Jason Chen
Alex Zelinsky

Department of Systems Engineering
Research School of Information Science and Engineering
The Australian National University
Canberra, Australia
Jason.Chen@anu.edu.au
Alex.Zelinsky@anu.edu.au

Programing by Demonstration: Coping with Suboptimal Teaching Actions

Abstract

The difficulty associated with programing existing robots is one of the main impediments to them finding application in domestic environments such as the home. A promising method for simplifying robot programing is Programing by Demonstration (PbD). Here, an end user can provide a demonstration of the task to be programed, with a PbD "interface" interpreting the demonstration in order to determine low-level control details for the robot. A key aspect of the interpretation process is to make it robust to the noise typically included in a demonstration by the human. In this paper we present a method to help identify and eliminate any noise present in the demonstration. Our method involves two steps. The first step uses the demonstration to build up a partial knowledge of the geometry present in the task. Statistical regression analysis is used on demonstrated trajectories to determine equations describing curved surfaces in configuration space. The second step in our method uses the geometric information obtained in the first step to determine if there are more optimal paths than those demonstrated for completing the task. If there are, our method proposes these as the appropriate control commands for the robot. We show the validity of our approach by presenting successful experiments on a realistic household-type task—changing rolls on a paper roll holder.

KEY WORDS—Programing by Demonstration, teaching by showing, hybrid dynamic systems, configuration space

1. Introduction

For over two decades now, the robotics research community has been searching for a simpler, more natural method of robot programing. Current methods of programing generally occur at a very low level—usually by writing computer code. An important potential domain for robot application is in domestic environments, especially the home. A simple and natural method for robot programing is essential in such a domain since end users will generally not possess knowledge of either computers or robotics. A method of programing that holds much promise in this regard is robot Programing by Demonstration (PbD). In PbD, the end user *demonstrates* the task to be programed. A PbD interface interprets the demonstration and determines the control details required by the robot to achieve the task. The method provides a simple and very natural programing interface for humans; humans often transfer skills between themselves by *showing* how something is done.

PdD is an active research area and many approaches have been presented. A number of authors have investigated using neural networks to learn from the demonstration a skill mapping that will complete the task. For example, Asada (1990) proposed a multi-layered neural network to learn non-linear compliance strategies used by a human in a chamferless peg-in-hole task. Pomerleau (1991) used a neural network to learn road following skills in an automated car-driving system. Kaiser and Dillman (1996) proposed a neural net approach to learn peg-in-hole and door opening tasks. Other authors in the PbD field have taken a more physics-based approach. They assume some generic model of the skill in the task, and then use the demonstration to fill in the missing "gaps". In some work, the model is quite specific to the task being programed. For example, Atkeson and Schaal (1997) derive a skill model relating angular velocity of a pendulum to the horizontal acceleration of the hand at its pivot in a pendulum swing-up and balance task. They use a human demonstration to help find optimal values for the robot of unknown parameters in the model. In PbD, other authors assume a generic model for skill relevant to a class of tasks. In many cases, the task class of interest is assembly. For example, Asada and Izumi (1989) propose the hybrid force–position control regime (Raibert and Craig 1981) as the skill model for

The International Journal of Robotics Research
Vol. 22, No. 5, May 2003, pp. xxx-xxx,

assembly. They use the demonstration in PbD to determine the unknown parameters of a hybrid force–position controller for a simple place-block-in-corner task. This type of approach is based around a continuous-time description of skill in assembly. An alternative approach is to model skill in an assembly task as a set of discrete states. Skill transfer to the robot can then be made as the sequence of task states used by the human in the demonstration. A number of different regimes for discretizing a task have been proposed. Hannaford and Lee (1991) propose a hidden Markov model approach, McCarragher (1994) proposes a hybrid dynamic system (HDS) as a means for discretizing assembly tasks, while Ikeutchi, Kawade, and Suehiro (1993) suggest the notion of discretization for assembly based on ten primitive contact formations.

The PbD research field is a rich and active one, and many interesting results have been obtained; for a more comprehensive review of the field, see Chen (2001). While PbD holds much promise, a well-known weakness with the approach is that the demonstration can often contain suboptimality. For example, Kaiser, Friedrich, and Dillmann (1995) recognize five sources of suboptimality that can exist in a demonstration: 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 (1996) identified that, in a pick-and-place task through a field of obstacles, a human will naturally introduce "noise" into the demonstration by using different paths to traverse regions where the gap between obstacles is large. Nechyba and Xu (1996) identified that skill models obtained from human-produced training sets can produce trajectories not characteristic of the source process or, even worse, that are potentially unstable. Witvrouw et al. (1996) found that a demonstration of a peg-in-hole task could contain actions by the demonstrator that were suboptimal, erroneous, or even unintended. 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. The solution is to identify and remove noise from the demonstration before any programing of the robot takes place.

In this paper we present a new, two-step approach for coping with demonstration suboptimality in PbD. The two steps are:

(i) determine the geometric properties of the task from demonstration;

(ii) determine optimized robot control commands based on the information obtained in (i).

Our methods for achieving both steps (i) and (ii) are new to the literature. Regarding (i), a number of competing works exist. Some approaches produce an "implicit" representation of task geometry. For example, Koeppe and Hirzinger (2000) train neural nets from demonstration to map sensor signals to appropriate force and velocity control commands. These nets implicitly describe the natural and artificial motion constraints existing in the task, and so do form a representation of task geometry. However, since the representation is implicit, it would be difficult to utilize this type of representation to identify and remove noise from a demonstration. Other approaches derive representations of task geometry that are explicit, but only qualitative. Skubic and Volz (1996) propose the automatic recognition of task object contact formations from demonstration using fuzzy logic. Similar work is proposed by Hovland and McCarragher (1997), and Witvrouw et al. (1996). These works derive a description of task geometry as a particular set of contact formations between task objects. However, since the description is only qualitative, it is less rich than the one we propose in this paper. In our approach, we derive an explicit and quantitative geometric representation of the task. Such a representation provides a rich source of information for building strategies to cope with demonstration suboptimality. Our approach is to use statistical regression analysis on demonstrated paths to derive the "configuration space" (Latombe 1991) for the task. Configuration space (C-space for short) can be viewed as a "geometric" representation of a task that focuses on the constraints on motion caused by one object in the task on another, rather than on the dimensions of the objects themselves.

Regarding step (ii) of our approach, a number of alternative approaches also exist. The problem is one of identifying and removing any noise in a demonstration in order to produce an optimized set of control commands for the robot. Tso and Liu (1997) proposed one approach to this problem, where their aim was to select the most consistent demonstration from among a number of demonstrations of a task. They identified that, although a human can execute varying motion across different demonstrations, all demonstrations will have in common a "core motion" required to complete the task. Their idea was to use a hidden Markov model to select the most consistent demonstration in the group. While this approach can select the demonstration containing the least amount of noise, it is not able to remove noise from a demonstration. This can be a disadvantage, for example, if all demonstrations contain noise.

Another type of approach is proposed by Kaiser and Dillmann (1996) and Witvrouw et al. (1996). In contrast to Tso and Liu (1997), this work is able to *remove* noise from a demonstration. The work uses thresholding, smoothing, and loop-removal techniques to filter out any noise in a demonstration. For example, Kaiser and Dillmann (1996) propose two types of noise removal methods for a peg-in-hole task. First, they remove ineffective actions, i.e., actions that changed the configuration of the peg by less than a pre-defined threshold. Secondly, they remove actions that were later corrected, i.e., those that were partly or fully negated at following time-steps. Removing actions that are negated at following time-steps is

essentially the process of removing loops from the path defined in C-space by the demonstration. Work by these authors has been successful in removing obvious flaws in the demonstration, however it is limited to deriving paths that contain only points existing in an original demonstrated path. As such, less obvious flaws cannot be removed, e.g., corrected actions that do not form explicit loops.

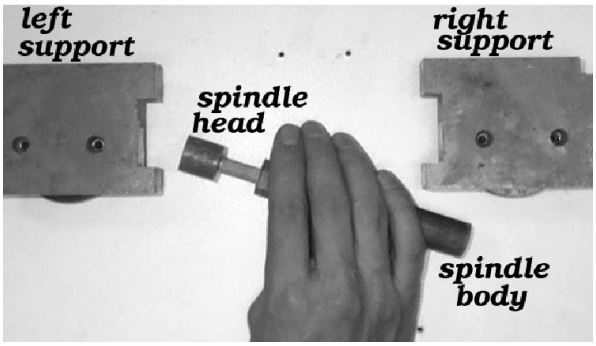
Delson and West (1996) propose a different approach. They address the case where all demonstrated paths start and end at the same point. They then identify an obstacle-free envelope formed by the outermost lying paths. A noise-free path is constructed to lie completely within the demonstration envelope. This approach can derive paths that contain segments that were not explicitly demonstrated, and, as such, it is capable of removing suboptimalities from demonstrated paths that work in Kaiser and Dillmann (1996) and Witvrouw et al. (1996) cannot. However, the approach assumes that the demonstrator will always pass on the same side of any obstacle in the workspace, and it is restricted to finding paths that lie within the obstacle-free envelope. The latter can be a limitation, for example, where only one demonstration is provided. In addition, the method is applicable for C-space of 2 or 3 dimensions. It cannot easily be extended to higher dimensions.

Our approach to step (ii) uses the knowledge of C-space gained in step (i) to make decisions about where a demonstrated path contains noise. Note that our approach to (i) produces only a partial representation of C-space—it provides a representation only in those regions of C-space that were visited by the demonstration. As such, well-known path planning methods (Latombe 1991) cannot be applied to derive a noise-free robot control command. With only a partial knowledge of C-space available, our method for step (ii) is to use the paths traced out in C-space by each demonstration to grow regions in the space where noise removal by traditional path planning techniques can by used. We use the simple and well-known "roadmap" approach to path planning (Latombe et al. 1996) in these regions to determine optimized robot control commands. Our method compares well to others. Like work in Delson and West (1996) it can derive paths containing segments that were not explicitly demonstrated. As such, it can remove a greater range of suboptimalities than for work in Kaiser and Dillmann (1996) and Witvrouw et al. (1996). Compared to Delson and West (1996) our approach has the following advantages: (a) it does not assume that all demonstrations pass between the same start and end points; (b) it can derive paths that lie outside the demonstration envelope; and (c) it can be applied to find paths in C-space of any dimension. A limitation of our approach compared to Delson and West (1996) is that care needs to be applied when tuning the parameters of the method, otherwise a robot control command that is non-obstacle-free may result.

Prior to presenting, in Sections 3 and 4, the details of our solution for achieving steps (i) and (ii), we first formulate in the next section a more precise statement of the problem to be solved for each. To this end, we introduce two approaches to modeling assembly tasks in PbD (note that our focus in this paper is specifically on PbD for assembly type tasks, i.e., tasks involving contact and constrained motion). The first modeling approach we adopt is C-space. We have already introduced the notion of C-space, however our aim in the next section will be to present in more detail a number of C-space concepts important for understanding work that follows. The second level of our modeling approach for assembly is to use a HDS to model the human skill in executing a task. It turns out that a HDS model will allow us to simplify the problem involved in step (ii) of our approach, since it allows the problem of removing noise from an entire demonstrated path to be simplified into one of independent noise removal from distinct segments of that path.

