about the boundedness and liveness of the simple coordination structures.

Theorem 1: The Petri net N, underlying the CS, is bounded if all Petri nets N_d, $N_c^i, i = 1, \ldots, n$ are bounded.

Proof: Reminding that $R(N, \mu)$ represents the reachability set of N and from the definition of the receiving and sending mappings we obtain that $\forall m \in R(N, \mu) \Longrightarrow m(p) \leq n_i$ if $p \in F$. Since Petri nets N_d, N_c^i, $i = 1, \ldots, n$ are closed subnets of N, it follows immediately that the restriction of $R(N, \mu)$ on P_d is a subset of $R(N_d, \mu_d)$ and the restriction of $R(N, \mu)$ on P_c^i is a subset of $R(N_c^i, \mu_c^i)$. Therefore, the boundedness of N_d, $N_c^i, i = 1, \ldots, n$ guarantees the boundedness of the underlying Petri net N. It is also easy to show that when $n_i = 1, i = 1, \ldots, n$, the safeness of N_d, $N_c^i, i = 1, \ldots, n$ will guarantee the safeness of N. Note that the boundedness of N guarantees the structural stability of the coordination level. Q.E.D.

Theorem 2: The Petri net N, underlying the CS, is live if all Petri nets N_d, $N_c^i, i = 1, \ldots, n$ are live.

Proof: By connection constraint (4d), for each C_i there exist transitions in N_d that take f_I^i and f_O^i as their input and output places in F, respectively. From the definition of simple coordination structures and constraint (Section V-A), t_f^i takes both f_O^i and f_{SI}^i as its output places. Since only t_f^i has its output place being the input place of t_s^i, it is guaranteed that if a number of tokens is displaced into f_I^i, the same number of tokens will appear in f_O^i when N_c^i is live. Therefore, in order to show N is live, we only need to show that N_d as a subnet of N is a live Petri net.

Let $m \in R(N, \mu)$ be an arbitrary marking; $R(N, m, k)$ be the set of markings reached from m by firing at most k transitions in T_d. Let m_d and $R_d(N, m, k)$ be the restrictions of m and $R(N, m, k)$ on P_d, respectively; and $R(N_d, m_d, k)$ be the set of markings reached from m_d by firing at most k transitions in T_d when N_d is considered as an independent Petri net. Let $T(k)$ be the set of transitions in T_d that are enabled under $R(N, m, k)$ and $T'(k)$ be the set of transitions in T_d that are enabled under $R(N_d, m_d, k)$ when N_d is considered to be independent. We first prove that $R_d(N, m, k) = R(N_d, m_d, k)$ and $T(k) = T'(k)$ for any $k \geq 0$.

When $k = 0$, $R_d(N, m, 0) = m_d = R(N_d, m_d, 0)$, and, obviously, $T'(0) \supseteq T(0)$. Let $t \in T'(0)$ be an enabled transition with respect to N_d. If $(t, f_I^i) \notin S_D$ and $(t, f_O^i) \notin R_D$ for all i, then it is clear that t is also enabled by m, therefore $t \in T(0)$ in this case. When $(t, f_I^i) \in S_D$, let s be the firing

sequence of transitions from μ to m, k_S^i be the number of transitions t' in s such that $(t', f_I^i) \in S_D$, k_R^i be the number of transitions t'' such that $(t'', f_O^i) \in R_D$, p_{IS}^i be the number of tokens in f_{IS}^i, p_O^i be the number of tokens in f_O^i, and k_f^i be the number of firings by transition t_f^i. Since there are n_i initial tokens in f_{IS}^i and f_{OS}^i, respectively, we have

$$p_{IS}^i = n_i - k_S^i + k_f^i$$
$$k_R^i \le k_f^i \le \min\{k_S^i, n_i + k_R^i\}.$$

However, by connection constraint (4c), in this case:

$$0 \le k_S^i - k_R^i \le n_i$$

that imlpies $\min\{k_S^i, n_i + k_R^i\} = k_S^i$. Therefore, t_f^i can be fired enough times such that $p_{IS}^i > 0$, which indicates that t is enabled under m, i.e., $t \in T(0)$. Similarly, when $(t, f_O^i) \in R_D$, we have

$$p_O^i = k_f^i - k_R^i$$

The constraint (Section IV-C) indicates in this case

$$1 \le k_S^i - k_R^i \le n_i + 1$$

which implies $\min\{k_S^i, n_i + k_R^i\} \ge k_R^i + 1$. Therefore, t_f^i can be fired enough times such that $p_O^i > 0$, hence $t \in T(0)$. In all the cases, $t \in T'(0) \Rightarrow t \in T(0)$, therefore $T'(0) = T(0)$.

Assume that $R_d(N, m, k) = R(N_d, m_d, k)$,$T'(k) = T(k)fork \le q$. Clearly, $R_d(N, m, q+1) = R(N_d, m_d, q+1)$ follows immediately from $T'(q) = T(q)$. Since $T'(q+1) \supseteq T(q+1)$, by the same procedure used in the proof of $T(0) = T'(0)$, we can show that $T(q+1) \supseteq T'(q+1)$. Hence $R_d(N, m, k) = R(N_d, m_d, k)$ and $T(k) = T'(k)$ for any $k \ge 0$.

Since N_d is live, every transition can be enabled by firing some transitions from m_d in N_d. Since $R_d(N, m, k) = R(N_d, m_d, k)$ and $T(k) = T'(k)$ for any $k \ge 0$, we see that the same transition can also be enabled after the same number of firings of transitions from m in N. Therefore, N is live.

Q.E.D.

The liveness of N insures the absence of deadlock in the coordination level.

B. Decision Making in the Coordination Structure

The decision making in the coordination level is achieved in three steps: task scheduling, task translation, and task formulation. Task scheduling is the process of identifying the appropriate tasks to be executed for the requested job. Once a task is located, task translation takes place by decomposing the task into a subtask sequence and, after assigned with real-time information through task formulation, executing subtasks or sending them to the corresponding infimal units. In terms of PNT, the problem of task scheduling and translation for a given task a is to find an enabled transition t such that the translation mapping $\sigma(t, a)$ is defined and then select the right translation string from $\sigma(t, a)$ for t.

Let Q_T and Q_D be two queues to be used to store the unexecuted tasks and the delayed tasks, respectively. Let $v = a_1 a_2 \ldots a_s$ be a task plan to be executed. Based on the execution rule of Petri nets, a simple and uniform scheduling procedure for task scheduling of the dispatcher and coordinators can be defined:

Scheduling procedure (SP):

1) $Q_T := a_1, a_2, \ldots, a_s, \quad Q_D := \phi$;
2) IF Q_T is empty THEN STOP;
3) $u := F(Q_T)$;
4) IF there exists a $t \in T(u)$ and t is enabled by firing t, GO TO 7;
5) IF there exists an internal operation sequence $e \in T_0^*$ such that a $t \in T(u)$ is enabled by firing e, THEN firing $e.t$, GO TO 7;
6) $I(Q_D, u)$, IF Q_T is empty THEN $Q_T := Q_D$ and $N(Q_D)$, GO TO 2;
7) IF Q_D is not empty THEN $Q_T := U(Q_D, Q_T)$ and $N(Q_D)$, GO TO 2.

where $F(Q)$ returns and deletes the first element of Q; $I(Q, u)$ inserts u to Q at the end of Q; $U(Q_1, Q_2)$ unifies Q_1 and Q_2 by placing Q_2 at the end of Q_1; and $N(Q)$ empties Q; Q, Q_1, and Q_2 are queue variables. $T(u) = \{t : \sigma(t, u)$ is defined$\}$ is the set of transitions that are capable of processing task u. $T_0 = \{t : \sigma(t, \lambda)$ is defined or $\sigma(t, \cdot)$ is not defined$\}$, called the set of internal operations, are the transitions that can be called to change the internal state of a transducer without processing any input task.

