2003

Virtual assembly with biologically inspired intelligence

Xiaobu Yuan
University of Windsor

Simon X. Yang
University of Guelph

Virtual Assembly with Biologically Inspired Intelligence

Xiaobu Yuan[1] and Simon X. Yang[2]

[1]School of Computer Science, University of Windsor, Windsor, Ontario, N9B 3P4, Canada
[2]School of Engineering, University of Guelph, Guelph, Ontario, N1G 2W1, Canada

Abstract

This paper investigates the introduction of biologically inspired intelligence into virtual assembly. It develops an approach to assist product engineers making assembly-related manufacturing decisions without actually realizing the physical products. This approach extracts the knowledge of mechanical assembly by allowing human operators to perform assembly operations directly in the virtual environment. The incorporation of a biologically inspired neural network into an interactive assembly planner further leads to the improvement of flexible product manufacturing, i.e., automatically producing alternative assembly sequences with robot-level instructions for evaluation and optimization. Complexity analysis and simulation study demonstrate the effectiveness and efficiency of this approach.

1 Introduction

Flexibility has been recognized as a desirable feature in manufacturing systems to ensure higher quality while maintaining an increasing diversity of products. Driven by the highly competitive market, researchers in a wide range of application areas, including engineering, economics, and management, have worked on the development of a variety of methods to achieve their specific goals and/or subgoals through this feature. In particular, flexibility in product design anticipates significant improvements in terms of responsiveness to the market, speed of product development, and saving in prototypes and verification testing [25].

Accordingly, virtual assembly fulfills design flexibility by working in the computer. It replaces physical objects with the virtual representation of machinery parts, and provides advanced user interfaces for human operators to design and generate product prototypes, to analyze and optimize manufacturing processes, and to verify and control work-floor implements directly in the computer-synthesized working environment. The elimination of physical prototyping and on-site verification makes virtual assembly a powerful tool to reduce the life-cycle of manufacturing and to adapt changes or introduce new products.

Presented in this paper is an approach that introduces biologically inspired intelligence into virtual assembly. The discussions start with a brief survey of the research topics directly related to this work. The focus then turns to the introduction of human expertise of mechanical assembly and the incorporation of a biologically inspired neural network into the development of a virtual assembly system. The benefits of flexibility become evident with the capability of an interactive assembly planner to produce alternative assembly sequences from a single user-defined sequence. Finally, this paper concludes with simulation study and discussions on experiment results.

2 Related Work

Robot-based mechanical assembly uses robotic manipulators to put together products from its component parts. Correspondingly, virtual assembly needs to deal with the issues of robot programming and assembly planning. In addition, it has to handle the challenges created by applying technologies such as virtual reality and neural networks.

2.1 Robot programming

Robotic manipulators fit machinery parts together by performing assembly tasks in specified sequences. Each task instructs a robotic manipulator to perform an assembly operation and to establish a mating relationship between objects. The old-fashion online robot programming requires human operators to physically move a teaching pendant. It records the actual movements of joints and uses them to control robots for high volume productions. Online programming suffers from the down-time of robot operations, the danger imposed upon human operators, and the difficulty of making adjustments for new products.

In comparison, offline programming promotes the development of automated manufacturing tools and allows for the integration of different technologies from a wide range of originally separated areas. There has been a number of methods developed for different applications [20]. They range from simple text-based programming interfaces to computer-aided production systems. The former requires long development time and expert programmers to visualize joint motions without a physical robot, and the latter provides a full set of tools to design and program the entire manufacturing process.

Graphical user interfaces present machinery parts onto the screen and provide electronic mice as an interactive device for object manipulation. They are more convenient to use than text-based interfaces, and are common in computer-aided design/manufacturing [2], including most of the commercial systems in the market such as `ROBOCAD` and `IGRIP`. The drawback, however, is that this type of human/computer interaction is two-dimensional in nature. Decomposition of three-dimensional assembly operations into two-dimensional sub-operations is a must.

