Architecture and Implementation of a Multisensor System for Robotic Assembly Environment

Yi-Tin Ho, *Li-Chen Fu* and *Han-Shen Huang*

Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

Abstract

To upgrade the robot intelligence adopted in a variant task environment, we have to integrate various sensors for environmental information and therefore we have to develop sensory data processing techniques or to find a negotiated solution from disparate sensory data. We propose a multisensor system architecture for the prototype of general configuration of sensor system designing, especially for robotic assembly environment. The utmost objective is to provide user (assembly system designer) with an environment without extra consideration of sensory module configuration or information flow.

1 Introduction

MANY specific robotic systems with multiple sensors integrated have been published [9–11]. Multisensor systems are surveyed in different areas of application; industrial tasks like material handling, part fabrication, military command and control for battle management, space, target tracking, inertial navigation, and remote sensing of coastal waters [7, 8]. With the aid of multisensor, robot would be endowed more intelligent to sense environment.

Robotic assembly systems are usually controlled by a hierarchical structural scheme with different levels, namely, task planning, subtask decomposition, trajectory planning, trajectory execution and manipulation, as depicted in Fig. 1.

At the task planning level, it performs the task such as assembly sequence planning to determie what sequence would be the most efficient (with cost and time consideration) way to sequentially complete the assembly task. A task command should always be decomposed into some subtask commands for ease of implementation. Trajectory planning, following the subtask decomposition level, is responsible for the task of planning trajectories for a robot arm. Trajectory execution takes charge of fulfilling trajectory plans and monitoring execution processes. The last level, manipulation, is always implemented in a closed-loop with sensor information about the environment and the robot actions.

Modern automatic assembly systems utilize many types of sensors [1, 3–5]. Various sensors are integrated to an assembly system for meeting the increasing need of automation. The robots employ the sensor as a measuring device; that is, it computes the sensor information and acts upon that information about working space [2]. With the hierarchical structure of the assembly system mentioned above, the functional structure of the integration becomes obvious. The sensor system should meet the needs of different levels in the assembly hierarchy. One must decide how automatic (intelligent) he or she wants the system to be when selecting an integration mechanism. The higher level information the sensor system can provide, the more intelligent the system is.

2 Problem Formulation

This paper aims to a multisensor system architecture which can be adopted in most of the assembly tasks, and to find general rules for organizing different sensors with necessary assembly information.

2.1 Different Phases in a Robotic Assembly Process

Robotic assembly process can be divided hierarchically into three levels, namely, task level, trajectory level, and manipulation level. In each level, a robot performs different kinds of work with the corresponding way of interaction with environment. They may use sensory data to accomplish more complicated tasks or to achieve higher-level machine intelligence. Before developing the sensor system, we have to investigate the flow of the assembly process and the necessary environmental information.

In temporal analysis, the task execution in a robotic assembly system can be divided into two phases, namely, *location/recognition (L.R.) phase* and *assembly (A.) phase*, which are discussed below.

1. (a) Location/recognition phase

   In *L.R. phase*, the main tasks of are to acquire as accurate information about the assembly work cell as possible. Furthermore, additional information on the task environment is also needed in this phase, such as object identification as well as the number and shapes of the existing obstacles.

   1.) Task level :
   The *L.R. phase* in this level is to accomplish

0-7803-2775-6/96 $4.00 © 1996 IEEE

the so-called pre-processing planning by making the necessary decisions, including determination of the suitable assembly types and of the robot actions to be taken for each particular part assembly.

2.) Trajectory level :
In the trajectory level, the main task is to determine the robotic motion trajectories so as to speed up, if possible, the robot transference while ensuring collision avoidance when in a workspace cluttered with obstacles.

3.) Manipulation level :
In *L.R. phase*, manipulation means the skills which should be chosen to perform the task determined at the task level.

2. (b) Assembly phase

The second phase of robotic assembly task is similar to the *L.R. phase*, which is divided into three levels with different types of jobs.

