Proceedings of the 1999 IEEE
International Conference on Robotics & Automation
Detroit, Michigan • May 1999

Agent-Based Planning and Control of a Multi-Manipulator Assembly System

Juan-Carlos Fraile
Automatic and Systems Engineering Dept.
University of Valladolid
Valladolid 47011 – Spain
jcfraile@dali.eis.uva.es

Christiaan J.J. Paredis, Cheng-Hua Wang
Pradeep K. Khosla
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890
{cjp, cwwang, pkk}@ices.cmu.edu

Abstract

This paper presents a distributed planning and control architecture for autonomous Multi-Manipulator Systems (MMS). The control architecture is implemented using an agent-based approach. A team of distributed and autonomous agents is deployed to model the flexible assembly system in such a way that the agents negotiate, collaborate, and cooperate to achieve the goals of assembly tasks.

The main focus of this paper is on assembly task allocation and assembly task execution. We describe the agent models and communication mechanism, and explain how they handle complex interactions among agents. A distributed trajectory planning approach based on artificial potential fields is also presented.

Experimental results show that our multi-agent planning and control framework is suitable for flexible robotic assembly tasks. Our approach addresses the issues of flexibility, scalability, reconfigurability, and fault-tolerance. We anticipate that the same approach can be applied to other flexible manufacturing environments.

1 Introduction and Related Work

To stay competitive in the current global economy, it is crucial for companies to react quickly to changing economic factors, such as customer demands, new technologies, and cost of materials/production. Such market changes often encourage the creation of new products or demand product improvements. Accommodating frequent product changes requires a manufacturing system be more flexible than currently prevalent hard automation systems. Consequently, there has recently been an increasing interest in flexible manufacturing and assembly systems [15], [16] and [17].

There are two important aspects to a flexible manufacturing system: automation hardware and the corresponding planning and control software. The latter is the heart of a flexible manufacturing system; an appropriate software architecture can improve system performance significantly. In this paper, we focus mainly on the software aspect.

Distributed Artificial Intelligence covers the intersection of Artificial Intelligence and Distributing Computing. Multi-Agent Systems (MAS) are commonly used in solving difficult problems in the areas of Distributed Artificial Intelligence. This paper takes the multi-agent approach in which a team of agents, each with only limited local knowledge and local information, collaborates to satisfy both local and global objectives. The overall behavior of the system emerges through the dynamic interactions of the agent's local behaviors.

Several other researchers have addressed the use of MAS for the control of robotic systems. Basran [2] considers a flexible agent-based robotic assembly cell. The agents use a contract-net protocol for negotiation and dynamic task allocation. Oliveira [9] presents a cooperative multi-agent system for robotic assembly cells using a blackboard architecture as an inter-agent communication mechanism. Nagata [8] proposes a rather original approach to assembly task planning for multiple manipulators by associating an agent to each of the parts to be assembled, while Lueth [7] and Ouelhadj [10] take the more traditional approach of associating an agent to each of the robots performing the assembly tasks. Overgaard [11], on the other hand, develops a multi-agent approach to grasping in which each agent controls only part of a robot. Huang [4] proposes an agent-architecture for autonomous distributed systems based on the control mechanism of organs in living systems, or animals in colonies or groups.

Motion planning is critical for assembly task execution. The computational complexity of finding a collision-free path/trajectory for a multi-manipulator systems grows exponentially with the total number of degrees-of-freedom (DOFs) of the system [6]. As a result, most practical approaches utilize only heuristic solutions to make the problem tractable [1], [3], and [12].

2 Problem Definition

We are developing an agent-based planning and control system for a flexible assembly system with multi-manipulators. The input of the system is the mechanical model of the product to be assembled. The output of the system is the final assembled product. The general flow diagram that indicates the global operation of the system is shown in Figure 1. This flow is mainly divided in two stages: an off-line stage and an on-line stage.

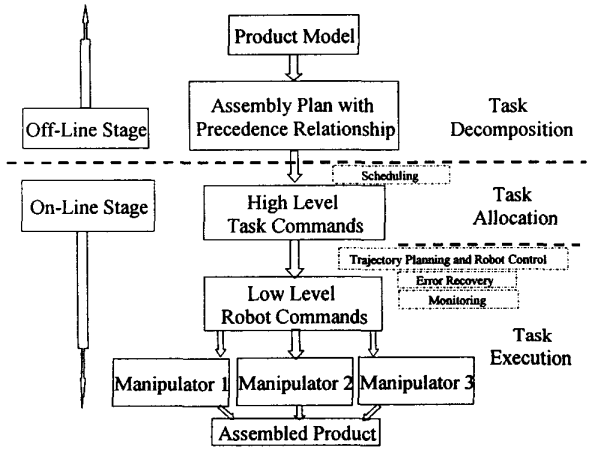

Figure 1: Global Operation Flow.

