Constructing Task-Level Assembly Strategies in Robot Programming by Demonstration

Jason Chen
Department of Information Engineering,
Research School of Information Science and Engineering,
The Australian National University, Canberra, Australia
Ph: int + 61 6125 8687, Fax: int + 61 6125 8660
E-mail: Jason.Chen@anu.edu.au

Keywords

Programming by Demonstration, Teaching by Showing, Hybrid Dynamic Systems, Task Level Planning

Abstract

Programming by Demonstration (PbD) is a technique for programming robots that holds much promise in making robots more accessible to ordinary, non-technical users. However, a well known difficulty with the method is that a human will often demonstrate the task to be programmed inconsistently or even erroneously, leading to the inclusion of what is essentially noise in the demonstration. A number of techniques exist in the literature for filtering out this type of noise, however most focus on very low level control command details. In this paper we propose a new, complimentary direction of research. We take a "task-level" view of the demonstration, and note that noise can exist at this level also. We propose a framework, based on a Hybrid Dynamic System modelling approach, to select the most optimal, task-level execution strategies that were demonstrated. We apply our framework to a real household task of inserting the compressible spindle of a paper towel holder into its supports. We conduct experiments to show that significant improvements in robot performance of the task can be achieved by a PbD regime that includes our method.

1 Introduction

One of the main impediments to an expanding role of robotics in society are the current difficult and unnatural programming interfaces available. Writing computer code and the use of teach pendants are currently the predominant methods for robot programming, and both have obvious deficiencies; requiring specialist knowledge and being: unnatural to

use, time consuming and/or not very powerful. A novel approach to robot programming is Programming by Demonstration (PbD), where the human programs the robot by demonstrating how the task is done. A PbD interface interprets the human's demonstration and generates the control details required by the robot so it too can complete the task. Such a programming interface is very natural for a human to use, it does not require specialist knowledge, and can potentially program very complex tasks.

Programming by demonstration is an active research area, and many interesting results have been obtained (see [10] or [9] for a literature review of the field). One of the key results identified with the approach is that the demonstration can often contain sub-optimality. A number of authors, including Dillman et. al. [18], Delson and West [19], Nechyba and Xu [33] and De Schutter et. al. [22], cite sources of sub-optimality that can arise in the demonstration. For example, Dillman [18] recognised five sources of sub-optimality: where the human demonstrates unnecessary, incorrect, or unmotivated actions, where there is choice of scenario regarding when to apply an action, and where the actions are demonstrated with the wrong intention (i.e. the user does not know enough about the task). Delson and West [19] identified that, in a pick and place task through a field of obstacles, a human will naturally introduce "wiggles" into the demonstrated paths used to traverse regions were the gap between obstacles is large. Suboptimal actions of this type obscure the skillful set of actions required to complete the task, and can be viewed as *noise*. Clearly, having the robot directly copy the demonstration will not be optimal, and in many cases will not result in the robot correctly achieving the task. The solution is to identify and remove noise from the demonstration before any programming of the robot takes place.

A number of approaches have been proposed in the literature to solve this problem.

Although these approaches have met with some success, we note that they all focus on identifying and removing noise from the control details that were demonstrated [27, 12, 22, 19, 10] (see [10] for a more in depth literature review of this work). In this paper we propose a new, complimentary direction of research. We show that, while humans can demonstrate the control details for a task inconsistently, they will also generally demonstrate a range of "task-level" execution strategies that vary in their optimality. That is, the demonstration will also contain noise at the task level, and an important part of PbD is to identify and remove it. Noise removal at the task level, and at the control-detail level, are both complimentary processes; in that they should both be conducted by a competent PbD system. However, we note that the potential for improved robot performance by optimising at the task level is great; finessing the control details of a poor task-level strategy will not lead to much better task performance. Given the complimentary nature of these approaches, we have formulated this paper as a companion paper to our previous work on optimising at the control-detail level (reported in [10]), This is in the sense that we apply our approach here to the same task and set of demonstrations as in that paper, so that the robot performance presented here is the combined result of optimising at both the control-detail, and the task levels.

The work we propose in this paper is limited in its applicability to assembly tasks, i.e. tasks involving contact and constrained motion between a workpiece and environment. We deal with finding optimal task level plans in a directed graph of contact formations between task objects that were visited in the demonstration. In some ways then, the work in this paper is also related to work by Xiao and Ji [32] and Lefebvre et. al. [30], where task level planning using contact formations is also addressed. The difference between that work, and

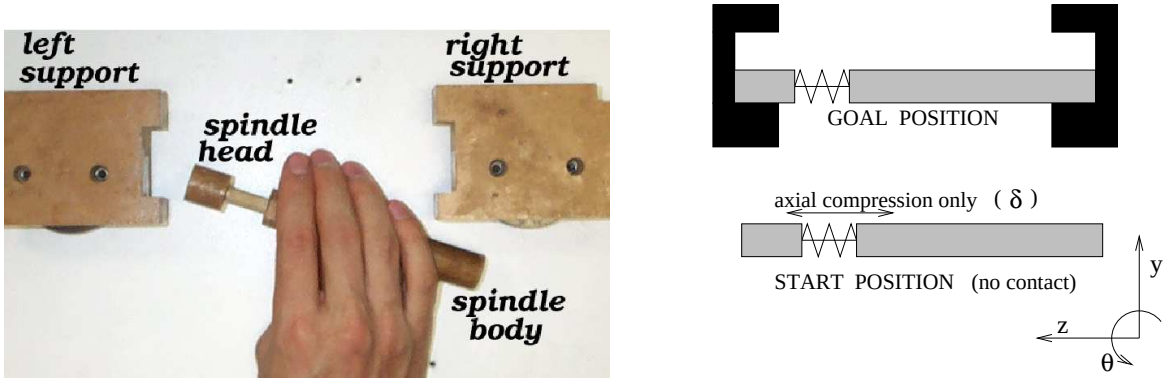

Figure 1: The spindle-assembly task chosen for PbD

our work here, is that they generate plans autonomously using algorithms based on the geometric properties of task objects. In this paper we generate plans based on the "hint" provided by the demonstrator.

This paper is set-out as follows. In Section 2 we formulate more precisely the problem to be solved. We show how a task-level execution strategy can be described as a sequence of discrete events in a Hybrid Dynamic System skill model. This description leads us to formulate our problem in this paper as finding a *desired event path* that results in optimal robot performance of the task. In Section 3 our solution to this problem is presented. Robot performance is broken down into four distinct areas, with the desired event path selected as the path predicted to result in the best robot performance of the task in the four performance areas on average. Section 4 presents the results of the experimental testing of our method on an actual set of demonstrations of a real assembly task. Finally, we state our conclusions for the work in Section 5.

2 Problem Formulation

We have introduced the idea of a *task-level strategy*, and how different task-level strategies can be used to complete a task. To specify more concretely what we mean in this regard, we now introduce the task to be used for experiments in this paper, and a set of six demonstrations of this task provided by a human in a PbD environment. Figure 1(a) shows the task we use for experiments; the Spindle Assembly Task. It involves inserting an axially compressible spindle between two fixed supports. Figure 1(b) shows the start and goal spindle configurations of the task. The start position sees the spindle removed from the supports so that no contact exists. The goal position sees the spindle lying between the front edges of the rebate in each support, ie. as shown in Figure 1(b). Note two things about the task. First, it is essentially a planar task, and we consider only demonstrated motions in the horizontal plane (i.e. the spindle has 4 degrees of motion freedom, each specified by one of x, y, θ and δ). Second, at least partial compression of the spindle must occur before its insertion between the supports can take place. This has the important consequence that completion of the task must follow a sequence of spindle head insertion into the left support, followed by compression, and then by spindle body insertion into the right support.