2. Problem Formulation

C-space was first introduced by Lozano-Perez (1983) in the early 1980s to simplify the modeling of tasks involving objects that both translate and rotate; for a more complete presentation on the concept of C-space than will be provided here, see Latombe (1991). The idea with C-space is to represent the configuration of an object with n degrees of freedom (DOF)—translational or rotational—in a physical workspace, as a single point in an n-dimensional *configuration space*. To help make our discussion more concrete, we introduce in Figure 1 the spindle-assembly task, the task adopted as the test bed for our work in this paper. The task requires the insertion of a compressible *spindle* between two, fixed *supports*, and is based on the household task of changing rolls on a paper roll holder. Four degrees of motion freedom exist for the spindle in the task: the position and orientation of the spindle body relative to the supports (y, z and θ), and the compression of the spindle head relative to the spindle body (δ). Then, C-space for the spindle-assembly task will be of dimension four, and a particular configuration of the spindle in the task will correspond to a single point in that space. Configuration space divides topologically into three distinct regions. Taking the spindle-assembly task as an example, the first, C_{free}, contains all points for configurations where the spindle is not in contact with the supports. The second region, C_{obs}, contains points for configurations where the spindle violates (i.e., passes into) the supports. Finally, the third region, C_{con}, contains points lying exactly on the boundary between C_{free} and C_{obs}. C_{con} corresponds in the physical workspace to where the spindle is exactly in contact with one or both supports. It defines a surface (or hyper-surface) in C-space consisting of interconnected patches of curved surface (or hyper-surface) called *C-surfaces*. Recall step (i) of our approach in this paper: to derive a representation of C-space for a task from demonstration. Then we note that finding a representation of C-space really means finding the equations of C-surfaces in C-space.



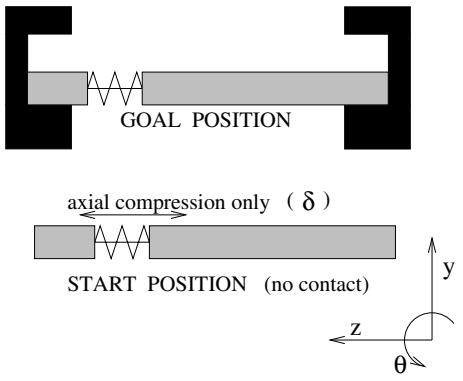


Fig. 1. The spindle-assembly task chosen for PbD.

Since we require some information about a C-surface before its equation can be derived (i.e., in our case, that the demonstration visit the C-surface), the problem we must solve for step (i) of our approach can be stated as: find the equation of every C-surface that was visited during the demonstration. We present a solution to this problem in Section 3.

Once a representation of C-space is found, recall that step (ii) of our approach is to use this information to remove noise from the demonstration. In C-space, the demonstration will define a path through the space. Since an assembly task typically commences with task objects not in contact, the start point of the path will lie somewhere in C_{free}. The final configuration in assembly usually involves task objects in contact, and so the end of the path will lie on one of the C-surfaces that makes up C_{con}. In between, the path will generally traverse a number of intermediate C-surfaces. The problem of removing noise from the entire path is difficult, but can be simplified by breaking down the path into segments (where each segment lies on a distinct C-surface), and then removing noise from each segment independently. Denote as *via-points* the points at which we segment the demonstrated path; that is, these points will lie at a boundary between C-surfaces in C-space, and will define the start and end points of a path segment. Call these points "start" and "end" via-points respectively. Then the problem to be solved for step (ii) in our approach is: find a noise-free robot control command that will traverse a C-surface between the start and end via-point on that surface. A solution to this problem is presented in Section 4.

Our approach has been to simplify the noise removal problem by breaking an entire demonstrated path down into segments and removing noise from these segments in an independent fashion. We note that an important and valid criticism of this approach is that "noise" in a demonstration may not be limited to existing only on individual C-surfaces. That is, it may be that a particular C-surface was visited by the demonstration only because the the demonstration contained noise. Clearly our approach here must be coupled with another, if comprehensive noise removal is to occur. Just such work exists, and has been presented in Chen and McCarragher (2000). A complete presentation of this work is beyond the scope of this paper. However, due to the importance of this work to our work here, we now provide a brief overview of it.

At the core of work in Chen and McCarragher (2000) is the idea of using a HDS to model the skill required for assembly tasks. In its most general form, a HDS involves a continuous-time system interacting with a discrete-event system (Brockett 1993). As applied in robotic assembly (McCarragher and Asada 1995), the continuous-time system represents the continuous-time dynamics of the spindle relative to the supports, i.e., in state space form as (i) a differential equation,

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{1}$$

describing the free-space dynamics of the spindle (where $\mathbf{x(t)}$ is the continuous-time system state vector $[y, z, \theta, \delta]^{\mathrm{T}}$, and $\mathbf{u(t)}$ is the control input vector), and (ii) a set of equations, each of the general form,

$$g(\mathbf{x}(t)) = \mathbf{0} \tag{2}$$

describing the constraints on the spindle's motion caused by any contact with the supports. In contrast, the discrete-event part of the HDS captures the discrete nature of assembly dynamics; to complete an assembly task, we must traverse through a sequence of distinct contact formations until the final "fully-assembled" contact formation is reached. In HDS literature, each distinct contact formation in the task is denoted as a *discrete state*. To help clarify the discussion, we introduce, in Figure 2, a set of demonstrations of the spindle-assembly task. Each of the six demonstrations in the figure is shown as a sequence of distinct contact formations, starting with the no-contact contact formation (labeled as discrete state number 2), passing through a series of intermediate contact formations, to finally reach the fully-assembled contact formation (labeled

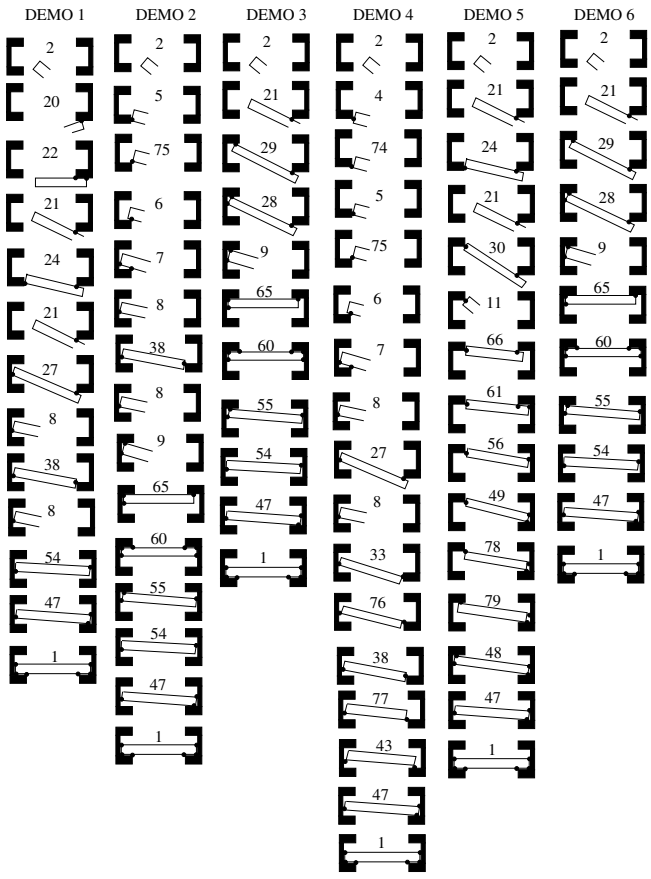


Fig. 2. The set of state sequences demonstrated by the human in the spindle-assembly task.

as state 1). One of the main advantages of the HDS modeling approach is that it provides a two-level description of the skill required for completing an assembly task. The continuous-time system describes the low, time-based-control level of demonstrator skill, while the discrete-event system can describe demonstrator skill at the "task level". Note how the paths of discrete states visited for the spindle-assembly task in Figure 2 can each be interpreted as a task-level *strategy* for completing the assembly. The idea in work in Chen and McCarragher (2000) was that noise removal at the task level could be realized by determining an optimal path of discrete states that would complete the task. To recognize the relevance of this work to our work here, an understanding of the relationship between HDS modeling and C-space is required. We know that each discrete state in the HDS model corresponds to a unique set of constraints on spindle motion. Recall our description of the C-space modeling approach—how each C-surface in C-space corresponds to a unique set of constraints on spindle motion. That is, there exists a one-to-one correspondence between C-surfaces in C-space and discrete states in the HDS model. Each discrete state in the HDS that defines contact (e.g., for the spindle-assembly task, all states shown in Figure 2, except state 2) will correspond to a distinct C-surface in C-space. The no-contact discrete state in the HDS (i.e., state 2 in the spindle-assembly task) corresponds to C_{free} in C-space. Then for our work in this paper, the idea in Chen

and McCarragher (2000) of removing noise from the demonstration at the task level really means selecting a sequence of C-surfaces over which a "noise-free" path through C-space should pass. That is, paths derived by that work determine which C-surfaces should be visited by the robot control command. Our focus in this paper is on removing noise from the segments of path lying on each of these C-surfaces.

3. Constructing Configuration Space

Recall the problem to be solved in the first step of our approach: find the equation of every C-surface that was visited during the demonstration. We note that C-surfaces in C-space for a task with n DOF will range in their dimension from "$n-1$" to zero.[1] Our first step in this section is to identify that any C-surface of dimension "$n-2$" and less can be described as the intersection of a number of "$n-1$"-dimensional C-surfaces. Call a C-surface of dimension "$n-1$" a *primitive C-surface*, and denote the mth primitive C-surface visited in the demonstration as c_m^*. Then our problem in this section really is to find an equation that describes the form of each *primitive* C-surface c_m^* visited in the demonstration. C-surfaces of dimension "$n-2$" and less can then be specified as a set of primitive C-surface equations.

Our approach is to derive equations for primitive C-surfaces by using the well-known statistical "fitting" technique of regression analysis. It is often the case that an unknown, or partially-known, process will generate data from which a description of the process needs to be gained. Regression analysis is a method to achieve just this end. The idea is to presume some generic model, and then to use the data generated by the process to identify the unknown parameters in the model. In our case, we wish to identify the form of a particular primitive C-surface c_m^*. A generic model for c_m^* can be derived; call it the "regression model". The data to determine the unknown parameters of that model must come from the demonstration; specifically the data will consist of the set of points defining the path traced out on c_m^* by the demonstrator. Call this set of points for c_m^* the "data set" of c_m^*. Then the two elements of our approach that require further explanation for each c_m^* are (i) how we determine its data set, and (ii) how we derive its regression model.

3.1. Determining a Data Set

The data set must be determined from our recording of the demonstration. Recall Figure 2, the set of demonstrations provided for this paper of the spindle-assembly task. We show, in Figure 3, the apparatus used to record these demonstrations. It includes a *polehemus* position sensor, used to record the position and orientation of the spindle head and body relative to the inertially fixed supports. Note how transmitters of the polehemus sensor have been attached to the spindle head and body, and the location of its inertially-fixed receiver (situated at the top of the figure). Force sensors mounted beneath each support were used to record the contact forces and moments applied by the demonstrator through the spindle. Note that, although we show sequences of distinct contact formations in Figure 2, a continuous stream of data was recorded from both the polehemus and force sensors for all demonstrations.

Only position data from each demonstration are used directly in the regression process. We denote as c_i the ith C-surface (of any dimension) to be visited in the demonstration set.[2] Our final requirement is to form a data set for each *primitive* C-surface from the stream of position data. However, as a means to this end, we first break up the data stream into segments, where each segment lies on a distinct C-surface c_i. Since any c_i can be visited more than once in the demonstration set, a number of "segments" may be derived for a particular c_i. We group these segments together to form a set of data points p_i for each c_i.