The information for task translation of a transition includes the task to be executed as by the translation mapping σ, the current relevant status of the transducer as specified by the input function I, and the feedback from the infimal units through the receiving mapping R. The transition will notify its task translation decision to those transitions specified by the output function O, and send it to the other units through the sending mapping S.

A passive approach for task translation has been adapted in the present work, i.e., a fixed number of different translations for a transition are predetermined and the translation is just to select one of them according to the situation encountered. Since it is impossible to get analytic formulas required for performance evaluation, a random strategy and learning approach have used for task translation. For a transition t, let $M_t = \Sigma|\sigma(t, a)|$ be the number of translations designed for t and N_t be the number of situations distinguished by the transition. Consider a matrix of subjective probabilities, $(p_{ij})_{M_t \times N_t}$. The decision rule of the task translation for choosing a translation is

Decision Rule (DR): When situation j, $j = 1, \ldots, N_t$, is observed, choose a translation string s_i using a random strategy with the subjective probability p_{ij}, $i = 1, \ldots, M_t$.

A random cost function is associated with each translation. After each execution of the action specified by s_i for situation j, the cost estimates and the subjective probabilities will be updated according to certain learning algorithm. This issue will be discussed in detail in the task simulation of Section V.

Using the execution rule of Petri nets, the scheduling procedure SP, and the decision rule DR, we can establish a uniform task execution procedure for the dispatcher and coordinators to manipulate the control and communication among them during the task processing of the coordination level [31].

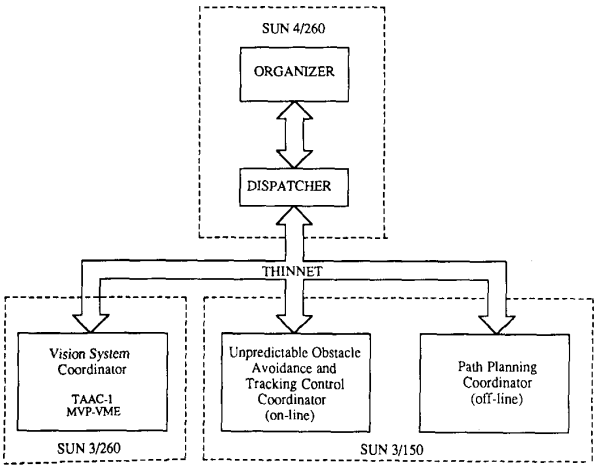


Fig. 2. Implementation of IMRS.

III. System Description and Petri Net Model of the Coordination Level

The mobile robot used for the IMRS is a Labmate of TRC. The robot will be able to perform motion tasks with the ability of avoiding unexpected moving objects in a structured environment. The IMRS is composed of three main levels namely, the *organization level*, the *coordination level*, and the *execution level*. The Organizer, in the organization level, issues higher level task plans in the form of ordered strings of tasks to the coordination level to accomplish the tasks assigned for IMRS. After receiving task plans from the Organizer, the coordination level translates them to real-time operation instructions and then gives these instructions interactively to the devices of the execution level. The execution level will execute the instructions and report the execution results to the coordination level.

The focus of this paper is on the coordination level, which consists of one dispatcher (DSP) and three coordinators: a path planning coordinator (PPC), an obstacle avoidance and tracking control coordinator (OATC), and a vision system coordinator (VSC), as depicted in Fig. 2. Since only task plans from the Organizer are concerned here, these plans have been generated by a given task grammar.

To facilitate the design and implementation of the mobile robotic system, the system specifications have been divided into two levels of abstraction. At the higher level of abstraction, the qualitative aspects of behavior of the system are represented by considering only the set of possible sequences of tasks and the precedence relationships among them. At the lower level, the quantitative aspects of behavior of the system are described by providing all the operational procedures invoked with the specific parameters. Since the interest here is focused on the coordination level of IMRS, only the higher level specification has been considered. The coordination structure described in Section II is used to specify the coordination level.

Four tasks for the coordination level have been defined: 1) wmu:3–D world memory update; 2) mod: moving obstacles detection; 3) pp: path planning; 4) $moac$: moving obstacle avoidance and tracking control. The one-step task plans from the Organizer have been generated by the grammar

$$G = (S, N, \Sigma_o, P)$$

where $\Sigma_o = \{wmu, mod, pp, moac\}$ is the set of terminal symbols; S is the start symbol; $N = \{S, A, B, C, D\}$ is the set of nonterminal symbols; and

$$P = \{S \rightarrow wmuA, A \rightarrow ppB, B \rightarrow modC, C \rightarrow moacD \mid moac, D \rightarrow B \mid S\}$$

is the set of production rules.

In order to be able to concurrently execute the present motion task and do path planning for the next motion task, two-step task plans are constructed using one-step plans by

$$TS = L(G) \| L(G)$$

where $L(G)$ is the language generated by grammar G and $\|$ is the concurrent operator between two languages [18]: TS is the set of task plans to be processed by the dispatcher.

A. The Dispatcher (DSP)

1) Unit Description: The dispatcher deals with the control and communication of the three coordinators. It is mainly concerned with the problem of which coordinator should be called for a given task (*task sharing*) and/or be informed by the result of recently completed execution (*result sharing*). The control and communication is achieved by translating a given task plan to a sequence of control commands with the necessary information and dispatching it to coordinators. The process of translating and dispatching is performed interactively between the dispatcher and the coordinators. After the completion of all the required tasks, the dispatcher will formulate the feedback information to the Organizer.

The dispatcher (as well as the Organizer) is physically sited on a Sun 4/260 workstation based on a 10 MIPS Sparc processor (RISC architecture) with 8 Mbytes of main memory and 892 Mbytes hard disk at CIRSSE. The command and data transmission between the dispatcher and coordinators is realized through Thinnet, a local network version of Ethernet.

2) Petri Net Model: In order to process the task plans generated by the grammar G, the dispatcher given by the Petri net in Fig. 6 has been designed (the links connected with any of the places I, IS, O, and OS are ignored at this point). A description of the individual transitions follows.

t_s	initialize the dispatcher when a task is received from the Organizer.
t_{wmu}	issue command to perform "3–D world memory update" process.
t_{pp}	issue command for path planning and provide constraints (space, kinematic, dynamic, etc.,) and optimization criterion
t_{mod}	issue command for moving obstacle detection within the specified space and provide accuracy and time requirements.

t_{moac} issue command for moving obstacle avoidance and tracking control.
t_{mte} perform motion task evaluation. In case the execution is failed, determine an appropriate sequence of transitions to continue the motion task.
t_{fbo} formalize feedback required by the Organizer.
t_{cmo} continue motion task along the original path.
t_{cmn} continue motion task along a new path.
t_f report the result of motion task execution to the Organizer.