Six demonstrations of the task were provided by the human, and are shown in Figure 2. We denote the demonstrations shown in the figure as D_1 to D_6 respectively. Note how we can present these demonstrations as sequences of distinct contact configurations between the spindle and supports. Each contact configuration can be thought of as a distinct *state* in the task. We have assigned each state a unique identifier, eg. state "1" for the goal configuration in the task, and state "2" for the starting configuration, etc. We will provide

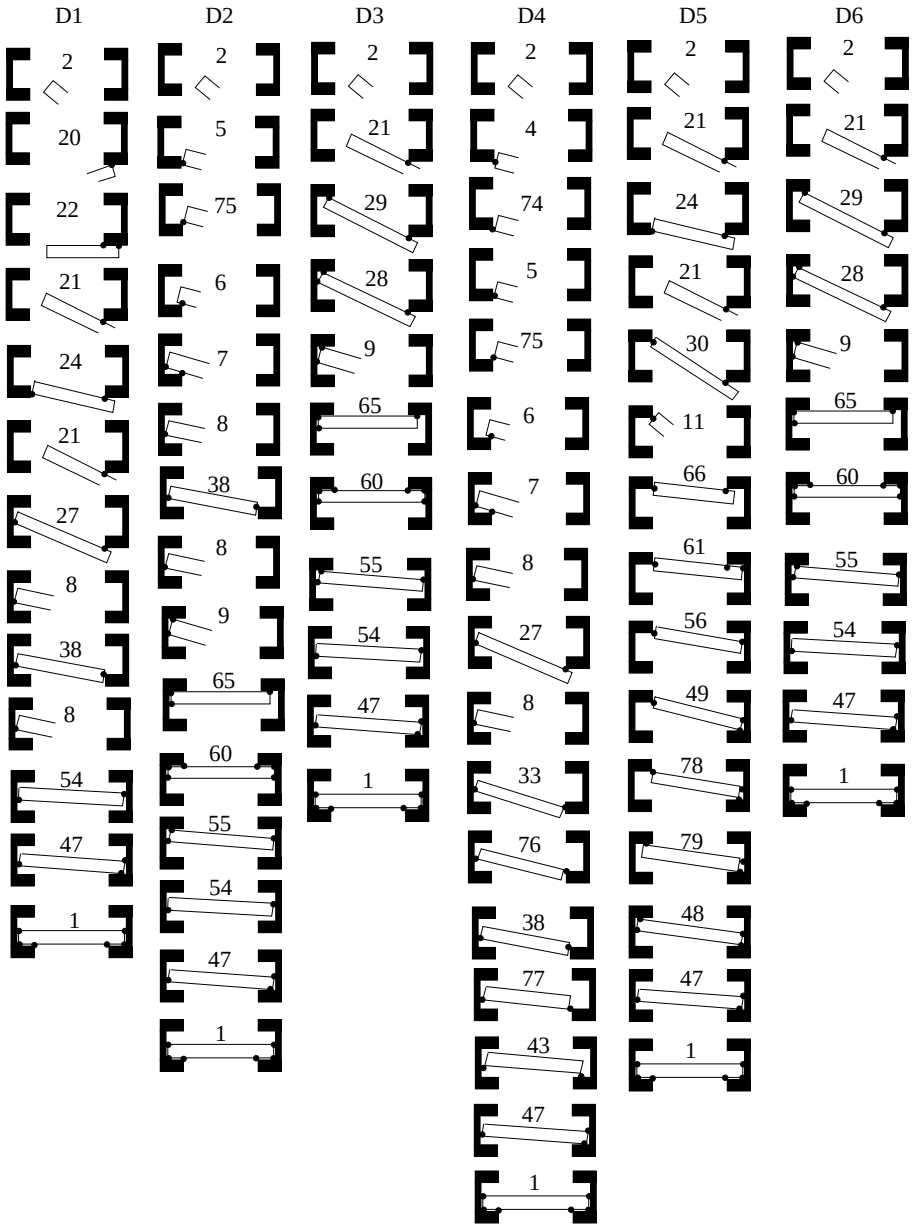

Figure 2: The set of state sequences demonstrated by the human in the spindle-assembly task

a more formal basis for this form of representation later in this section, however, we first continue with the important issue of presenting how a human can demonstrate a number of different task-level strategies in robot PbD.

The demonstration set D_1 to D_6 contains a number of different task-level strategies. We note that two main phases of the spindle-assembly task exist, (i) spindle head insertion into the left support, and (ii) spindle body insertion into the right support. We group the strategies used by the demonstrator in each of these phases into categories in Table 1. The demonstrator has used three main strategies to insert the spindle head into the left support. First he has taken the "direct-head" approach, where the spindle head is moved directly towards the rebate in the left support, ie. the state sequence 5-75-6-7 more-or-less allows the spindle to move in a "straight line" between the spindle's start configuration and its inserted configuration in state 8. Second, he has demonstrated a "guided" approach. Here the contact in state 21 is used as a guide to achieve spindle head insertion, i.e. the demonstrator slides the spindle edge along the bottom left corner of the right support until spindle-head insertion is achieved. Third, a "delayed" approach was demonstrated. Here, contact between the spindle head and left support was maintained in state 11 while the spindle body was inserted into the rebate of the right support. That is, the final insertion of the spindle head into the rebate of the left support was *delayed* until the spindle body was fully inserted into the right support.

Table 1 also shows the strategies used by the demonstrator to insert the spindle body into the right support. Note that at this stage in the task the spindle head is in contact with the left support, however the demonstrator needs to bring the spindle body into contact with the right support. Table 1 classifies, into three main types, the strategies used by the