2.2 Assembly planning

Given a robotic manipulator and the design of a product, the objective of assembly planning is to determine feasible and optimal assembly sequences for the robotic manipulator to assemble the product from its component parts [12]. The assembly of m machinery parts by a robotic manipulator of k degrees of freedom creates a configuration space of $m + k$ degrees of freedom. A point in the configuration space defines a particular state of a workcell. Considering the original layout of the workcell and the final scene of an assembled product as the initial and destination points in the configuration space, a feasible assembly sequence then corresponds to a continuous, safe, and realizable path that connects the two points [21].

Consequently, assembly planning faces the problem of path searching in an $(m + k)$-dimensional configuration space. Assembly planning in theory is impossible as the required computation time grows exponentially with the dimension of the configuration space. In practice, it is common to reduce the complexity by making assumptions. Free-flying objects, for instance, simplify the problem into a logical planning problem [26]. However, it is only a subproblem of assembly planning as it leaves robotic execution out of assembly plans. Among the approaches that work on the complexity problem, the most influential are the numerical potential field, connectivity characterizing, and sequential frameworks [7].

Assembly planning at its current stage has to leave the actual motion of objects out of the planning process. The majority methods classified under interactive planning, such as the one in [5], are in fact dealing with mechanical disas-

sembly. Disassembly planning is fundamentally different from assembly planning as it provides only a subset of assembly sequences and considers a restricted set of allowable moving directions [19]. Other methods simplify mechanical assembly to pick-and-place operations as a way of avoiding motion planning, yet assembly planning without the actual movement of machinery parts is hardly useful in practice.

2.3 Virtual reality

The recent development of virtual reality has changed the way that human operators interact with computers. New devices, such as head-mounted display and data glove devices, become affordable and more reliable. They extend user interface from the classical two-dimensional to a three-dimensional space. Virtual reality provides a new means of immersing production engineers in a computer-synthesized working environment for them to interact with the virtual machinery parts, specifying and visualizing assembly activities. Though virtual reality technology is still under development, researchers have already begun investigating its application in industry manufacturing.

The most straight forward application includes those that explore the sensing capability of data glove devices. By requiring human operators to put on data gloves while manipulating physical objects, a paper in [23] presented a method to collect human movement of objects for the integration of task planning and execution. A couple of others used the sensory reading of finger joint bendings to map grasping operations from a user to the manipulator [11, 29]. Another practice in this category deals with "teaching by showing", and developed methods of transforming human operations to symbolic assembly commands with data glove devices [16].

Other applications emphasize more on the advantages that virtual reality offers to human/computer interaction. Data glove devices in this case become an instrument to construct human hands into the virtual environment. It makes possible for direct manipulation of virtual objects [18, 24]. In addition, virtual reality may go beyond reality by allowing object manipulation at the conceptual level, i.e., to act at a distance [9]. An increasing number of projects has started paying attention to the overall environment that could eventually allow production engineers to plan, evaluate, and verify the assembly of mechanical products in the computer [10].

2.4 Neural Networks

Path planning is a typical application area of neural networks in robotics. Most models use global methods to search the possible paths in the entire workspace [1]. As a result, they suffer from the same computational complexity as assembly planning does. Other models also have the problem of undesired local minima, which may create traps in some cases such as concave U-shaped obstacles [28]. Path planning with penetration growth distance shows the advantage of searching over collision paths [17], and has the capability of generating optimal, continuous robot paths. Unfortunately, the neural-network approaches that work with static environments only are not suitable for interactive applications such as virtual assembly.

There is a number of models developed for real-time motion planning through learning. For instance, combining an adaptive sensory-motor mapping model and an online visual error correction model may produce the trajectory of robot manipulators at run time [14]; and dynamic navigation of a mobile robot without any collisions is possible through unsupervised learning [15]. Since learning-based path planning cannot perform properly in fast changing environments, virtual assembly cannot use the neural networks whose operation relies on explicit learning or cost optimization.

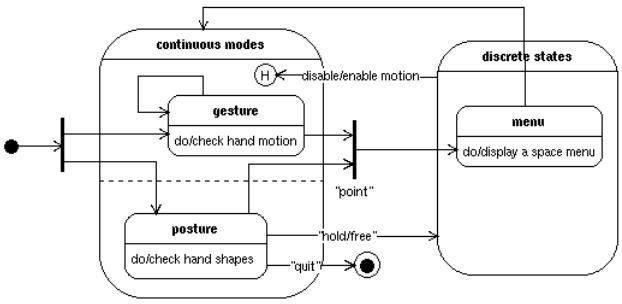


Figure 1: State Transition Diagram of "*Hand*"

3 The Incorporation of Intelligence

The advantage of virtual assembly does not come from a simple imitation of the activities that take place in the physical world during mechanical assembly. Instead, its approach of working in the computer creates the opportunity of integrating a wide range of features into a seamless environment. This section discusses the design of a hand-based user interface and the construction of a biologically inspired neural network. The next section then discusses their application in knowledge extraction and real-time path planning in a virtual assembly system.