3) Task Translation and Specification: The input alphabet of the dispatcher is Σ_o, and the output alphabet is

$$\Delta_o = \Sigma_c = \Sigma_p \cup \Sigma_m \cup \Sigma_v$$

where

$$\Sigma_p = \{\text{path}\}$$
$$\Sigma_m = \{\text{freemove}, \text{move}\}$$
$$\Sigma_v = \{\text{sendinfo}, \text{detection}, \text{terrain}\}$$

are the input alphabets to the path planning, obstacle avoidance and tracking control, and vision coordinators. The tasks in these alphabets will be defined later in the specification of the corresponding coordinators.

The translation mappings in the dispatcher are:

$$\sigma(t_{wmu}, wmu) = \{\text{terrain}(mmo_i, \text{art}_i) \quad i = 1, 2, \ldots\}$$

where mmo_i: minimum magnitude of the objects to be considered. art_i: accuracy requirements about detected objects in terrain exploration mode:

$$\sigma(t_{pp}, pp) = \{\text{path}(mss_i, prp_i, dct_i, gcc_i, poc_i) \quad i = 1, 2, \ldots\}$$

where mss_i is the margin of safe distance from static obstacles. prp_i is the preferred region for the path. dct_i is the dynamic constraints of the robot expressed in terms of bounds in torque. gcc_i is the geometric constraint in terms of bounds in curvature of $F(s)$; and poc_i is the parameters of optimization criteria. That is, values for the parameters of the following cost function:

$$J = \int_0^{T_f} (C_t + C_u \cdot \|u\|^2) \quad dt.$$

where C_t and C_u are weighting factors for time and energy, respectively, and u is the control torque:

$$\sigma(t_{mod}, mod) = \{\text{detection.sendinfo}(ard_i, ptc_i), \text{sendinfo}(ard_i, ptc_i) \quad i = 1, 2, \ldots\}$$

where ard_i is the accuracy requirements about detected objects in moving object detection mode and ptc_i is the upper bound of processing time (results must be reported within that time).

$$\sigma(t_{moac}, moac) = \{\text{move}(msm_i, ftp_i), \text{freemove}(ftp_i) \quad i = 1, 2, \ldots\}$$

where msm_i: margin of safe distance with moving obstacles and ftp_i is the requirement about final execution time and position.

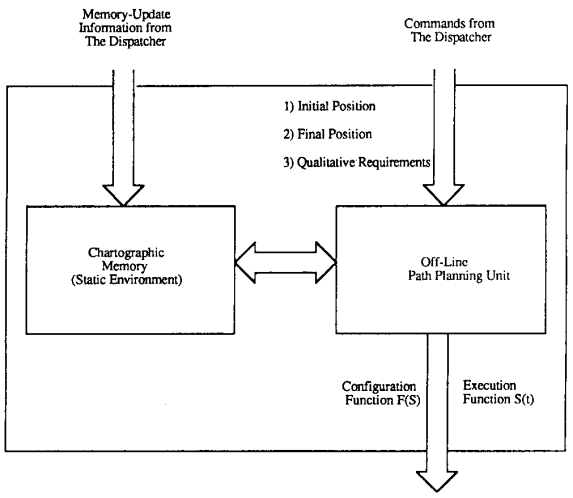


Fig. 3. The path planning coordinator.

Note that there are two tokens in the initial place p_s. This will make the dispatcher capable of processing the concurrent task sequences specified by $TS = L(G) \| L(G)$.

B. The Path Planning Coordinator (PPC)

1) Unit Description: The path planning (off-line) coordinator will solve the problem of finding a path between a start point and a goal point, avoiding collisions with the static obstacles and optimizing over some criteria (e.g., time) under certain constraints (e.g., input constraints). A macroscopic structure of this coordinator is given in Fig. 3. The cartographic memory stores the geometric description of the environment pertinent to the motion tasks.

A path is specified by its geometry through a *configuration function*

$$F(s) \in \Re^2 \qquad s \in [0, 1]$$

and its motion trajectory through a nominal *execution function*

$$s_n(t) \qquad t \in [0, T_f]$$

where T_f is the total motion time. For reasons explained later, it is also necessary to associate $F(s)$ with two more functions, namely, $L(s)$ and $R(s)$, denoting the distances of the obstacle free segment of the line perpendicular to $F(s)$ on the left and right of $F(s)$, respectively, as s goes from 0 to 1. This is done in order to have a notion of the free space along $F(s)$.

Several approaches are available for the calculation of $F(s)$ and$s_n(t)$ by the path planning coordinator [4], [7], [9]. The 2-D version of the *Extended VGraph Algorithm* by Chung and Saridis [4] has been used here.

After the completion of path planning, the result will be sent to the obstacle avoidance and tracking control coordinator for execution.

It is obvious that the complexity of the problem is big and the solution is obtained in an off-line fashion. This coordinator is physically realized, at CIRSSE, on a Sun Sparc station with 8 Mbytes of main memory.

2) Petri Net Model: The task to be processed by this coordinator is a single task $\Sigma_p = \{\text{path}\}$. The Petri-net model of the Planer is in Fig. 7. The transitions of the coordinator are as follows:

- t_xs initialize the coordinator.
- t_{cpm} choose path from menu when there exists a $F(s)$ in the menu of predetermined paths that satisfies the imposed requirements mss_i, prp_i, gcc_i (defined in Section III–A).
- t_{gso} build Grown Space Obstacles.
- t_{vg} construct VGraph.
- t_{fp} find a collision-free configuration function $F(s)$ (i.e., a collision free path) from VGraph with the requirements mss_i, prp_i, gcc_i (defined in Section III–A).
- t_{ef} determine execution function $s_n(t)$ (i.e., motion wrt time along $F(s)$) based on the constraints dct_i and poc_i.
- t_{fap} find an alternate path. In case that no trajectory was found to satisfy requirements, perform planning with minimum possible loss.
- t_f acknowledge results of path planning to the dispatcher and reset the input semaphore and the start place p_s.

3) Task Translation and Specification: The translation mappings are

$$\sigma(t_{cpm}, path) = \{\text{SearchMenu}\}$$
$$\sigma(t_{gso}, path) = \{\text{BuildGSpacsObstacles}\}$$

$$\sigma(t_{vg}, \lambda) = \{\text{BuildVGraph}\}$$
$$\sigma(t_{fp}, \lambda) = \{\text{SearchVGraph}\}$$
$$\sigma(t_{ef}, \lambda) = \{\text{Trajectory}\}$$

where SearchMenu, BuildGSpacsObstacle, BuildVGraph, SearchVGraph, and Trajectory are executable programs.

C. The Obstacle Avoidance and Tracking Control Coordinator (OATC)

1) Unit Description: The obstacle avoidance and tracking control coordinator supervises the execution of the motion task of the robot. The structure of this coordinator can be seen in Fig. 4. Three basic operations are involved in the coordinator:

1) utilization of the sensory input to adjust for local calibration errors with respect to the world,
2) collision avoidance with the *unpredictable* objects of the environment.
3) tracking control of the actual vehicle.