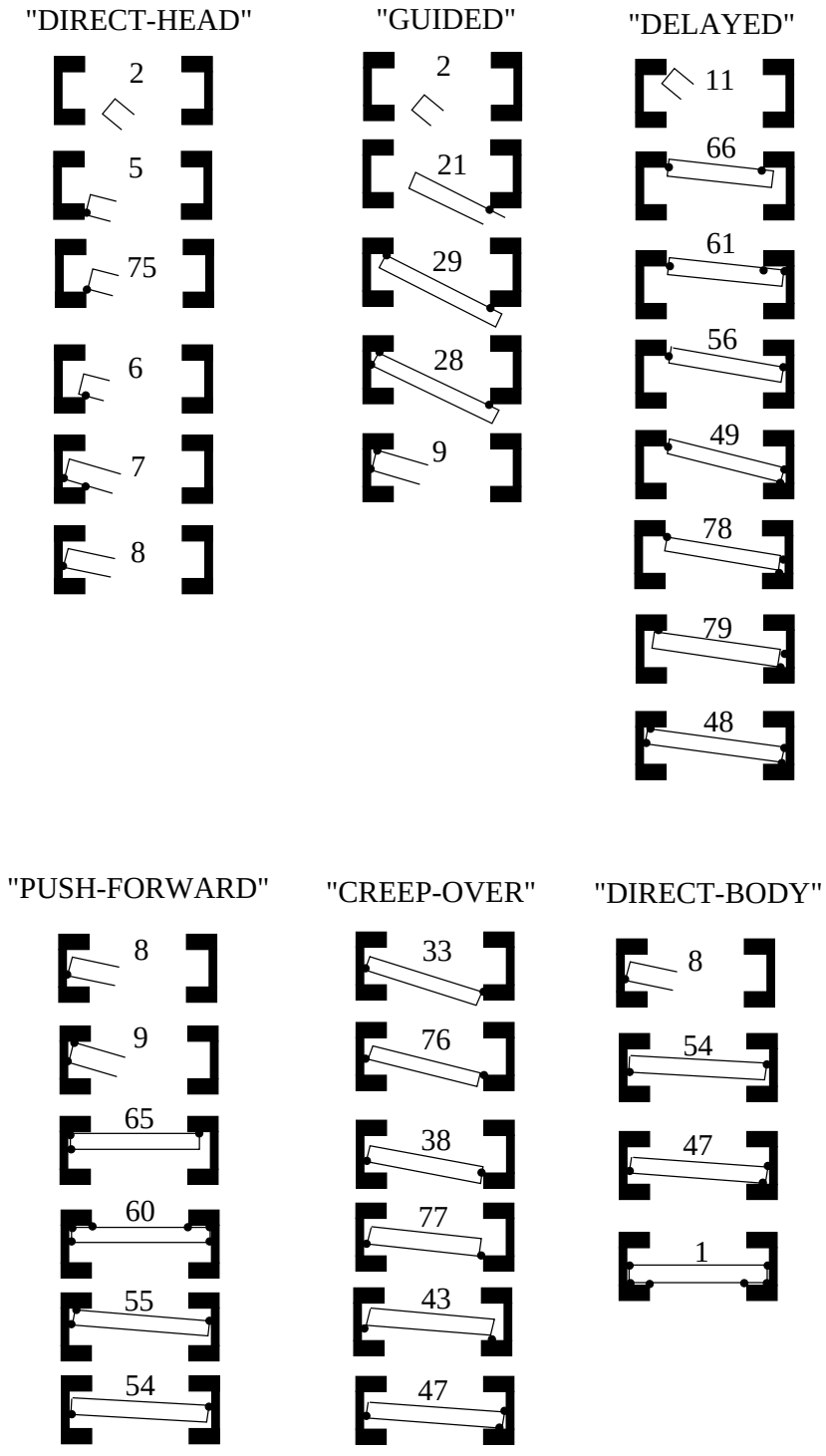

Table 1: The task-level execution strategies demonstrated for the spindle-assembly task

demonstrator to make this contact. First, he has used the "push-forward" approach, where contact is first made in state 60. Using this strategy, the demonstrator pushes the spindle body forward, pivoting about a contact between the spindle-head and left support until state 60 is achieved. Second, he has used a "creep-over" approach. Here, contact between the spindle body and right support is first made in state 33. Insertion of the spindle body into the rebate is then achieved by "creeping-over" the front lip of the right support (i.e. through states 76, 38, 77, and 43) and into the rebate. Third and finally, the demonstrator has used the "direct-body" approach. Contact is made by moving the spindle body directly toward the base of the rebate in the right support, i.e. without touching either side of the rebate.

The preceding discussion shows how many different task-level strategies can be demonstrated by the human. Then an important problem in PbD is to select for the robot a set of strategies that will see it best perform the task. In this paper we propose one possible solution to this problem. We formulate this solution within an approach to modelling a robotic assembly process as a Hybrid Dynamic System (HDS). Hybrid Dynamic System theory involves the study of dynamic systems that consist of two component subsystems; one discrete in nature, and the other continuous in nature [23]. Quite a bit has been written about applying Hybrid Dynamic System theory as a means to model robotic assembly [16, 4, 6, 8], including in the context of PbD [10]. In essence, the assembly process is interpreted in this work as, at the discrete level, a sequence of discrete events that occur asyncronosly through time when objects in the task move from one contact configuration to another. In contrast, the continuous system describes the dynamics of task objects in the usual way, i.e. as a differential equation relating the time rate of change of the pose of

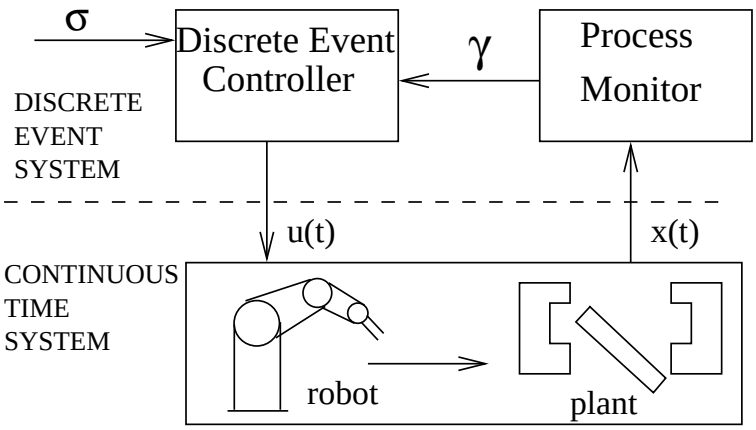

Figure 3: The Hybrid Dynamic System skill model for assembly tasks

an object with its existing pose and the control input.

A useful outcome of this dual description of assembly dynamics is that control of the assembly process can be undertaken at two levels, i.e. in contrast to the usual approach of modelling and control in the continuous time domain only. We show in Figure 3 the details of the control regime first proposed by McCarragher [4] for the control of robotic assembly in a Hybrid Dynamic system setting. The dashed line in the figure marks the interface between the discrete-event and Continuous Time Systems. In the Discrete Event System there exists the Process Monitor (PM) and Discrete Event Controller (DEC). The role of the PM is to recognise from the Continuous Time System state vector $\mathbf{x}(t)$ when a discrete event has occurred. Upon recognising the occurrence of an event, the PM outputs the new state γ of the Discrete Event System to the DEC, which then determines the command $\mathbf{u}(t)$ for output to the Continuous Controller (CC) in the Continuous Time System. Note that command $\mathbf{u}(t)$ is chosen with the purpose of triggering the next event in a *desired event path*, where in Figure 3 we show this desired event path as σ. The Continuous Controller

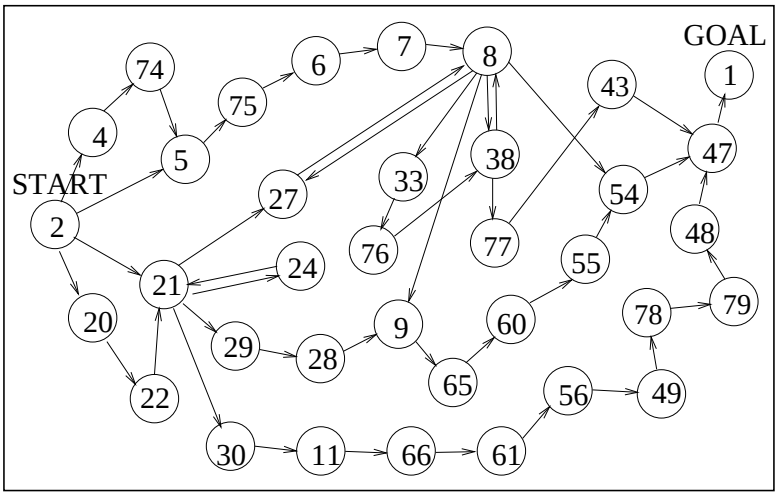

Figure 4: The directed graph $\mathcal{A}$ of states and events present in the demonstration set D_1 to D_6

receives the command $\mathbf{u}(t)$ from the DEC, and applies it via the robotic mechanism to modify the configuration $\mathbf{x}(t)$ of the task. The loop is completed with the PM monitoring when the next discrete event occurs.