1.) Task level :
Similar to the *L.R. phase*, the task level here is to process the highest level decisions, such as determination of the assembly (part mating) strategy , and of the assembly sequence, etc.

2.) Trajectory level :
In the *assembly phase*, trajectory represents more delicate robotic motion than in *L.R. phase*, especially when it is a rigid arm. Some methods also have to be employed to avoid any damage such as collision.

3.) Manipulation level :
The mating operation is taken in this level. There may be lots of mating types, but we concentrate on the insertion operation for peg-in-hole type assembly, which is the majority in assembly processes.

2.2 Sensory Functions Required in Each Processing Phases

To provide helpful information for the operations described in the previous section, several observations should be made for an assembly process in each processing level.

The extremely critical information of the task level is the object identification In the trajectory level, we have to carry the part ready for assembly to the mating point (configuration), from which the mating process proceeds. Such mating point is usually determined by soft or compliant contact detected by force monitoring. In the manipulation level of the *L.R. phase*, both the buffering or grasping operations need the pose information (location and orientation) of the object. In the *A. phase*, a match of the contact pattern between some contact information and the modeled force pattern should be accurately reached in the manipulation level.

After the above discussion, we are now ready to propose an architecture of a multisensor system.

3 Multisensor System Architecture

Our multisensor system is hierarchically divided into three levels, the organization level, the coordination level, and feature extraction level. The organization level mainly deals with the planning tasks, the coordination level supervises all the data flow and command execution, and the feature extractor in the feature extraction level can be viewed as a complete sensor equipped with various functions.

A simple diagram that shows the hierarchical structure of the sensor system is given in Fig. 2.

The *organizer*, which is the highest level in the sensor system, is responsible for decomposing tasks instructed by the planner into a number of elementary subtasks, each is with some prior knowledge, and will then pass them as a sequence of input commands to its successor, namely, the *coordinator*. After execution of the given commands, the *coordinator* may or may not feedback the result or other information to the *organizer* for modifying or updating the knowledge base.

The *coordinator* is an intermediate level of mechanism serving as a supervisor of the feature extractors between the *organizer* and the HFE structure. Essentially, the way the *coordinator* executes the organizational commands is through passing message to the HFE structure and incidentally supervising the progress.

HEF(Hierarchical Feature Extraction) Structure is the lowest level in the hierarchical multisensor system, and is the practical interface with the working environment, which is originated from the logical sensor structure proposed by Henderson [6]. Here, we keep the main characteristics of the logical sensor structure, namely, the abstraction of sensor devices and equipment with algorithm module, and refine the conceptual development so as to lead to a feature extractor.

Each feature module in the HFE structure is an independent controller to its succeeding sub-feature extractors. There are command line and message channel connecting the feature module and its successors hierarchically. The lowest level module (feature extractor) in the HFE structure is the physical sensor device driver, which works with the simplest hardware control, i.e, to switch on, to switch off, or to get an image. The feature extraction modules which are connected hierarchically through command lines and message channels will then be called the HFE (hierarchical feature extractor) structure as a whole.

The benefit of using such structure is that when the *coordinator* requires a piece of specific information, it communicates with the appropriate feature module without concerns about the lower level or the lowest level of processing actions. The communication can be at any point in the structure, so that the whole HFE structure behaves just like a dedicated module. The HFE structure which incorporates the *coordinator* is depicted in Fig. 3.

3.1 Flexibility in HFE Structure

Here, we take the concept of the logical sensors (Henderson, 1988) to construct our feature extraction unit (of feature extractor). Each unit contains four parts, the input command, the communication, the processing program, and the selector part. They will be described below and is depicted in Fig. 4.

1.) Input command : which is used to actuate the unit. This enabling command may be a very simple one without any parameter attached, or with some instructive parameters like the buffer number to be selected to store the processed image data.