3.1 Hand-based human/computer interaction

The main difficulty of introducing human expertise into assembly planning roots in the inconsistent ways of object manipulation in the physical and virtual worlds. Window-based human/computer interaction is not sufficient for the task of specifying three-dimensional assembly operations, and the use of virtual reality devices does not need to limit to data collection only. By properly exploiting the object manipulation capability of data glove devices, an operator may handle the virtual representation of machinery parts in a similar way as in the physical world, and eventually become capable of engaging in the planning activities.

In a virtual environment, all objects are graphics models that follow no physical restrictions. The hand cannot pick up any objects in the virtual environment as they tender to run through each other like ghost objects. A hand-based user interface uses the continuous measurement from a space tracker to determine the position and orientation of the hand. It also uses the sensory inputs from a data glove device to signal the bending of figure segments in terms of joint angles. A combination of different hand postures may activate different control commands or initiate the gesture sampling for motion specification [22].

In particular, a closing hand triggers the `hold` command to make a selection or uphold an action; an open hand implies `free`, which releases an item in possession or frees the hand from constraints; a `point` sign invokes items for selection; and an "OK" sign means time to `quit`, i.e., to terminate a running process. Shown in Fig. 1 is a state transition diagram [4] of the hand. The posture and gesture start as two concurrent states. They function in the super state of 'continuous modes', generating control commands and motions.

Before the hand picks up an object, the `hold` command works primarily with `free` for hand adjustment. It releases the hand from the continuous control mode. The motion of the hand in the physical world no longer contributes to the motion in the virtual environment. In such a way, the hand is able to prepare itself to a desired position or orientation. Object manipulation begins with a `point` command, which prompts up a space menu in the virtual environment and activates menu selection. A casting ray from the hand then highlights the menu item it hits, which becomes selected when followed immediately by `hold`.

If the selected item is labeled as "reference", the control goes into a remote selection mode, in which a user selects graphics features to define reference coordinate systems. The first feature selected by a combination of `point` and `hold` commands is the start point of an axis and the

second one is the end point. The first pair of features defines the major axis **z**; and the second pair defines the minor axis **x**. Reference definition operates in an "action-at-a-distance" fashion, which is available only in a virtual environment as physical limits does not permit conceptual-level object manipulation.

Alternatively, the selection of another menu item "motion" starts physical-level object manipulation. It checks for the intersection between the hand and the machinery parts to see if the hand touches any object. A `hold` posture grabs the object in touch, and makes it to move together with the hand until a `free` posture releases it. In either the "reference" or "motion" mode, a `quit` command returns the control to the menu state. One of them is in active until the selection of menu item "done". During the interaction, remote selection reduces required motion, and direct manipulation maps grab-move-release actions to the actual operations.

In addition to the listed items, two assembly operations are implied in gestures at the release of an object. They are `slide` and `screw`. The `slide` command slides an aligned part into its position. It triggers the sliding motion of an object along the major alignment axis. This command is activated with a pushing gesture when the object reaches to a proper alignment. The `screw` command differs from `slide` in the way it becomes active. It triggers the rotation of an object down to its alignment, which happens with a proper alignment and a twisting gesture. Both sliding and screwing operations result in a full alignment of objects by the computer.

3.2 A biologically inspired neural network

The original model of the biologically inspired neural network used electrical circuit elements to describe a patch of membrane in biological neural systems [8]. Let V_m be the voltage across the membrane, and C_m be the constant membrane capacitance. The following state equation then describes the dynamics of V_m.

$$C_m \frac{dV_m}{dt} = -(E_p + V_m)g_p + (E_{N_a} - V_m)g_{N_a} - (E_K + V_m)g_K \quad (1)$$

In the equation, E_K, E_{N_a}, and E_p are parameters that represent respectively the saturation potentials for the potassium ions, sodium ions, and passive leak current in the membrane. Correspondingly, the conductance in each of the three channels is g_K, g_{N_a}, and g_p.

