Abstract

This report reviews ongoing research in modeling and control of assembly systems. A modeling framework is developed for representing assembly tasks as a collection of discrete operations with precedence relations reflecting physical constraints. Devices are mapped into the subtask description leading to an *operation precedence graph* which is useful for analyzing alternative system configurations and supervisory control structures. Underlying continuous processes determine the tolerance requirements for accomplishing the subtasks. Since reducing uncertainty in the system configuration is the fundamental objective in assembly tasks, two alternatives for representing uncertainty are defined and illustrated by examples. These representations of uncertainty are discussed with respect to alternative feedback control strategies for satisfying tolerance constraints. The concluding section identifies directions for future research.

1 Introduction

As assembly automation systems become more complex, the analysis and design of these systems requires more sophisticated tools to achieve desired performance, including speed, reliability, and flexibility. Current research efforts in the Flexible Assembly Laboratory at CMU are focussed on developing representation and modeling tools to be used as a basis for automated planning, design, and programming of assembly systems and their supervisory controls. This report reviews several aspects of this research in progress, and describes the components and representation of the assembly task which lends itself to a supervisory control structure expressed in terms of discrete events and continuous space task constraints.

Automated assembly has been studied from a variety of different perspectives. Assembly requires acquisition, orientation, and mating of parts in a predefined pattern. Engineering approaches to assembly have focussed on the development of mechanisms and devices to accomplish these goals [1],[2]. A number of studies in the robotics literature [3] have viewed assembly from the robot planning point of view as an extension of blocks of world problems in AI. Task-oriented programming languages [4], [5] have viewed assembly as a hierarchy of discrete operations. This type of hierarchical representation was used by Sanderson and Perry to describe assembly work cell configurations.

None of these approaches provide a level of description which easily supports the simulation and evaluation of control structures in a real-time sense. In particular, the tradeoffs between speed, reliability, and flexibility are not accessible in conventional representations. Reliability of assembly is achieved by constraining or sensing part positions and may be modeled by various uncertainty measures. We have previously described the use of entropy measures [7], [8] to characterize the reduction of uncertainty. In this report, we introduce the use of tolerance sets as a method for modeling uncertainty which lends itself to real-time modeling applications and the evaluation of sensor-based control loops in relation to speed and reliability.

This report is organized as follows. Section 2 describes a high-level view of assembly tasks and systems in which discrete operations are the basic modeling units. Much of the system organization and supervisory control structure is determined by breaking down the task into discrete operations. The actual assembly process evolves in continuous space and time, and the continuous aspects of the assembly problem are discussed in Section 3. In particular, task constraints in continuous space are defined in terms of tolerance sets* and methods for modeling uncertainty in the system configuration are introduced. Section 4 focuses on the issue of sensory feedback for supervisory

control of assembly systems. The evolution of uncertainty as the process progresses in time is addressed with respect to alternatives for integrating real-time sensory information into the control loop. Directions for future research are summarized in the concluding section.

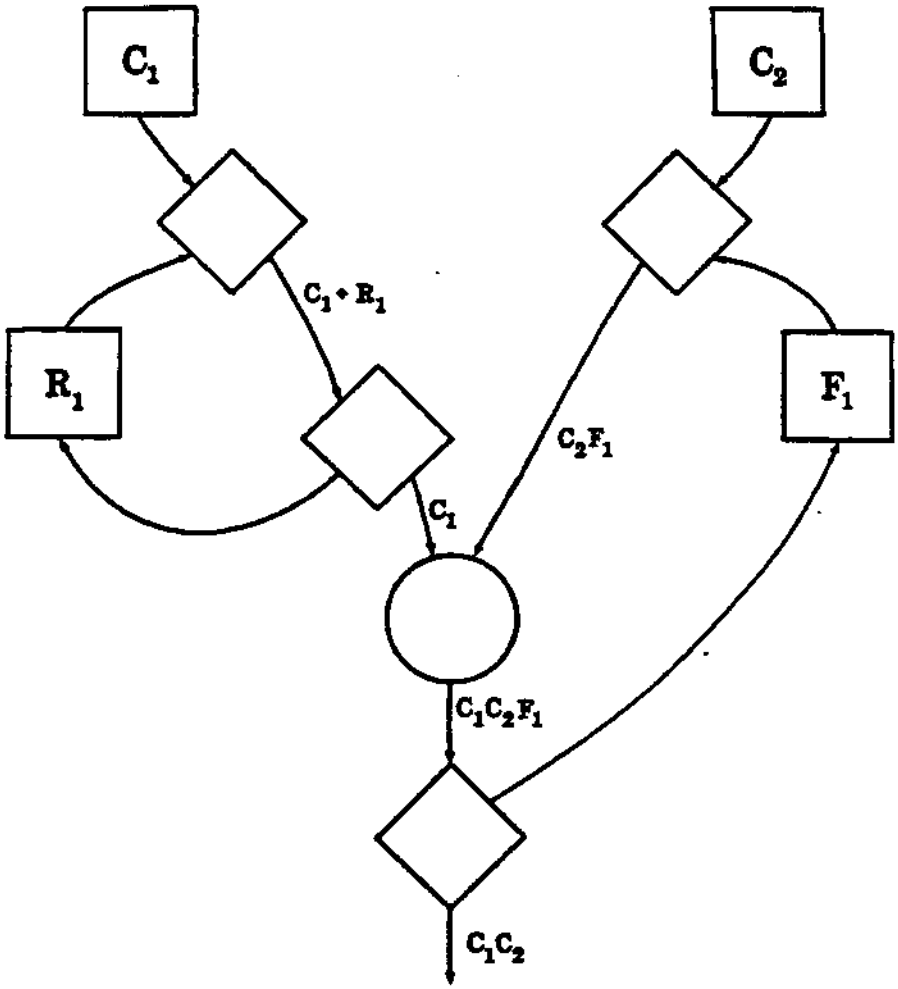
2 Discrete Operation Models of Assembly Systems

This section presents a description of assembly tasks and systems in terms of discrete operations involving the assembly components and system devices. For a given assembly task, a graph is drawn representing the precedence relations among the operations. This model is used as a framework for relating discrete task descriptions to underlying continuous processes, and to formulate the problem of selecting system devices to execute the task.

Terms used in this section are defined as follows:

device	Physical element in the assembly *system* such as a fixture, gripper, tool, etc.
component	An individual part in the assembled *product*.
subassembly	An ensemble of connected components for which the configuration can be specified by less parameters than required to specify the configurations of the individual components.
entity	An independent object at some point in the assembly procedure. An entity may be *simple*, that is, an individual component or device; or *compound*, that is, a collection of attached devices and/or components (subassemblies) which are identified as a single object.

An assembled *product* is composed of a set of individual *components* connected according to a specified set of geometric relations; any subset of connected components is a *subassembly*. A device is an element of the assembly *system*, not the *product*. An *entity* may be any mutually attached subset of components or devices; entities may be *merged* or *split* by an operation. A *subtask* is an *operation* which involves only components, not devices. In this representation, the assembly task is described by a progression of compound entities. Devices are added and withdrawn from these progressive compound entities as successive operations are performed. Figure 1 is a schematic description of an assembly task using these definitions. As discussed in the next section, it is often convenient to describe the task in terms of component-component operations or *subtasks*, then map device assignments into the subtask *precedence* graph as a separate step.



C_1 , C_2 - components

R_i , F_i - dericei

O • difici • component operation

O - component * component operation

Figu re 1: Schematic description of assembly tasks

2.1 Task Description

We first describe the assembly task independent of the system devices which accomplish the assembly. In this description, the assembly a sequence of operations involving components and subassemblies. A decomposition into subtasks, each involving only two subassemblies are used here. We assume there is at least one such decomposition of the task into subtasks.

The physical constraints of the assembly impose precedence relationships on the sequence of subtasks since physical access restricts the order in which parts may be mated. As an example, consider the flashlight assembly in fig. 2 where the subtasks are identified as $T_1, \ldots, T_7$. Any sequence of steps to assemble the flashlight consists of these discrete subtasks, each of which can be defined in terms of an operation involving two components or subassemblies. Certain subtasks must be performed before other subtasks. For example, the bulb must be inserted into the reflector (T_2), and the lens into the cap (T_1) before the reflector-bulb subassembly can be inserted into the cap-lens subassembly (T_3).

There are, however, options in the ordering of some of the subtasks for the flashlight assembly. This is illustrated in fig. 3 which shows a *subtask precedence graph* (SPG) for assembly sequences for the flashlight, where the graph nodes are subtasks and the directed branches indicate the fixed precedence relations among the subtasks. The point at which each component enters the assembly is also indicated. These graphs represent the differences in the precedence relations when the end is connected to the casing first, fig. 3(a), vs. first connecting the cap-reflector subassembly to the case, fig. 3(b). From these graphs, it is evident that the selection of a particular assembly sequence with its corresponding precedence relations imposes constraints on the structure of the system which will accomplish the assembly. For example, the amount of parallelism in the precedence graph dictates the extent to which the assembly can be decomposed into simultaneous operations at independent stations. The specific ordering of subtasks also has implications for the device operations including fixturing, manipulation, and sensing required to do each step of the assembly.