2.) Communication : which behaves as a specific I/O port of the feature extraction unit. The way of the communication can be a procedure invoking with call by reference parameters, or with a mailbox for sending message to and/or receiving messages from the other unit (remote procedures) in a distributed environment. These parameters to be passed are determined when the unit is created.

3.) Selector : which is to select the proper set of inputs. Each feature extraction unit contains two kinds of input, one is the kind of complementary sub-feature input and the other is the kind of competitive inputs. Generally, the selector is responsible for selecting a suitable input based on some specific criterion (e.g., within certain time bound) from a set of competitive inputs.

4.) Program : which constitutes the main part of the unit. The unit calculates the target feature from the set of inputs selected by the selector following the instructions given by the input commands which come either from the *coordinator* or from a parent feature extraction unit.

3.2 Implementation and Performance

We use object-oriented programming to singularize the flexibility and the extensibility of our sensor system. The "overload" characteristic of C++ enables the transparency of the invoking of sensing functions. We can use the same command name to invoke each feature extraction unit. The communication portion of a unit is easily performed by object pointers. A piece of programming codes of typical logical sensor followed the feature extraction unit (see Fig. 4) is shown below, and the performance is discussed at the end of the section.

```
class logical_unit {
 private data :
    character string : sensor_name[15];
    number of competitive subfeature : npara;
    number of complementary subfeature : nsub
     /* to record how many subfeaure input is
connected */
    subfeature pointers :
          logical_unit *p[max_sub_unit];
     /* the linkage to the subfeatures */
 public data and function :
   virtual actuating function : sensing;
     /* same actuating name in each feature
extractor */
 };
```

It's hard to evaluate the performance of a complicated sensor system, but we can investigate this problem more detailedly by analysis of the influence factors like response time and accuracy.

A logical sensor in the hierarchical structure can be modeled as

$$x_i = \Psi_i(x_{i1}, x_{i2}, \cdots, x_{ik}),$$

where Ψ_i is the uncertainty model of ith logical sensor, and x_{ij} is the information provided by the jth sub-logical sensor (successor) for constituting the result of the ith logical sensor, and

$$x_{ij} = \begin{cases} y_k & \text{if sensor } i \text{ is a pure algorithm module} \\ \tilde{y_k} & \text{if sensor } i \text{ is an atomic sensor} \end{cases}$$

where the *atomic sensor* means the lowest level nodes in the HFE structure ,i.e. the hardware device driver, it can be analyzed more delicately by

$$\tilde{y_k} = S_m(t, p)$$

where t is the transducer portion of the mth physical sensor, and p represents the sensing portion. For example, a CCD camera have an intensitive array-like CCD chip as the sensing portion, and with RS-170 as the transducer portion.

The cost of response time can be formularized by the cost function

$$cost_i^T(t_i) = \begin{cases} f_i(t_i) & \text{if } t_i \le t_i^c \\ \infty & \text{if } t_i > t_i^c \end{cases},$$

where t_i is the time period that s_i takes to respond with a command from the coordinator. The function f_i is monotonically increasing with respect to the time period t_i, and t_i^c is the hard time constraint that bounds the response time. And the average cost can be defined by

$$E[cost_i^T] = \begin{cases} \int_0^{t_i^c} f_i(t_i)p_i(t_i)dt_i & \text{if } P[t_i > t_i^c] = 0 \\ \infty & \text{otherwise} \end{cases},$$

where p_i represents the probability that the response time is equal to t_i. And the total cost of the responding time can be formulated as

$$Cost^T = \sum_{i=1}^{N} w_i E[cost_i^T],$$

where N is the total number of logical sensors, w_i is the weighting factor indicating the relative importance (or the rate of usage) of the corresponding logical sensor (feature extractor).