After setting C_m to *1*, a shunting equation takes its form by substituting $E_p + V_m$, g_p, $E_{N_a} + E_p$, $E_K - E_p$, g_{N_a}, and g_K with notions x_i, A, B, D, S_i^+, and S_i^- respectively.

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)S_i^e(t) - (D + x_i)S_i^i(t) \quad (2)$$

In particular, x_i is the neural activity of the i-th neuron in the two-dimensional membrane. Parameters A, B, and D are three nonnegative constants describing the passive decay rate, the upper and lower bounds of x_i respectively. S_i^e is the excitatory input and S_i^i is the inhibitory input to the neuron.

The equation in (2) is useful for understanding the real-time adaptive behavior of individuals to complex and dynamic environmental contingencies [6]. In fact, this general shunting model works with any discrete neural network in a high-dimensional space provided the topological organization of the network characterizes the problem domain. In mechanical assembly, the task space is M-dimensional if the robotic manipulator it uses has M degrees of freedom. This application then needs an M-dimensional discrete neural network to model the robot operations.

Suppose a neuron N_q locates at a point q in the M-dimensional network, $q=<q_1, \cdots, q_M>$. It connects to all its n direct neighboring neurons N_{p_j}, $p_j=<p_{j1}, \cdots, p_{jM}>$ and $1 \le j \le n$. Following the notation in (2), x_q and x_p denote the neural activities of N_q and N_p respectively. A modification to (2) produces the following shunt-

ing equation that defines the dynamics of N_q.

$$\frac{dx_q}{dt} = -Ax_q + (B - x_q)([I_q]^+ + \sum_{j=1}^{n} \omega_{qp_j}[x_{p_j}]^+) - (D + x_q)([I_q]^- + \sum_{j=1}^{n} \omega_{qp_j} c[x_{p_j} - s]^-) \quad (3)$$

In the equation, I_q is the external inputs to N_q. The two functions $[x]^+$ and $[x]^-$ result in $max\{x, 0\}$ and $max\{-x, 0\}$, respectively. Parameters A, B, and D represent the passive decay rate, the upper and lower bounds of N_q, respectively. Parameter c is a constant in the range *[0,1]*, and s is an adjustable safety factor. Especially, the symmetric weights ω_{qp} are determined by a monotonically decreasing function $f(|q-p|)$ of the Euclidean distance between p and q. For instance, $f(a) = \mu/a$, if $0 < a < r_0$ for two positive constants μ and r_0. Otherwise, $f(a) = 0$.

Every dimension of the M-dimensional neural network maps to one particular joint of the M-link robotic manipulator. In the k-th dimension, the density of neurons depends on the increment of the k-th joint, and the number of neurons covers the entire range of the k-th joint. The dynamics of neuron activity x_q in (3) portrays the operation of assembly tasks in such a way that the location of neuron N_q in the network stands for a particular joint configuration of the robotic manipulator. Neuron N_q connects only to its neighboring neurons in the network. A change from a neuron to any of its neighboring neurons triggers a change in the set of joint incremental values.

For any task in an assembly sequence, the neuron in the neural network that maps to the joint configuration of the robotic manipulator picking up a machinery part is a starting neuron N_s, and the one that maps to the completion of the assembly task by the robot manipulator is the target neuron N_t. All the other neurons classify into two types. One type, denoted by set $\{N_c\}$, includes all the neurons whose location maps to a robot configuration that causes a collision between objects or objects and the robotic manipulator; the other, $\{N_f\}$, counters in the rest of neurons that lead to the collision-free movements of the robotic manipulator.

Different external inputs to the neurons then distinguish one type from another. The input I_q in (3) to neuron N_q is a large positive constant V, $V >> B$, if N_q happens to be the target N_t. I_q changes to -V if N_q is an element in $\{N_c\}$. Otherwise, I_q is *0* for all the neurons in the set of $\{N_f\}$ (Fig. 2). As the starting neuron must be an element of $\{N_f\}$, its external input is always *0*. In addition, the stimuli within the receptive field of neuron N_q also include a sum of the weighted neural activities from its direct neighbors. In such a way, it allows the network to propagate positive neural activity through excitatory connections, and to restrain the negative activities through inhibitory connections.