As an analytical tool for representing SPG's we define the subtask precedence matrix (SPM) P with elements defined as

$$P_{ij} = \begin{cases} 1 & \text{if } T_j \text{ immediately precedes } T_i \text{ in the SPG} \\ 0 & \text{otherwise.} \end{cases}$$

For example, the SPM corresponding to the end-first assembly sequence SPG in fig. 3a is given in fig. 4. The SPM is useful for generating certain properties of the assembly sequence. Let the components of the *task state vectors* t be defined by

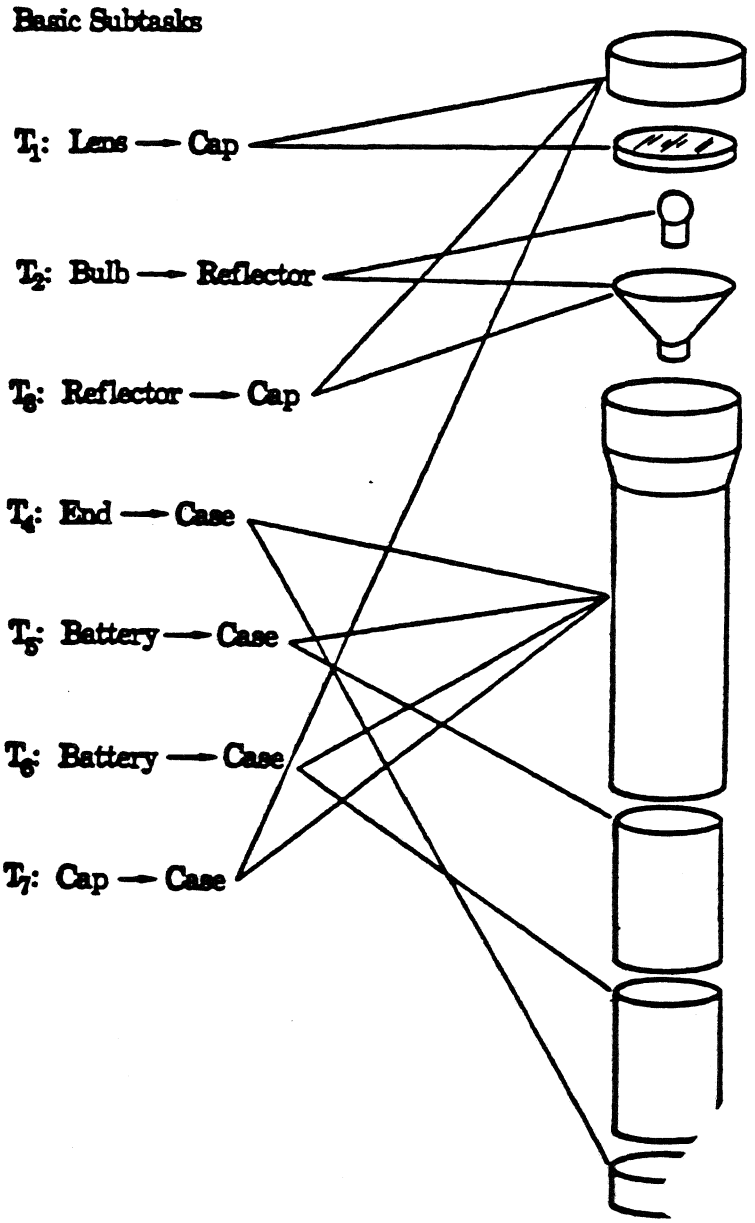


Figure 2: Flashlight assembly subtasks

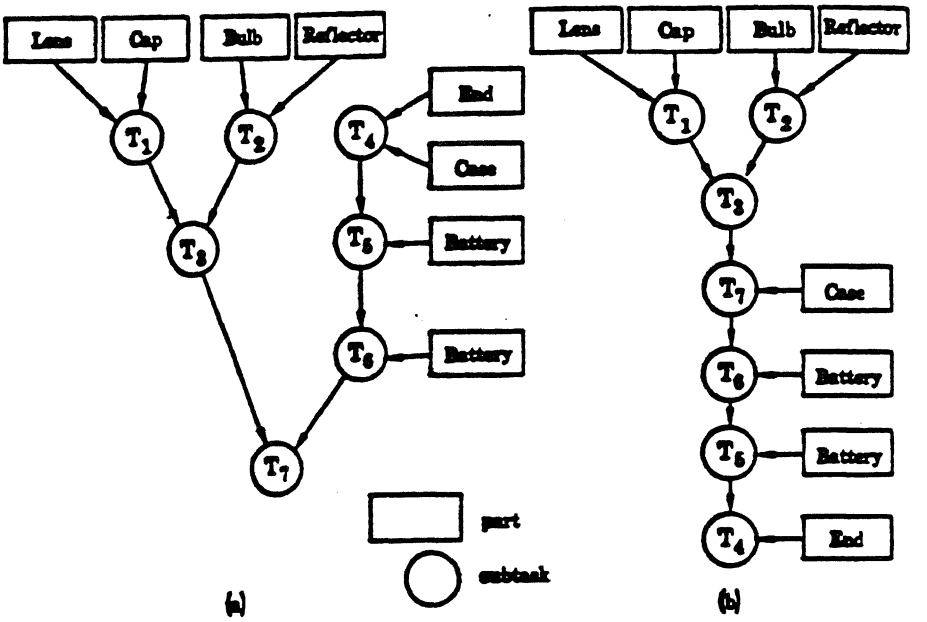


Figure 3: Two alternative subtask precedence graphs for the flashlight assembly

```
      \ancestor:1 2 3 4 5 6 7
subtask\
   1            0 0 0 0 0 0 0
   2            0 0 0 0 0 0 0
   3            1 1 0 0 0 0 0
   4            0 0 0 0 0 0 0   = P
   5            0 0 0 1 0 0 0
   6            0 0 0 0 1 0 0
   7            0 0 1 0 0 1 0
```

Figure 4: Subtask precedence matrix for SPG in fig. 3a

$$t_i = \begin{cases} 1 & \text{if subtask } T_i \text{ is completed} \\ 0 & \text{if subtask } T_i \text{ is not completed.} \end{cases}$$

At the task level, t represents the current state of the assembly process.

The following useful properties are derived from the above definitions:

Property 1 t is a valid task state if and only if $tP \wedge t^c = 0$.

Proof: t is a valid task state if f for each $t_j = 1$(subtask T_j completed.) Since the SPG is a tree, each column of P has exactly one nonzero element. Thus, the nonzero elements of the vector tP correspond to the ancestors of all completed tasks. This implies t is valid if f $t_k = 1$. For each nonzero element of tP, that is, $tP \wedge t^c = 0$.[1]

Property 2 The set of subtasks immediately precedent to task state t is given by the set of nonzero components of $tP + t$.[2]

Proof: The nonzero elements of tP correspond to the ancestors of all completed tasks. The set of tasks which could have been most recently completed are the tasks which are not ancestors to any completed tasks. This set corresponds to the nonzero elements of tP + t.

Property 3 Subtasks T_i can be completed immediately following state t if and only if $t_i^c \cdot \prod_{j=i}^{n} (P_{ij} t_j)^c = 1$

Proof: T_i can be completed iff t_i = 0 and all ancestors of T_i are completed, that is $t_j = 1$ if $P_{ij} = 1$. Thus, T_i can be completed iff $t_i^c = 1$ and $P_{ij}\ t_j^c = 0$ for all js i.e., iff $t_i^c \prod_{j=i}^{n} (P_{ij} t_j^c)^c = 1$.

From the SPG it is clear that all binary vectors t do not correspond to a feasible task state. Property 1 gives an explicit test for a valid task state in terms of the SPM. Properties 2 and 3 can be used to generate all possible task states. Property 2 gives an expression for finding the complete set of possible *precedence states* for a given state t. Thus, starting with the final state t = [1,...,1] all possible state sequences can be generated backward in time. Similarly, Property 3 gives an expression for computing the *subsequent states* which can immediately follow a given state t.

The number of possible task states for an SPG can be computed by the following algorithm:

1. Assign a weight of 2 to each subtask node which is a "leaf" in the SPG (note the SPG is always a *tree*).

---

[1] t^c is defined as the one's complement of the vector t, ' denotes vector (or matrix) transpose, and indicates component-wise and of the vector arguments.

[2] Binary vector addition is defined componentwise with 1 + 1 = 0 in properties 2 and 3 (i.e., + = exclusive or).

2. The weight assigned to each non-leaf node is equal to the product of the weights from the precedent nodes plus one.

3. Progressing tfirough the branches of the tree, the total number of possible states for the SPG is equal to the weight of the root node, which corresponds to the final subtask in the assembly sequence.

Applying these steps to the SPGs in fig. 3 gives 21 and 9 possible task states for the assembly sequences in fig. 3(a) and fig. 3(b) respectively. In this example the more parallel structure of end-first SPG leads to a higher number of possible states.

At the task level, assembly system design involves the evaluation of alternative precedence graphs, for a given task. Automatic generation and evaluation of alternative precedence graphs based on a geometric description of the assembly would be a useful tool for task planning and system design. As illustrated in fig. 5, a computer program for this purpose would abstract from a CAD database the information necessary for generating the alternative precedence graphs. Methods must be developed to efficiently generate acceptable assembly sequences that meet system design requirements. Currently, research is underway to determine criteria for selecting alternative precedence graphs. These criteria will involve aspects of the more detailed assembly/task models described in the following sections.

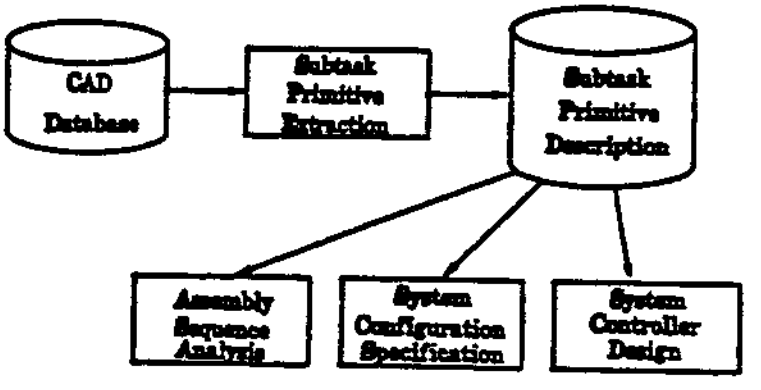


Figure **5:** Use of **suNask** description primitives for Design

2.2 Operation Description

The SPG representation in section 2.1 describes only the joining of components as subtasks of the overall assembly task. Design of the assembly system involves assigning devices to each subtask and implementing a control algorithm to sequence and synchronize the discrete operations. Assigning of devices introduces additional types of operations. Each operation can be classified as one of the following types:

C-C Operation: An operation joining or separating two components or subassemblies. The C-C (component-component) operations are the subtasks defined in the previous section.

D-C Operation: An operation involving the joining or separation of a device and a component, such as grasping, fixturing, etc.

D-D Operation An operation between two devices, as when a manipulator acquires a tool.

A particular implementation of the assembly system results in a sequence of *operations* with a precedence graph which includes the D-C and D-D operations. Figure 6 illustrates an operation precedence graph (OPG) for the implementation of the lens-cap subassembly for the flashlight in figure 2. The point at which specific devices enter and leave the assembly process are indicated on this graph. We note the following relationships between the OPG and the corresponding part of the SPG (fig. 3) for this subassembly:

- The basic structure of the SPG is obtained when only the C-C nodes are retained in the OPG.
- In the OPG all D-C operations appear as nodes along *branches* of the original SPG.
- D-D operations introduce new branches to the original SPG.