In order that these operations can be implemented in real time the following items must be obtained:

1) the information from the off-line planning, i.e., functions $F(s)$, $s_n(t)$, $L(s)$, $R(s)$, must be available to the coordinator,

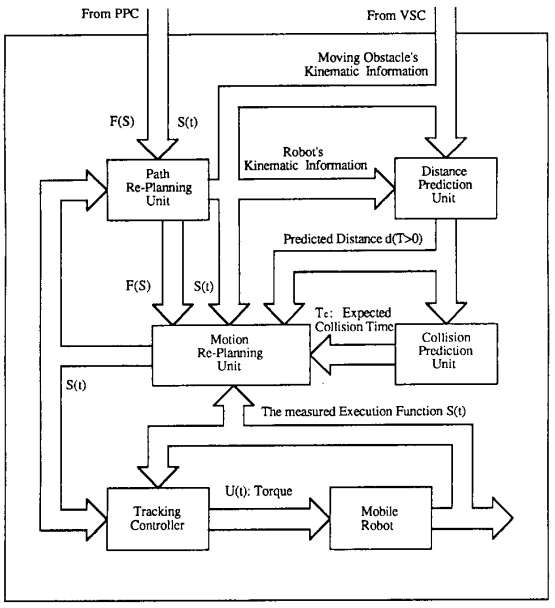


Fig. 4. The obstacle avoidance and tracking control coordinator.

2) some modeling simplifications, regarding the dynamics of the mobile robot and the shape of the objects, have to be made for the computation load reduction purpose.

In order to plan the path and control the vehicle efficiently, the dynamics of the mobile robot has to be fully taken into account. If the velocity $\dot{s}$ is bounded at each point of the trajectory to avoid sliding and skidding, a complete dynamic model of the mobile robot system is of the form [12]

$$(M + I \cdot f^2(s)) \cdot \ddot{s} + I \cdot f(s) \cdot f'(s) \cdot \dot{s}^2 = F$$

where M is the mass of the robot, I is the moment of inertia of the robot around the axis vertical to the moving plane, $f(s)$ is the curvature of the trajectory, and u is the input force into the system that is actually bounded by

$$U_{\min} \leq u \leq U_{\max}.$$

This dynamic model satisfies the nonholonomic constraints of the motion of the robot, because these were considered during the off-line planning, when $F(s)$ is defined. Furthermore, the velocity $\dot{s}$ has to be bounded at every point s of the trajectory so that skidding is avoided. Rolling without slipping is satisfied by appropriately steering of all the wheels of the mobile robot. This is assured during the off-line planning stage.

For the representation of moving objects, it is assumed that the shape of objects is satisfactorily described by their convex polyhedral hulls [10], or mathematically, by

$$A \cdot x \leq b$$

where $x \in \Re^3, A \in \Re^{l \times 3}$, $b \in \Re^l$ and l denotes the number of planes of the object. Although such a description

may be considered as crude, it satisfies the collision avoidance requirements and assists the reduction of computation load.

The major units in this coordinator are described briefly in the sequel:

The distance prediction unit takes as input from the vision system coordinator, the description of the polyhedral hull form and the prediction of the rotational and translational motion of a detected moving object. Combining them with the corresponding information of the robot, it predicts the minimum distance

$$d(t) = \min_{x,y}\{\|x - y\| : \quad x \in C_r(t), y \in C_o(t)\}$$

between the robot C_r and the moving object C_o, during some finite time horizon T_h [7].

The collision prediction unit determines the likelihood of having collision within time T_h, as well as the expected collision time T_c. This is done by an enhanced version of the accelerated expanding subinterval random search (AESRS) [11].

The motion replanning unit calculates a new execution function $s(t)$so that the expected collision can be avoided, while $s(t)$ stays as close as possible to $s_n(t)$. An optimal control strategy (OCS) proposed by Kyriakopoulos and Saridis [12] has been used by this unit for motion replanning. Furthermore, to enable the on-line implementation of OCS and to accelerate its convergence, a strategy that is based on the heuristic notion of either accelerate or deccelerate so that avoidance is guaranteed, was implemented. This strategy, the minimum interference strategy (MIS), is used to find an initial guess for the new motion function. Generally, MIS is computationally efficient and in most of the cases gives a close to optimal solution. Its output can be used as an initial guess to OCS.

When replanning of $s(t)$ does not guarantee collision avoidance, a local perturbation on $F(s)$ within the bounds $L(s)$ and $R(s)$, will be performed by the path replanning unit.

This coordinator is implemented, at CIRSSE, on a Sun Sparc workstation.

2) Petri Net Model: The tasks to be processed by this coordinator are $\Sigma_m = \{\text{freemove}, \text{move}\}$. The Petri-net is given in Fig. 8. The transitions of the coordinator are as follows:

- t_s initialize coordinator.
- t_{np} execute nominal plan with the specification ftp_i (defined in Section III–A) in case of no moving obstacle.
- t_{dp} combine the motion information about a moving obstacle and the robot and determine the time horizon T_h of interest during which the predicted minimum distance between the object and the robot is sought.
- t_{cp} predict collision and estimate the collision time T_c, by the accelerated expanding subinterval random search (AESRS).
- t_{dmp} perform motion replanning with the specifications msm_i and ftp_i (defined in Section III-A). Combine information about geometry $F(s)$ of path, current execution function $s(t)$, predicted minimum distance $d(t)$, expected collision time T_c and moving objects' predicted trajectory, determine $s(t)$ that can prevent collision. First execute MIS to get a collision free initial guess and then feed the result to OCS.
- t_{pr} perform path replanning with the specification msm_i. If there is no $s(t)$ that prevents collision, alter locally the geometry $F(s)$ of the path. t_{tc}: optimally track the desired trajectory, as expressed by both $F(s)$ and $s(t)$. t_f: report execution results to the dispatcher.
- t_f report execution results to the dispatcher.

3) Task Translation and Specification: The translation mappings are:

$$\sigma(t_{np}, \text{freemove}) = \{\text{DownloadCurrentTraj}\}$$
$$\sigma(t_{cp}, \lambda) = \{AESRS\}$$

$$\sigma(t_{dp}, move) = \{\text{DetermineTimeHor.PredictDistance}\}$$
$$\sigma(t_{mrp}, \lambda) = \{MIS.OCS\}$$

$$\sigma(t_{pr}, \lambda) = \{\text{GetFeasSpace.Replan}\}$$
$$\sigma(t_{tc}, \lambda) = \{\text{InvKinematics.Trackangles}\}$$

where DownloadCurrentTraj etc., are are executable programs.

D. The Vision System Coordinator (VSC)

1) Unit Description: The vision system coordinator (on-line) performs two basic operations: detection of obstacles, and description of the detected obstacles as convex polyhedra.

This coordinator is hosted by a Sun 3/260 workstation with 8 Mbytes main memory, 280 Mbytes of Hard disk. A MVP-VME and a TAAC-1 boards are connected to the Sun-3 VME bus through a bidirectional host bus interface. Two Javelin cameras are employed by the coordinator. Fig. 5 gives the architecture of the vision system.