The power of the Hybrid Dynamic control regime just described comes from its two-tiered nature. Control elements in the discrete event system form a meta controller which take a high level point of view of the process, i.e. they control the sequence of discrete events needing to occur for task completion to be reached. The elements in the Continuous Time System focus at a lower level on controlling the task configuration so that the next desired discrete event occurs. [1] Given our interest is to formulate a task level plan for the robot from the task level plans demonstrated by the human, our focus in this paper will be on the Discrete Event part of the HDS. It is this component of the HDS that allows us

[1] It is interesting to note that research by Hogan into human manipulation competency [20] suggests that humans approach assembly in a similar fashion, i.e. with an abstract, task-level plan, which is then implemented through manipulation at a more concrete, continuous level. This research suggests that a HDS is indeed a valid means to model robotic assembly

to describe, and therefore capture, demonstrated strategies at the task level. Toward this end, an important and useful representation of the Discrete Event System is as a directed graph. Each node in the graph represents one of the discrete states in the task, while each arc represents a discrete event that can occur to take the task between these states. For example, we show in Figure 4 the directed graph formed from the demonstrated sequences of events and states of Figure 2. Denote the graph in Figure 4 as $\mathcal{A}$. Then, it can be seen that the problem of determining an optimal task-level strategy for the robot really means finding a path of discrete events traversing in $\mathcal{A}$ between the initial unassembled state (State 2) to the final fully-assembled state (State 1) which see the robot perform in some optimal fashion. Recall from Figure 3 that such a path will form an input into the DEC component of the discrete event control regime, and that we denoted it as σ. Our purpose in this paper is to present a method for deriving σ.

3 Selecting a Desired Event Path

Selecting a desired event path σ means predicting the performance of the robot for every event in $\mathcal{A}$. Accurate prediction of how the robot will perform each event means that selecting σ is the relatively straight-forward process of choosing a path in $\mathcal{A}$ that passes between the start and goal states, and that contains the most optimally performed events. However, in order to present measures that predict optimal robot performance of an event, we must first define what we mean by optimal robot performance.

3.1 Defining Optimal Robot Performance

We define optimal robot performance as performance with the following characteristics:

1. low execution time

2. high reliability

3. low control effort

Our aim is to select events for σ that are performed by the robot with these three characteristics. However, in addition, we note the capability of the human to analyse his performance of the task. A human will routinely identify and repeat often any strategies for task execution that he finds works well. We make the assumption that strategies found to work well by the human will also work well for the robot [2]. So, in addition to selecting events that see the robot perform with characteristics 1.,2., and 3., our aim is also to select events for σ that were:

4. demonstrated often

3.2 The Path Selection Framework

Recall that each arc in directed graph $\mathcal{A}$ represents an event that was demonstrated. Our idea for determining σ is to assign to each arc in $\mathcal{A}$ a cost, where the cost assigned reflects how well the robot will perform the event represented by that arc. A high cost is assigned to the arcs of events predicted to result in poor performance, while a low cost is given to the arcs of events predicted to be performed well. We then determine σ by conducting a search in $\mathcal{A}$ for the minimum cost path between the start and goal states in the task. Since a minimum cost path is selected, only the events resulting in the best possible robot performance will be included in σ.

[2]Note that this assumption requires the relative level of performance of each event by the human and robot to be correlated, something we will not prove. However, we include this area in our set of performance areas since, at least from the outcomes of our experiments, a correlation seems to exist that makes this performance area a useful one to include.

The obvious question with the approach regards how the cost of each arc should be determined. Let $\tau_k^{\mathcal{A}}$ denote an event that is represented by one of the nodes in $\mathcal{A}$ (i.e. the kth event demonstrated in the task) and let $\gamma_{\check{\tau}_k}^{\mathcal{A}}$ and $\gamma_{\hat{\tau}_k}^{\mathcal{A}}$ be the states between which $\tau_k^{\mathcal{A}}$ takes the assembly process, i.e. the occurrence of $\tau_k^{\mathcal{A}}$ triggers a change in discrete task state from $\gamma_{\check{\tau}_k}^{\mathcal{A}}$ to $\gamma_{\hat{\tau}_k}^{\mathcal{A}}$. Then, a cost is assigned to the arc in $\mathcal{A}$ representing $\tau_k^{\mathcal{A}}$ as follows. First, individual costs are assigned to $\tau_k^{\mathcal{A}}$ in each of the four performance areas: time, reliability, control effort, and number of times demonstrated. Sections 3.2.1, 3.2.2, 3.2.3 and 3.2.4 describe for each performance area how this step is achieved. An overall cost is then calculated from the costs assigned in the individual performance areas. Section 3.2.5 describes the details of how the overall cost is calculated.

3.2.1 Time

Strictly speaking, an event in the HDS is instantaneous and takes zero time. It is the infinitesimal amount of time taken to pass between two distinct contact formations in the task. For our work here, we are interested in how long an event takes to *achieve*. That is, how long after the previous event in the assembly process does it take for us to have event $\tau_k^{\mathcal{A}}$ occur. Three components to this time exist:

1. *time for discrete event control in* $\gamma_{\check{\tau}_k}^{\mathcal{A}}$: how long for the DEC to determine a control command $u(t)$ for use in state $\gamma_{\check{\tau}_k}^{\mathcal{A}}$ that will cause event $\tau_k^{\mathcal{A}}$ to occur.

2. *time for continuous control in* $\gamma_{\check{\tau}_k}^{\mathcal{A}}$: time for the CC/manipulator to traverse $u(t)$.

3. *time to process monitor* $\tau_k^{\mathcal{A}}$: time for the PM to recognise that $\tau_k^{\mathcal{A}}$ has occurred.

We note that item 2 forms the dominant component in the time required to achieve $\tau_k^{\mathcal{A}}$. That is, the majority of time required by the robotic system to achieve event $\tau_k^{\mathcal{A}}$ will result from having to change the physical configuration of the task. Work by Hovland and McCarragher [7] showed that the time required for process monitoring is small. They used a multilayer perceptron neural net as a PM, and showed that process monitoring in real time is possible, (i.e. for a reasonably complex asymmetrical insertion task, events were recognised in the order of 1.5 milliseconds on standard computer hardware). The time required for discrete event control may be greater. However, deciding on a continuous control command for each event in $\mathcal{A}$ need not occur in real time. Indeed, a more sensible approach is to do the processing offline. An appropriate $u(t)$ for each event in $\mathcal{A}$ would then be available for use by the robot at execution time.

The dominance of the CC in determining the time required to achieve $\tau_k^{\mathcal{A}}$ means that our metric for predicting performance in the time area is simplified. We identify that the time required to achieve event $\tau_k^{\mathcal{A}}$ will depend mainly on the amount the task configuration is required to change for the event to occur. A good measure of the amount the task configuration must change is the "distance" along the path traversed by the demonstrator to achieve this event in the Configuration Space [31] representation of the task. Note that where more than a single path for event $\tau_k^{\mathcal{A}}$ was demonstrated, the average length of the paths were taken. Denote this average length for event $\tau_k^{\mathcal{A}}$ as L_{av}^k. Then our metric for calculating a cost Ct_k in the time performance area for $\tau_k^{\mathcal{A}}$ is:

$$Ct_k = L_{av}^k \tag{1}$$

3.2.2 Reliability

The reliability of execution by the robotic system will depend on the reliability of each of the components in the system. That is:

1. The reliability of the PM in identifying $\tau_k^{\mathcal{A}}$ correctly when it occurs
2. The reliability of the DEC in selecting an obstacle-free $u(t)$ that will cause $\tau_k^{\mathcal{A}}$ to occur
3. The reliability of the CC in precisely following $u(t)$ so that an erroneous event (i.e. one that is not $\tau_k^{\mathcal{A}}$) does not accidently occur.