Before moving on to how we form data sets for our primitive C-surfaces from p_i, we first address the valid question of how segmentation of the position data stream was achieved. The force data recorded from the demonstration form a critical part of the segmentation process. Recall how each c_i in C-space corresponds to a particular contact configuration in the task. Work by Skubic and Volz (1996) has looked at automatically identifying from demonstration when changes in contact configuration (i.e., c_i) in a task occur. This work does not require an a priori known task model, but is by no means mature, and further work is required. However, the thrust of their findings was that force and moment signatures could be used to identify when contact configurations change. Then our method for segmentation in this paper should be based on force. Work in Skubic and Volz (1996) looked at identifying the changes automatically; here we implement a force-based approach manually. To give a more concrete idea of how the segmentation process proceeded, we provide an example. Figure 2 shows that, in demonstration 1, the assembly process moved into state 24 from state 21. That is, part of our stream of position data contains points lying on the C-surfaces corresponding to states 24 and 21. The separation of these points into those lying on the C-surface of state 24, and those lying on the C-surface of state 21, was achieved by noting the time at which a spike in the z-direction force data occurred. Of course, the force data contained many spikes, and, in general, it was difficult to know which parts of the force data corresponded to which state changes. To assist the human

1. A C-surface of dimension "$n-1$" corresponds to configurations in the spindle-assembly task where the spindle has lost one degree of motion freedom due to contact with the supports. A C-surface of dimension zero (i.e., a point in C-space) corresponds to a configuration where the spindle has no motion freedom (e.g., the goal configuration in the spindle-assembly task, as shown in Figure 1).

2. Note the difference between our notation here of c_i to denote a C-surface of any dimension, and our previous notation of c_m^* to denote a *primitive* C-surface.

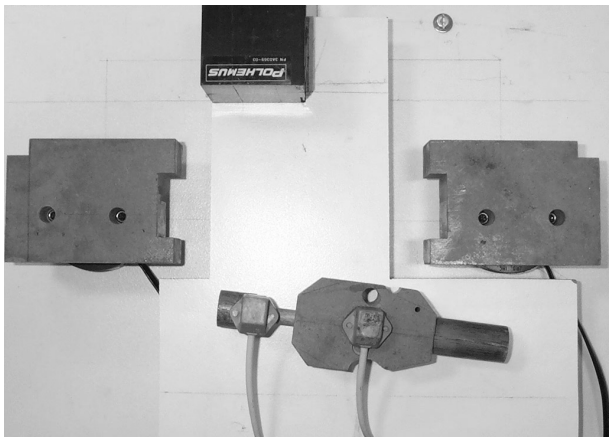


Fig. 3. Apparatus used to capture human's demonstration of the spindle-assembly task.

operator in the segmentation process, we made a time-stamped video recording of each demonstration. We were then able to coarsely determine from the video when a state change occurred, and then use the force data to determine exactly when it occurred.

With a p_i for each c_i determined, the next step is to combine p_i to form data sets for primitive C-surfaces. Where c_i is a primitive C-surface[3] corresponding to c_m^*, then all data points in p_i should be included in the data set of c_m^*. Where c_i is a non-primitive C-surface (i.e., it is of dimension "$n - 2$" or less), recall that non-primitive C-surfaces are defined by the intersection of a number of primitive C-surfaces. Denote as the *primitive set*, the set of primitive C-surfaces that intersect to form a particular non-primitive C-surface c_i. Then the p_i for this c_i should be included in the data set of all primitive C-surfaces that make up its primitive set.

3.2. Deriving a Regression Model

The regression model is an equation of known form, but is one containing parameters that are initially unknown. In our application, the regression model is an equation describing the generic form of a primitive C-surface. It turns out that no single regression model will generically describe the form of all primitive C-surfaces in C-space. The number of regression models will depend on the number of distinct *contact types* that exist in a task. For example, Lozano Perez (1983) identified that in a planar, 3-DOF task involving polyhedral objects, two distinct contact types exist: edge–vertex contacts (i.e., an edge of the manipulated object in the task in contact with a vertex of the environment), and vertex–edge contacts (a vertex of the manipulated object in contact with an edge of the environment). In this case, two regression models are required: one for each contact type. McCarragher (1996) identified that for a 6-DOF task involving polyhedral objects, three contact types exist: surface–vertex, vertex–surface and edge–edge. In this case three distinct regression models are required. The spindle-assembly task is similar to the planar, 3-DOF task analyzed by Lozano Perez (1983), in that it is planar and involves polyhedral objects. However, the task analyzed by Lozano Perez (1983) contained a manipulated object consisting of a single, rigid body. In contrast, the spindle-assembly task contains a manipulated object consisting of two rigid-body objects with a single degree of freedom between them. Then, a total of four distinct regression models are required for the spindle-assembly task. It turns out that two of these are identical to the regression models required for tasks analyzed by Lozano Perez (1983). Assuming that we have a manipulated object consisting of the spindle body alone, the first of these two contact types is where a spindle-body *vertex* makes contact with an *edge* of a support. We show in Figure 4(a) how the primitive C-surface (it will be a primitive C-surface since the single point contact results in the loss of one degree of spindle-body motion freedom) of this contact type is described by the vector equation:

$$({}_{\mathbf{a}}\mathbf{A} + {}_{\mathbf{b}}\mathbf{C} - {}_{\mathbf{a}}\mathbf{B}) \cdot {}_{\mathbf{a}}\mathbf{n} = \mathbf{0}. \tag{3}$$

Here we adopt the notation ${}_aA$ to mean vector A given with respect to frame F_a, ${}_bC$ to mean vector C with respect to frame F_b, etc. The second of these contact types is where an *edge* of the spindle-body makes contact with a *vertex* of a support. We show in Figure 4(b) how the primitive C-surface corresponding to this contact type is described by the vector equation:

$$({}_{\mathbf{a}}\mathbf{A} + {}_{\mathbf{b}}\mathbf{B} - {}_{\mathbf{a}}\mathbf{C}) \cdot {}_{\mathbf{b}}\mathbf{n} = \mathbf{0}. \tag{4}$$

3. c_i is a C-surface of any dimension that was visited by the demonstration; some of these will have dimension "$n - 1$", i.e., they will be primitive C-surfaces.

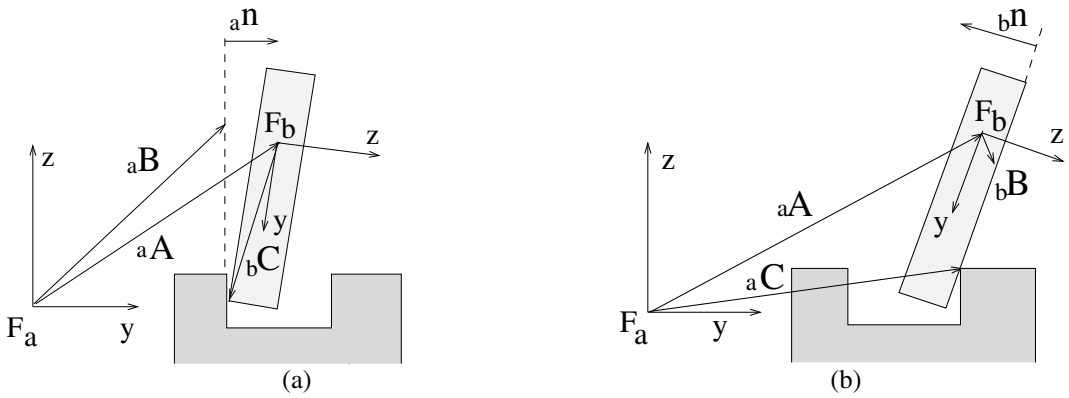


Fig. 4. Regression model derivation for contact type where the spindle body is in contact with the right support.

Both equations (3) and (4) can be expanded to give scalar equations of the form

$$\phi_1(y, z, \theta; b, c, d, e, f) = 0 \tag{5}$$

where y, z, and θ are the position and orientation of the spindle body, and b, c, d, e and f are the regression model's unknown parameters. Note that the parameters in eq. (5) have physical meaning. That is, in Figures 4(a) and (b), pair (c, d) defines a vector C that locates the position of the vertex in each contact, pair (b,b) defines a vector B that locates the line in space of the edge in each contact, and finally, pair (e,f) defines the direction of a unit vector **n** that lies normal to the edge in each contact. The fact that these parameters had physical meaning meant their true values could be obtained by measurement; something we use later to verify the accuracy of parameter estimates obtained by regression analysis.

Equations (3) and (4) form the set of possible regression models for the primitive C-surfaces that exist in a planar task involving a single, rigid-body, manipulated object. For the spindle-assembly task, two additional regression models are required. These additional models correspond to contact types where a spindle head *edge* is in contact with a *vertex* of a support, and where a spindle head *vertex* is in contact with an *edge* of a support. Both case give rise to a scalar equation of the form

$$\phi_2(y, z, \theta, \delta; b, c, d, e, f) = 0; \tag{6}$$

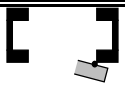
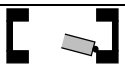
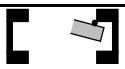
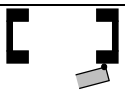
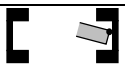
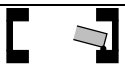
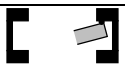
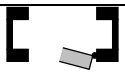
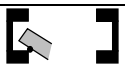
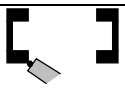
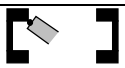
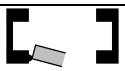
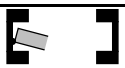
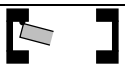
see Chen (2001) for the details of the analysis. Here the additional variable δ in eq. (6) compared to eq. (5) describes the position of the spindle head relative to the spindle body.

3.3. Results

Once the model and data set for a c_m^* are found, the mechanics of regression are straightforward; details can be found in Chen and Zelinsky (2001) or Chen (2001). Recall the set of demonstrations that were provided of the spindle-assembly task (shown in Figure 2). It turned out that fourteen distinct primitive C-surfaces for the task were visited in this demonstration set. We show in Table 1 the regression results for each of these primitive C-surfaces. Recall our notation of c_m^* as the mth distinct primitive C-surface visited in the demonstration. Column 1 in the table indicates the primitive C-surface (i.e., c_1^* to c_{14}^*) for which the results were obtained. Column 2 in the table shows the underlying contact formation corresponding to each c_m^*. Columns 4–8 show two rows of parameter values for each c_m^*. The upper row shows the parameter estimates obtained by regression analysis, where the hat on each symbol b, c, d, e, f signifies that each value is an *estimate*. The lower row shows a set of parameter values obtained by measurement, i.e., the bottom row shows the true parameter values. Note that the units for each parameter value shown is meters (recall our description of the physical meaning of these parameters in Section 3.2). Finally, column 9 in the table lists the states from which data in the data set were obtained (refer to Figure 2 to see the contact configuration corresponding to each one of these states). Recall in Section 3.1 our need to form a data set for each primitive C-surface. Column 9 in the table gives an indication of how much data were available for the data set of each primitive C-surface.