The Matrox MVP-VME board is used as a frame grabber and to perform basic image processing operations such as filtering, edge detection, thinning, thresholding, etc. Since it is assumed to use the convex polyhedra hulls to describe the various objects, an algorithm was developed to process their visual information. A Hough transformation is performed to map the straight lines of the Cartesian space to points in the parameter space, thus facilitating the tracking of the moving objects in a fast mode [1], [6]. A stereo matching algorithm has been developed that matches object's features (edges and vertices) extracted from the left and right view. Since the information on the moving points is computed with noise, a Kalman filter has been added to yield optimal estimates of tracking. The Hough transformation, stereo matching of the features and model construction are all performed on the TAAC-1 at a speed of 10 MIPS.

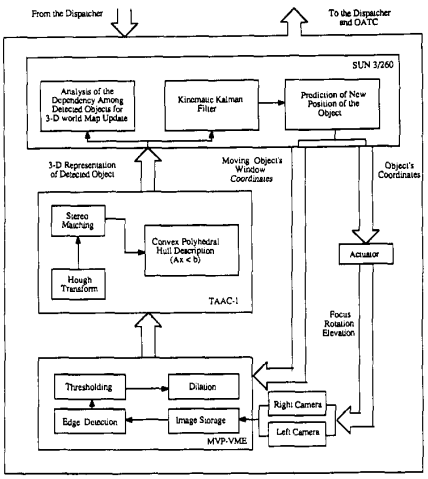


Fig. 5. Vision system coordinator.

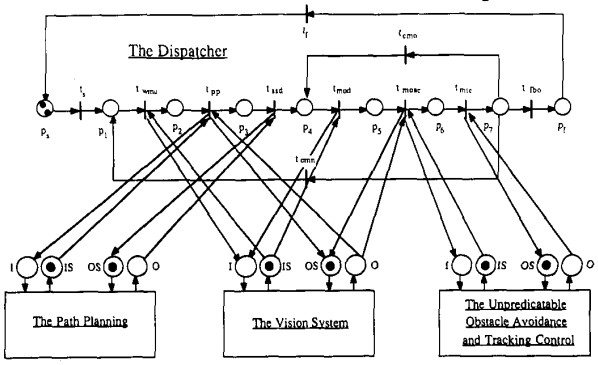


Fig. 6. Coordination structure for the mobile robot system.

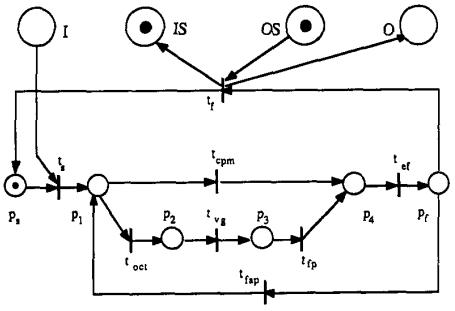


Fig. 7. Petri net model for the PPC.

In the case of terrain exploration, similar operations have to be performed for the static environment. This information is used for the world map update. The features of the "new" static objects, are compared to those included already in the cartographic memory map. Decision is made on whether or not these objects should be incorporated into the map.

The result of this coordinator is transmitted to the path planning coordinator or the obstacle avoidance and tracking control coordinator.

2) Petri Net Model: The tasks to be processed by this coordinator are Σ_v =sendinfo, detection, terrain. The Petri-net of the vision coordinator is given in Fig. 9. The transitions of the coordinator are as follows:

- t_s initialize the coordinator and pass task related information.
- t_{cal} calibrate the coordinate systems and find the transformations.
- t_{cte} set to the terrain exploration mode with the requirements mmo_i and art_i (defined in Section III-A).
- t_{ote} perform the terrain exploration operations. Depending on the input (Fig. 9), either it updates the map with information coming from the vision system, or it searches the map for information requested by the path planning coordinator.
- t_{smd} set to the moving object detection mode. with the requirements ptc_i and ard_i (defined in Section III-A).
- t_{ipo} grab frames using both cameras and perform edge detection, thinning, thresholding, and dilation. These operations can be performed in parallel for the right and left images.
- t_{fe} calculate the analytic representation of the object's features (edges, vertices) using Hough transform, on each image.
- t_{sm} match the features extracted from the right and left image using stereo matching and calculate the range by triangulation.
- t_{kf} use Kalman filter to obtain the translational and rotational kinematic parameters of the moving object.
- t_{ot} extend filtering process to predict the motion of the objects in order to focus, rotate and elevate the cameras to track the object, and move processing window in image processing level.
- t_{mp} acknowledge availability of extracted parameters of moving object so as to continue to track object (bidirectional arc).
- t_f report information to the dispatcher.

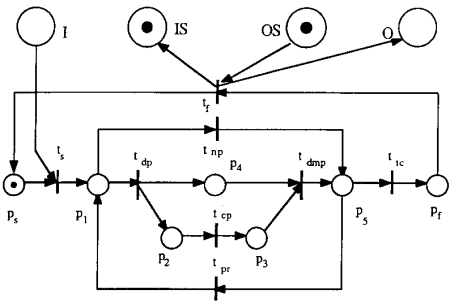


Fig. 8. Petri net model for the OATC.

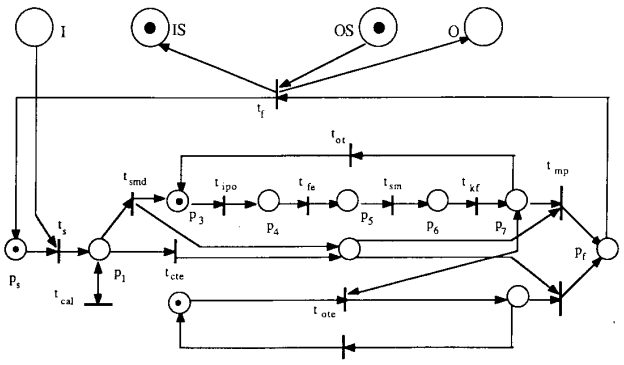


Fig. 9. Petri net model for the vision controller.

3) Task Translation and Specification: The translation mappings for this coordinator are

$$\sigma(t_{cte}, \text{terrain}) = \{\lambda\}$$
$$\sigma(t_{smd}, \text{detection}) = \{\lambda\}$$
$$\sigma(t_{mp}, \text{sendinfo}) = \{\text{SockSequentPacket}\}$$

$$\sigma(t_{ote}, \lambda) = \{\text{ExtractEdges.FormObject.MatchObject}\}$$

$$\sigma(t_{ipo}, \lambda) = \{\text{GrabFrame.Convolve.Threshold}\}$$
$$\sigma(t_{sm}, \lambda) = \{\text{StereoMatch.Triangulation}\}$$

where ExtractEdges, FormObject, MatchObject, GrabFrame, Convolve, Threshold, StereoMatch, Triangulation, SockSequentPacket are executable programs.

E. Integration and Execution

The final coordination structure for IMRS is obtained by integrating the PNT's for the dispatcher and the coordinators through the sending and receiving mappings, which have already been specified in the individual PNT's (Fig. 6–9). The input point (I), input semaphore (IS), output point (O), and output semaphore (OS) are physically realized through common memory located on the Sun 4/260.