The OPG provides a structure for addressing several design issues, including:

Performance evaluation. For a fixed sequence and assignment of devices, the operation time for the overall assembly process can be evaluated based on times for each of the operations. Stochastic models can also be used at this level for average performance analysis.

Resource Allocation. If there are alternatives for real-time or off-line assignment of devices or the sources of components to the operations, these options can be evaluated to optimize the overall assembly time. Graphically, alternative assignments might be represented as shown in fig.7 where incoming simple entities can be selected from a predetermined set of alternatives for each operation.

Supervisory Control Design. At the highest level, the control of the assembly system involves initiating the operations in the correct sequence. The OPG is a useful

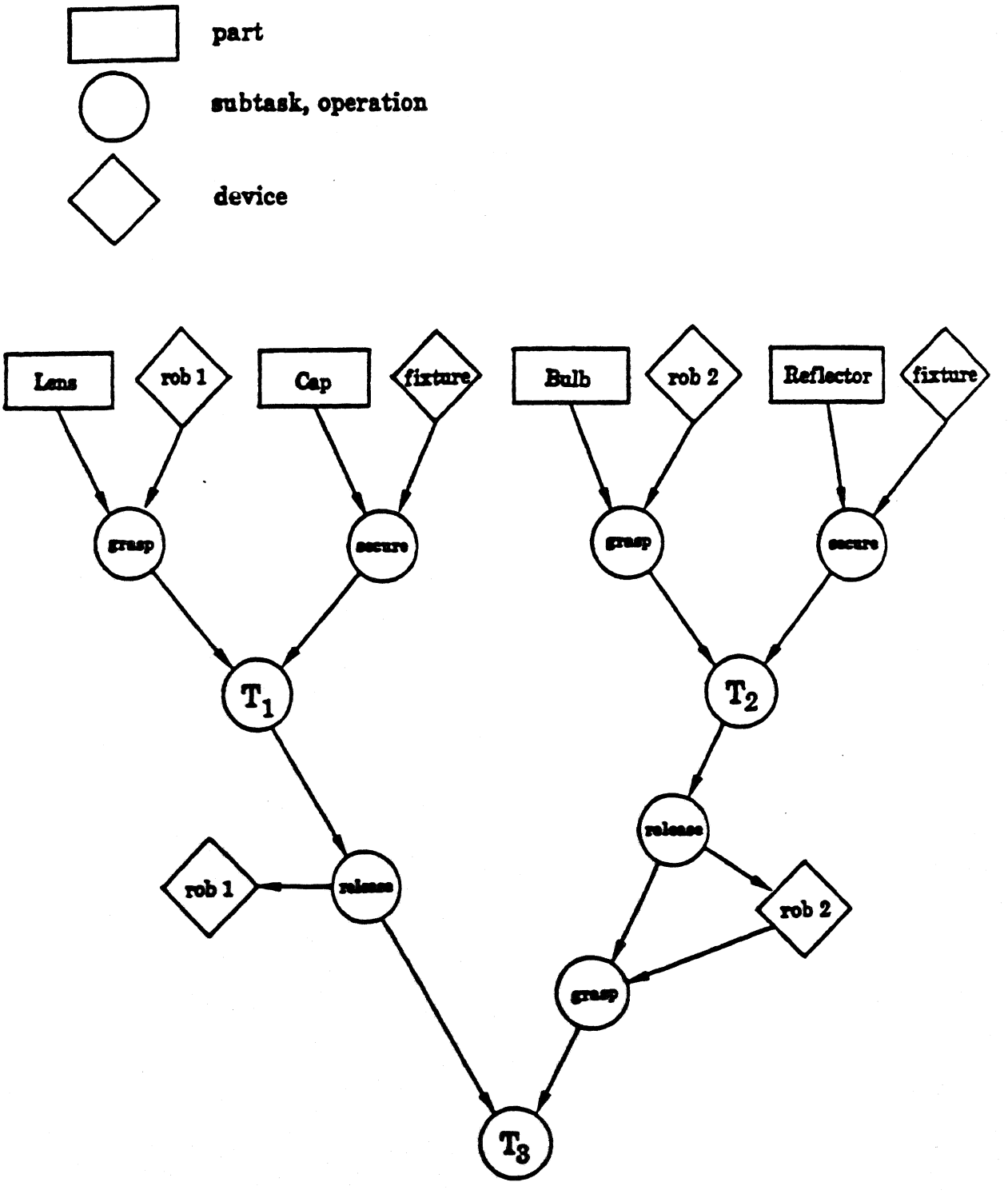


Figure 6: Operations precedence graph for flashlight lens-cap subassembly

representation of the system at this level for identifying and monitoring the state of the system. This provides a context for modeling the dynamic resource allocation problem. Furthermore, the structure of the OPG is directly related to the required structure of the supervisory control. For example, distributed monitoring and control can be implemented for independent operations on parallel branches of the OPG.

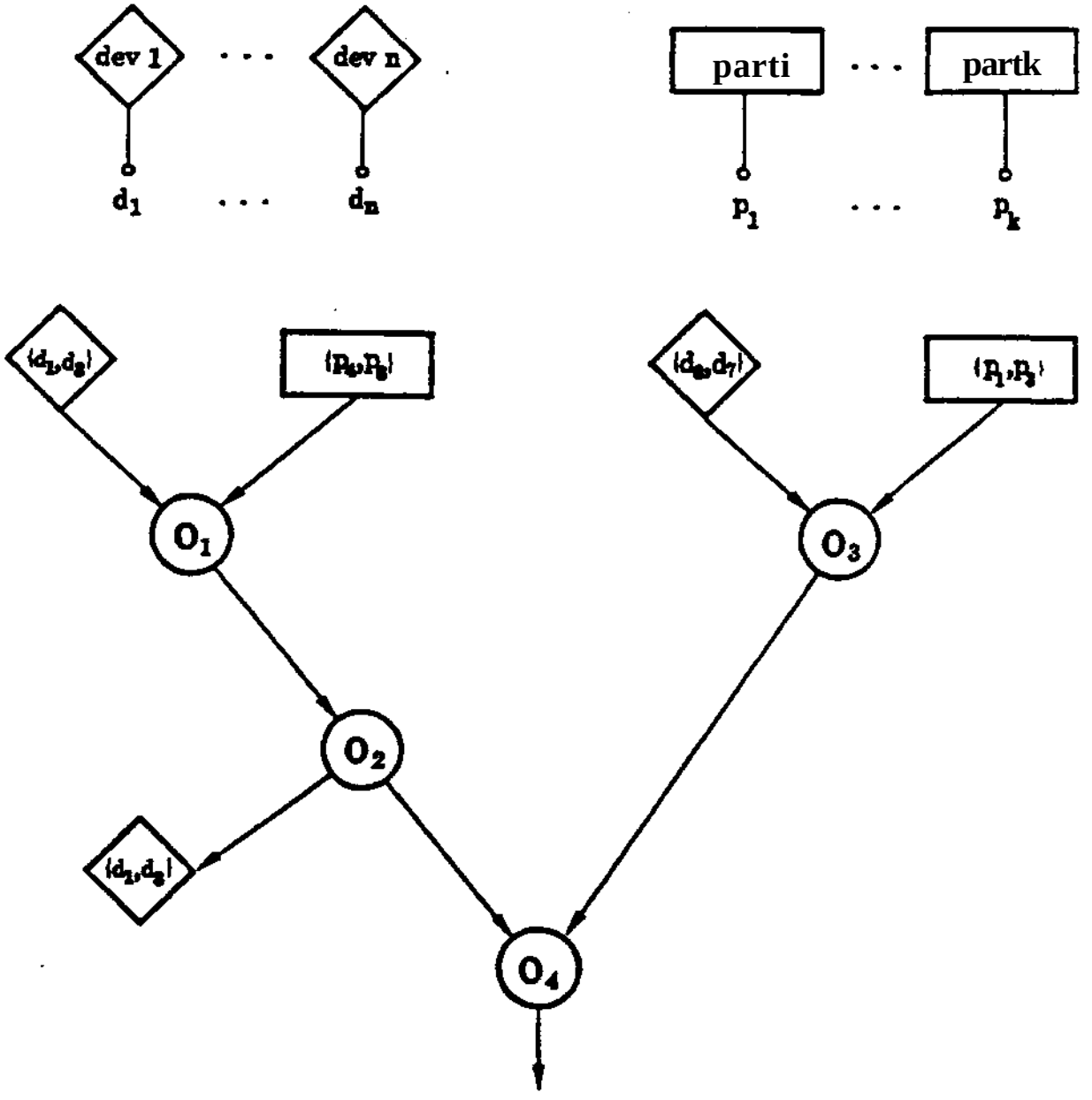


Figure 7: Representation of alternative entity assignments

3 Modeling Continuous Processes in Flexible Assembly

The operations precedence graph in section 2.2 describes the assembly process as a partially ordered set of *discrete* operations. This level of description is adequate for planning and evaluating the feasibility of an assembly strategy, but does not describe the process in sufficient detail.to study alternative sensing and control strategies to optimize dynamic performance and reliability. For these purposes we introduce, in this section, a continuous-parameter process-level description of assembly operations to represent the propagation of uncertainty as the assembly task progresses.

As described in the previous section, the execution of assembly tasks can be decomposed into discrete operations, each involving two entities. The objective of sensing, computation, and manipulation in assembly is to assure the tolerance requirements are satisfied for each discrete operation. Thus, the assembly process can be viewed as a sequence of continuous-parameter operations which are defined by relationships between the configurations of pairs of entities.