Our experience is that the main source of execution failure in robotic systems stems from item 3. Regarding item 1, Hovland and McCarragher [7] showed that a 95 percent success rate for Process Monitoring can be achieved, even with quite small training sets for the Multi Layer Perceptron. Results from our experiments (which we will present later in Section 4) using this type of PM are consistent with their findings. Regarding item 2, results presented in [10] for a DEC showed that appropriate setting of parameters in that method resulted in every command produced by the DEC being valid and obstacle-free (we discuss more fully in Section 4 the details of this DEC method). Failure caused by item 3 results mainly from flexibility and backlash in the robot and workpiece (i.e. the spindle). The relative positioning of the workpiece and the environment can then not be precisely known or controlled. Failure by this mode is well known in robotics, especially for tasks involving contact between task objects [11]. We adopt the well known hybrid [3] force-position control regime first introduced by Raibert and Craig in [17] as our CC. A strength

[3] where *hybrid* in this case refers to the combination of force and position control, in contrast to our previous use of the term in *hybrid dynamic system*, which referred to the combination of discrete event and continuous time systems

of the hybrid-force position control regime is that it uses constraints provided by contact in the task to guide assembly motion. That is, if many constraints are present, then motion can be guided, and more reliable execution of the task achieved. We therefore propose the inclusion of $dof_{\gamma^A_{\tilde{\tau}_k}}$; the number of dofs of the spindle in state $\gamma^A_{\tilde{\tau}_k}$, as one of the components in our reliability measure Cr_k. If the spindle motion is not constrained in a particular state, then the events that cause the assembly process to "leave" that state are deemed less attractive from a reliability point of view. However, we must temper the desire for events out of highly constrained states to be included in σ with the realisation that progress must be made towards the fully assembled state. For example, states 60, 48, etc. in Figure 2 allow no progress towards the fully assembled state since the spindle is fully constrained in these states. We therefore multiply $dof_{\gamma^A_{\tilde{\tau}_k}}$ with $1/L^k_{av}$ (where we introduced L^k_{av} in Section 3.2(a)) in Cr_k to reflect that fact that we desire states that constrain motion, but that also allow the spindle configuration to change so that progress towards full assembly can occur. The final component of Cr_k we propose is $S_{\gamma^A_{\tilde{\tau}_k}}$, the number of neighbouring states to state $\gamma^A_{\tilde{\tau}_k}$ in $\mathcal{A}$. That is, we calculate the cost in the reliability area for event $\tau^{\mathcal{A}}_k$ as:

$$Cr_k = dof_{\gamma^A_{\tilde{\tau}_k}} \times \frac{1}{L^k_{av}} \times S_{\gamma^A_{\tilde{\tau}_k}} + 1 \tag{2}$$

where we multiply $dof_{\gamma^A_{\tilde{\tau}_k}} \times 1/L^k_{av}$ with $S_{\gamma^A_{\tilde{\tau}_k}}$ because incorrect traversal of $\gamma^A_{\tilde{\tau}_k}$ will cause an event other than $\tau^{\mathcal{A}}_k$ to occur. That is, we penalise, in Cr_k, the events out of states where it is possible for many alternate events to occur. We add the value 1 so that a state with zero spindle degrees of freedom is not assigned a zero cost. An arc with zero cost in $\mathcal{A}$ can cause problems for our search algorithm when trying to find the minimum-cost path.

3.2.3 Control Effort

By control effort we mean the computational effort in controlling the assembly process. Recall how the HDS modelled assembly skill as the repeated application of the same skill "loop", i.e. recognising a contact formation, determining a command to reach the next desired contact formation, applying that command, recognising the new contact formation, etc, etc. For each loop, computational effort is required. The PM must process force and position data in order to recognise the event, a control command $\mathbf{u}(t)$ must be selected by the DEC, and the CC must control the robot so that $\mathbf{u}(t)$ is physically achieved in the workspace. Clearly, from a computational point of view, the less times we need to repeat this loop, the better. Then a good measure of control effort for task completion using a certain event path is the number of events in that path. That is, a cost in the control-effort area Ce_k for transition $\tau_k^{\mathcal{A}}$ is calculated simply as:

$$Ce_k = 1 \tag{3}$$

Then, any event path selected by our method will have a cost in the control-effort performance area equal to the length of the path itself.

3.2.4 Number of Demonstrations

Determining a cost Cn_k for $\tau_k^{\mathcal{A}}$ in the number of demonstrations performance area is straight forward. The cost should be set inversely proportional to N_k, the number of times transition $\tau_k^{\mathcal{A}}$ was demonstrated. It should be set *inversely* proportional to N_k so that $\tau_k^{\mathcal{A}}$ with good performance in this area (i.e. those demonstrated often) are assigned a low cost in $\mathcal{A}$, while

those demonstrated less often are assigned a high cost. Then the metric for calculating a cost in the number-of-demonstrations performance area Cr_k for $\tau_k^{\mathcal{A}}$ we propose as:

$$Cn_k = (N^{max} - N_k)^{dof} + 1 \tag{4}$$

where N^{max} is the number of times the most demonstrated transition was demonstrated. We raise $(N^{max} - N_k)$ to the power of dof (the dof of the task, i.e. four in our case) so that the cost assignment is not linear with respect to $(N^{max} - N_k)$. That is, events repeated more often are given a much lower cost than infrequently demonstrated events. This cost assignment structure is necessary so that a short path with infrequently demonstrated events is not selected over a longer path with more frequently demonstrated events. Analysis suggests that the dof of the task is a good value to use in the indexation, i.e. because the length of path through $\mathcal{A}$ will generally increase as the dof increases. Note also in Equation 4 that we have added a value of 1. This is to ensure that Cn_k can never equal zero, i.e. for the case when $\tau_k^{\mathcal{A}}$ is the most demonstrated event in the task. An arc with zero cost in $\mathcal{A}$ can be problematic for a search algorithm when searching in for a minimum-cost path.

3.2.5 Determining an Overall Cost

We identified four performance areas in which we categorise robot performance. Note then that a transition may result in good robot performance in one area and poor performance in another. A method is required to calculate the overall performance of an event. To determine the overall performance of an event we introduce the following measure:

$$C_k = W_t Ct_k + W_r Cr_k + W_e Ce_k + W_n Cn_k \tag{5}$$

where C_k is the overall cost calculated for $\tau_k^{\mathcal{A}}$, and W_t, W_r, W_e, and W_n are a set of weights. The weights allow the overall cost of a transition to be modified according to what aspect of performance is deemed important by the demonstrator. For example, if the time weight W_t is set to a value greater than the other weights, then the overall cost C_k will be determined more by the event's cost in the time performance area. If we adopt a greater W_t value for all transitions in $\mathcal{A}$, then a search through $\mathcal{A}$ for the least cost path will result in a path with low execution time. In this way, our event-path selection framework allows robot performance to be tuned according to what is required. Note that by increasing the value of the other weights in the set, paths can be selected that cause the robot to perform reliably, with low in control effort, or to copy the demonstrator.

4 Experimental Results

Our aim in this section is to present the paths selected by our framework, and to confirm that they did cause the robot to perform the task in the fashion predicted. Figure 4 presents the four paths selected; labelled respectively as the "time", "reliability/number-of-demonstrations", "control-effort", and "even" paths. Note that the even path is the result of an even emphasis being given to all weights in Equation 5, so that a mode of task execution that is a compromise across all performance areas should be produced.