This neural network overcomes the drawbacks of other neural networks. It uses only local connections among neurons, and the computational complexity linearly depends on the neural networks size. Moreover, the underlying dynamic neural activity operates without explicitly searching over the free workspace or the collision paths, without explicitly optimizing any cost functions, without any prior knowledge of the dynamic environment, and without any learning procedures. It is therefore useful for planning real-time optimal robot motion in dynamics situations without the need of any learning procedures.

4 A Virtual Assembly System

The front-end of a virtual assembly system should be able to allow operators to specify and evaluate assembly operations directly in a the virtual environment. Its final output is a set of optimized instructions that contains all the details to control robotic manipulators implementing optimized assembly tasks.

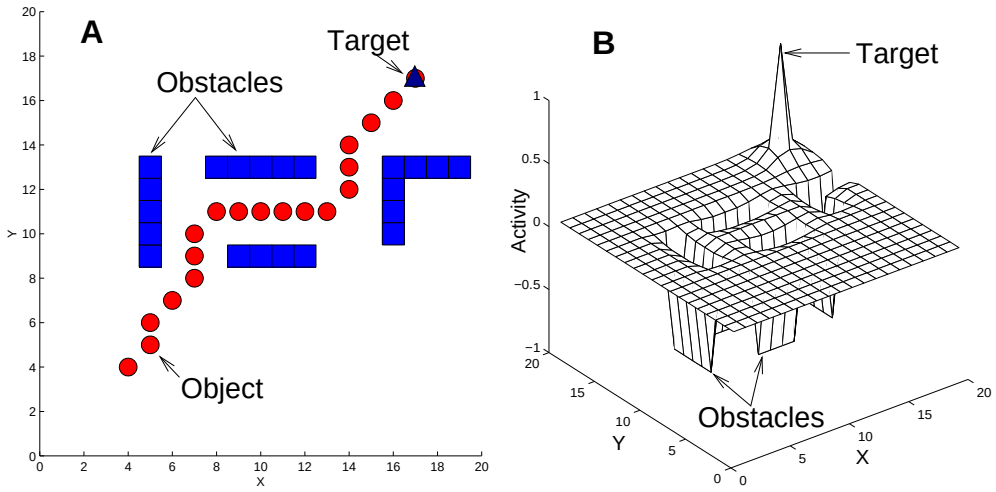


Figure 2: Collision-Free Path at Real Time

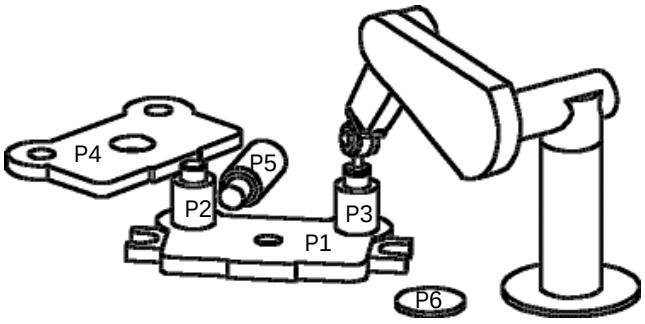


Figure 3: The Assembly of a Die-Set

4.1 Virtual programming

As shown in Fig. 3, a robotic manipulator is simply a set of graphics models in the virtual assembly system, each of which represents a rigid segment of the physical manipulator. The connection of segments forms a linkage of joints, and the numbering of joints goes from the fixed base to the end. For any two adjacent joints, a homogeneous transformation $m_{i-1}^{i} = R_z(\theta_i)T_z(d_i)T_x(a_i)R_x(\alpha_{i-1})$ determines their relationship, $0 < i \leq M$, where M is the linkage's degree of freedom [13].

Every assembly task involves three stages of positioning [27]. The first stage specifies the pre-assembly position of an object. It positions a virtual robotic manipulator and prepares its end effector for object handling. The second stage takes care of the in-assembly motion of an object. The practical nature of mechanical assembly requires a robotic manipulator to move the object along a safe and realizable path through the environment. The last stage of object positioning places the object under manipulation at its destination.