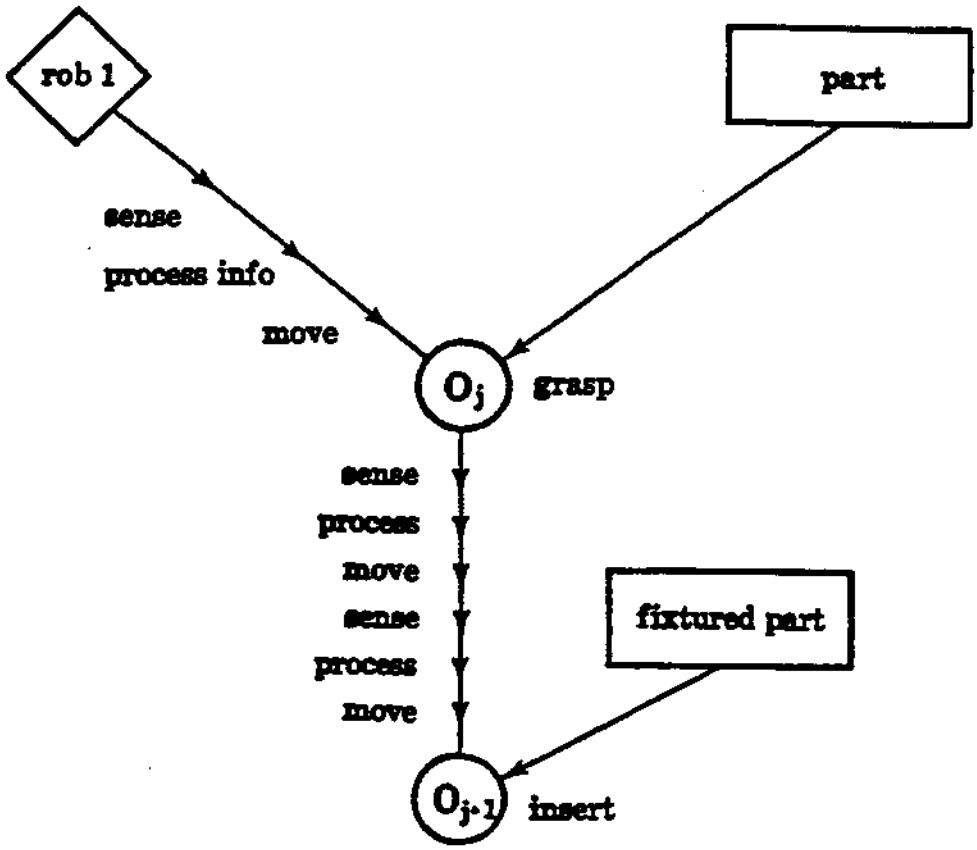
In the following section the continuous configuration space and tolerance set for a two-entity operation are defined and illustrated by a simple example. Since the objective of the sensing, control and manipulation in assembly is to eliminate uncertainty in the system configuration, a fundamental modeling issue is the representation of uncertainty. In section 3.2, alternative approaches to modeling uncertainty in the system configuration are discussed and illustrated.

3.1 Configuration Space and Tolerance Sets

In the context of the OPG, the continuous processes in assembly task execution occur along the *branches* between operation nodes of the graph, as shown in fig. 8. The processes along each branch can be described in the *configuration space* for the two entities involved in the operation. Let the configuration (position and orientation) of entity / be specified by the vector $c_i \in C_i$ where C_i is the set of configurations for entity /. For a discrete operation involving entities /˙and / the *tolerance set* $T(ZC = C_i \times C_j$ is the set of configurations for which the operation can be successfully performed. The goal of the continuous processes along each of the branches in the OPG is to obtain a configuration for the pair of entities which is in the tolerance set for the target operation node.

The problem of performing an assembly operation can be stated formally as: *(Given an Initial configuration* $c^\circ = (c^o_{|B} \; c^o_j) \in C$ *coordinate the manipulation, sensing and computational processes to establish a final configuration* $/ = (Z_i \; d_j \in 7)$.

For example, consider the operation of grasping an object where the uncertainty in position of the



FIgu re 8: Continuous processes between discrete operations

object must fall within the grasping envelope of the hand for it to successfully acquire the object. This operation ts illustrated in fig. 9: a gripper which has a maximum opening *g* is to acquire a part of width a. The tolerance set is determined by the grasping envelope *g* and the part size (a).

Specifying the configuration in terms of the horizontal positions ($*_r *_2$) of the gripper and part, respectively, the tolerance set in the two-dimensional configuration space is given by

$$T=\{(x_1,x_2) \mid |x_1-x_2|<0.5(g-a)\}. \quad (1)$$

The set *T* is shown in fig. 10 along with the configuration point ($\overset{0}{X}_r x^\wedge$ corresponding to the configuration of fig. 9. A similar tolerance relation occurs for parts mating problems where the mating envelope of the parts determines the tolerance set in the joint configuration space for the parts.

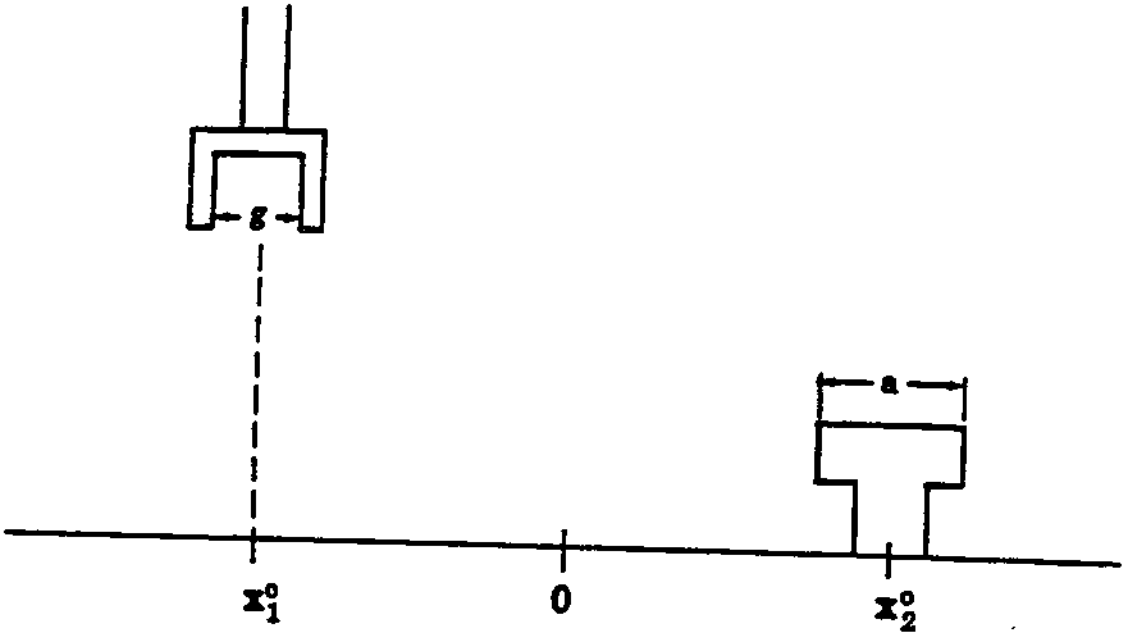


Figure 9: Operation example: grasping a part

This example illustrates how the specific operations and entities in the discrete description dictate the constraints and objectives for the continuous processes. Each node of an OPG corresponds to a tolerance *set* in the configuration space of the two entities involved in the operation. The objective of the sensing, manipulation and other processes which occur along the branches leading to the successful execution of the operation is described in terms of this set. Given a set of operations and devices, the tolerance sets map continuous-space constraints onto the OPG, and provide the primaiy point of connection between the continuous parameter and discrete operation descriptions cf assembly systems.

It is difficult to formulate a configuration space description of the tolerance set for most red assembly operations. Several issues must be addressed in order to develop a modeling methodology which adequately represents the physical constraints while admitting computationally feasible solutions to the design and control problems arising in assembly automation. This approach leads to a number of research issues, including:

- methods for extracting tolerance information from CAD database representations of assembly components and devices,
- efficient schemes for representing complex spatial constraints,

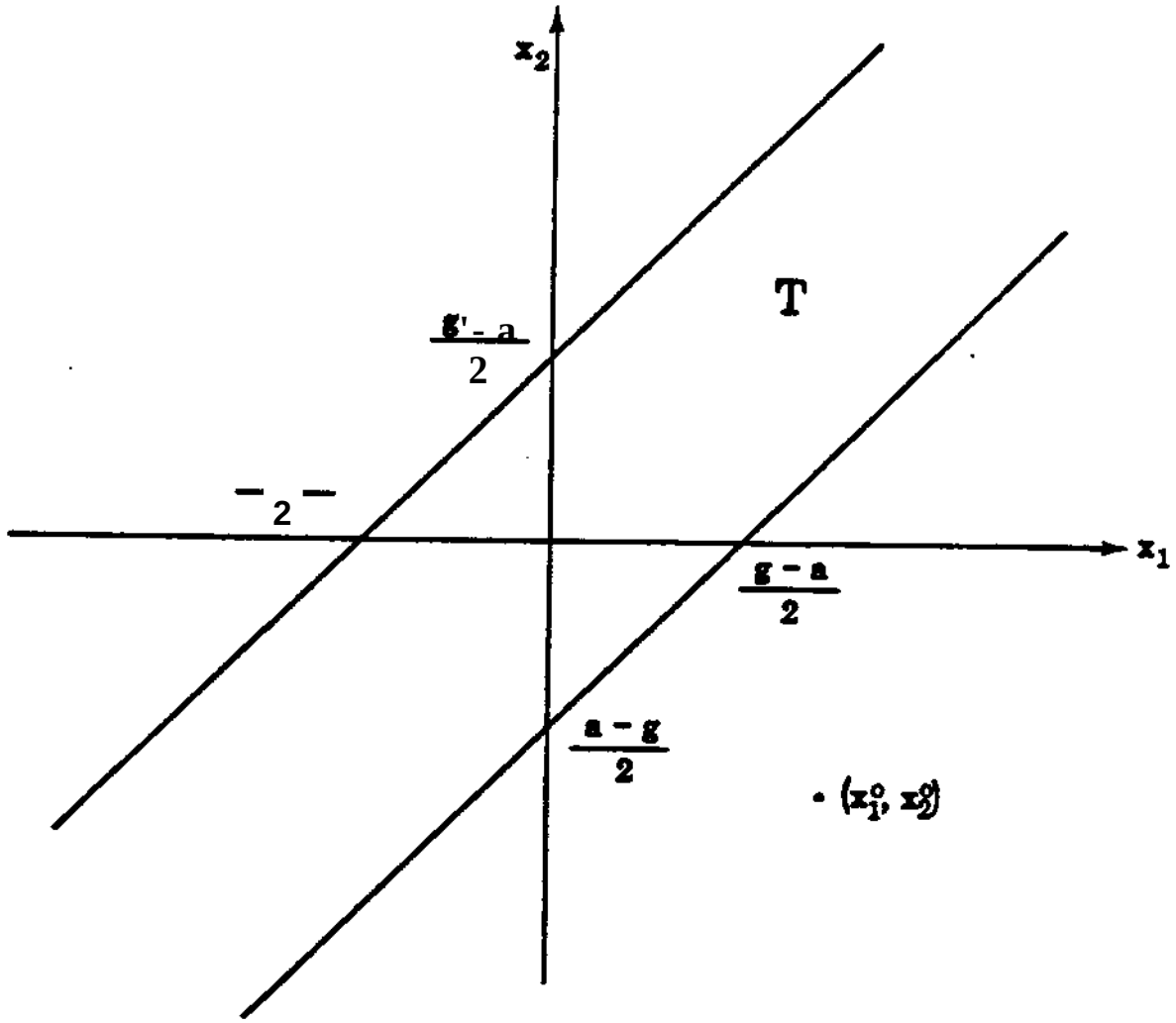


Figure 10: Tolerance set for operation of fig. 9

- "orthogonal" parameterizations of geometric models which permit decompositions of tolerance specifications into several independent coordinates, and
- approximation methods for simplifying configuration and tolerance specifications while guaranteeing the integrity of the model.