Prior to discussing the merits of these paths and how the robot performed them, we introduce the experimental apparatus on which testing was carried out. Recall from Figure 3, the components in the robotic system: the manipulator, CC, DEC and PM. Each needed

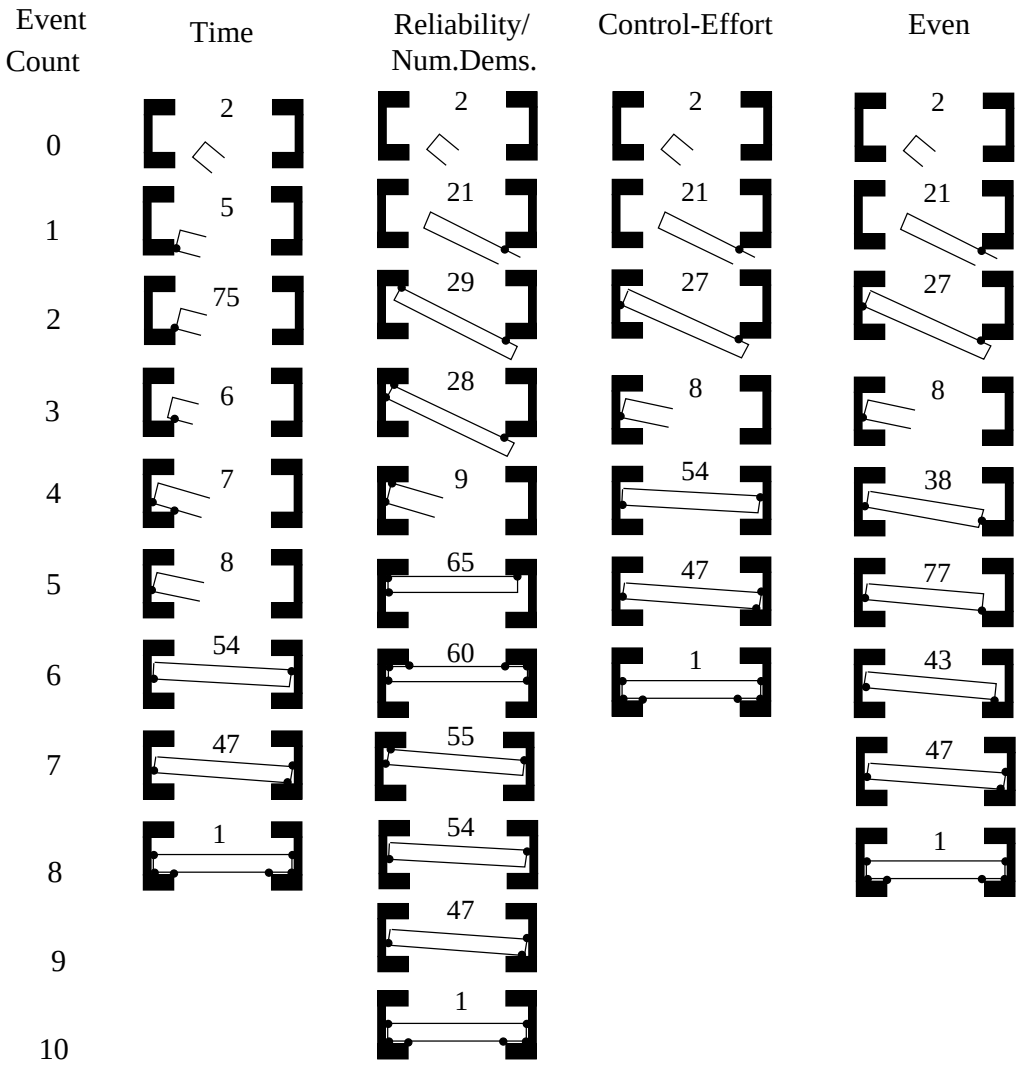

Figure 5: Event paths selected by our framework

to be implemented in order for experiments to be conducted. We show in Figure 4 the manipulator used in experiments, and we have already noted that we adopt the well known hybrid force-velocity control method first proposed in [17] as a CC. Adopting this control regime for our CC means that the DEC must output a $u(t)$ consisting of both velocity and force components. A number of generic DEC regimes have been proposed in the literature. For example McCarragher proposes a DEC based purely on velocity control [4], while Austin and McCarragher [1] propose a DEC that provides a partitioned force and velocity control command, as required by our CC. However this work is not directly applicable in our experiments, since it bases its force and velocity command synthesis on a geometric model of the task, rather than on trajectories provided by a human in a demonstration. We implement a DEC that also outputs a partitioned force and velocity command suitable for our CC, but that derives these commands based on demonstrated force and velocity trajectories. The details of this work can be found in [9] and, in particular, the details of the velocity command synthesis part of the work can be found in [10]).

The remaining component in the robotic system is the PM. A number of generic approaches for implementing a PM exist in the literature, for example: based on neural nets [7], HMM's [8], energy analysis [25], on the theory of polyhedral convex cones [26], and others [5, 24, 14, 15]. For experiments in this paper we implemented the neural net approach of [7]. Note that this method is not demonstration based, i.e. it does not formally synthesise a PM from demonstration. Work that addresses this issue can be found in [28, 29, 3, 21]. Our aim here is to achieve a functional PM so that we can test our task-level strategy selection framework. The details of the method by Hovland can be found in [7], and the implementation details of the method for the spindle assembly task, in [9]. In brief, the method

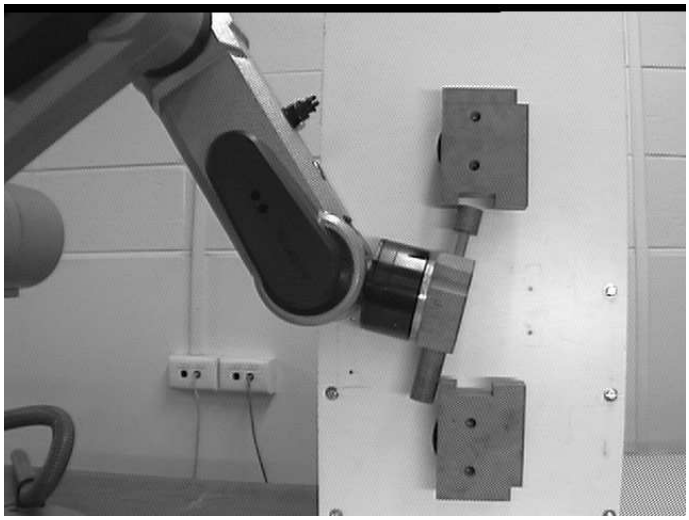

Figure 6: The Scorbot Eshed robot on which experiments were conducted

Path Name	Weight Emphasised	State Sequence	Successful Executions	Execution Time (secs)	Computation Time (secs)
Time	W_t	2-5-75-6-7-8-54-47-1	6/10	34.3	1.04
Reliab./N.Dems.	W_r, W_n	2-21-29-28-9-65-60-55-54-47-1	10/10	52.7	1.29
Control Effort	W_e	2-21-27-8-54-47-1	7/10	39.1	0.81
Even	equal	2-21-27-8-38-77-43-47-1	8/10	42.0	1.12

Table 2: Results of implementing the paths selected by our framework on the robot

uses a multilayer perceptron neural net. The net is trained by inputing position and force signals from the different state transitions [4] in the task into the input nodes, and clamping the output node representing the transition being trained high (with the remainder of output nodes clamped low). Backpropogation is then used to determine appropriate weights. During the execution of the task, the transition experienced is recognised according to the output node that becomes active.