In the virtual assembly system, a human operator signals the pre-assembly position by grasping an object with his hand after touching the object. This operation ensures the human hand actually reaches to the object, which not only helps to physically bring the end effector of a robotic manipulator to the object but also makes possible for the mapping of human grasping to the manipulator. Due to the lack of physical restrictions in virtual environments, the post-assembly position only requires the operator to bring the moving object close to its destination. A `free` command releases the object, and the system completes the desired alignment with pre-defined coordinate systems.

The configuration of the robotic manipulator for the pre- and post-assembly positions corresponds to the starting neuron N_s and the target neuron N_t in the neural network respectively. Suppose N_p is the neuron whose neural activity

x_p yields the biggest value among all the v neighboring neurons of N_s, i.e., $x_p = max\{x_{sj}, j = 1, 2, \cdots, v\}$. Given the range of $[J_{ks}, J_{ke}]$ for the k-th joint of the robotic manipulator and a total number of t_k neurons in the k-th dimension of the network, an update from neuron N_s to neuron N_p activates the robot joints for an increment of $< r_1(p_1 - q_1), \cdots, r_M(p_M - q_M) >$, $r_k = (J_{ke} - J_{ks})/t_k$.

By following the gradient ascent rule and adaptively changing the current configuration, the neural network globally guides the robotic manipulator move around in the working environment while avoiding possible collisions. The movement of the robotic manipulator at the same time brings the object under manipulation from its starting position to its resting position. The generated collision-free path then determines the in-assembly position of objects. In addition, it helps to decide the precedence relationship of objects for the construction of the deduction matrix, which is to be discussed in the next subsection.

When the human operator finishes with all the machinery parts, his operation of putting the objects together into a product actually defines a sequence of assembly tasks. Suppose a product P consists of n component parts $P_i, 1 \leq i \leq n$, and the user-defined sequence is the kth sequence in a total of m possible sequences to produce P. The user-defined assembly sequence then takes a form in the following format.

$$P_0 \overset{\tau_1(k)}{\rightarrow} P_1(k) \cdots \overset{\tau_i(k)}{\rightarrow} P_i(k) \cdots \overset{\tau_n(k)}{\rightarrow} P_n(k) \quad (4)$$

where P_0 is a stable workstation, and $\tau_i(k)$ defines the transformation of the ith object $P_i(k)$, $1 \leq i \leq n$. The index k implies that the order of objects and their associated transformation can be different in different sequences, i.e., $P_i(k) \neq P_i(k')$ and $\tau_i(k) \neq \tau_i(k')$ when $k \neq k'$.

4.2 Interactive Assembly Planning

An interactive assembly planner consists of four steps of *sequence programming*, *constraints deduction*, *sequence generation*, and *interactive evaluation* in the specified order (Fig. 4). In the first step, the operator exercises his expertise of mechanical assembly through virtual programming to define an assembly sequence as in (4). The second step then extracts human expertise from the sequence and uses this knowledge to construct a deduction matrix with precedence constraints.

An initialization first sets all the elements of an $n \times n$ deduction matrix to *-9*. The row of this matrix stands for the parts from $P_0(k)$ to $P_{n-1}(k)$, and its columns cover from $P_1(k)$ to $P_n(k)$, $1 < k < m$. Each element in a row of the deduction matrix implies if an object must be in place before the other objects. A test run of the neural network with the object in its resting position then tells if such a precedence relation exist when there are no collision-free paths to move the other objects. Otherwise, the successful generation of a collision-free path means that there is no precedence constraint between this object and another object.

Constraints deduction starts with the diagonal elements $m(i, i)$, $1 < i < n - 1$. Its value changes to *1* if $P_i(k)$ and $P_{i+1}(k)$ makes a face contact or if path generation fails to bring $P_i(k)$ to its destination after $P_{i+1}(k)$ is in place first. Otherwise, $m(i, i)$ is *0*. Afterwards, constraints deduction of the upper-right region of the matrix always tries to decide their precedence relationship first through symbolic inference. For an element $m(i, j)$, $0 \leq i < n - 1$ and $i < j < n$, its value changes to *-1* if both $m(i, i + 1)$ and $m(i + 1, j)$ are either *1* or *-1*. When symbolic reasoning cannot reach a conclusion, the neural network is once again used to decide for $m(i, j)$.