The off-line stage performs assembly task decomposition. It produces a preliminary assembly plan that consists of a sequence of assembly operations and the precedence relationships among them. The input to this off-line stage is the mechanical model of a product that is composed of parts. It then generates a preliminary assembly plan based on assembly accessibility and stability. The assembly operations that make up a preliminary assembly plan are task level operations. Currently we have implemented two such operations: "Pick/Place Part" and "Insert Part." The preliminary plan is used as the input to the on-line stage.

The on-line stage focuses on task allocation and task execution. In this stage, agents of a team collaborate and cooperate to achieve common goals. The task allocation phase assigns assembly operations to appropriate resources (manipulators). The task level operations are also mapped into low-level operations that manipulators can understand and execute. In the agent system, several special agents are added during task execution: a collision avoidance agent (for collision-free trajectory planning), a fault tolerance agent (for fault recovery), and a monitoring agent (for interleaving planning and execution).

This paper focuses on the on-line stage. We present an agent based planning and control architecture for task control and coordination. The multi-agent paradigm has been adopted as it provides the following advantages:

- **Homogenous Framework**: All system components are considered and modeled in a uniform and homogenous manner
- **Modularity and Scalability**: Different modules are developed and implemented independently. This approach can simplify agent component development/maintenance and make addition/ removal of agents much easier.
- **Dynamic Reconfigurability**: The capability to quickly reconfigure a system enables timely response to changing market conditions
- **Distributed Control**: Each agent has local control with the capability to coordinate its activities through messages-passing communication. Agents can reside on different computers and manipulator controllers.
- **Fault Tolerance**: Techniques for fault detection, failure recovery, and execution monitoring during task execution is provided to make system performance more robust.

3 Hardware Description

The multi-manipulator assembly system developed at the Automatic and Systems Engineering Department of the University of Valladolid consists of the following components (Figure 3). There are three 5 DOF Scorbot ER-IX manipulators in a triangular configuration. These manipulators share a common workspace where they can simultaneously perform assembly operations. In the

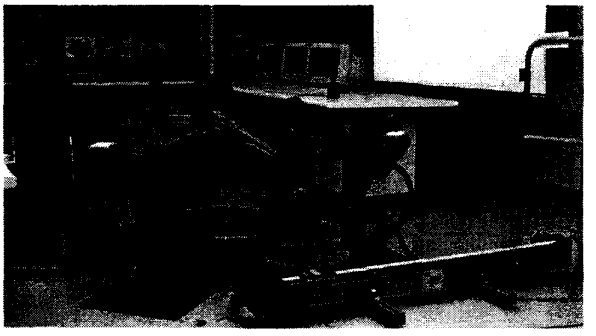

Figure 2: The MMS in Our Laboratory.

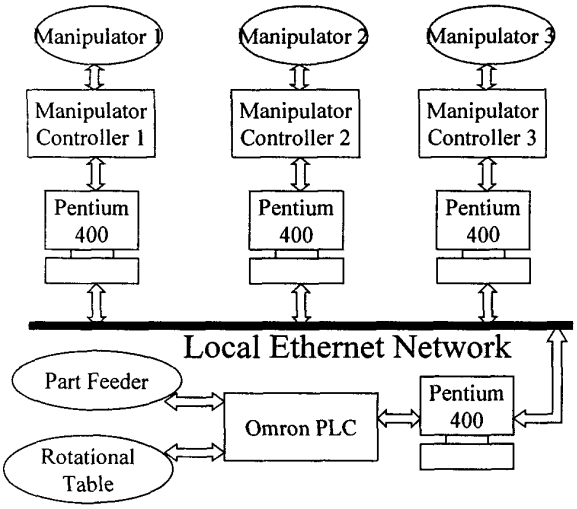

Figure 3: Hardware Architecture.

middle of the workspace there is a servo-controlled rotating worktable on which the assembly is built. A linear conveyor system serves as a part feeder to the manipulators. A buffer area is provided to store parts temporarily before they are assembled.