[4] produced, for example, by manually backdriving the robot

With the experimental apparatus introduced, we now present our results. Ten attempts at each of the selected paths presented in Figure 4 were performed by the robotic system just described. Table 2 shows the results of these experiments. The layout of the table is straightforward, with its first three columns describing the path for which the results were obtained, and its last three columns presenting the reliability, execution-time, and computation-time results respectively. Two important questions exist concerning the path selection and robot execution results presented in Figure 4 and Table 2: (a) do the paths selected by our framework make sense given the task-level strategies that were available in the demonstration ? (recall that the task level strategies available in the demonstration were presented in Table 1), and (b) did the robot perform these paths in the manner (i.e. fast, reliably, etc) that was predicted ?

Figure 4 shows that the time-path has used the direct-body and direct-head assembly strategies. The reason is because both these methods use an essentially "straight-line", direct approach to inserting both the spindle head and body. Our use of path length as the basis for the metric in this performance area has caused these strategies to be selected over others (i.e. the guided, push-forward, creep-over, and delayed strategies) which all have longer path lengths in comparison. Table 2 confirms the validity of choosing this basis for our time metric. The time path required 34.3 seconds, compared to 39.1 seconds for the next longest path (the control effort path) up to 1 minute and 9.6 seconds for path with the longest execution time (the delayed path). This result suggests that our path length metric is a valid mode for predicting execution time.

In contrast to the time path, the reliability path contains the guided strategy for spindle head insertion, and the push-forward strategy for spindle-body insertion. These strategies

were selected because of the guided motion each path provides. Our assertion was that the robot can traverse a state more reliably when its motion is more guided. For spindle-head insertion, the guided strategy contains much more guided motion than either the direct-head or delayed assembly approaches; principally due to State 21, where a dof of 3 exists. This is in contrast to the direct-head and delayed approaches, where most of the "distance" travelled occurs in State 2 (the no contact state), where no guiding occurs. With regard to spindle body insertion, the push-forward approach on average results in a lesser degree of spindle freedom compared to alternative paths. To see why the push-forward approach was selected over the creep-over approach, recall that spindle-body insertion involves pivoting the spindle about a point in contact with the left support so that it can be passed around the front lip of the rebate in the right support. The push-forward approach was demonstrated using State 9 (spindle dof of 2) to achieve this pivoting motion, while the creep-forward approach was demonstrated with State 8 (spindle dof of 1) for the same motion. That is, the push-forward approach was selected because State 9 encodes a greater constraint and guides motion more than in State 8. Experimental results confirmed the path selections made by our framework, and the underlying reasons for why these sections were made. Reliability ranged from six successful executions in ten attempts for the time path, to ten from ten attempts for the reliability path. Failures in experiments were caused mainly by the CC. Flexibility in the spindle and robot arm, along with backlash in the robots joints, and the quite fine tolerances required in some areas of the task, caused the CC to sometimes not achieve an event in a path. Three of the four failures for the time path occurred when the assembly process accidently moved from State 8 into State 9 during the pivoting motion of the spindle. In comparison, the reliability path used State 9 instead of State 8 for this

pivoting motion, which constrains the spindle head in both the y- and z-directions, rather than just in the z-direction. That is, experimental experience suggests that the basis for our reliability metric is correct; constraining spindle motion leads to more reliable execution. The other failure for the time path was caused by the spindle head clipping the front edge of the rebate in the left support during spindle head insertion. In contrast, this did not occur for the reliability path, which used State 21, instead of State 8, for spindle head insertion. The reason it did not occur was because the contact in State 21 provided a constraint on the spindle motion during the spindle head insertion phase. Again, the success with respect to reliability of the more highly constrained State 21 over the less constrained State 2, suggests that the basis for our reliability metric is correct.

The control-effort path uses the guided approach for spindle-head insertion and the direct-body approach for spindle-body insertion. Understanding why these strategies were selected is straightforward. Recall that our metric in this performance area was based on the number of events in an event path. Studying $\mathcal{A}$ in Figure 4 reveals that the control-effort path is the path between the start and goal states in the task with the least number of events. Column six in the Table 2 shows the computation time required for control purposes for each path. Processing in experiments was conducted on a Motorola 68040 based VME board. The times shown are the sums of the times required be each of the PM, DEC and CC functionalities. A set of valid $\mathbf{u}(t)$ by the DEC is achieved offine, so the computation time required to select the correct $\mathbf{u}(t)$ is negligible. Notice the relationship between the number of events in each path and the computation time required, i.e. the longer the path, the greater the computation required. We found that each event required roughly the same computation time to process monitor, so the more events in a path,

the greater computation required by the PM. We also found that paths with more states required greater computation by the CC. These findings suggest that the number of events in a path is a valid metric for predicting control effort.

The path selected in the number-of-demonstrations performance area is the same path as the reliability path. It was selected because of the frequency with which the guided approach to spindle head insertion and the push-forward approach to spindle-body insertion were demonstrated. The number-of-demonstrations performance area is different to the others, in that we do not need experimental results to validate the path selected; we saw in Section 4 how this path *did* copy the demonstrator. Our experiments here need to validate our assertion that copying the demonstrator leads to more optimal robot performance of the task. Experiments suggest that our assertion is a valid one. We have seen that the number-of-demonstrations path included the guided approach to spindle-head insertion and the push-forward approach to spindle-body insertion. One of the reasons identified by the human for using these strategies was their reliability. We have noted the reasons for the reliability of these strategies above, i.e. they allow assembly motion to be "guided". However, the human identified another, additional reason why these paths encode reliable execution. Note how event (2-21) provides a reliable way to make initial contact between the spindle and the right support. The large, flat, contact area provided by the spindle's side means the initial contact into State 21 can be achieved even if some uncertainty exists about the spindle's position relative to the support. Reliable robot execution of the number-of-demonstrations path has resulted in part from the inclusion of event (2-21) in the path. That is, in this case, copying the demonstrator has led to more optimal robot performance of the task, even though the reason for the more optimal performance was not explicitly

identified and included in the path selection system. This fact gives strength to the inclusion of the number-of-demonstrations performance area our framework. It can capture for the robot, aspects of task performance identifiable by the intelligent human system that are not modelled directly by metrics in other performance areas.

The even path uses the guided approach for spindle-head insertion, and the creep-over approach for spindle-body insertion. This selection of strategies represents a balanced approach to the assembly. The guided approach is not as optimal in the time area as the direct-head method, however it is not that much more time consuming, and is more reliable. The creep-over approach to spindle-body insertion is also a balanced selection. It is not as fast as the direct-body approach but is more reliable. It is not as reliable as the push-forward approach (because, as noted previously, State 8 is used for pivoting rather than State 9), but it is shorter in distance (and therefore faster) than that method. The length of the path is not as short as the control-effort path, but is shorter than the reliability path. This even path also contains events that were used often by the demonstrator, eg. (2-21), (27-8), and (47-1). Experimental results in Table 2 confirm that this path does result in balanced robot performance, which is really to be expected given the metrics performed in the individual performance areas as required.

So in summary, experiments have seemed to confirm the validity of our approach. At the base of the success was the design of metrics which accurately predicted how the robot would perform different demonstrated events. Also a success was the inclusion of weights in the process to allow different aspects of performance to be emphasised in the selection process. Success in both these areas meant that our system could choose the best demonstrated strategies for the robot in a number of distinct performance areas, and could avoid selecting

strategies resulting in suboptimal robot performance of the task. These results confirm the utility of including our method in any PbD system. They confirm that a significant improvement in the quality of programming can be achieved by a PbD system that includes such a method.