For each of PNT's for the dispatcher and coordinators, the following execution procedure has been used to control its operation:

Task Execution Procedure

1) Receive a task plan $v = v_1 \dots v_m$;
2) Use SP to find an executable task v_i and a sequence of transitions $t_{i1} \dots t_{in}$ to execute it;
3) For $k = i_1$ to i_n do:
   a) Call DP to select a translation (subtask plan) u_k among $\sigma(t_k, v_i)$ for the execution of t_k;
   b) Remove tokens from the input places and displace tokens to the output places of t_k according to the input and output functions, the sending and receiving mappings;
   c) Send u_k to the input point of the coordinator specified by the sending mapping;
   d) Evaluate the execution result of u_k based on the feedback from the output points of C_k;
   e) Update the performance estimates and subjective probability for t_k according to the learning algorithms used;
4) Remove v_i if it has been successfully executed;
5) Stop if no task left, go to step 2 otherwise.

where scheduling procedure SP and decision rule DR have been described in Section II.

IV. Task Simulation Scenario

The purpose of this section is to simulate the task process using the Petri net transducer model of the coordination level of IMRS described in the previous section. The dynamics of task process in IMRS have been simulated through traveling of tokens in the places of the coordination structure. The motion of tokens is controlled by the SP. The information associated with the tokens is determined by the DR and execution results. A linear refinement learning algorithm has been used in the simulation to improve the performance estimates and subjective probabilities in DR.

A simple scenario has been proposed for the simulation (Fig. 10): A mobile robot in a structured environment, is asked to move from a starting position A to a final B. While it moves, it should be able to avoid unexpected moving obstacles. The environment is divided into two regions, $R_i \quad i = 1, 2$. In the region R_i, it is assumed that a moving obstacle may come with probability p_i^o, through any of the entrances and at any instant during the motion of the robot. An obstacle appearing at R_i will move randomly toward one of the inside doors in the same region, with either a "slow" or a "fast" speed. The probability that a moving object in R_i moves slowly is p_i^s. Note that only one moving obstacle may be allowed to show up in a region during the one motion task.

The model described in Section III-C has been used to represent the dynamics of the robot Labmate in this simulation. Input to the vision system is the simulated images created by setting intensity values contaminated with white noise to all the points of the image plane that correspond to objects (represented by convex polyhedral hulls). The origin of the vision coordinate system is taken at the center of the left image plane. The vision coordinator is asked to extract the edges of objects and to calculate their 3-D representation. Output of this coordinator is the coordinate information of moving objects required by obstacle avoidance and tracking control.

TABLE I
EVENTS, AVERAGE MOTION TIMES, AND RELIABLITITES FROM SIMULATION

	R_1	R_2	$R_1 \cup R_2$
Number of Motion Tasks	38	12	50
Number of Predicted Collisions	28	10	38
Number of Collisions	2	1	3
Average Motion Times (s)	9.89	11.64	10.30
Estimated Motion Reliabiltiies	0.764	0.366	0.671

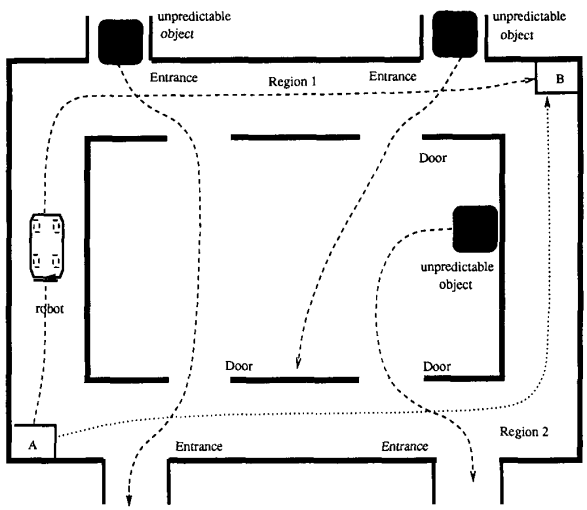


Fig. 10. Layout of the work cell for motion task simulation.

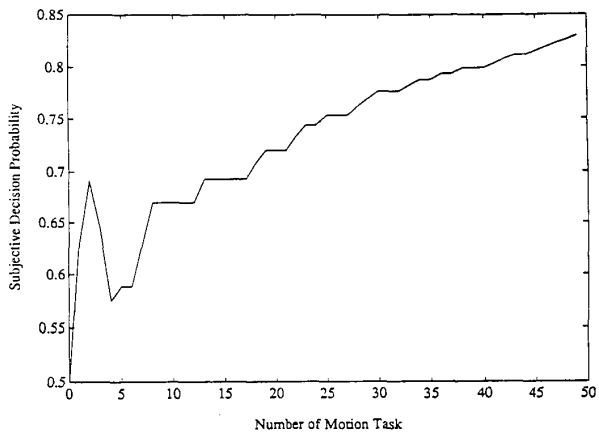


Fig. 11. Learning curve of region R_1.

V. RELIABILITY MEASURES AND LEARNING PROCESS

The only learning performed in the task process is to let the dispatcher decide the preferred region R_1 or R_2 in which the path planner should find a path. This decision is made by the transition t_{pp} of the dispatcher (i.e., by setting $prp = 1$ or 2). Let p_r^i be the subjective probability to select R_i, $i = 1, 2$ ($p_r^1 + p_r^2 = 1$). To learn the optimal decision, a reliability function has been used as the performance criterion [15]. The reliability of the decision made by t_{pp} is defined to be the probability for the robot to move from A to B within the planned time T_f. The reliability of the decision is estimated and updated by the following algorithm:

$$\tilde{R}_i(k_i + 1) = \tilde{R}_i(k_i) + \beta(k_i + 1) \cdot [R_{\text{obs}}(k_i + 1) - \tilde{R}(k_i)]$$

where R_{obs} is the observed value for reliability, $R_{\text{obs}} = 1$ when the motion is succeed within T_f, $R_{\text{obs}} = 0$, otherwise. $\tilde{R}$ the reliability estimate, and k_i the number of times region R_i has been chosen. After updating the performance estimate, the subjective probability is updated by the algorithm:

$$p_r^i(k + 1) = p_r^i(k) + \gamma(k + 1) \cdot [\xi_i(k) - p_r^i(k)],$$

$$\xi_i(k) = \begin{cases} 1 & \text{if } \tilde{R}_i = \max_j \tilde{R}_j \\ 0 & \text{otherwise} \end{cases}$$

where k is the number of times the transition t_{pp} has been fired. The $\beta(k)$ and $\gamma(k)$ are any series that satisfy Dvoretsky's convergence condition [22].