The agent-based control system is implemented in a distributed fashion on four Pentium 400 PCs. Each manipulator has its own low-level controller that is connected to one of the PCs over a serial RS-232 line. The rotating worktable and part feeder are controlled by an Omron-PLC, which is connected to the fourth PC. All four PCs are further linked through a local Ethernet network.

4 Agent Types

The multi-agent paradigm is based on the premise that complex problems can be partitioned into simpler sub-problems that have limited mutual dependencies. The fewer dependencies the system has, the more successful the agent-based approach can be. Therefore, problem decomposition and design of individual agent become the most important software design decisions for agent-based system. We have adopted a hybrid decomposition approach [14], which combines robot-based and task-based components. Our agent architecture is shown in Figure 4.

We have designed the following agents in our system:

- **Scheduler Agent** is responsible for scheduling the detailed assembly operations and assigning them to available resources (manipulators).
- **Manipulator Agents** are responsible for the low-level control of individual manipulators, and for converting task level assembly operations to low-level robot commands.
- **Auxiliary Component Agents** are responsible for the control of the auxiliary components, i.e. part feeder and the rotation table in our system.
- **World State Agent** contains both static and dynamic information of all components in the system. Static information includes geometric, kinematic, and dynamic information of all system components. Dynamic information includes current component configuration and any dynamic state information.
- **Trajectory Planning Agent** is responsible for generating the collision-free motion of the three manipulators on the fly.

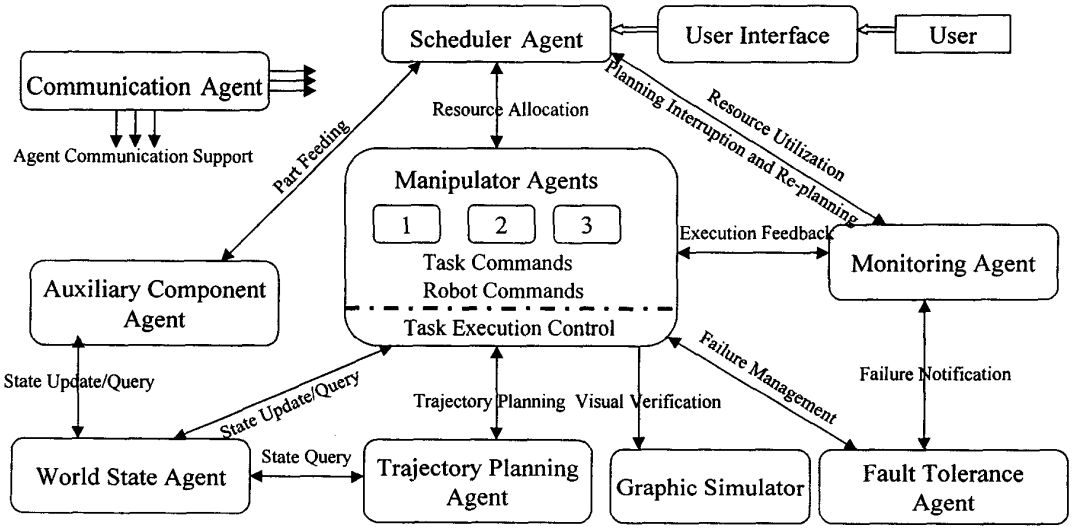

Figure 4: Agent System Architecture.

- **Fault Tolerance Agent** manages and coordinates the failure recovery process.
- **Communication Agent** handles message passing and message dispatching in the agent system through a blackboard architecture
- **Monitoring Agent** monitors task planning and execution to improve resource utilization. It notifies the fault tolerance agent in case of system failures.
- **Graphic Simulation Agent** provides visual verification of task execution.

5 Agent Model

A typical agent architecture in our system is illustrated in Figure 5. The agent model is based on the ARCHON project [13]. Agents are involved in two types of activities: internal activities that lead the agent to achieve its own goals, and external activities through which the agent communicates and collaborates with others agents to achieve common goals. To support both internal and external activities, a typical agent contains the following modules:

- **Communication Module**

The communication module is responsible for managing and controlling communications among agents. Both direct peer-to-peer communication and communication through blackboard architecture is supported. The communication module itself consists of a low-level network interface and a higher-level communication protocol.

The communication module manages two types of messages: state messages and commands messages. State messages contain dynamic state information of system components, while command messages contain other information, such as geometric information, kinematic and dynamic performance, or execution commands. Command communications are implemented by priority messages, buffers and without time restrictions. A blackboard architecture manages the command communications.