The parameter estimates determined by regression can be seen to range in their accuracy. In some cases the estimates were excellent, e.g., c_{13}^*, c_1^*, c_{12}^*, c_{14}^*, c_9^*, c_{10}^*. These parameter estimates were generally within the range of millimeters from true parameter values. In contrast, Table 1 shows that results obtained for some c_m^* were not particularly accurate, e.g., c_2^*, c_3^*, c_4^*, c_7^*. Clearly, the cause of these less accurate estimates needed to be found. Two requirements existed for accurate parameter estimates. The first was a sufficient amount of data, i.e., that the system of equations formed by the data set in the regression analysis is sufficiently overconstrained. In our case, the position sensor was capable of data output at a rate of 120 Hz, so sufficient data were generally available for all

Table 1. Results of Regression Analysis For the Spindle-Assembly Task

c_m^*	Contact Formation	$\hat{b}$ / b	$\hat{c}$ / c	$\hat{d}$ / d	$\hat{e}$ / e	$\hat{f}$ / f	Data set from states ...
c_1^*		0.008 0.011	0.524 0.525	0.227 0.223	0.019 0	-0.998 1	D_1: 22,21,24,21,27 D_2: 27 D_3: 21,29,28 D_4: 27 D_5: 21,24,21,30 D_6: 21,29,28
c_2^*		0.002 0.011	0.612 0.540	0.316 0.223	0.992 1	0.127 0	D_1: 38 D_2: 38 D_4: 76,38,77
c_3^*		-0.023 0.011	0.523 0.570	0.310 0.223	0.730 0	0.634 1	D_5: 66,61
c_4^*		0.466 0.525	0.086 0.086	0.016 0.011	-0.972 -1	0.234 0	D_1: 20,22
c_5^*		0.183 0.213	0.096 0.086	0.002 0.011	0.033 0	0.999 1	D_1: 54,47,1 $D_2/D_3/D_6$: 55,54,47,1 D_4: 47,1 D_5: 79,48,47,1
c_6^*		0.581 0.540	0.078 0.086	-0.004 0.011	0.996 1	0.088 0	D_1: 47,1 D_2: 47,1 D_3: 47,1 D_4: 77 43,47,1 D_5: 49,78,79,48,47,1 D_6: 47,1
c_7^*		0.447 0.570	-0.067 0.086	-0.118 0.011	-0.997 -1	0.079 0	D_2: 65,60 D_3: 65,60 D_6: 65,60
c_8^*		0.191 0.223	0.118 0.086	0.043 0.011	0.001 0	0.996 1	D_4: 33,76
c_9^*		0.018 -0.011	0.564 0.540	0.372 0.373	0.019 0	-0.998 -1	D_2: 75,6,7 D_4: 75,6,7
c_{10}^*		0.024 0	0.512 0.525	0.353 0.373	-0.999 -1	-0.105 0	D_4: 4,74
c_{11}^*		-0.030 0	0.538 0.570	0.393 0.373	-0.985 -1	-0.120 0	D_5: 11,66,61,56,49,78
c_{12}^*		0.366 0.373	-0.007 0	-0.014 -0.011	0.087 0	-0.998 -1	D_1: 24 D_2: 5,75 D_4: 74,5,75 D_5: 24
c_{13}^*		0.377 0.383	0.005 0	-0.012 -0.011	0.009 0	-0.997 -1	D_1:27,8,38,8,54,47,1 D_3D_6: 28,9,65,60 55,54,47,1 D_2: 7,8,27,8,38,8,9,65,60,55, 54,47,1 D_4: 7,8,27,8,33,76,38,77,43,47,1
c_{14}^*		0.599 0.560	-0.006 0	-0.014 0.011	-0.996 -1	0.084 0	D_2: 9,65,60,55 D_3: 9,65,60,55 D_5: 79,48 D_6: 9,65,60,55

c_m^*. The second requirement for accurate parameter estimation is a good range of data, i.e., that the demonstrator traces out paths over a wide range on the C-surface. It turned out that this was the reason for less accurate estimates in our case.

There were two reasons why a path of limited range may be traced out on a c_m^* in the demonstration. The first was because the c_m^* was only briefly visited, e.g., in the spindle-assembly task—c_3^*, c_4^*. For example, c_4^* was briefly visited; existing in only two states (20 and 22) in demonstration 1. This has resulted in the less accurate estimates shown for c_4^* in the table. What is required for this type of c_m^* is a larger demonstration set so that more paths on distinct parts of the C-surface become available. That is, the demonstration must contain sufficient information about a region in C-space if our method is to determine an accurate representation of the region. The second reason for limited path range on a c_m^* was because the geometry of the task limited the range of motion that could be demonstrated, i.e., that a c_m^* only exists over a small region

in C-space. For example, the final column in Table 1 shows that c_6^* was visited often in the demonstration. Then we would expect a good range of paths on this primitive C-surface, and a set of precise parameter estimates to match. However, there is a difference between the parameter estimate values and their true values for c_6^* of up to 10 percent. In this case, the range of motion that can be demonstrated is naturally limited by the geometry of the task, i.e., the spindle cannot move very far from an orientation aligned with the task's z-axis since it is lying between the rebates in each support. Although parameter estimates for this type of c_m^* can, in some cases, differ by up to 15 to 20 percent from the true values, these estimates still do in fact provide an accurate description of the c_m^* over the limited range of motion allowed by the task. That is, for noise removal purposes these parameter estimates provide a useful description of the c_m^*. Noise removal means deriving noise-free paths that lie on a c_m^*. Parameter estimates that describe a c_m^* well over the limited range allowed by the task are useful because our derived noise-free path will move onto a new C-surface before reaching regions on the c_m^* described badly by the parameter estimates. The majority of the c_m^* with less accurate parameter estimates in Table 1 do so for this reason, e.g., c_2^*, c_5^*, c_6^*, c_7^*, c_8^*, c_{11}^*. So many c_m^* of this type exist for the spindle-assembly task because only a very limited range of spindle motion is possible when it is close to fully inserted. Note how the c_m^* just listed correspond to contact formations occurring when the spindle is lying somewhere between the rebate in each support.

To finalize this section, we draw two conclusions. First, demonstration followed by regression analysis is a valid method for generating a description of C-space. In the majority of cases, a valid and useful representation of C-space will be obtained. A less accurate representation can be obtained for regions that were visited infrequently in the demonstration; these cases require further demonstrations that visit the region more thoroughly. Secondly, in cases where motion is limited in the demonstration due to task geometry, the parameter estimates obtained for a C-surface can be significantly different to true values. However, they still constitute an accurate C-space description of the C-surface over the limited range that it exists, and are useful for path planning purposes. With our description of C-space derived, we turn our attention now to formulating noise-free, robot control commands.

4. Deriving Robot Control Commands

Recall from Section 2 the problem to be solved here. Each c_i will, in general, have a number of path segments from the demonstration that traverse it. We denote as $\hat{\mathbf{u}}_{ij}$ the jth demonstrated path segment existing on c_i. In Section 2 we noted that (a) $\hat{\mathbf{u}}_{ij}$ will have start and end points (we called these via-points) lying on the boundary of c_i, and (b) that $\hat{\mathbf{u}}_{ij}$ may not define an efficient path due to the possible inclusion of noise in the demonstration by the human. Then our problem in this section is to derive an efficient, noise-free robot control command that passes between the start and end via-points of $\hat{\mathbf{u}}_{ij}$. We denote such a command, for $\hat{\mathbf{u}}_{ij}$, as $\mathbf{u}_{ij}$. We now commence the presentation of our method with an overview.

4.1. Overview

Work in Section 3 did not derive a full knowledge of C-space, and so not all boundaries on a particular c_i are guaranteed to be known. To help show what we mean, we present in Figure 5 an example two-dimensional (2D) C-surface where boundaries A, B and C were visited in the demonstration and are known, while D is unknown. The figure shows that some portion of a known boundary of c_i may exist behind an unknown boundary, e.g., C3 and B3, and hence does not really divide an "obstacle-free" region on the C-surface from an "obstacle-defining" one. We note that a boundary on a C-surface is guaranteed to divide obstacle-defining and obstacle-free regions along any segment traversed in the demonstration, e.g., segments 1, 3, and 5 in Figure 5. We call such segments, *boundary segments*, and observe that if the C-surface is of finite size (as is usually the case), then a point immediately in front of the boundary segment will be obstacle-free, e.g., points Q_1, Q_2, and Q_3. We use this observation as the basis for growing obstacle-free regions on the C-surface. We grow a free region in front of each boundary segment of the C-surface. If the region is grown very small, then we are guaranteed that it will be obstacle-free. However a small region is of limited use for path planning purposes. Hence we grow a region of useful size and accept that the region will only likely be obstacle-free. We call such a region a *likely-free region*. Once a likely-free region is identified, we use a roadmap type approach to path planning, similar to that presented in Latombe et al. (1996). We randomly generate points within the region, and use a simple path planner to create a connectivity graph $\mathcal{L}$ that records which points have an obstacle-free path between them. Besides points in likely-free regions, we also know that points in demonstrated paths interior to known boundaries are obstacle-free, e.g., points in segments 2 and 4. We call such segments *interior segments*. We use the same simple path planner to create a graph $\mathcal{D}$ that records the connectivity between points in different interior segments. Finally we combine graphs $\mathcal{L}$ and $\mathcal{D}$ into a graph $\mathcal{K}$ that represents the connectivity of all obstacle-free points on the C-surface. We identify the nodes in $\mathcal{K}$ that represent the start and end via-points of $\hat{\mathbf{u}}_{ij}$. We search for the minimum cost path between these nodes to give the final noise-free path $\mathbf{u}_{ij}$. Four main steps are involved in our method:

- creating boundary segments;
- growing likely-free regions;
- creating interior segments;

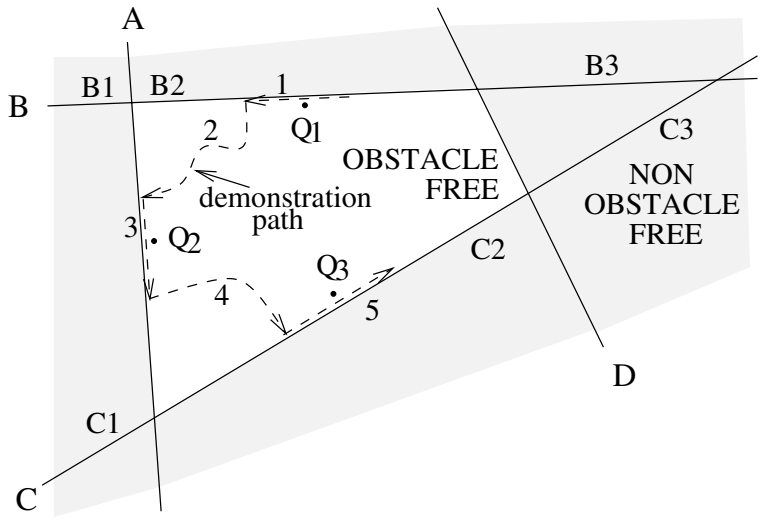


Fig. 5. Demonstration segments identify regions on a C-surface that are likely to be obstacle-free.

- creating a connectivity graph $\mathcal{K}$, and searching $\mathcal{K}$ for our final noise-free path.

We present the details of each step in the following four subsections.

4.2. Creating Boundary Segments

Three steps are required to create boundary segments for c_i: (i) finding boundary C-surfaces to c_i; (ii) identifying *raw* boundary segments for the boundary C-surfaces found in (i); and (iii) projecting points in raw boundary segments to create a set of *clean* boundary segments. To achieve (i), recall that each c_i has associated with it a "primitive set", the set of primitive C-surfaces that intersect to define it. We denote the primitive set for c_i as Ωi. Then we choose a *boundary C-surface* of c_i as any other C-surface visited in the demonstration that has a primitive set Ω_{bnd}, where $\Omega_{bnd} \supset \Omega_i$. Step (ii) is then straightforward. We select as a raw boundary segment, any path demonstrated on a boundary C-surface. Step (iii) is required because points in raw boundary segments will not generally lie exactly on the C-surface of the boundary state. That is, the regression analysis of the previous section derived parameters resulting in primitive C-surface equations that *best fit* raw demonstration data. We create clean boundary segments by solving a simple optimization problem: for each point in the raw boundary segment, we find the point on the boundary C-surface lying closest to it. From now on, we refer to clean boundary segments simply as *boundary segments*.