The environment in the simulation has been assumed to be $p_1^o = 0.5, p_2^o = 0.8$, $p_1^s = 0.8$, $p_2^s = 0.2$. Unitary initial reliability estimates, uniform initial subjective probabilities and $\beta(n) = \gamma(n) = 1/(10 + n)$ have been used. The motion task from A to B within T_f=10.02 s has been repeated 50 times in the simulation. The simulation results are described in Figs. 11 and 12 and Table I. The learning curve in Fig. 11 (i.e., the subjective decision probability p_r^1 versus the number of motion tasks performed) indicates that the optimal region for the motion is region R_1, agrees with the observation that $\tilde{R}_1 > \tilde{R}_2$. Fig. 12 presents the distribution of the motion times from A to B over the 50 motion tasks. Table I gives the average motion times from A to B in the two regions, the estimated reliabilities of accomplishment of motion tasks within the specified time T_f, and enumeration of events that occurred during the simulation. Such events are motion tasks performed on each region, collisions predicted and collisions happened on each regions.

VI. CONCLUSION

This paper has presented an application example of the general coordination theory for intelligent machines developed by Wang and Saridis [30] in the area of intelligent robots. The effort here has been focused on the specification and integration of path planning, control, and vision systems that are necessary for the autonomous operation of an intelligent mobile robot. The goal guiding this work is to use the formal model to explicitly represent the task precedency and information dependency among the individual systems in the coordination level of the IMRS. This work is only the first step toward the completion of the IMRS at the NASA Center for Intelligent Robotic Systems for Space Exploration (CIRSSE) at Rensselaer Polytechnic Institute.

The Petri net model established has provided an analytic framework for not only task simulation, but also performance

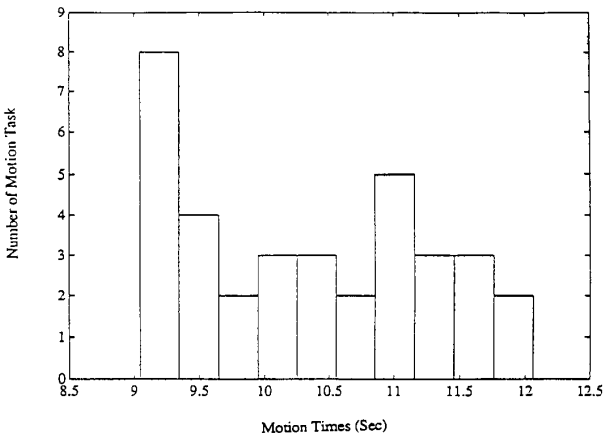
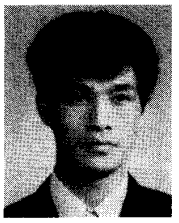


Fig. 12. Distribution of motion times in simulation.

evaluation. The task simulation can be used to test and verify the correctness and other qualitative properties of the system softwares and control algorithms. The performance evaluation can be employed to improve the system performance and to guide the hardware selection. The simulation program developed here will be expanded to include a Monte Carlo work cell generator to model the environment uncertainty and the SILMA CimStation to visualize the task process. The performance evaluation can be conducted by associating execution time with transitions of Petri net. The timed Petri net obtained for the coordination structure can be used to evaluate various performance indices to measure the efficiency of operations and the responsibility of decisions in the coordination level of IMRS, as have been demonstrated [13], [19].

REFERENCES

[1] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recog.*, vol. 13, pp. 111–122, 1980.
[2] S. Bonner and R. B. Kelley, "Planning 3–D collision-free paths, CIRSSE Tech. Rep. 29, Rensselaer Polytechnic Inst., Troy, NY 1989
[3] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robotics Automat.*, vol. RA-2, Mar. 1986.
[4] C. H. Chung and G. N. Saridis, "Obstacle avoidance path planning by the Extended VGraph algorithm," CIRSSE Tech. Rep. 12, Rensselaer Polytechnic Inst., Troy, NY 1989.
[5] J. L. Crowley, "Asynchronous control of orientation and displacement in a robot vehicle," *1989 Int. Conf. Robotics Automat.*, Scottsdale, AZ, 1989.
[6] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, Jan. 1972.
[7] E. G. Gilbert and D. W. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles," *IEEE J. Robotics Automat.*, vol. RA-1, Mar. 1985
[8] C. Isik and A. M. Meystel, "Pilot level of a hierarchical controller for an unmanned mobile robot," *IEEE J. Robotics Automat.*, vol. RA-4, no. 3, June 1988.
[9] Y. Kanayama and S. Yuta, "Vehicle path specification by a sequence of straight lines," *IEEE J. Robotics Automat.*, vol. RA-4, June 1988.
[10] K. J. Kyriakopoulos and G. N. Saridis, "An efficient minimum distance and collision estimation technique for on-line motion planning of robotic manipulators," NASA–CIRSSE Tech. Rep. 19, Rensselaer Polytechnic Inst., Troy, NY, 1989.
[11] ——, "Minimum distance estimation and collision prediction under uncertainty for on-line robotic motion planning," *IFAC'90 Tallin Congress*, 1989.
[12] ——, "A supervisory control strategy for navigation of mobile robots in dynamic environments," Ph.D. thesis, Rensselaer Polytechnic Inst., Troy, NY, Mar. 1991.
[13] A. H. Levis, "Human organizations as distributed intelligence systems," *IFAC DIS'88*, 1988.
[14] T. Lozano-Perez and M. Wesley, "An algorithm for planning collision free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, 1979.
[15] J. E. McInroy and G. N. Saridis, "Reliability analysis in intelligent machines," NASA–CIRSSE Tech. Rep. 39, Rensselaer Polytechnic Inst., Troy, NY, and accepted for publication by *IEEE Trans. Syst., Man, Cybern.*
[16] H. P. Moravec, "Rover visual obstacle avoidance," in *Proc. 7th Int. Joint Conf. Artificial Intell.*, Vancouver, BC, Canada, 1981.
[17] F. R. Noreils and R. G. Chatila, "Control of mobile robot actions," *IEEE 1989 Intern. Conf. Robotics Automat.*, Scottsdale, AZ, 1989.
[18] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1981.
[19] J. Robinson and A. A. Desrochers, "Performance analysis of the CIRSSE testbed control architecture," NASA–CIRSSE Tech. Rep. 47, Rensselaer Polytechnic Inst., Troy, NY, 1989.
[20] A. C. Sanderson and L. S. Homem de Mello, "Real-time planning and intelligent control," NASA–CIRSSE Tech. Rep. 31, Rensselaer Polytechnic Inst., Troy, NY, 1989.
[21] G. N. Saridis, "Foundations of intelligent controls," *Proc. IEEE workshop on Intelligent Contr.*, pp. 23–27, Rensselaer Polytechnic Inst., Troy, NY, 1985.
[22] G. N. Saridis and J. H. Graham, "Linguistic decision schemata for intelligent robots," *Automatica*, vol. 20, no. 1, pp. 121–126, 1984.
[23] G. N. Saridis and H. E. Stephanou, "A hierarchical approach to the control of a prosthetic arm," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 407–420, 1977.
[24] G. N. Saridis and K. P. Valavanis, "Analytic design of intelligent machines," *Automatica*, vol. 24, pp. 123–133, 1988.
[25] R. P. Sobek and R. G. Chatila, "Integrated planning and execution control for an autonomous mobile robot," *Int. J. Artificial Intell. in Eng.* vol. 3, Apr. 1988.
[26] M. Takano, S. Odaka, T. Tsukishima, and K. Sasaki, "Study on mobile robot navigation control by internal and external sensor data with ultrasonic sensor," in *Proc. IROS'89*, Tsukuba, Japan, 1989.
[27] A. Tsolkas, "Visual motion detection using Hough transform," Master's thesis, Rensselaer Polytechnic Inst., Aug. 1990.
[28] F. Y. Wang and G. N. Saridis, "A formal model for coordination of intelligent machines using Petri nets," in *Proc. 3rd IEEE Int. Intell. Contr. Symp.*, Arlington, VA 1988.
[29] ——, "A coordination model for intelligent machines," *CIRSSE-TR-89-15*, Rensselaer Polytechnic Inst., Troy, NY, 1989.
[30] ——, "A coordination theory for intelligent machines," accepted by *IFAC J. Automatica*, 1989.
[31] ——, "The coordination of intelligent robots: A case study," in *Proc. 4th IEEE Intern. Intell. Contr. Symp.*, Albany, NY, 1989.