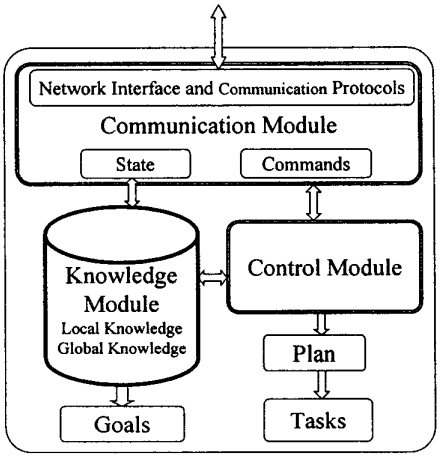

Figure 5: Agent Model.

- **Knowledge Module**

The knowledge module provides an agent with an information source about the agent itself and the environment (others agents). This module has a knowledge base that stores necessary data and knowledge for the agent to perform its activities. The knowledge base includes two kinds of knowledge:

1. **Local knowledge base**: It contains information related to agent capabilities and its own state information.
2. **Global knowledge base**: It contains information of other agents, with which the agent will work. The agent collaborates and cooperates with these agents during the task allocation and task execution stage.

- **Control Module**

The control module determines the agent's behaviors. It operates in a goal-driven fashion. When an agent is assigned a set of tasks, every task in the set is converted into a set of goals. The agent can achieve the goals as long as its capabilities can meet the requirements. Agent goals are mapped into a collection of plan scripts, which can be achieved by the control module. The scripts are coded with the necessary parameters and low level robot commands. The control module can run either independently or cooperatively with others agents. In our system the control module consists of a set of C++ programs which were developed according to the object-orientation programming (OOP) paradigm.

All agents in the MMS have their three modules located in PC-Pentium , except the manipulator agents that have its control module placed into the manipulator controller.

6 Communications Among Agents

The ability to represent, query, and manipulate knowledge is crucial to agent-based systems. Finin et al [5] propose KQML, a novel language for such purpose. We have adopted the KQML format to represent our message format and protocol. There are two types of communications in our system: commands and state communications.

- **Command communications *(non-periodic communication)***

A blackboard architecture has been implemented inside the communication agent to manage the command communications. The message is handled according to its

priority and time stamp. A high priority command message will be sent before a lower priority message. If two messages have the same priority, the one with earlier time stamp will be sent first.

Agents communicate with each other by posting command messages on the blackboard. The communication agent is responsible for handling message dispatching sending the message to its recipient, and sending the reply back to its sender. Figure 6 shows how this kind of communications is achieved.

The messages that fall into the category of command communications are:

1. **Control commands**: Commands related to the control of the system components (manipulators, rotation table, and part feeder).
2. **Information request commands:** Query commands regarding the agent's state and knowledge.
3. **Failure commands:** Failure report

- **State communications between world state agent and manipulator agent *(periodic communications)***

During the task execution stage, manipulator agents need up-to-date state information of other manipulator agents to plan collision-free trajectories. This information must be updated at each time instant to reflect the current states of the manipulators. This state information is used by the trajectory planning agent during task execution.

As we expect, there is intensive message passing between the world state agent and manipulator agents, particularly during the trajectory planning stage. This communication bottleneck can dramatically degrade system performance. To overcome this problem, we have developed a periodical and point-to-point state communication between manipulator agents and world state agent (Figure 7). For every small time interval (a few milliseconds) the world state agent will request the latest state information from all manipulator agents. Manipulator agents reply by sending their current state information to world state agent. The world state agent then updates all state information in it.

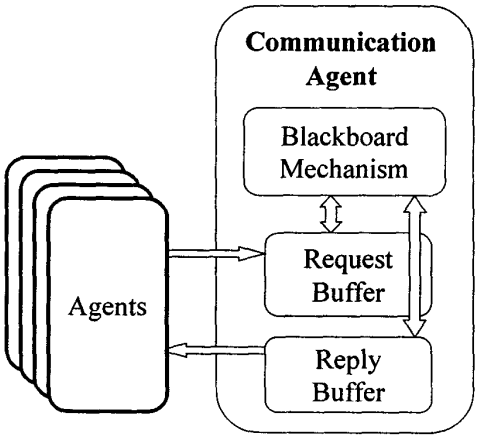

Figure 6: Command Message Managed by the Communication Agent.

Whenever an agent requests the state information of some manipulator, it requests the information from the world state agent instead of communicating directly with the manipulator agent. This approach reduces communication overhead and simplifies state information management.