3.2 Modeling Uncertainty

As described in section 3.1, the control objective in assembly is to bring the joint configurations of entity-pairs into specified tolerance sets for the discrete operations. The formal specification of this problem for each operation involves the continuous configuration space C and the tolerance subset T. A hard automation system for executing an assembly operation uses fixtures and devices to constrain the configurations of the entities so that a fixed sequence of actuator commands guarantees the tolerance requirements are met. For hard automation, sensors are used merely to indicate success or failure of the operation.

In *flexible* assembly systems, the variety of parts to be handled, and assemblies to be produced dictate that real-time sensory feedback must be used to eliminate uncertainties in the positions and orientations of the entities. For system design and control, it is necessary to quantitatively represent this uncertainty as well as the information available from actuators, sensors, and computational processes in the system. The set of actuators could include simple relays and solenoids as well as sophisticated manipulators with local servo-controllers. Similarly, the set of sensors could include limit switches as well as vision systems with independent processors. Computations for estimating the system configuration can range from simple scaling operations to sophisticated recursive algorithms for updating an internal geometric model.

There are several approaches to modeling uncertainty, including bounding sets, probability distributions, fuzzy sets, and symbolic representations for rule-based reasoning. Each representation is suited to a particular class of problems. In the design of assembly systems, since tolerances are of primary importance, we propose the use of bounding sets as a natural representation. On the other hand, for real-time decision and control it may be useful to include some type of probabilistic information to implement efficient search algorithms when the configuration is not known. In this section we compare the use of bounding sets versus probability distributions for modeling assembly operations for two examples.

Example 1: Sequential Placement

Consider the transfer of a single part through a series of stations by consecutive robots, one robot

between each station. For simplicity we consider the representation of this problem in a single dimension using the following definitions:

$x(k)$: part position at stage k

$y(k)$: measured position at stage k

$e_y(k)$: error in measurement y_k

$a(k)$: acquire (pick) command at station k

$e_a(k)$: error in execution of acquire command $a(k)$

$m(k)$: move (place) command from stations k to $k+1$

$e_m(k)$: error in execution of move command m_k

d_k: distance from stations k to $k+1$

w_k: 0.5 [(width of gripper k) [(width of part)]

These definitions are illustrated in Fig. 11

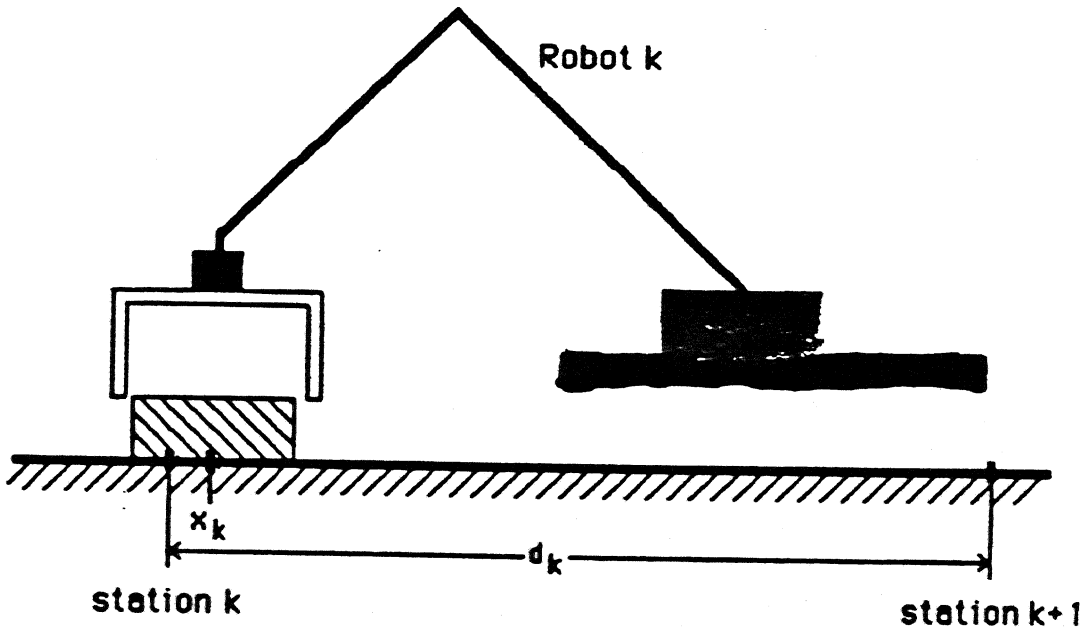


Figure 11: Sequential placement example: transfer of part by robot k from station k to station $k+1$.

The commands $a(k)$,$m(k)$ are positions for the end-effector to acquire and place the part,

respectively. The actual location of the end-effector when each command is executed is given by $a(k)+e_a(k)$ and $m(k)+e_m(k)$, respectively. The k^{th} robot successfully acquires the part from station k when

$$|x(k)-a(k)-e_a(k)| \le w_k, \tag{2}$$

that is, when the gripper jaws are opened so that the part is between the jaws. Assuming (2) is satisfied, the move command places the part at

$$x(k+1)=x(k)-a(k)-e_a(k)+m(k)+e_m(k). \tag{3}$$

The sensed location of the part at each stage is given by

$$y(k)=x(k)+e_y(k). \tag{4}$$

Modeling the uncertainty in this example requires the representation of knowledge about the values of the errors $e_a(k),e_m(k),e_y(k)$. The formulation of the control problem, that is, how to choose the command values $a(k),m(k)$, depends on the method of modeling this uncertainty.

A Probabilistic Model:

Let the errors $e_a(k),e_m(k),e_y(k)$ be independent, zero mean, gaussian random variables (rv's) with variances $\sigma^2_{ea},\sigma^2_{em},\sigma^2_{ey}$, respectively, for all k. Also, let the initial position $x(0)$ be a zero mean ($\mu_x(0)=0$), gaussian rv, independent of the errors, with variance $\sigma^2_x(0)$. In general, the commands $a(k),m(k)$ are also rv's since they may depend on the measurements $y(k)$. From equation (3) the position $x(k)$ of the part at each stage will be a rv with mean $\mu_x(k)$ given by

$$\mu_x(k)=\sum_{j=0}^{k-1}[\mu_m(j)-\mu_a(j)] \tag{5}$$

where $\mu_m(k),\mu_a(k)$ are the mean values of the commands $m(k),a(k)$, respectively. The variance of $x(k)$, $\sigma^2_x(k)$, is generated by the recursive relation

$$\sigma^2_x(k+1)=\sigma^2_x(k)+\sigma^2_m(k)+\sigma^2_a(k)+2(\sigma^2_{x,m}(k)-\sigma^2_{x,a}(k)-\sigma^2_{m,a}(k))+\sigma^2_{ea}+\sigma^2_{em}, \tag{6}$$

where $\sigma^2_{x,m}(k),\sigma^2_{x,a}(k),\sigma^2_{m,a}(k)$ denote covariances of the rv's indicated by the subscripts.

The probability of successful acquisition of the part at stage k, denoted by $p(k)$, is given from equation (2) as

$$p(k)=\text{Prob}\{|x(k)-a(k)-e_a(k)| \le w_k\}. \tag{7}$$

A reasonable control objective in this modeling framework is to maximize the probability of successfully transferring the part through a sequence of *N*+ 1 stations. This is stated formally as

$$\max n^{*}L_{f=0}rt/>- \qquad (8)$$

Alternative control strategies for this stochastic problem are discussed in Section 4.

A.Bounding Set Model:

We now consider a bounding set model for the sequential placement problem. Knowledge of the initial position *x(0)* and the errors $e_a(k), e_m\{k|e_y(k)$ is represented by sets of possible values for each variable. To describe the model in terms of general set notation we first introduce some definitions and notation.

Bounded Variable (bv): An uncertain real-valued variable specified in terms of a set of possible values.

Bounding Set: The set of possible (real) values for bv v denoted by 5(v). tn tfce following it will be assumed all bounding sets are closed and connected.

Joint Bounding Set: The set of n-vectors of possible simultaneous values for am ordered set of *n* bv's, ($v_1, \ldots, v^$, denoted by $B(v_{lt}\ldots .v^$, referred to as an *n^{th}-order bounding set.*

Conditional Bounding Set: The bounding set for bv v_1 given the value of bv v_2=TJ, denoted by $B(v_1/v_2=i|)$. This bounding set for v_1 is given by

$$B(v_1/v_2=\eta)=\{v_1|(v_1,\eta)\in B(v_1,v_2)\}.$$

independent bounded variables: A set of *n* bv's

$v_{1\S}\ldots,v^$ are *independent* if $B\{v_{l\%}\ldots ,v^= -S(v_2)x - \bullet \bullet xB(vJ.$

Sum of sets: For sets *A*9*B of* n^m-order real-valued vectors, the sum *A*+ *B* is defined as

$$A+B=\{^\ x=a+b \quad \text{for} \quad atAMB\}. \qquad (9)$$

Scalar multiplication: For set *A* of n^-order real-valued vectors, and real constant c, the product *cA* is defined as

$$cA=\{x|x=ca \quad \text{for} \quad a\in A\}. \tag{10}$$

For our example, the bounding sets for the robot command errors $e_a(k),e_m(k)$ will be denoted E_{ea},E_{em}, respectively, representing the positioning accuracy of the robot. The measurement errors $e_y(k)$ are in a bounding set E_{ey} representing the resolution and accuracy of the sensors and the initial position $x(0)$ is in bounding set $X(0)$. It is assumed $e_a(k),e_m(k),e_y(k),x(0)$ are independent bv's. Due to the possible dependence of the the commands $a(k),m(k)$ on the measurements $y(k)$, the commands, and consequently the $x(k)$ for $k\geq 1$, are not necessarily independent bv's with respect to the initial position or the error bv's. We assume, however, that commands $a(k)$ and $m(k)$ are independent of the current command errors and future command and measurement errors.