Fei-Yue Wang (S'89–M'90) was born in Qingdao, China, in November 2, 1961. He received the B.E. degree in chemical engineering from Shandong Institute of Chemical Engineering, Qingdao, China, the M.S. degree in mechanics from Zhejiang University, Hangzhou, China, and the Ph.D. degree in computer and systems engineering from Rennselaer Polytechnic Institute, Troy, New York, in 1981, 1984, and 1990 respectively.

From 1984 to 1986 he was an instructor at the Department of Mechanics, Zhejiang University, China. In 1986, he was awarded Pao Yu-Kong and Pao Zao-Long Scholarship for Chinese Students for his academic achievement. Currently, he is an Assistant Professor at the Department of Systems and Industrial Engineering, the University of Arizona, Tucson. In his previous work in mechanics and applied mathematics, he contributed significantly to the development of theories of shell, plate, plane, three-dimensional elasticity, and micropolar elasticity for both isotropic and anisotropic elastic materials. He is the translator of the book *Buckling of Elastic Structures* (by J. Roorda) and the author of more than 50 journal articles, conference papers, and technical reports in the areas of applied mathematics, mechanics, computer science, control, communication, robotics, and intelligent machines. His fields of interest include robotic systems and computer integrated manufacturing, intelligent control and intelligent machines, and theoretic computer science.

Dr. Wang is a member of Sigma Xi, Chinese Society of Mathematics and Mechanics, and the ASME.

Konstantinos J. Kyriakopoulos (S'86–M'90–S'91–M'91) was born in Athens, Greece, on September 19, 1962. He received the Diploma in mechanical engineering with Honors from the National Technical University of Athens (TNTUA), Greece in June 1985.He received the M.S. degree and Ph.D. degrees both in computer and systems engineering, in 1987 and 1991, from Rensselaer Polytechnic Institute, Troy, NY.

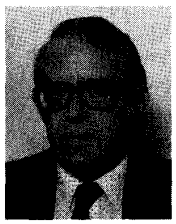
Since 1988 he has been doing research at the NASA Center of Intelligent Robotic Systems for Space Exploration. From 1986 to 1990 he held various teaching and research assistantship positions. His research interests are in the area of optimization theory with applications in robotic motion planning and control.

He is a member of the Technical Chamber of Greece. He was awarded the G. Samaras award of academic excellence from NTUA, the Bodosakis Foundation fellowship (1986–1989) and the Alexander Onasis Foundation Fellowship (1989–1990).

Athanasios Tsolkas (S'85–A'88) was born in Athens, Greece, on August 29, 1964. He received the Diploma in electrical and computer engineering from the National Technical University of Athens (NTUA), Greece, in 1988. He received the M.S. degree in computer and systems engineering in 1990 from Rensselaer Polytechnic Institute, Troy, NY.

During the academic year 1989–1990 he was a Research Assistant at the NASA Center of Intelligent Robotic Systems for Space Exploration. His research interests are in the are of Computer Vision and Computer Architectures.

He is a member of the Technical Chamber of Greece.



George N. Saridis (M'62–SM'72–F'78) was born in Athens, Greece. He received the diploma in mechanical and electrical engineering form the National Technical University of Athens, Greece, in 1955, and the M.S.E.E. and Ph.D. degrees from Purdue University, West Lafayette, IN, in 1962 and 1965, respectively. In 1988 he was certified as a Manufacturing Engineer for Machine Vision by the Society of Manufacturing Engineering.

Since September 1981 he has been a Professor of the Electrical and Computer Science Engineering Department and Director of the Robotics and Automation Laboratory at Rensselaer Polytechnic Institute. In 1973 he served as a Program Director of System Theory and Applications at the Engineering Division of the National Science Foundation. Since June 1988, he has been the Director of the NASA Center for Intelligent Robotic Systems for Space Exploration at Rensselaer Polytechnic Institute, Troy, NY. From 1955 to 1961 he was an instructor in the Department of Mechanical and Electrical Engineering for the National Technical University of Athens. From 1963 until 1981, he was with the School of Electrical Engineering, Purdue University. He was an instructor until 1965, and Assistant Professor until 1970, an Associate Professor until 1975 and a Professor of Electrical Engineering until 1981.

Dr. Saridis is a Fellow of the IEEE and a member of Sigma Xi, Tau Beta Pi, Eta Kappa Nu, the Academy of Sciences of New York, and the American Association of University Professors, the American Society of Engineering Education and the American Association for the Advancement of Science. In 1972–1973 he was the Associate Editor and Chairman of the Technical Committee on Adaptive Learning Systems and Pattern Recognition of the Society of Control Systems of the IEEE, Chairman of the 11th Symposium of Adaptive Processes, IEEE delegate to the JACC in 1973 and 1976, and Program Chairman of the 1977 JACC. In 1973 and 1979 he was elected as a member of the ADCOM and in 1986 he was appointed member of the Board of Governors of the Society of Control Systems of the IEEE. From 1970 to 1981 he was appointed Chairman of the Education Committee of the above society. He was the International Program Committee Chairman of the 1982 IFAC Symposium on Estimation and System Parameter Identification as well as the 1985 IFAC Symposium on Robotic Control. In 1974 and 1981 he was appointed Vice-Chairman of the IFAC International Committee on Education and from 1981 to 1984 he was the Survey Paper Editor for *Automatica*, the *IFAC Journal*. He is the series Editor of the JAI Publications on Annuals on Advances in Robotics and Automation. In 1986 he was appointed Chairman of the Control Systems Society's Committee on Intelligent Controls and Chairman of the Awards Committee of the Robotics and Automation Society. From 1983 to 1984 he was the Founding President of IEEE Council of Robotics and Automation. Dr. Saridis is also the author of the book, Self-Organizing Control of Stochastic Systems, editor of the book Annual on Advances in Automation and Robotics, Vol. 1 and 2 and co-editor of the books, *Fuzzy and Decision Processes*, *Proceedings of the 6th Symposium of Identification and Systems Parameter Estimation*, and *Proceedings of the 1985 SYROCO*. He has also written more than 300 book chapters, journal articles, conference papers and technical reports. He is the recipient of the IEEE Centennial Medal Award in 1984 and the IEEE Control Systems Society's Distinguished Member Award in 1989.