- **State communications between world state agent and others *agents (non-periodic communication)***

Whenever a non-manipulator agent changes its state, it sends its current state to the world state agent. This is done in a non-periodic fashion to reduce unnecessary state polling form the world state agent. Figure 7 shows how the world state agent manages the state communications.

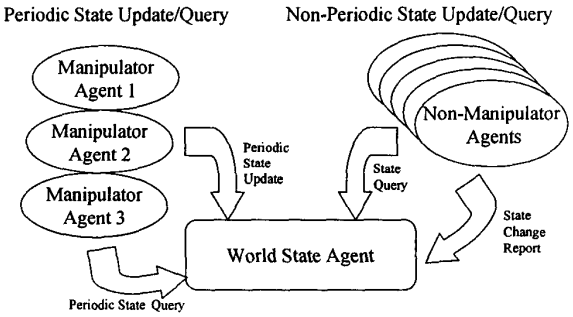

Figure 7: State Messages and World State Agent.

7 On-line Collision-free Trajectory Planning

As the three manipulators share the same workspace, collisions among these manipulators are possible. To avoid collisions between the robots as well as between the robots and the environment, we have developed an agent-based collision avoidance system. The system is based on the concept of *Manipulator Incremental Motion* (MIM). As illustrated in Figure 8, the trajectories of each manipulator are divided into small steps (a few millimeters along the trajectory). The computation of these steps is coordinated by the Trajectory Planning Agent and is performed in near real-time during the assembly execution. This allows us to produce collision-free task execution, even when the trajectories of the robots are not known in advance.

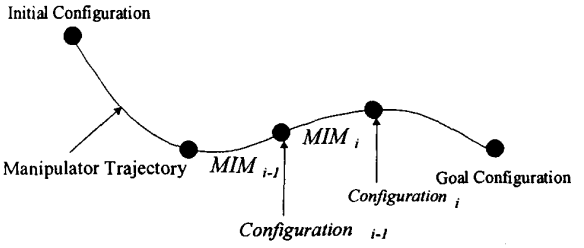

Figure 8: Manipulator Incremental Motion (MIM).

We have developed a distributed trajectory planning system in which each of the manipulator agents computes its incremental motion. The Trajectory Planning Agent assigns a priority to each of the manipulators to compute the next manipulator incremental motion in the order that has been determined previously.

The priority management is both dynamic and reactive. It is dynamic since the priority is calculated on the fly at each configuration along the path. It is reactive because it takes into account the System State at each time instant. The following factors are considered to compute the priority for each manipulator:

- *Assembly sequences.*
- *Distance between the manipulator and its goal configuration.* The closer to the goal configuration the manipulator is, the higher its assigned priority.
- *Distance between manipulator and parts.* A higher priority will be assigned to the manipulator that is closer to the parts that are about to be assembled. The part feeder will deliver the parts into the system.
- *Manipulator priority history.* In order to achieve load balancing for manipulators, the manipulator, which has been utilized frequently, will be assigned a lower priority.
- *Manipulator failure.* If a manipulator breaks down during execution, it will be assigned a high priority so that the following trajectory planning will first consider the broken manipulator and plan accordingly.

Once the priority of each manipulator is determined the next manipulator incremental motion of the manipulator will be computed.

The calculation of manipulator incremental motion of the manipulators is based on an artificial potential field technique [6]. This technique uses artificial potential (forces) to model the trajectory-planning problem. There are two kinds of potentials: attractive potential generated by the goal configurations of manipulators and repulsive potentials generated by obstacles. These two potentials encourage a manipulator to move towards its goal configuration and keep it from moving towards the obstacles. Time is considered as another independent variable to determine the motion of the manipulators.

The following information is needed in order to calculate the manipulator incremental motion of a manipulator:

- Current and goal configuration of the manipulator
- Geometric models of workspace, manipulators, part feeder, rotation table, storage buffer, and parts that have been assembled.

This approach can be treated as a search problem, which aims to find a shortest collision-free trajectory. We have developed a heuristic search approach to generating collision-free trajectories. For each manipulator at its current configuration, all feasible states are generated. Each state is represented by the following parameters:

- Distance between the current configuration and the goal configuration of the manipulator
- The states of the all system components

This method uses both, the configuration space (C-space) and the Cartesian workspace (W-space) representations. The potential function "g" is a function of two dependent variables, *Do* and *Dr*, where:

- *Do:* the distance from the manipulator to its goal configuration. 'Do' is defined within the manipulator configuration space.
- *Dr*: the minimum distance between the manipulator and all the obstacles in the system.