From equation (3), the general expression for the bounding set $B(x(k))=X(k)$ is given by the recursive relation

$$X(k+1)=B(x(k)-a(k)+m(k))+E_{ea}+E_{em}, \tag{11}$$

The bounding set $B(y(k))=Y(k)$ is given from equation (4) as

$$Y(k)=X(k)+E_y, \tag{12}$$

and the conditional bounding set of $x(k)$ given measurement $y(k)=\nu$ is

$$B(x(k)/y(k)=\nu)=[E_y+\{+\nu\}]\cap X(k). \tag{13}$$

Equations (12)and (13) can be viewed as projections of the joint bounding set for $(x(k),y(k))$. This is illustrated in fig. 12 where $X(k)$ is an interval on the real line (the horizontal axis) and the measurement error (independent of $x(k)$) is in the interval $E_y=[-\varepsilon_y,\varepsilon_y]$, which generates from equation (4) the joint bounding set $B(x(k),y(k))$. In fig. 12, the bounding set for $y(k)$ (12) is the projection of $B(x(k),y(k))$ onto the vertical axis and the conditional bounding set $B(x(k)/y(k)=\nu)$ is the projection of $B(x(k),y(k)=\nu)$ onto the horizontal axis.

For this example, where all of the variables are scalars and the bounding sets are connected, the ranges of values can be specified by upper and lower bounds. (Note from equation (9) that the sum of connected sets is always connected.) The relations in equation (11)and (12) imply recursive equations for the bounds of the connected sets which will be discussed in section 4.2.

In this modeling framework successful acquisition of the part at stage k is guaranteed when (from equation (2))

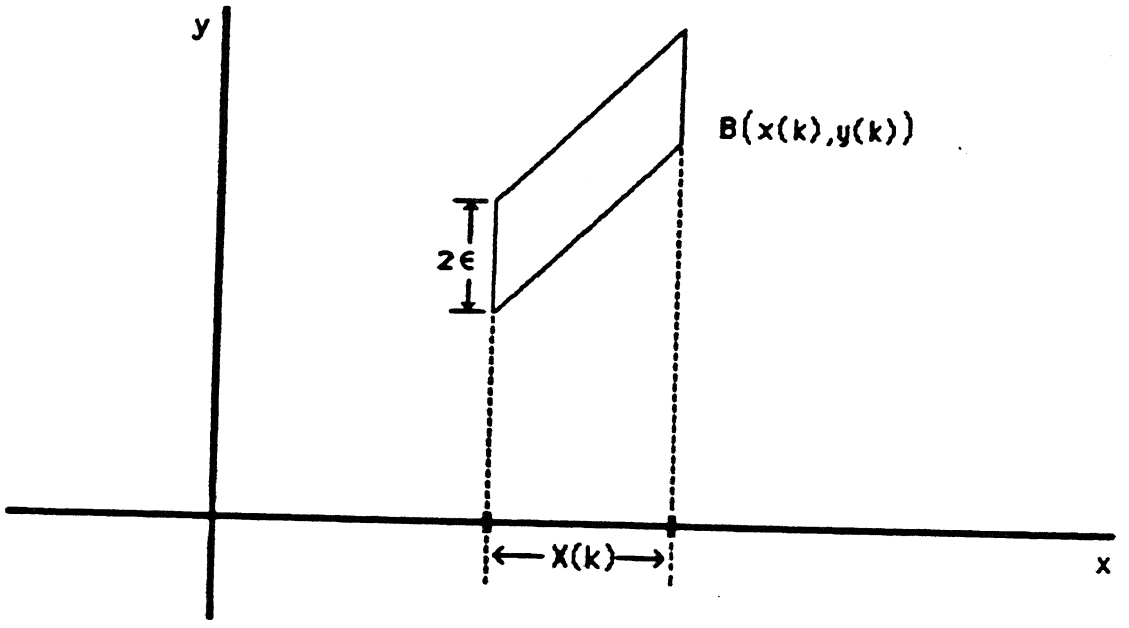


Figure 12: Joint Bounding Set $B(x(k),y(k))$ for $X(k),E_y(k)$ given as intervals

$$B(x(k)-a(k)-e_a(k))\subset[-w_k,w_k]. \tag{14}$$

A control objective for this problem would be to successfully transfer the part through $N+1$ stations. From equation (14) this objective is stated as

$$B(x(k)-a(k)-e_a(k))\subset[-w_k,w^k] \text{ for all } k=0,\ldots,N. \tag{15}$$

Various control strategies for satisfying equation (15) are discussed in Section 4.2.

Example 2: Stacking Blocks

Consider the problem of stacking blocks illustrated in fig. 13. For simplicity we consider only a single horizontal dimension for this problem using the following variables:

$p(k)$: command position for placing the k^{th} block,

$e_p(k)$: error in execution of $p(k)$,

$x(k)$: center-of-mass of the k^{th} block after placement,

w_k: 0.5 × width of block k,

m_k: mass of block k,

$y(k)$: measured location of block k,

$e_y(k)$: error in measurement y_k.

It is assumed the robot successfully acquires each block and places it on the stack with horizontal position $p(k)+e_p(k)$ for block k. Also, the blocks are symmetric with

$$x(k)=p(k)+e_p(k) \tag{16}$$

and the edges of the k^{th} block are at $x(k)\pm w_k$. The measured location of block k is given by

$$y(k)=x(k)+e_y(k). \tag{17}$$

From static analysis, the condition for successful stacking of to k^{th} block is

$$\max\{l_{1,k},\ldots,l_{k,k}\}\le m_k x(k)\le \min\{u_{1,k},\ldots,u_{k,k}\}, \tag{18}$$

where for $m=1,\ldots,k$

$$l_{m,k}=(x(m-1)-w_{m-1})(\textstyle\sum_{j=m}^{k} m_j)-\sum_{j=m}^{k-1} m_j x(j), \tag{19}$$

and

$$u_{m,k}=(x(m-1)+w_{m-1})(\textstyle\sum_{j=m}^{k} m_j)-\sum_{j=m}^{k-1} m_j x(j). \tag{20}$$

We now consider models for the sources of uncertainty in this example which are $x(0)$, the position of the first block, $e_p(k)$, the error in the placement of block k, and $e_y(k)$, the measurement error.

A Probabilistic Model:

Suppose $x(0),e_p(k),e_y(k)$ are independent, zero mean, gaussian random variables with variances $\sigma_x^2(0),\sigma_{ep}^2,\sigma_{ey}^2$, respectively. Since the command positions will, in general, depend on the measured values, the k^{th} command $p(k)$ is a random variable with mean $\mu_p(k)$ and variance $\sigma_p^2(k)$. It is assumed that $p(k)$ is independent of $e_p(k)$ and all future position errors, which implies from equation (16)

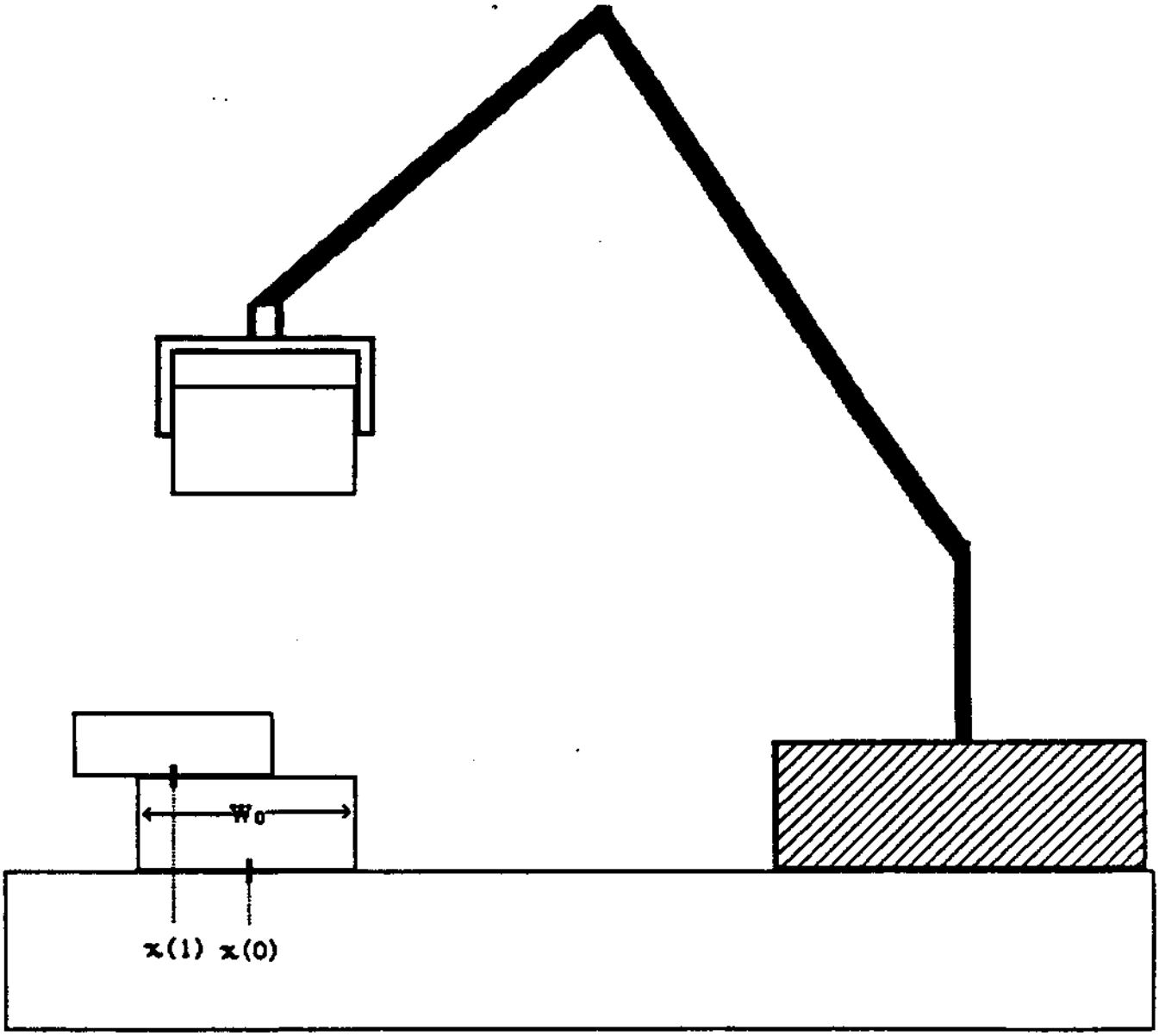


Figure 13: Example 2: Stacking of blocks with single dimension of error

$$\mu_x(k)=\mu_p(k) \tag{21}$$

and

$$\sigma_x^2(k)=\sigma_p^2(k)+\sigma_{ep}^2. \tag{22}$$

In contrast to the previous example, the tolerance requirements at each stage are dependent on the results of the previous stages. The lower and upper bounding variables $l_{m,k}$ (19) and $u_{m,k}$ (20) are random variables with means determined by the mean values of the positioned blocks and variances which may include covariance dependencies among the block positions if the commands are dependent on sensor information. A natural control objective in this framework is to stack N blocks successfully with a given probability. However, even the evaluation of the probability of success at each stage (the probability of satisfying equation (18)) is an involved computation when feedback is used to determine the control values.