Similarly, the sensor uncertainty (inaccuracy) can be modeled in the same way

$$cost_k^U(\vec{x_k}, \vec{z_k}) = \begin{cases} g_k(\|\vec{x_k} - \vec{z_k}\|) & \text{if } |x_{kj} - z_{kj}| \le L_{kj}^c \\ \infty & \text{otherwise} \end{cases},$$

where $\vec{x_k}$ represents a vector of natural state, $\vec{z_k}$ represents the estimated vector, L_{kj}^c represents the hard error constraint. The average and the total cost of sensor uncertainty can be formulated as

$$cost_k^{\tilde{U}} = E[cost_k^U] = \begin{cases} \int_Z g_k(\|\vec{x_k} - \vec{z_k}\|)p_k(\vec{z_k})d\vec{z_k} \\ \qquad \text{if } \sum P[|x_{kj} - z_{kj}| > L_{kj}^c] = 0 \\ \infty \qquad \text{otherwise} \end{cases}$$

$$Cost^U = \sum_{k=1}^{N} \alpha_k cost_k^{\tilde{U}}$$

The hard constraints in both models represent the bound of respond time and error which would cause failure, for example, a part on a running conveyor belt would crash down without being grasped in time, or the opened jaws of a manipulator would run into a damage hit with the part if the pose error is too large.

We can use the performance evaluation analysis as a criterion to evaluate the efficiency of a sensor system, but it's difficult to tell what kind of architecture for a sensor system is better or worse. It depends on individual application task. But in the general environment of automatic robotic assembly task, the proposed architecture would works well and give the assembly designer confidence to follow with the rules. In our laboratory, several sensing units are implemented by Borland C++ with proposed structure. The sensing time is listed below (unit in seconds).

Feature	Average sensing time	Condition
Speed	4.97	speed of object = 30.97 mm/sec
Orient	0.84	
Location	0.29	
Obj. Pose	1.15	

It can be compared with the time of sensing with non-structured programming of sensor system, whose results are listed at the end of next chapter. It reveals that the sensing speed is slowed down by constructing the structure, but the performance is acceptable compared to the non-structured programming.

4 Experiment

In this section, we present an automatic robotic assembly cell which is set up in our Intelligent Robotic Laboratory.

4.1 System Configuration

The employed equipment in our assembly cell is depicted in Fig. 5. The main piece is the five-axis AdeptOne robot arm with all revolute joints. This robot is mainly responsible for the assembly process, whose reachable workspace covers almost the whole working range of the cell. Referring to the configuration shown in Fig. 5, the conveyor belt sitting in the middle of the work cell transports the assembly parts into the cell. It is the main transferring mechanism in our system.

Two pairs of optical switches are placed across the conveyor belt as our proximity sensors, which are responsible for monitoring the presence of a part at some location of the conveyor belt, and calculates the speed of the movement of the conveyor belt.

In the workspace, there are three CCD cameras connected to DT-2861 image processor board. They provide gray-level images with 512*512 pixel resolution of the scene. One of them is called overhead camera, which is located right above the pairs of optical switches in order to detect the presence of the incoming part. Another is called side camera, which is located right above the end of the conveyor belt in order to monitor all events which take place on the conveyor belt. The third one is the eye-in-hand camera, which is equipped on the end of the third link of the AdeptOne arm. The block diagram of the visual operation is depicted in Fig. 6.

A force/torque sensor, Lord FT-75/250, is equipped onto the wrist of the AdeptOne arm, which is shown in Fig. 7. It detects the force and torque information as a contact sensor, and can analyze force and torque in Cartesian coordinate.

The image data and the signals of the optical switches are connected to a PC-486 through some hard-wires communication time seems to dominate the overall computation task in the assembly process. The AdeptOne robot is under control from a dedicated robot controller.

4.2 System Control Scheme

The functional architecture of the whole assembly system is similar to the sensor system, that is described in the last section. It contains an assembly planner or state monitor, and a task coordinator which instructs the other two execution modules, the sensor system and the robot motion system. The block diagram is shown in Fig. 8.