5 Conclusion

We have presented a system for selecting execution strategies at the task-level for robotic assembly tasks from demonstration. We made the point at the outset of the paper that such a process has largely not been addressed in the literature to date, and that most work in the area looks to optimise low level control details; an important part of the PbD process, but something of limited utility if the execution strategy at the task-level is suboptimal. We identified that a Hybrid Dynamic System is a good model for describing, and therefore capturing, human-demonstrated assembly strategies at the task level (although we note that other task-level representations for, in particular, robotic assembly exist [13, 2], and an interesting direction of future work would be to investigate whether valid methods of demonstration optimisation exist under these regimes also). Our proposed task-level strategy selection system was based on a set of simple but general metrics that predict robot performance in four distinct performance areas: time, reliability, control effort and number of demonstrations. Our system was applied to demonstration data from a real and non-trivial assembly task. We saw how quite a number of different task level strategies existed in this demonstration data, confirming our claims that robot command optimisation at the task level is a necessary part of PbD. Our system identified a number of these demonstrated strategies as being more optimal than others. It constructed event paths from the most opti-

mal strategies in each performance area according to what aspect of performance was being emphasised at the time. We confirmed the validity of the constructed paths with experiments. We saw that strategies selected to be optimal in a particular performance area did in fact lead to this type of robot performance of the task, and for the reasons predicted by our metrics. In addition, experiments confirmed that strategies not selected by our system were indeed less optimal than those selected. Our system forms a new and complimentary approach to existing approaches in the literature to the problem of removing noise from the demonstration in PbD. It provides a means for lifting the focus of the optimisation process from the control level details to the task level.

References

[1] D. J. Austin and B. J. McCarragher. Hybrid force/velocity discrete event controller synthesis for assembly tasks with friction. In *Proceedings of IEEE Conference on Robotics and Automation*, San Francisco, USA, April 2000.

[2] B.Hannaford and P.Lee. Hidden markov model analysis of force/torque information in telemanipulation. *The International Journal of Robotics Research*, 10(5):528–539, October 1991.

[3] B.J.McCarragher. Force sensing from human demonstration using a hybrid dynamical model and qualitative reasoning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 557–563, San Diego, CA, USA, May 1994.

[4] B.J.McCarragher and H.Asada. The discrete event control of robotic assembly tasks. *ASME Journal of Dynamic Systems, Measurement and Control*, 117(3):384–393, September 1995.

[5] T.J. Debus, P.E. Dupont, and R.D. Howe. Contact state estimation using multiple model estimation and hidden markov models. *International Journal of Robotics Research*, 23(4/5):399–414, 2004.

[6] D.J.Austin and B.J.McCarragher. Force control command synthesis for constrained hybrid dynamic systems with friction. *International Journal of Robotics Research*, 20(9):753–764, 2001.

[7] Geir.E.Hovland and B.J.McCarragher. Combining force and position measurements for the monitoring of robotic assembly. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'97), Grenoble, France*, pages 654–660, September 1997.

[8] Geir E. Hovland and Brenan J. McCarragher. Hidden markov models as a process monitor in robotic assembly. *International Journal of Robotics Research*, 146(17):266–267, October 1997.

[9] J.Chen. Coping with demonstration suboptimality in robot programming by demonstration. *Unpublished Ph.D. thesis, Department of Engineering, FEIT, Australian National University, Canberra, Australia*, 2001.

[10] J.Chen and A.Zelinsky. Programming by demonstraton: Coping with suboptimal teaching actions. *The International Journal of Robotics Research*, 22(5):299–319, May 2003.

[11] J.J.Craig. *Introduction to robotics: mechanics and control.* Addison-Wesley, second edition edition, 1989.

[12] M. Kaiser and R. Dillman. Building elementary skills from human demonstration. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 2700–2705, April 1996.

[13] K.Ikeuchi, M.Kawade, and T.Suehiro. Toward assembly plan from observation, task recognition with planar, curved and mechanical contacts. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2294–2301, 1993.

[14] T. Lefebvre, H. Bruyninckx, and J. De Schutter. Polyhedral contact formation modeling and identification for autonomous compliant motion. *IEEE Transactions On Robotics And Automation*, 19(1):26–41, 2003.

[15] T. Lefebvre, H. Bruyninckx, and J. De Schutter. Polyhedral contact formation identification for autonomous compliant motion: Exact nonlinear bayesian filtering. *IEEE Transactions On Robotics*, 21(1):124–129, 2005.

[16] Brenan J. McCarragher and Haruhiko Asada. The discrete event modeling and trajectory planning of robotic assembly tasks. *Journal of Dynamic Systems, Measurements and Control*, 117(3):394–400, October 1995.

[17] M.H.Raibert and J.J.Craig. Hybrid position/force control of manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102/127:126–133, June 1981.

[18] H.Friedrich M.Kaiser and R.Dillmann. Obtaining good performance from a bad teacher. In *Workshop: Programming by Demonstration vs Learning from Examples; International Conference on Machine Learning*, California, USA, July 1995.

[19] N.Delson and H.West. Robot programming by human demonstration: Adaptation and inconsistency in constrained motion. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, 1996.

[20] N.Hogan. How humans adapt to kinematic constraints. In *Proceedings of the 7th Yale Workshop on Adaptive Learning Systems*, May 1992.

[21] P.Sikka and B.J.McCarragher. Learning to recognize discrete state transitions in assembly. In *Proceedings of the Australian Robot Association's National Conference on Robots for Australian Industries*, Melbourne, July 1995.

[22] Wim Witvrouw Qi Wang, Joris De Schutter and Sean Graves. Derivation of compliant motion programs based on human demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2616–2621, April 1996.

[23] R.W.Brockett. Hybrid models for motion control systems. In H.L.Trentelman and J.C.Willems, editors, *Essays on Control: Perspectives in the Theory and Its Applications*, chapter 2, pages 29–5. Birkhauser, Boston, MA, 1993.

[24] J.De Schutter, H.Bruyninckx, S.Dutr, J.De Geeter, J.Katupitiya, S.Demey, and T.Lefebvre. Estimating first order geometric parameters and monitoring contact transitions during force controlled compliant motion. *International Journal of Robotics Research*, 18(12), 1999.

[25] S.Dutre, H.Bruyninckx, and J.De Schutter. Indentification and monitoring based on energy. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 1333–1338, Minneapolis, Minnesota, USA, April 1996.

[26] S.Hirai. Identification of contact states based on a geometric model for manipulative operations. *Advanced Robotics: The International Journal of the Robotics Society of Japan*, 8(2):139–155, 1994.

[27] S.K.Tso and K.P.Liu. Demonstrated trajectory selection by hidden markov model. In *Proceedings of the International Conference on Robotics and Automation*, pages 2713–2718, Albuquerque, New Mexico, April 1997.

[28] Marjorie Skubic and Richard A. Volz. Identifying contact formations from sensory patterns and its applicability to robot programming by demonstration. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 458–464, 1996.

[29] Marjorie Skubic and Richard A. Volz. Learning force based assembly skills from human demonstration for execution in unstructured enviroments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1281–1288, May 1998.

[30] T.Lefebvre, H.Bruyninckx, and J.De Schutter. Task planning with active sensing for autonomous compliant motion. *Journal of Robotics Research*, 24(1):61–81, 2005.

[31] T.Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computing*, C32:108–120, February 1983.

[32] Jing Xiao and Xuerong Ji. A divide-and-merge approach to contact motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, California, USA, May 2000.

[33] M.C.Nechyba Yangsheng Xu. On the fidelity of skill models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, USA, April 1996.