A Bounding Set Model:

Using the definitions related to bounding sets given for the previous example, we let E_{ep}, E_{ey} denote the bounding sets for the command errors $e_p(k)$ and the measurement errors $e_y(k)$. The initial position $x(0)$ is in the bounding set $X(0)$ and it is assumed the $x(0), e_p(k), e_y(k)$ are independent bv's. As in the previous example we assume that the position command $p(k)$ is independent of the current command error and the future command and position errors. Thus, the bounding set for the k^{th} position is given by

$$X(k)=P(k)+E_{ep} \tag{23}$$

where $P(k)$ is the bounding set for the command $p(k)$.

In the bounding set approach, the objective of the control is to guarantee at each stage that the tolerance requirement (18) is satisfied. To state this in terms of the bounding sets, let $L_{m,k}, U_{m,k}$ be the bounding sets for the bv's $l_{m,k}, u_{m,k}$ in (19),(20), respectively. Letting l_k be the least upper bound for the set $L_{1,k} \cup \cdots \cup L_{k,k}$ and u_k be the greatest lower bound for the set $U_{1,k} \cup \cdots \cup U_{k,k}$, then satisfaction of (18) requires

$$m_k X(k) \subseteq T_k \tag{24}$$

where T_k is the interval

$$T_k = V_k, u_k). \quad (25)$$

Note that evaluation of equation (24) is considerably easier than evaluating the objective in the probabilistic formulation. The use of sensor information in the bounding set formulation for control is also more straightforward for bounding set models.

4 Supervisory-Level Feedback Control

Section 4.1 describes the basic structure of a supervisory control system and identifies, in general terms, the types of control schemes used at the supervisory level. Section 4.2 deals with the evolution of uncertainty in the on-line knowledge of the system configuration as sensor information is incorporated into the control decisions.

4.1 Discrete Supervisory Control Structures

An assembly operation is accomplished through the coordination of a set of actuators. The supervisory controller issues actuator commands based on the processed sensory information and knowledge of the previous command history. The objective of the supervisory control is to successfully implement the sequence of subtasks. Thus, the system must be designed to provide enough feedback information to identify successful completion of subtasks and satisfaction of tolerances. The nature of the feedback scheme depends on several features: the type of decision space at each point in the feedback process, the sensor signal space, and the algorithmic structure for determining the actuator actions.

In this section we describe the feedback loops in a sensor-based flexible assembly system. As illustrated in fig. 14, a control process in the system combines information from sensors, computations and internal timers to issue commands to actuators or other processes in the system. We define each of the components in fig. 14 as follows:

sensors: physical devices with output signate dependent on the configuration of the system.

actuate rs: physical devices that effect changes in the system configuration.

control processes: analog or digital combinatorial operations (including simple computations) on sensor and memory data to issue actuator commands, initiate computations, or set timers. Control processes are initiated by signals from sensors, computational processes, timers or. other control process.

computational processes: information processing algorithms which are not simple combinatorial operations and generally do not have a fixed operating time. The results of computations can update internal parameters and initiate control processes.

timer processes: internal timers used to initiate and terminate computational and control processes.

Interactions among these components are indicated in fig. 14. This disgram of the system is not necessarily explicit in a given implementation, but it serves to identify the different ways in which sensory information is used to control a robotic system.

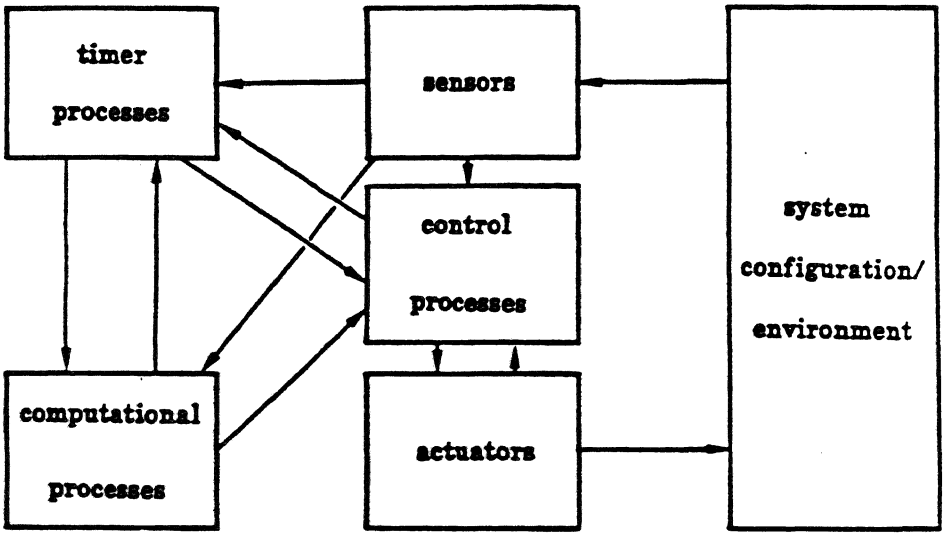


Figure 14: Components of sensor-based flexible assembly systems

A control process is characterized by the following features:

1. **initiating process:** the sensor, computational or timer processes which can initiate the control process.

2. **terminating process:** the sensor, computational or timer processes which can terminate the control process.

3. **inputs:** information from sensors, computations, memory and timers used to compute the control process commands.

4. **outputs:** commands issued to actuators or to initiate computational, timers or other control processes.

5. **input-output relation:** the relationship between the control process inputs and outputs.

Once the first four aspects of the control process have been specified, the controller design problem is reduced to the determination of the input-output relation. This is the design problem normally addressed in control engineering. For real-time supervisory control, however, the system structure is perhaps more critical in determining the overall system performance, and systematic methods for this level of design have *not* been developed.

Supervisory control involves multistage decision problems which can be hierarchically classified into *command sequence* decisions and *command value* decisions. The command sequence decision refers to the selection of the actuator to receive a command. This is a discrete decision from the set of available actuators and in many cases it is determined *a priori.* The command value decision is also from a discrete space in most applications; however, it can be modeled as a continuous value decision when the command space is large, as in the case of position values for a robot manipulator. We classify various supervisory control schemes according to the variety of options for on-line implementation of the command-sequence and command-value decisions. In this framework four classes of control structures are:

fixed sequence-fixed value: the mode of control most common in so-called hard automation. The supervisor is a sequencer which issues predetermined actuator commands in a fixed order.

variable sequence-fixed value: a more sophisticated version of the sequential controller, the actuator sequence depends upon logical relations among the sensor feedback signals. This type of control is required for coordinating concurrent processes with varying performance times or for implementing error recovery routines.

fixed sequence-variable value; supervisory commands are issued in a fixed order to* the actuators, but the values change based on sensory information or an internal model of the configuration. Many robotic manipulators operate in this mode in current applications,

variable sequence-variable value: the supervisor decides, on-line both the actuator to receive a command, and the command to be issued. Applications of robots to changing tasks in uncertain environments require this mode of control.

This section has been descriptive since there are presently no standard analytical formulations of the control problem at this level. One modeling approach currently being pursued is the use of modified Petri nets which represent the discrete sequencing structure of the control interfaced with a lower-level model of the continuous processes. The following section focuses on fixed sequence-variable value control to simplify the discussion of uncertainty as the feedback process evolves.

4.2 Evolution of Uncertainty in the Feedback Process

The uncertainty in the system configuration changes as actuator commands are executed and sensor information becomes available. Analysis of the uncertainty as a function of time and the use of feedback information in real-time control decisions depend on the way in which the uncertainty is modeled as discussed in section 3.2. In this section alternatives for incorporating feedback information are discussed in the context of Example 1 from section 3.2.

4.2.1 Open-Loop Control

Fixed sequence-fixed value supervisory control is essentially open-loop. For the Sequential Placement Problem in section 3.2, a fixed sequence of acquire and move commands $\{(a(0),m(0)),(a(1),m(1)),\ldots\}$ results in ever-increasing uncertainty in the location of the part due to the accumulation of errors in the execution of the commands. Since no sensor information is being used, the most reasonable open-loop control sequence is given by the recursion

$$m(k)=a(k+1)=a(k)+d_k, \qquad a(0)=0. \tag{26}$$

Probabilistic Model: Since the commands are deterministic, the position of the part at each stage is a Gaussian rv with mean given from equations (5) and (26) as

$$\mu_x(k)=\sum_{j=1}^{k} d_j \tag{27}$$

and variance given from (6) as

$$\sigma_x^2(k)=\sigma_x^2(0)+k\{\sigma_{ea}^2+\sigma_{em}^2\}. \tag{28}$$

The probability of successfully acquiring the part (7) can be computed easily for each stage since $x(k)-a(k)-e_a(k)$ is a zero mean Gaussian rv with variance $\sigma_x^2(k)+\sigma_{ea}^2$. The only options for improving the performance of the open-loop control are to decrease the variance of the initial position or improve the accuracy of the robot.