When both, the priority and the best manipulator incremental motion , have been calculated at each time instant, the complete motions of the manipulators can be obtained.

8 Conclusions

In this paper, we present a distributed planning and control architecture for autonomous multi-manipulator systems using a multi-agent paradigm. We have focused on the on-line stage (task allocation and task execution) of assembly operations. The task execution part has been completely implemented and tested with our hardware system. All agents except the fault tolerance agent have been designed and implemented and are currently being tested.

The experimental results demonstrate that our multi-agent architecture is an adequate framework for flexible robotic assembly. The system is modeled by a team of autonomous agents that develop cooperative strategies to achieve the common goals. Agents communicate with

designated communication format and protocol to exchange information.

We anticipated that our approach can be applied and extended to other flexible manufacturing systems. We are currently applying and extending the planning and control strategy to another flexible assembly system with four manipulators (RobotWorld by Automatix, Inc.) at Carnegie Mellon University. The preliminary experiments have shown some promising results.

Acknowledgements

This research has been funded in part by the Research Program TAP 95-0092 of the "Comisión Interministerial de Ciencia y Tecnología (CICYT)" of the Spanish Government, by DARPA under contract ONR #N00014-96-1-0854 and by the Institute for Complex Engineered Systems at Carnegie Mellon University.

References

[1] J.Barraquand, B.Langlois, and J.-C.Latombe. "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man and Cybernetics*, No 2, pp. 222-24, 1992.

[2] J.S. Basran, E.M. Petriu, and D.C. Petriu. "Flexible agent-based robotic assembly cell," *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pp. 3461-3466, 1997.

[3] B. Faverjon, and P. Tournassoud. "A local based approach for path planning of manipulators with a high number of degrees of freedom," *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, pp. 1152-1159, 1987.

[4] S.Y. Huang, and Umetani Y. "A constructing scheme for autonomous distributed control systems with multi-agent society," *Proceedings of the International Symposium on Autonomous Decentralized Systems*, pp. 17-24, 1997.

[5] Tim Finin, Richard Fritzson Don McKay and Robin McEntire. "KQML as an Agent Communication Language," *The Proceedings of the Third International Conference on Information and Knowledge Management* (CIKM'94), ACM Press, November 1994

[6] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[7] T.C. Lueth, and T. Laengle. "Task description, decomposition and allocation in a distributed autonomous multi-agent robot system," *Proceedings of the 1994 IEEE International Conference on Robots and Autonomous System*, pp 1516-1523, 1994.

[8] T. Nagata, and J. Hirai. "Distributed planning for assembly tasks by multiple manipulators," *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp.3522-3529. 1994.

[9] E. Oliveira. "Cooperative multi-agent system for an assembly robotics cell," *Robotics and Computer Integrated Manufacturing*, Vol. 11, No 4, pp. 311-317, 1994.

[10] D. Ouelhadj, C. Hanachi B. Bouzouia. "Multi-agent system for dynamic scheduling and control in manufacturing cell," *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, pp. 2128-2133, 1998.

[11] L. Overgaard, Nelson B. and Khosla P. "A multi-agent framework for grasping using visual servoing and collision avoidance," *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pp.2456-2461, 1996.

[12] P. Pal, and K. Jayarajan. "Fast path planning for robots manipulators using spatial relations in the configuration space," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 668-673, 1993.

[13] Thies Wittig, (ed.) *ARCHON: an architecture for multi-agent systems*, Ellis Horwood, 1992.

[14] E.S. Tzafestas, "Agentifying the process: task-based or robot-based decomposition?" *Proceedings of the 1994 IEEE International Conference on Systems, man and Cybernetics*, pp. 582-587. 1994.

[15] F. Choobineh and R. Suri (eds.), *Flexible Manufacturing Systems: Current Issues and Models*, Industrial Engineering Management Press and, Norcross: Georgia, 1986.

[16] P.F. Muir, A.A. Rizzi, and J. Gowdy, Minifactory: A Precision Assembly System Adaptable to the Product Life Cycle, in *Architectures, Networks, and Intelligent Systems for Manufacturing Integration*, B. Gopalakrishnan et al. Eds., Proceedings of the SPIE, Vol. 3203, pp. 74-80, 1997.

[17] Sandia National Laboratory, *AMPS: Agile Manufacturing Prototyping System*, http://www.sandia.gov/AMPSfact.html, 1997.