Similar to the coordinator in a sensor system (see Fig. 2), the assembly task coordinator mainly deals with how to interpret the input commands and how to supervise various modules. In the last section, the organizer of the sensor system receives commands from the planner and issues the translated instructions to the lower modules, but there the planner in fact consists of a real assembly planner and a task coordinator, as shown in Fig. 8, in a practical assembly system. The combination of the assembly planner and the task coordinator instructs the sensor system to fetch the necessary features. After that, the task coordinator would evaluate or judge the fetched features, and then command the robot motion system to complete a process. In Fig. 8, the sensor system may fetch the information either from the environment or directly from the investigation of the result of robot motion. By the cooperation between the sensor system and the robot controller, the assembly process is completed by this closed-loop control scheme.

The starting point of the assembly work is in the task coordinator, wherefrom the data, after inference, flow into the planner, sensor system, and the robot motion system. The task coordinator plays the role of the central supervisor to negotiate among various modules. We proceed to describe the details of the process flow in the *L.R. phase* and the *A. phase* introduced in section 2 according to the processing order.

4.2.1 Location/Recognition Phase

In the *L.R. phase*, we complete all the work except the mating part. The robot motion module will perform dynamic grasping of the incoming part on the running conveyor belt, carry the part to a designated location in order to perform a more precise

grasping, and then finally carry the precisely held part into the assembly site. These jobs are shown in Fig. 9.

After determining the speed of the moving object, the robot arm can track it along the conveyor belt and smoothly grasp the object which is with slight friction with the belt. The dynamic grasping procedure is depicted in Fig. 10.

Before the tracking operation, we have to find the precise 3D location of the object center and the orientation of the object. In our experiment, we simply use the 2D area information, which comes from the overhead camera, to recognize the incoming part and the height of this part which can be easily got from the knowledge base.

4.2.2 Assembly Phase

The parts used in the assembly task along with the fixture board are pictured in Fig. 11, which includes a base, a mainbody, a slide bar, and a round peg, which have to be mated together in a specific order.

The flow chart of the assembly process is depicted in Fig. 13, where the thin lines with arrows represent the processing flow and the bold lines with arrows relate the working process with the system supporting modules. For example, the '*Take to the mating point*' process is supported by both the sensor system module and the robot motion module, and these two modules are coordinated by the task coordinator. And the flow chart also shows that 'Take to the mating point' is the second step after '*Carry into assembly cell*'.

Take to the mating point

In a static environment, the mating points can be well defined prior to the processing. However, in a dynamic environment, it needs to employ some locating techniques with visual or contact sensors.

4.3 Results

The assembly task performed by the 5-axis AdeptOne robot described in section 4.2 is implemented with a special robot language, System V, on AdeptOne supervisor and Borland C++ on PC-486.

The assembly time is related to robot motion speed under the consideration of robot accuracy and safety. The average and the worst assembly time of three parts is listed below (unit in seconds).

	Total time	Pure mating time (sec)
mainbody	33.471	12.057
	*35.214	*13.086
slide bar	46.929	24.986
	*47.729	*25.786
round peg	14.907	6.425
	*32.286(3)	*22.600(3)
	*36.229(4)	*26.557(4)

The total time counts from the instant of part's appearance on the conveyor belt to the instant of completion of the assembly task. The numbers prefixed by asterisk (*) denote the time obtained for the worst case, but those without the prefixes denote the average time. In the last row with round peg, (n) means that square searching strategy is applied up to n counts. Clearly, we can see that it needs about 4.42 seconds for one more trial count.

The sensing time is listed below, and we can compare the result with the one listed in the last section, which is implemented with structured sensor system architecture. It reveals that the structuralization and the systemization seem to cost more (in time) than a simple intelligent integrated piece of programming.

Sensing term	*Sensing time (sec)*
Speed	4.943
area, orientation and location	0.856
Refining scan	4.848

5 Conclusion

Sensors incorporated in robotic assembly cells usually suffered from the problems of nonstructured management which results in lack of reusability and extensibility. We analyze the robotic assembly task into several processing levels and propose a multisensor system architecture with associated feature templates for related usage in corresponding levels. A structured sensor system with benefit of reusability, flexibility or extensibility is introduced and implemented.