We show as an example in Figure 6 the results of creating boundary segments for the C-surface corresponding to state 8 of the spindle-assembly task. We have denoted this C-surface as c_5; it was the fifth distinct C-surface visited in the demonstration set. The figure shows points on the boundary surfaces of c_5, i.e., on C-surfaces corresponding to states 7, 33, 9, and 54 in the task. Note that we present boundary segment points in the figure as spindle configurations in the physical workspace rather than as single points in C-space. This is necessary due to the difficulties involved with graphically presenting points in a four-dimensional space. Note also that only a subset of points in each boundary segment are presented for reasons of clarity, where the subset was chosen to reflect the range of points existing in the boundary segment. The main thing to notice about the examples in Figure 6 is that points in each boundary segment result in a precise contact between the spindle and supports. That is, configurations shown in Figure 6 do not see the spindle lose contact with, or pass into, the supports. This is a consequence of the projection step in the process, i.e., we ensured that points in the final boundary segments lay exactly on the boundary C-surface to c_i.

4.3. Growing Likely-Free Regions

We grow a likely-free region by generating a region of points on our C-surface c_i immediately in front of a boundary segment. Each point in the region is determined as follows. First, a point R in the boundary segment is randomly selected. Next, a distance value dst[4] is randomly chosen using the uniform probability distribution over the interval $(0, \mathsf{md}]$. Parameter md denotes a maximum distance value, and determines how big the likely-free region is grown. A point in the likely-free region is then determined by generating a point Q that (a) lies a distance dst from R, and (b) lies exactly on our C-surface

4. Note that we use symbols in plain upright text to denote parameters of our $\mathbf{u}(t)$ derivation method.

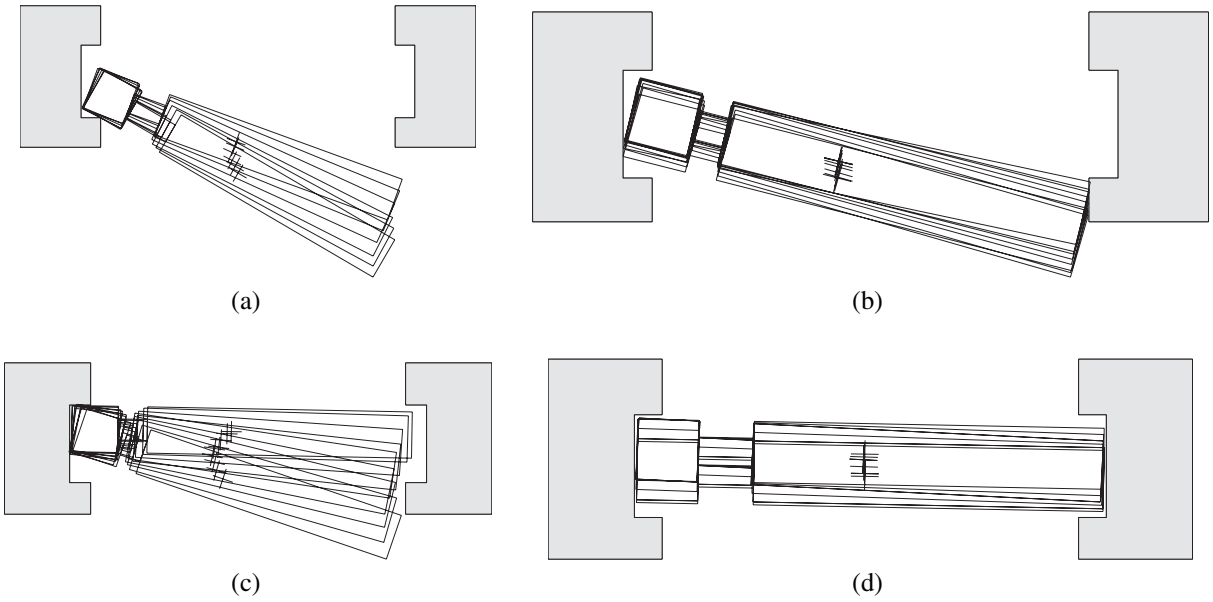


Fig. 6. Four examples of boundary segments for state 8 in: (a) state 7; (b) state 38; (c) state 9; (d) state 54.

c_i. We know that, in addition to conditions (a) and (b), point Q must also lie within known C-surface boundaries. We refer to the region on c_i lying within known boundaries as the *bounded region* for c_i. For example, we present in Figure 7 a 2D C-surface with six boundaries c_1 to c_6, each described by equations $\phi_1 = 0$ to $\phi_6 = 0$, respectively. Here, the bounded region consists of the union of regions labeled 1, 2, and 3. Note how the bounded region defines a set of points on c_i that are obstacle-free, i.e., we wish to generate only Q lying within the bounded region of c_i. We now present the details of how we ensure that only Q lying inside the bounded region are generated.

Let $\phi_{bnd} = 0$ be the equation of the C-surface c_{bnd}, a boundary C-surface to c_i. That is, equation $\phi_{bnd} = 0$ defines a known boundary on our C-surface c_i. Then we identify that any point Q we generate will satisfy one of the following conditions:

$$-\mathsf{eps} < \phi_{bnd}\,|_Q < \mathsf{eps} \tag{7}$$

$$\phi_{bnd}\,|_Q \leq -\mathsf{eps} \tag{8}$$

$$\phi_{bnd}\,|_Q \geq \mathsf{eps} \tag{9}$$

where $|_Q$ denotes the equation ϕ_{bnd} evaluated at point Q, and where eps is a parameter of small value. If eq. (7) is satisfied, then Q lies on, or very close to the boundary, while if eq. (8) or eq. (9) are satisfied, Q lies to one side of the boundary. We are never interested in Q that satisfy eq. (7). A Q lying exactly on the boundary is not obstacle-free, and a Q lying very close to the boundary may not be obstacle-free (recall that the equations we derived for primitive C-surfaces in Section 3 are best estimates that contain some errors). Rather, we are interested in generating Q that satisfy either eq. (8) or eq. (9), where our choice between eqs. (8) and (9) will depend on which side of the boundary is obstacle-free.

In general, there will be more than a single known boundary to our C-surface c_i. Let α denote generally the number of known boundaries on c_i. We denote the equations of boundaries 1 to α respectively as

$$\phi_{bnd\_1} = 0, \quad \phi_{bnd\_2} = 0, \quad \ldots \quad , \phi_{bnd\_\alpha} = 0. \tag{10}$$

We note that a Q which satisfies one or more equations in eq. (10) will lie on a boundary of the bounded region. We have seen that we are not interested in such Q. Rather, we want Q that lie on the obstacle-free side of all boundaries by at least a distance eps. That is, we must form a set of inequalities by *casting* each equation in eq. (10) to be an inequality of the form (8) or (9). We call such a set of inequalities a *boundary inequality set*. Then we note that a boundary inequality set defines a region on our C-surface c_i. For example, one boundary inequality set for the boundary equations $\phi_1 = 0$ to $\phi_6 = 0$ of our example in Figure 7 would be

$$\begin{aligned} &\phi_1 \geq \mathsf{eps}, \quad \phi_2 \leq -\mathsf{eps}, \quad \phi_3 \leq -\mathsf{eps}, \\ &\phi_4 \geq \mathsf{eps}, \quad \phi_5 \leq -\mathsf{eps}, \quad \phi_6 \leq -\mathsf{eps}. \end{aligned} \tag{11}$$

Then the region on the 2D C-surface in Figure 7 that is defined by the boundary inequality set in eq. (11) is region 1. That is, a point Q is guaranteed to lie in region 1 if it satisfies all inequalities in eq. (11). We note that, in general, the bounded

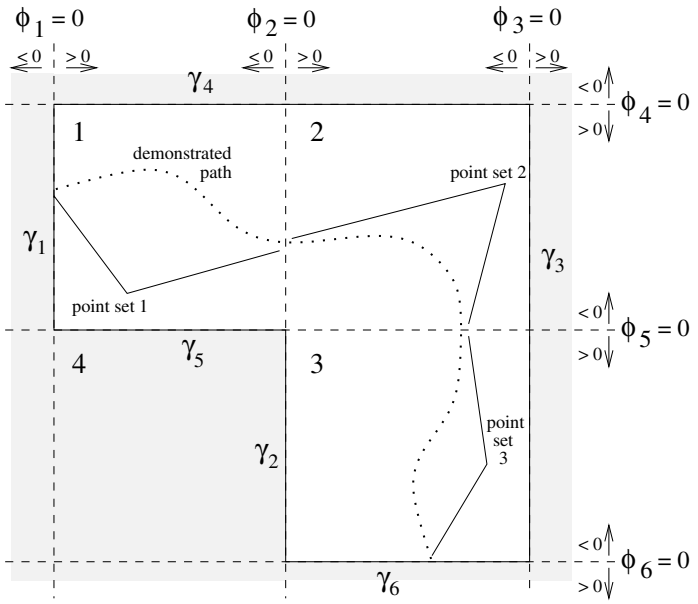


Fig. 7. Example of how demonstrated points determine a set of valid bounded subregions.

region on c_i cannot be specified by a single boundary inequality set. We denote as a *bounded subregion* the region defined by a single boundary inequality set, i.e., region 1 in Figure 7 is a bounded subregion. Then, we identify that the bounded region of a C-surface can be specified as a union of bounded subregions. For example, the bounded region in Figure 7 is given by the union of bounded subregions 1, 2, and 3. A point Q can then be tested to see if it lies within the bounded region by testing to see if it lies within any of its component bounded subregions. The question, then, is how to select the set of bounded subregions that make up our bounded region. Note that not all bounded subregions that can be generated for a certain set of boundary equations will be obstacle-free. For example, region 4 in Figure 7 is a valid bounded subregion, however region 4 is not obstacle-free.

Our solution is to use the set of demonstration points on the C-surface to determine what bounded subregions are valid. Let μ be a possible bounded subregion for c_i, and Υ be the boundary inequality set that defines μ. Then we say that μ is a valid bounded subregion if there exists a point in the demonstration set on c_i which satisfies all inequalities in Υ. For example, in Figure 7 the demonstrated points in point set 1 show region 1 to be a valid bounded subregion because they satisfy all inequalities in eq. (11). Similarly, demonstration point sets 2 and 3 show regions 2 and 3 as valid bounded subregions. Region 4 is not included as a valid bounded subregion because it does not contain any demonstrated points. This approach is a conservative solution because we may miss some valid bounded subregions in which no demonstrated points exist. However, we take this approach because it guarantees that we do not generate Q lying in bounded subregions that define obstacles.

We now present some examples of the likely-free region generation process. We show in Figure 8 the points in four likely-free regions that were generated on c_5, the C-surface of state 8 in the task. The likely-free regions shown were grown from boundary segments in state 7 (Figure 8(a)), in state 38 (Figure 8(b)), in state 9 (Figure 8(c)), and in state 54 (Figure 8(d)). Note that, for clarity, we present only a subset of all points generated for each likely-free region. There are five main things to note about the spindle configurations in Figure 8.

(1) All configurations correspond to points in C-space that have been generated a distance eps away from boundaries on c_5. For example, in Figure 8(c) all configurations maintain some distance between the base of the spindle and the lower support.

(2) There exist, for all boundary segments in Figure 8, precise contacts between the spindle head and the rebate in the top support. This results because we enforced condition that points Q were generated lying exactly on a c_i.