Bounding Set Model: Under open-loop control the bounding set for the part position is given from (11) as

$$X(k)=X(0)+kE_{ea}+kE_{em}+\{\sum_{j=1}^{k} d_j\}. \tag{29}$$

and the acquisition of the part at stage k is guaranteed when

$$X(k)+E_{ea}\subseteq[-w_k,w_k].$$

Increasing the number of stages for which the acquisition is successful can be accomplished only by decreasing the sizes of $X(0)$, E_{ea}, and E_{em}.

4.2.2 Memoryless Feedback Strategies

It is not uncommon in robotic systems to use "sense and move" strategies which use only the current sensor data. These strategies are *memoryless* in that the commands at each stage do not depend on information from previous measurements. In these strategies, the process of estimating the system configuration is usually separated from the control law, that is, the rule for choosing the command for a given configuration is independent of the method for computing the configuration. Since the commands are not specified a *priori,* the feedback strategy can only be specified in the context of a particular scheme for modeling the uncertainty in the system. We consider below memoryless strategies for the Sequential Placement Problem.

Probabilistic Model: Suppose the current measurement *y{k)* (4) is assumed to be the best estimate of the part position at each stage and the-commands are chosen according to the control law

$$a(k) = y(k), \qquad m(k) = y(k) + d_k.$$

For this memoryless strategy the mean and variance of the part position at each stage given the measurement *y(k)* are *y(k),* a^2_{ey}, respectively. The probability of successfully acquiring the part at each .stage (7) is improved significantly in comparison to the open-loop approach because $x(k) - a(k) - e_a(k) = -e_y(k) - e_d(k)$. Thus, the probability of success at each stage is determined by a set of independent, identically distributed random variables. The performance of this strategy can be improved by improving the accuracy of the sensor and robot and is independent of the variance of the initial condition.

Bounding Set Model: Assuming the measurement error is in the interval $E_y = [-e_y, e_y]$, the control strategy is to let *a(k)=)(k)* and $rrt(k) = y(k) + d_{kS}$ as in the probabilistic case. This reflects the assumption that the best way to satisfy the tolerance requirement (14) is to command the robot to go to the center of the estimated bounding set *X(k).* Using only the current measurement to estimate *X(k)* results in a larger bounding set than the exact expression for *B(x(k)/y(k))* (13).

In contrast to the open-loop case, the bounding set for *x(k)* after each measurement^^)}+ E_y is of a constant size determined by the bounding set for the measurement error. The part is acquired successfully at each stage provided (from (13))

$$E_y + E - (a) \subseteq [-w_k, w_k].$$

As for the probabilistic case, the success of this feedback control scheme is Independent at each stage and does not depend on the initial bounding set *X(Q).*

4.2.3 Feedback With Memory

As a third class of feedback strategies, consider schemes for the Sequential Placement Problem which determine the best estimate of the system configuration based on all past measurements, commands, and *a priori* knowledge of the initial part position. What is meant by the "best estimate" must be defined within the context of the method for modeling uncertainty. The commands are then chosen as if the estimate were the true configuration, thus separating the problems of estimation and control.

Probabilistic Model: Since the recursive equations for the part position are linear, a Kalman filter can be used to provide the best estimate of $x(k)$ given all past information including the current measurement $)\{k)$. Letting $x(k/k)$ denote this estimate and using the control law $a(k)=x(k/k\backslash m(k)=x(k/k)+dp$ we have the recursive relations

$$x(k/k)=(l - o\backslash\{k- 1/k- iWjMH- y^{*}\sim 1) + d_{kmml}) + (v\hat{}k- Vk- l)/\hat{}y(k)$$

where $a_x^2(k/k)$ is the variance of the location of the part at stage k given by

$$\sigma_x^2(k/k)=\frac{\sigma_{ey}^2(\sigma_x^2(k-1/k-1)+\sigma_{em}^2+\sigma_{em}^2)}{\sigma_{ey}^2+\sigma_x^2(k-1/k-1)+\sigma_{em}^2+\sigma_{em}^2}$$

Unlike the memoryless feedback schemes, the uncertainty in the part position is not constant at each stage and, in general, improves as more on-line measurements become available. Note that the part position variance is independent of the measurement values. Since the control is based on the best mean-square error estimate of the part position, the position at each stage is a Gaussian random variable and the probability of successfully acquiring the part at each stage is better than for the memoryless case. Since the estimation at each stage is given by a simple recursive equation, the computational burden is very little for this simple example.

Bounding Set Model: The "best" estimate of the system configuration in this context is the center of the bounding set for the position at each stage given all previous measurements and commands, which we will denote by the same notation used in the probabilistic case, $x(k/k)$. The control law will be as before, namely, $'\hat{}k) = x(kJk\backslash m(k)=-x(k/k) + d..$ To develop the recursion relations for computing $x(k/k)_t$ let $X(k/j)$ be the bounding set for $x(k)$ given the measurements (and controls) for stages 1 , . . . J. From (3) and the fact that $m(k)\sim a(k)=d_k$ we have

$$X(k/k\text{-}l)=X(k\text{-}yk\text{-}l)+\{d. \quad .l+E+E, \tag{30}$$

Since $e(Jfc)$ is independent of $jcfJfc)$, the joint conditional bounding set for $x(k),y(k)$ given all measurements and controls up through stage fc-1 is known. Thus, by analogy to (13), we have

$$X(k/k)=(E_y+\{y(k)\})\cap X(k/k-1). \tag{31}$$

The desired estimate at each stage is simply given by

$$x(k/k)=0.5(glb\{X(k/k)\}+lub\{X(k/k)\} \tag{32}$$

where $glb\{\cdot\}$,$lub\{\cdot\}$ denote the greatest lower bound and least upper bound, respectively, of the set arguments.

These relations can be converted directly into the following recursion equations for the glb's and lub's for the bounding sets. Using the notation $\bar{\eta}$ = lub$\{B(\eta)\}$and $\underline{\eta}$ = glb$\{B(\eta)\}$ for bv η, we have from (30), (31),

$$\bar{x}(k/k)=\min\{\bar{e}_y(k)+y(k),d_{k-1}+\bar{x}(k-1/k-1)+\bar{e}_a(k-1)+\bar{e}_m(k-1)\}$$

and

$$\underline{x}(k/k)=\max\{\underline{e}_y(k)+y(k),d_{k-1}+\underline{x}(k-1/k-1)+\underline{e}_a(k-1)+\underline{e}_m(k-1)\}.$$

The estimate (32) is then given as

$$x(k/k)=0.5(\bar{x}(k/k)+\underline{x}(k/k)).$$

Although this estimate is not generated by strictly linear equations, the computations are no more complex than those required for the Kalman filter in the probabilistic approach. The use of all prior information in computing $x(k/k)$ results in bounding sets at each stage which are subsets of the bounding sets obtained in the memoryless approach. Thus, the number of stages for which the part is acquired successfully is potentially larger using the estimation scheme developed above.

5 Summary of Research Issues

This report reviews ongoing research in modeling and control of assembly tasks and systems in the Flexible Assembly Laboratory at CMU. The results described for modeling and representation of assembly tasks provide the framework for the development of systems design tools, and raise a number of important issues for continuing research, summarized as follows.

- **Descriptive Primitives.** It is necessary to develop a general taxonomy of assembly components, devices and operations that is detailed enough to be useful, yet concise enough to facilitate computationally efficient search algorithms for use in system design and optimization procedures.

- **Design Criteria.** significant performance measures to evaluate alternative system implementations must be translated into a general method of representing performance primitives for the operation. Each level of the system design, the performance evaluation should be commensurate with the level of detail in the operation description.

- **Methods for Representing Uncertainty.** The appropriate method for representing

uncertainty depends on several factors including the nature of the *a priori* information, the control objectives, and the computational aspects of estimation and control. Both the probabilistic and bounding set methods have useful features. Research in this area should focus on criteria for choosing the appropriate representation of uncertainty, and techniques for combining methods such as the probabilistic and bounding set approaches.

- **System and Control Structures.** The OPG is clearly related to the possible alternatives for physical implementation of the assembly system. A method for classifying and identifying these structures will be developed.

- **Complexity of Feedback Strategies.** Another direction for further work is to determine the implications of various system structures for flexible assembly system design. For example, simplicity of design may suggest that it is advantageous to use simple feedback loops whenever possible with higher level control processes serving only to initiate and terminate the lower-level control processes. These higher level processes might have simple binary sensory inputs, such as limit switches. It is of interest to determine the types of assembly problems for which this an appropriate design philosophy.

- **Real-time Resource Allocation.** For flexible systems with alternative resource assignments for on-line implementation, decision strategies are being investigated which use feedback information to optimize the system performance.

References

1. J.L. Nevins, D.E. Whitney and S.C. Graves, "Programmable Assembly System Research and its Application -- a Status Report", *RI-SME Technical Paper (MS82-125)*, 1982.

2. S.H. Hopkins, et al, "Semiautonomous Systems in Automatic Assembly", *IEEE Proceedings*, Vol. 132, No. 4, July 1985, pp. 174-177, Pt. D

3. T. Lorano-Perez, *Task Planning,* M. Brady, et al, editors, 1975.

4. R.J. Popplestone, A.P. Ambler, I.M. Bellos, "An Interpreter for a Language Describing Assemblies", *Artificial Intelligence*, Vol. 14, No. 1, January 1980, pp. 79-107.

5. L. I. Lieberman, M.A. Wesley, "AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly", *IBM J. Research and Development*, Vol. 21, No. 4, April 1977, pp. 321-333.

6. A.C. Sanderson and G. Perry, "Sensor-based Robotic Assembly Systems: Research and Applications in Electronics Manufacturing", *Proceedings of the IEEE Special Issue in Robotics*, July 1983.

7. A.C. Sanderson, "Parts Entropy Methods for Robotic Assembly System Design", *Proceedings of the IEEE International Conference of Robotics*, March 13-15 1984.

8. D.M. Hamilton, "An Optical Tracking Algorithm and Parts Entropy Modeling for the Motor-Pigtail Problem", Tech. report, Department of Electrical and Computer Engineering and Robotics Institute Technical Report, Carnegie-Mellon University, November 1984, M.S. Project Report

Modeling and Control of Assembly Tasks and Systems

Bruce H. Krogh and Arthur C. Sanderson

CMU-RI-TR-86-1

Flexible Assembly Laboratory
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

July 1985

This work has been supported in part by the Xerox Corporation, the IBM Corporation, and ITie Robotics Institute Carnegie-Mellon University.

Table of Contents

List of Figures