References

[1] John W. Cook, "Applying Sensors to Automatic Assembly Systems", *IEEE transactions on Industry Applications. Vol. 27, No. 2*, March/April 1991

[2] Rex Maus and RandallAllsup, "Robotics : A Manager's Guide", John Wiley and Sons, Inc. 1986

[3] M. Myrup Andreasen, S. Kahler, T.Lund "Design for Assembly", IFS Publications, UK 1988

[4] IEEE "Proceedings ofInternational Conference on Computer Integrated Manufacturing", Rensselaer Polytechnic Institute, Troy, New York, May 1988

[5] IEEE "Proceedings of International Conference on Computer Integrated Manufacturing", Rensselaer Polytechnic Institute, Troy, New York, May 1992

[6] Tom Henderson and Esther Shilcrat, "Logical Sensor Systems", *Journal of Robotic Systems*, Vol. 1, No. 2, pp.169-193, 1984

[7] Norbert Roth and Peter Mengel, "Sensor Integration - Getting The Whole Picture", *Sensor Review*, Vol. 12, No. 1, pp.28-33, 1992

[8] Mohan Manubhai Trivedi, Mongi A. Abidi, Richard O. Eason, Ralph C. Gonzalez, "Developing Robotic Systems with Multiple Sensors", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 6, pp. 1285-1300, November/December 1990

[9] M. A. Abidi, R. C. Gonzalez, "The Use of Multisensor Data for Robotic Applications", *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 2, pp.159-177, April 1990