(3) All configurations lie in a region generally close to the boundary state. Recall that configurations lying "far" from a boundary state are undesirable because they may violate unknown boundaries on a c_i. Note in Figure 8

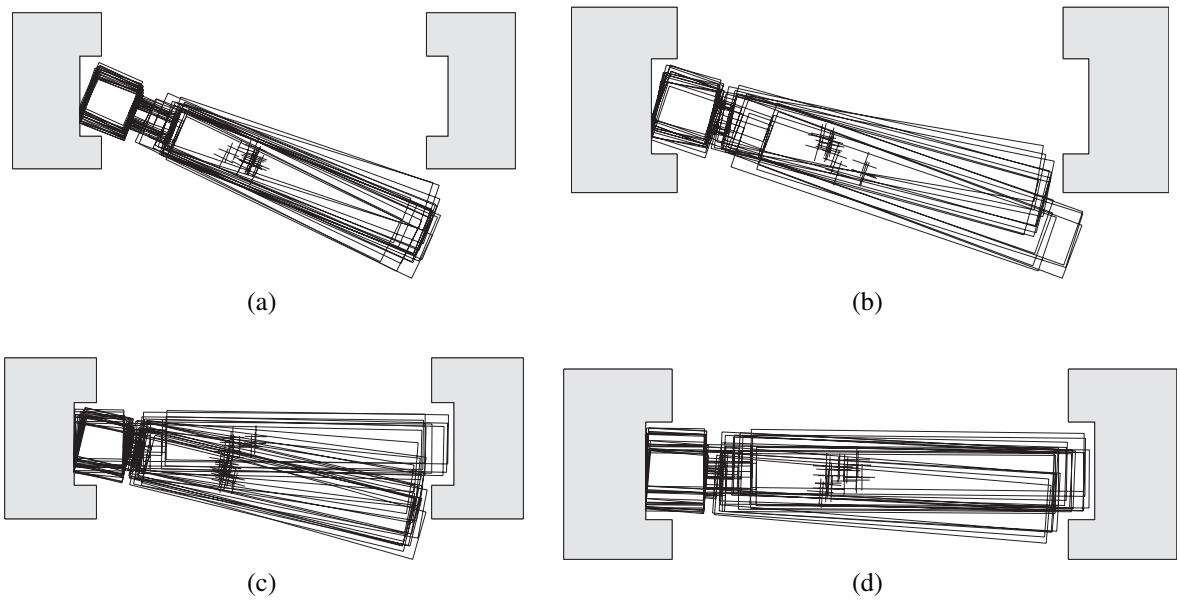


Fig. 8. Spindle configurations generated for a number of likely-free regions in state 8. (a), (b), (c) and (d) correspond to likely-free regions generated from boundary segments in states 7, 38, 9, and 54, respectively.

how we do not produce such configurations. This is a consequence of limiting points generated on c_5 to lie a distance less than md from boundary segments.

(4) All configurations correspond to points in C-space that do not violate known boundaries. For example, note how in Figure 8(c) there exists no configuration where the bottom of the spindle passes into the lower support, even though such configurations are "close" to our boundary segment in state 9. This occurs because our method only generates points for a c_i on the obstacle-free side of known boundaries.

(5) Finally, note how each of the bounded regions in Figure 8 define, by themselves, only a small obstacle-free region on c_5, but that together they provide an obstacle-free region covering most of the area of interest on this C-surface.

4.4. Creating Interior Segments

We know that, in general, a number of demonstrated paths will exist on our C-surface c_i; we have denoted the jth demonstrated path on c_i as $\hat{\mathbf{u}}_{ij}$. Our aim here is to derive, for each $\hat{\mathbf{u}}_{ij}$, an *interior segment*. There are two requirements on interior segments to which $\hat{\mathbf{u}}_{ij}$ do not comply: (i) points in $\hat{\mathbf{u}}_{ij}$ will not lie exactly on c_i (i.e., work in Section 3 derived equations that best fit the data points); and (ii) the start and end points in $\hat{\mathbf{u}}_{ij}$ do not lie exactly on the boundaries of c_i. We achieve (i) by projecting points in $\hat{\mathbf{u}}_{ij}$ onto c_i using the same simple optimization process as described in Section 4.2. We achieve (ii) by projecting the start and end points of $\hat{\mathbf{u}}_{ij}$ simultaneously onto c_i and the relevant boundary C-surface.

In Figure 9 we show an example of the interior segment generation process. The figure shows three interior segments that were produced on c_5. An interior segment is shown that passes through state 8: between states 7 and 27 in Figure 9(a), between states 27 and 38 in Figure 9(b), and between states 38 and 54 in Figure 9(c). There are three things to note about the spindle configurations in each interior segment.

(1) There exist precise contacts between the spindle head and the rebate in the top support. Precise contacts result here because we projected points in raw interior segments to lie exactly on c_5.

(2) All configurations (except the start and end configurations) in each interior segment correspond to points on c_5 that lie within known boundaries. As such, the interior segments shown do not accidentally cause spindle-support collisions that would see the assembly move into an incorrect state.

(3) The start and end configurations of each interior segment lie exactly in boundary states to state 8. For example, the start and end configurations in Figure 9(c) correspond to points lying exactly on the C-surfaces of state 38 and 54. Start and end configurations lying in boundary states result because we projected start and end points of raw interior segments onto the C-surfaces of boundary states to state 8.

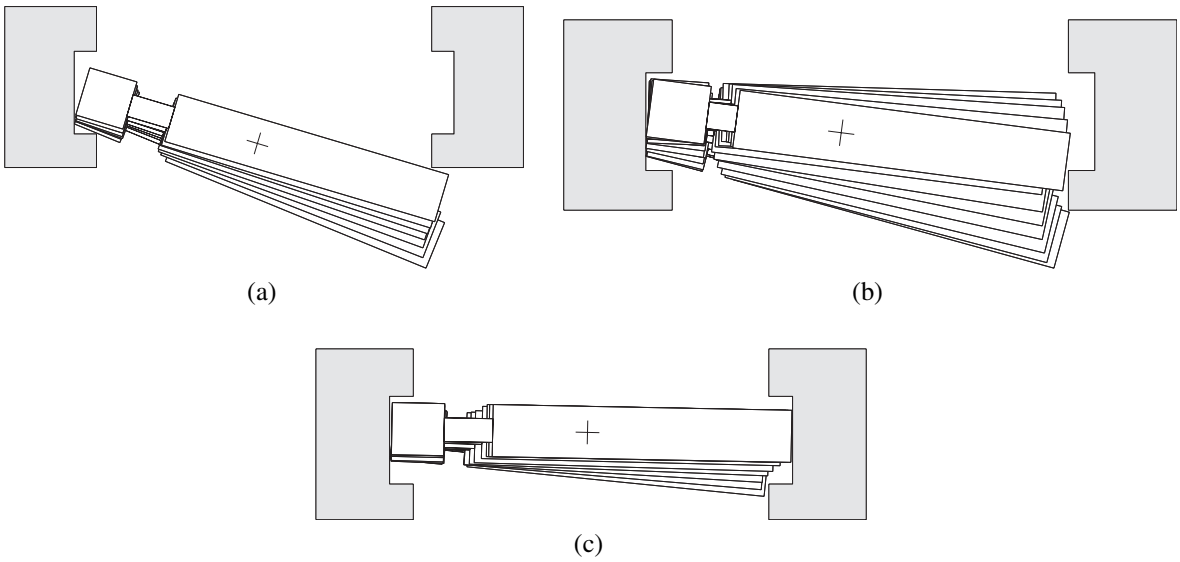


Fig. 9. Three interior segments determined for state 8. The interior segments were derived from paths demonstrated in state 8, i.e., from demonstration 4 (7-8-27) in (a), from demonstration 2 (27-8-38) in (b), and from demonstration 1 (38-8-54) in (c).

4.5. Creating a Connectivity Graph

The previous two sections have been devoted to generating points on our C-surface c_i. Two types of points were generated: points in likely-free regions, and points in interior segments. We show in Figure 10 a possible outcome of the point generation process for a simple 2D planar c_i. The unshaded region in the figure corresponds to the bounded region of our 2D C-surface. Three likely-free regions on our bounded region have been identified (labeled A, B, and C). In addition, two interior segments (labeled η_1 and η_2) were also generated. Our aim in this section is to create a graph $\mathcal{K}$ that represents the connectivity between all generated points on a c_i. For example, in our example in Figure 10, we wish to create a $\mathcal{K}$ that represents the connectivity between all points lying in likely-free regions A, B, and C and interior segments η_1 and η_2. Such a graph should have a node to represent each point in a likely-free region or interior segment. It should have arcs existing between nodes whose points are connected by an obstacle-free path. In addition, we assign to each arc in the graph a value equal to the cost of traversing the obstacle-free path it represents. We construct this graph $\mathcal{K}$ in three steps: (i) we create a graph $\mathcal{L}$ representing the connectivity between points in likely-free regions; (ii) we create a graph $\mathcal{D}$ representing the connectivity between points in interior segments; and (iii) we combine graphs $\mathcal{L}$ and $\mathcal{D}$ into $\mathcal{K}$.

4.5.1. Creating $\mathcal{L}$

We first create a point set consisting of all points in all likely-free regions on c_i. The nodes in $\mathcal{L}$ are created on a one-to-one basis with points in the point set. To construct arcs in $\mathcal{L}$ we must determine two things for each pair of points in the point set:

(a) if the points are connected;

(b) the cost of traversing between connected points.

We say that two points Q_1 and Q_2 in the point set are connected if two conditions are satisfied. First, that the straight-line path between Q_1 and Q_2 is obstacle-free, i.e., that every point on such a line lies within the bounded region of the C-surface. For example, points Q1 and Q2 in Figure 10 have a straight-line path between them that lies fully inside the bounded region. Secondly, that the distance between Q_1 and Q_2 does not exceed a "maximum-connected-distance" parameter mcd_l. We apply the second condition for the following reasons.

(1) Note that our point set will consist of points from distinct likely-free regions. We do not want to connect points in likely-free regions that lie far apart, e.g., between points in likely-free regions A and C in Figure 10. Using this approach we are less likely to connect points on c_i between which an unknown boundary exists.

(2) Intermediate points on the straight line between Q_1 and Q_2 will not in general lie exactly on the C-surface due to its curvature (eqs. (5) and (6) are not linear in the parameters of C-space). However, if Q_1 and Q_2 do not lie too far apart, then intermediate points will lie close enough to the C-surface for the purpose of checking whether they are obstacle-free.

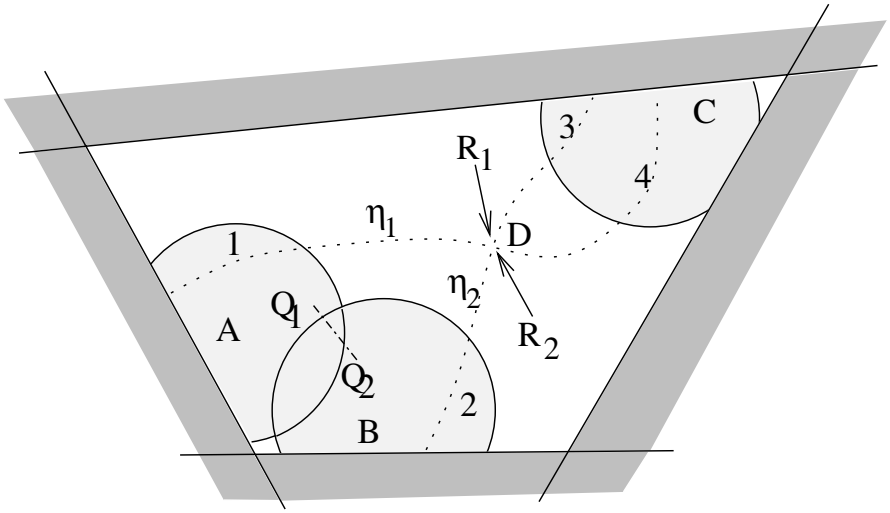


Fig. 10. Example of points generated by our method for a simple, planar C-surface.

(3) Applying such a condition is advantageous from a computational point of view. The number of points to be tested for connection increases rapidly as the allowed distance between the points increases.

(4) Finally, testing for connectivity only between points lying a distance less than mcd_l apart does not really detract from the end performance of our method. That is, points lying far apart that really should be connected, (e.g., those existing in the same, or overlapping, likely-free regions), will be connected efficiently at the final stage of the process; in general, they will be connected by a path that passes through a sequence of intermediate points, where intermediate points are separated by a distance of less than mcd_l.

Our second requirement for creating the arcs in $\mathcal{L}$ was (b) to determine their cost. We denote as ϖ_1 and ϖ_2 the nodes in $\mathcal{L}$ that represent points Q_1 and Q_2, and as ϑ the arc in $\mathcal{L}$ that connects ϖ_1 and ϖ_2. Then we calculate the cost of ϑ as the distance between Q_1 and Q_2. Distance is the appropriate measure of cost here, since we desire that a minimum cost path in the final connectivity graph $\mathcal{K}$ represents the shortest distance path on our C-surface.

4.5.2. Creating $\mathcal{D}$

We create $\mathcal{D}$ in two steps:

(i) create a distinct connectivity graph for each interior segment on c_i (e.g., distinct connectivity graphs for η_1 and η_2 in Figure 10);

(ii) combine graphs constructed in (i) into $\mathcal{D}$.

Step (i) is straightforward because the connectivity of points in any interior segment is known, i.e., apart from the start and end points, each point is connected to two other points, the previous and following points in the path. We create a graph for each interior segment with nodes and arcs that reflect this connectivity. We assign as costs to the arcs in the graph, the distance between each sequential point pair in the interior segment.

In step (ii) we must combine the graphs created in step (i) into a single graph $\mathcal{D}$. The process of combining these graphs means deciding if and where interior segments on c_i *intersect*, That is, we should create a connecting arc between two graphs $\mathcal{D}_1$ and $\mathcal{D}_2$ derived in step (i) when the interior segments η_1 and η_2 represented by these graphs are found to intersect, e.g., at point D in Figure 10. We say that η_1 and η_2 intersect at points R_1 (in η_1) and R_2 (in η_2), when the distance between R_1 and R_2 is less than a parameter mcd_d. If this is the case, an arc is connected between the node in $\mathcal{D}_1$ that represents R_1 and the node in $\mathcal{D}_2$ that represents R_2. Then $\mathcal{D}$ is created by repeating this process for every possible R_1 and R_2 in our set of interior segments on c_i.

4.5.3. Creating and Searching $\mathcal{K}$

We create $\mathcal{K}$ by combining graphs $\mathcal{L}$ and $\mathcal{D}$. The idea is to connect points in interior segments with those in likely-free regions where an interior segment passes through a likely-free region (e.g., along segments labeled 1,2,3 and 4 in Figure 10). We create a $\mathcal{K}$ that represents such connectivity as follows. For each point R in each interior segment η, find a point Q in any likely-free region that lies within a distance mcd_l away. If such Q exist, then for each Q found, create an arc between the node that represents point R in $\mathcal{D}$, and the node

that represents point Q in $\mathcal{L}$. Set the cost of the arc to the distance between the points R and Q.

Once $\mathcal{K}$ is achieved, then the solution to our problem of deriving a noise-free control command $\mathbf{u}_{ij}$ is straightforward. $\mathcal{K}$ represents the connectivity of obstacle-free points on c_i. Two of the nodes in $\mathcal{K}$ are guaranteed to represent the start and end via-points of $\hat{\mathbf{u}}_{ij}$. Then to derive $\mathbf{u}_{ij}$, we simply search in $\mathcal{K}$ for the minimum cost path between the nodes that represent the start and end via-points of $\hat{\mathbf{u}}_{ij}$. Each node in this minimum cost path will represent a distinct point on c_i, with $\mathbf{u}_{ij}$ formed as the sequence of these distinct points.

4.6. Setting Parameters to Appropriate Values

The method we have presented for determining a $\mathbf{u}_{ij}$ is based around five parameters, i.e., md, dst, eps, mcd_l and mcd_d. A key issue in ensuring good performance from our method regards setting these parameters to appropriate values. To this end, we now introduce a number of simple metrics that can be used to set parameter values. These metrics were used to determine parameter values for experiments in this paper.

Parameter **md** is the most important parameter in our method. Recall that the value of md determines how large a likely-free region is grown. A balance must be struck in setting md between the benefit of large likely-free regions for path generation purposes, and the risk of generating points that lie on the non-obstacle-free side of unknown boundaries. We note that a good gage of the value of md can be made by looking at the length of the boundary segment from which the likely-free region is grown. The probability of an unknown boundary passing close to our boundary segment, but not passing through it, decreases as the length of the boundary segment increases. That is, the larger the boundary segment, the larger md should be made. For our experiments in this section md for each boundary segment was set to a value of the length of the boundary segment multiplied by a constant value three. We found empirically that the value three provided conservatively sized likely-free regions that were still large enough to be useful for path planning purposes. Once md is determined, parameter **dst** is also determined. Recall that we set the value of this parameter as a random value between md and zero.

The purpose of parameter **eps** is to ensure we do not generate points in likely-free regions close to, or exactly on, known boundaries to a C-surface. The reason for this approach was because the regression analysis of the previous section provided only an estimate of the true equation for each boundary on our C-surface, i.e., the equation will contain some error. A good estimate of the amount of error is provided by the root mean squared error (RMSE), a metric often calculated in regression analysis for determining the "goodness of fit". Essentially the RMSE is the average distance between a fitted surface and the raw data points from which the surface was regressed. A RMSE value can be determined for each boundary on our C-surface. Then, for our experiments in this section, we set eps equal to three times the RMSE value determined for a particular boundary when growing a likely-free region from that boundary. Three times the RMSE will encompass 99.8 percent of the raw data points if the error in the data points is a normally distributed random variable, and so provides a conservative distance away from the boundary from which points in our likely-free region can be generated.

Parameter **mcd_l** had the purpose of deciding when two points in the same, or distinct, likely-free regions should be tested for connection. Recall that we did not want to connect points in likely-free regions that lay far apart, i.e., because of the risk that an unknown boundary to the C-surface may exist between them. Our aim was to connect points in the same likely-free region, or points in distinct likely-free regions lying close together or intersecting. Then, a value for mcd_l for a C-surface that we found worked well was 5 percent of the average size of likely-free regions on the C-surface, i.e., 0.05 times the average value of md for the likely-free regions on the C-surface.

Parameter **mcd_d** was used to determine if two interior segments on a C-surface intersected. The value we used for this parameter was the maximum distance between any two sequential points in an interior segment on the C-surface. That is, even if interior segments happened to intersect where sequential points in one of them lay the maximum distance apart, the method would still correctly identify the intersection.

4.7. Results

With a method for determining a value for each parameter in our approach, we can now present the results for deriving a set of $\mathbf{u}_{ij}$ for the spindle-assembly task. For the experiments reported on in this paper, we derived a $\mathbf{u}(\mathbf{t})$ for every $\hat{\mathbf{u}}_{ij}$ in the demonstration set. A full set of results for these experiments can be found in Chen (2001). However, in this paper, we show only a subset of the results presented there. We show (in Table 2) the results of deriving $\mathbf{u}_{ij}$ for the $\hat{\mathbf{u}}_{ij}$ existing in demonstration 1 and segments of demonstrations 2 and 5. Note that we have selected for presentation in Table 2 a subset of results from these experiments that will allow us to draw out, in a more compact discussion, all the important conclusions identified for the method in Chen (2001).

Table 2 presents the experimental results in five columns. Column 1 shows the $\mathbf{u}_{ij}$ for which the result was obtained. Columns 2 and 3 indicate the state number in the HDS, and the C-surface c_i, on which each particular $\mathbf{u}_{ij}$ was derived. Columns 4 and 5 show the two important results of the experiments. First (in column 4) how much noise removal did $\mathbf{u}_{ij}$ achieve, and second (in column 5) did $\mathbf{u}_{ij}$ really define an obstacle-free path? Regarding column 5, noise removal means that the derived path encoded a more direct route than the original demonstrated path $\hat{\mathbf{u}}_{ij}$. In some cases, derived $\mathbf{u}_{ij}$ achieved significant noise removal compared to the demonstrated path, e.g., $\mathbf{u}_{4,1}$, $\mathbf{u}_{5,1}$, $\mathbf{u}_{7,1}$, $\mathbf{u}_{1,2}$, etc. For these cases we have entered a "Yes" in column 5 of the table.

Table 2. Results of Our u(t) Derivation Method For Demonstrations 1 and 3 of the Spindle-Assembly Task

$\mathbf{u}(t)$	State Number	c_i	Noise Removed?	Obstacle Free?
Demonstration 1				
$\mathbf{u}_{1,1}$	2	C_{free}	No noise	Yes
$\mathbf{u}_{2,1}$	20	c_2	No noise	Yes
$\mathbf{u}_{3,1}$	22	c_3	No noise	Yes
$\mathbf{u}_{4,1}$	21	c_4	Yes	Yes
$\mathbf{u}_{5,1}$	24	c_5	Yes	Yes
$\mathbf{u}_{4,2}$	21	c_4	No noise	Yes
$\mathbf{u}_{6,1}$	27	c_6	No noise	Yes
$\mathbf{u}_{7,1}$	8	c_7	Yes	Yes
$\mathbf{u}_{8,1}$	38	c_8	No noise	Yes
$\mathbf{u}_{7,2}$	8	c_7	No noise	Yes
$\mathbf{u}_{9,1}$	54	c_9	Yes	Yes
$\mathbf{u}_{10,1}$	47	c_{10}	No noise	Yes
Demonstration 2				
$\mathbf{u}_{1,2}$	2	C_{free}	Yes	Yes
$\mathbf{u}_{11,1}$	5	c_{11}	Yes	Yes
$\mathbf{u}_{12,1}$	75	c_{12}	No noise	Yes
$\mathbf{u}_{13,1}$	6	c_{13}	No noise	Yes
$\mathbf{u}_{14,1}$	7	c_{14}	No	Yes
$\mathbf{u}_{7,3}$	8	c_7	Yes	Yes
$\mathbf{u}_{15,1}$	38	c_{15}	No noise	Yes
$\mathbf{u}_{7,4}$	8	c_7	Yes	Yes
$\mathbf{u}_{16,1}$	9	c_{16}	No	Yes
...	...	...	...	...
Demonstration 5				
$\mathbf{u}_{28,1}$	30	c_{28}	No	Yes
$\mathbf{u}_{32,1}$	56	c_{32}	No	Yes

Other demonstrated $\mathbf{u}_{ij}$ more or less followed exactly the demonstrated path. In some of these cases this was sensible, since the demonstrated path was noise-free. We have labeled these cases in the table with "No noise", e.g., $\mathbf{u}_{1,1}$, $\mathbf{u}_{2,1}$, $\mathbf{u}_{3,1}$, $\mathbf{u}_{4,2}$, etc. In the remainder of cases, the demonstrated path did contain noise, however the derived $\mathbf{u}_{ij}$ still followed the demonstrated path. We have labeled these cases in column 5 of the table with a "No", e.g., $\mathbf{u}_{14,1}$, $\mathbf{u}_{16,1}$, $\mathbf{u}_{28,1}$ and $\mathbf{u}_{32,1}$. Our main interest here is with those $\mathbf{u}_{ij}$ labeled in column 5 with a Yes or a No; i.e., did our method remove noise if it existed? Notice how noise has been removed for $\mathbf{u}_{ij}$ derived in states that (a) were visited often in the demonstration, and (b) correspond to C-surfaces that have many boundary C-surfaces.[5] States 2, 21, 8 and 54 each have properties (a) and (b). Notice then how the $\mathbf{u}_{ij}$ derived for noisy demonstrated paths in these states, i.e., $\mathbf{u}_{4,1}$, $\mathbf{u}_{7,1}$, $\mathbf{u}_{9,1}$, $\mathbf{u}_{1,2}$, etc., have each had noise successfully removed. In contrast, Table 2 shows that noise was not generally removed for $\mathbf{u}_{ij}$ derived in states visited only once in the demonstration set, e.g., states 30 and 56. Neither was it removed for states that were visited more than once, but that had C-surfaces with only a small number of boundary C-surfaces, e.g., states 9 and 7 (c_{16} has only two boundary C-surfaces, the C-surfaces of states 28 and 65, while c_{16} had no boundary states). The dependence of our method on properties (a) and (b) is a logical outcome, since the C-surfaces with these properties will contain a larger number of both interior segments and likely-free regions. In these cases, the likelihood of finding a $\mathbf{u}_{ij}$ that encodes a more direct route on the C-surface than the demonstrated path is greater. That is, where our method has sufficient "information" on a C-surface, noise will be removed when deriving $\mathbf{u}_{ij}$. Otherwise, the method will revert to using the original demonstrated path, even if it contained noise.