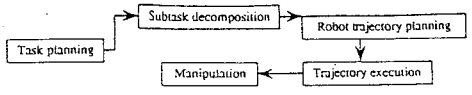

Figure 1: Robotic Assembly Control Flow

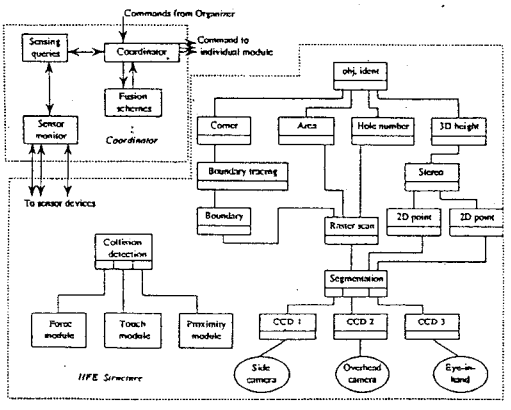

Figure 3: The Coordinator and the HFE structure

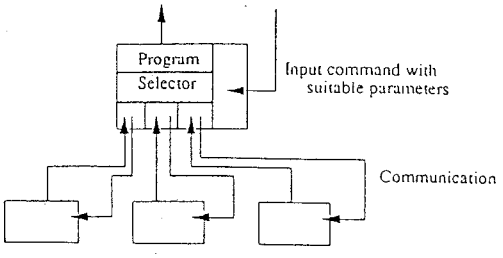

Figure 4: Feature Extraction Unit in HFE Structure

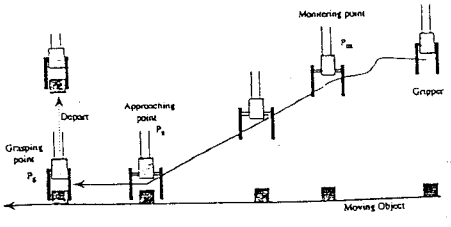

Figure 10: Dynamic Grasping of Moving Object

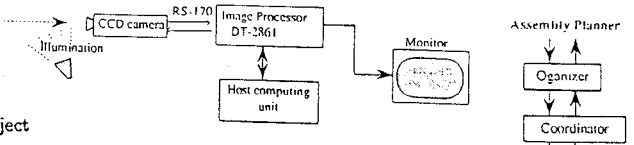

Figure 6: Camera Module

Figure 2: Hierarchical Sensor System

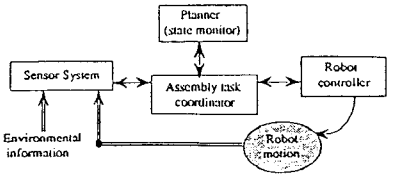

Figure 8: The Block Diagram of the Assembly System

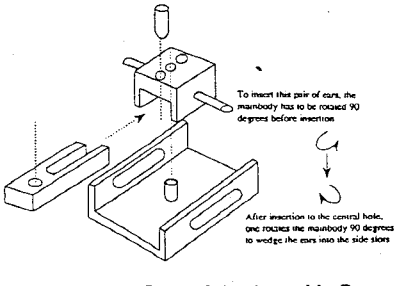

Figure 12: The Mating Steps of the Assembly Part

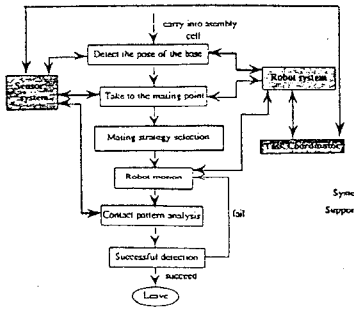

Figure 13: Assembly Flow Chart

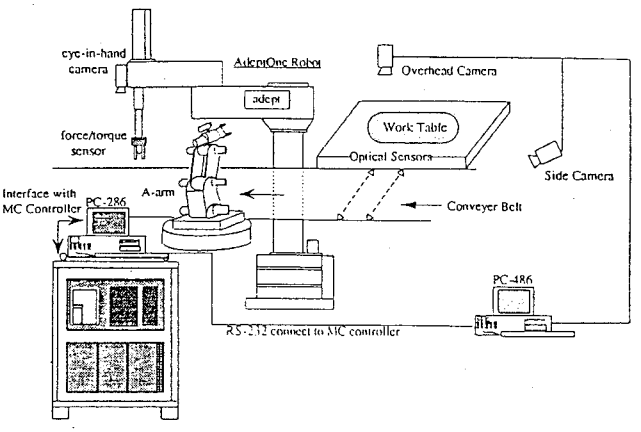

Figure 5: Configuration of the Assembly Cell

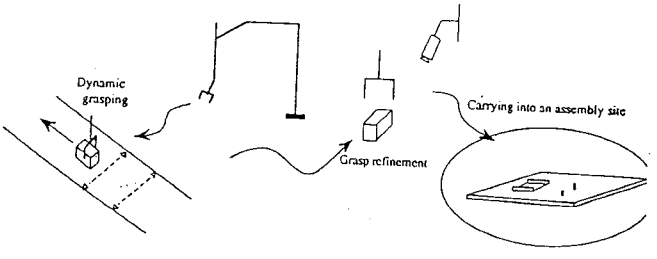

Figure 9: Working Situation of a Robot in the *L.R.phase*

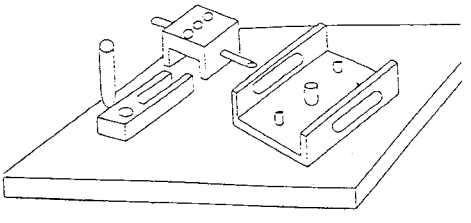

Figure 11: The Assembly Parts and the Fixture Board

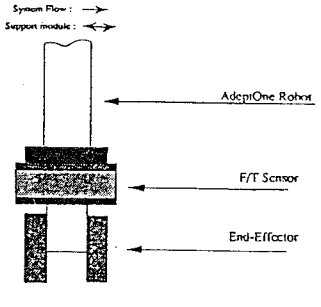

Figure 7: Wrist Force/Torque Sensor Configuration