The second important question regarding each $\mathbf{u}_{ij}$ shown in Table 2 was whether it defined an obstacle-free path.

5. Recall our definition of a *boundary C-surface* in Section 4.2.

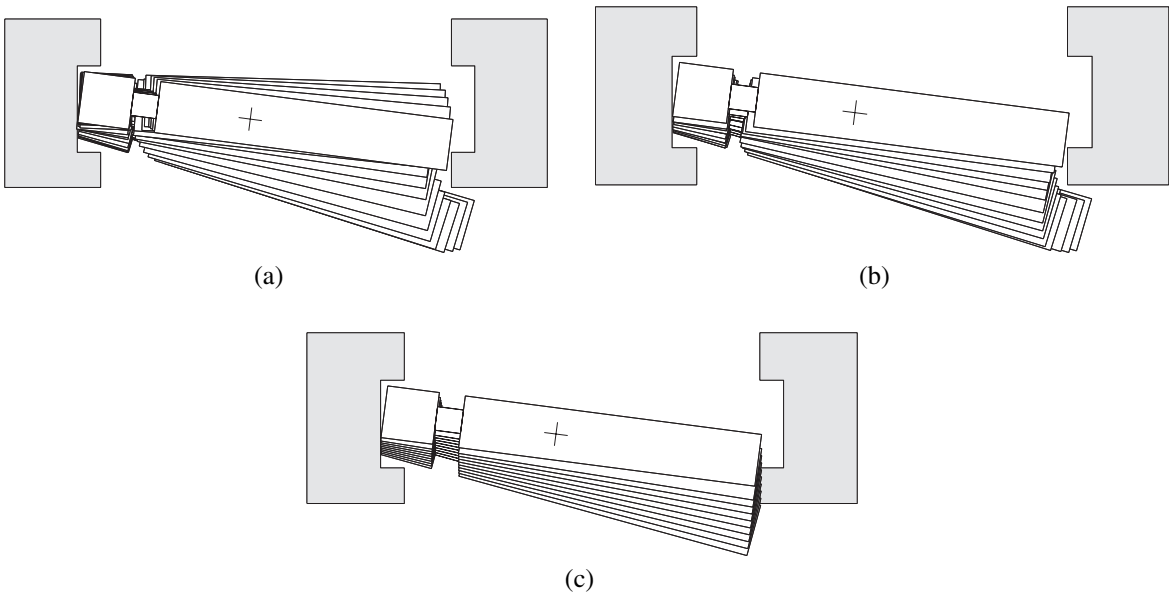


Fig. 11. An example of the process showing (a) an original demonstrated path containing noise, (b) the noise-free path derived by our method, and (c) the shortest length path.

Remember that our method derives $\mathbf{u}_{ij}$ that are likely obstacle-free. We show in column 6 of Table 2 the result for each $\mathbf{u}_{ij}$ in this regard. A key finding in these experiments was that each $\mathbf{u}_{ij}$ did in fact define an obstacle-free path. We have seen that three distinct types of $\mathbf{u}_{ij}$ exist in Table 2. First, $\mathbf{u}_{ij}$ derived where no noise existed. This type of $\mathbf{u}_{ij}$ follows a demonstrated path, and we know that demonstrated paths are obstacle-free. Secondly, $\mathbf{u}_{ij}$ derived in states that were visited often in the demonstration, and had many boundary states. It would be expected that $\mathbf{u}_{ij}$ for this type of state are obstacle-free. The presence of many boundary states means that likely-free regions will generally cover the C-surface well in regions where we wish to generate $\mathbf{u}_{ij}$. For example, for state 8 we know that boundary segments exist in states 7, 27, 33, 9, 38 and 54, and we saw in Figure 8 how the likely-free regions generated from these segments "cover" the C-surface of state 8 very well. Then for this type of $\mathbf{u}_{ij}$, if the demonstrated path on which a $\mathbf{u}_{ij}$ is based does contain noise, the good coverage of points of the C-surface means that an alternate, "noise-free" path for $\mathbf{u}_{ij}$ will most likely be found. The third and final type of $\mathbf{u}_{ij}$ in Table 2 is those where noise existed but was not removed. We noted how our method reverted to using the underlying demonstrated path for this type of $\mathbf{u}_{ij}$. That is, although these $\mathbf{u}_{ij}$ contain noise, they follow a demonstrated path, and will therefore be obstacle-free.

We have talked in general about the results of our method for deriving $\mathbf{u}_{ij}$. To illustrate more concretely how the method worked in practice, we now provide the derivation of $\mathbf{u}_{1,7}$ as an example. $\mathbf{u}_{1,7}$ encodes a path in state 8 between states 27 and 38. We show in Figure 11(a) how $\hat{\mathbf{u}}_{1,7}$ contains significant noise in that it is overly long. The demonstrator has moved the base of the spindle body toward the back of the rebate in the bottom support, only to then reverse this motion to enter state 38. This is typical of the type of noise introduced into a demonstration by a human. The $\mathbf{u}_{ij}$ derived for $\hat{\mathbf{u}}_{1,7}$ by our method is shown in Figure 11(b). Note how this path follows a much more efficient route than the original demonstrated path. In fact, the derived $\mathbf{u}_{ij}$ encodes a path that is just over half the length of the original demonstrated path. As an interesting comparison, we show in Figure 11(c) the path resulting by simply connecting the start and end points of $\hat{\mathbf{u}}_{1,7}$ by a straight line (in C-space) path. Note how this path is not valid since it results in the spindle body passing-into the right support.

We have seen how for $\mathbf{u}_{ij}$ our method will remove noise if sufficient information on a C-surface exists. If sufficient information does not exist, then it reverts to using the underlying demonstrated path. This type of outcome is attractive since, as we have seen, it tends to produce $\mathbf{u}_{ij}$ that are obstacle-free. An important influence on achieving this outcome was the conservative value we set for parameter md. Recall how we chose in Section 4.6 a metric that resulted in conservative values for md. Conservative values of md mean our method will only remove noise if sufficient information exists. The alternative approach of setting md aggressively may result in an increased number of demonstrated $\mathbf{u}_{ij}$ where noise is successfully removed, e.g., noise-free paths may be derived for $\mathbf{u}_{14,1}$, $\mathbf{u}_{16,1}$, $\mathbf{u}_{28,1}$, etc. However, the risk with such an approach is that $\mathbf{u}_{ij}$ can then become non-obstacle-free, since points in

likely-free regions would be generated far from their boundary segments. We prefer, and recommend, the approach of setting md conservatively. In this way, the method has the tendency to produce $\mathbf{u}_{ij}$ that are obstacle-free, and will only remove noise in cases where it has sufficient information to do so.

Our approach has been presented for PbD of a planar 2D task. However, we would like to reinforce that the approach is applicable to higher dimensions, including full 3D 6-DOF tasks. This result stems from the likely-free region generation process. Demonstrated points on one C-surface are used to generate points on another of dimension one higher. No built in limitation exists in the method on the dimension of these C-surfaces.

5. Conclusion

We have introduced a new method for coping with demonstration suboptimality in PbD. The first part of the approach was to derive an explicit and quantitative representation of a task geometry from demonstration. The step is a logical one because it provides a basis for making sensible decisions about which human actions represent noise, and which represent skillful execution of the task. Results have shown that a valid representation of task geometry could be obtained. The essential result was that sufficient information in the demonstration was required for the method to determine an accurate representation. The second part of this paper was concerned with deriving noise-free paths from noisy demonstrated paths. To achieve this end, we presented a path planning technique that does not require a full knowledge of C-space, but rather uses a partial knowledge of C-space and a set of known-obstacle-free, but noisy, paths to determine a more optimal set of paths. An important part of the work was that paths derived are not guaranteed to be obstacle-free. However, we have shown that with conservative tuning of parameter values, the method would tend to revert back to using the original suboptimal paths in regions where sufficient information was not available, while still providing a useful set of more-optimized control commands for the robot than those provided directly by the demonstration.

References

Asada, H. May 1990. Teaching and learning of compliance using neural nets: Representation and generalization to nonlinear compliance. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1237–1244.

Asada, H., and Izumi, H. 1989. Automatic program generation from teaching data for the hybrid control of robots. In *IEEE Transactions on Robotics and Automation* 5:163–173.

Atkeson, C. G., and Schaal, S. May 1997. Learning tasks from a single demonstration. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pp. 1706–1712.

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

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

Chen, J. R., and McCarragher, B. J. April 2000. Programming by demonstration: constructing task level plans in a hybrid dynamic framework. In *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 1402–1407.

Chen, J. R., and Zelinsky, A. May 2001. Generating a configuration space representation for assembly tasks from demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea.

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

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

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

Hovland, G. E., and McCarragher, B. J. September 1997. Combining force and position measurements for the modeling of robotic assembly. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'97)*, Grenoble, France, pp. 655–660.

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

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

Koeppe, R., and Hirzinger, G. 2000. A signal-based approach to localization and navigation of autonomous compliant motion. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Latombe, J.-C. 1991. *Robot Motion Planning*, Kluwer Academic, Dordrecht.

Latombe, J., Kavraki, L. E., Svestka, P., and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in

high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4):566–580.

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

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

McCarragher, B. J. 1996. Task primitives for the discrete event modeling and control of 6-DOF assembly tasks. In *IEEE Transactions on Robotics and Automation* 12(2):280–289.

McCarragher, B. J., and Asada, H. 1995. The discrete event modeling and trajectory planning of robotic assembly tasks. *Journal of Dynamic Systems, Measurements and Control* 117(3):394–400.

Nechyba, M. C., and Xu, Y. April 1996. On the fidelity of skill models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, MN.

Pomerleau, D. A. 1991. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation* 3(1):88–97.

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

Skubic, M., and Volz, R. A. 1996. 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*, pp. 458–464.

Tso, S. K., and Liu, K. P. April 1997. Demonstrated trajectory selection by hidden Markov model. In *Proceedings of the International Conference on Robotics and Automation*, pp. 2713–2718, Albuquerque, NM.

Witvrouw, W., Wang, Q., De Schutter, J., and Graves, S. April 1996. Derivation of compliant motion programs based on human demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2616–2621.