At the end of Step 2, all elements along the diagonal and in the upper-right region of the deduction matrix are reset from *-9* to *1*, *0*, or *-1*. Among them, an element $m(i, j)$ of a value *1*, $0 \leq i < n - 1$ and $i \leq j < n$, tells that the assembly of $P_i(k)$ must be done before the assembly of $P_j(k)$. Consequently, $P_j(k)$ becomes a child node of $P_i(k)$ in a tree $\mathbf{G}$ of assembly sequences. In Step 3 of the interactive planner, se-

quence generation employs a recursive procedure **Link_Node**(.) to construct the tree according to the elements of the deduction matrix.

An assembly tree represents assembly sequences with a simple directed graph [3]. In the tree, a path from the root node to any leaf node defines an assembly sequence. When Step 3 finishes the completion of **G**, the virtual assembly system is able to generate different sequences from the user-defined sequence, and evaluates them with predetermined measurements. In practice, different criteria are possible, such as the number of involved robotic manipulators, the degrees of required freedom, the number of primary operations, and the length, linearity, and realizability of assembly sequences [26]. When properly applied, they help to filter out the majority but the most promising designs for user verification.

5 Analysis and Simulation

Virtual assembly applies virtual reality techniques for the development of computer tools that help product engineers to make assembly-related decisions through abstract analysis, predictive models, robot visualization, and data presentation without physically realizing the product and its supporting processes. Experiments have been conducted to examine the operation of the virtual assembly system and the performance of biologically inspired intelligence. Tests covered a variety of machinery sets that involve different object shapes and different task difficulties.

Provided in this section are two groups of results for the assembly of a die-set. The first group goes through the process of virtual assembly to demonstrate the capability of producing alternative assembly sequences from a single assembly sequence. The role of the biologically inspired neural network in virtual assembly is then demonstrated in the second group with details of neural activities and joint displacement of two robotic manipulators during collision-free path generation. A subsequential discussion on the experimental results gives analysis of this work.

5.1 The assembly of a die-set

A typical die-set for coin-making consists of three principal components. They are basically a punch holder (P_4), a die holder (P_1), and two guideposts (P_2 and P_3), as shown in Fig. 3. It also has a stamping punch (P_5) and a metal plate (P_6). Equation in (5) gives a feasible sequence of assembling the die-set from its five components and placing the metal plate for stamping.

$$Tab \xrightarrow{\tau_1} P_1 \xrightarrow{\tau_2} P_2 \xrightarrow{\tau_3} P_3 \xrightarrow{\tau_4} P_4 \xrightarrow{\tau_5} P_5 \xrightarrow{\tau_6} P_6 \qquad (5)$$

In the sequence, the assembly tasks applied upon the parts are as the following, where each of the tasks τ_k, $0 < k \leq 6$, defines the the pre-assembly and post-assembly positions of object P_k.

- τ_1: place P_1 with its half-open hole facing up;
- τ_2: stand up P_2 by inserting it into a guidehole;
- τ_3: stand up P_3 at the other guidehole of P_1;
- τ_4: sit the two corner-holes of P_4 on the guideposts;
- τ_5: insert P_5 half-way through the bigger hole of P_4;
- τ_6: slide P_6 on top of the half-open hole of P_1.

After the human operator finishes defining the assembly sequence of (5) through virtual programming, interactive assembly planning takes place to determine the precedence relationship between objects. Following the order of object manipulation, it checks all parts one by one for precedence constraints. The first object P_1 is always on the table P_0, therefore $m(0,0)$ in Table 1 is *1*. As P_2 is in a hole of P_1 through alignment, the *1*-labeled $m(1,1)$ indicates that τ_2 can take place only after τ_1. For a similar reason, the values of both $m(3,3)$ and $m(4,4)$ are *1*.

However, there is no reference-alignment relationship between either the pair of P_2 and P_3 or the pair of P_5 and P_6. The values of $m(2,2)$ and $m(5,5)$ have to come from the test run of path generation between P_3 or P_6 and P_2 or P_5 with the neural network, and the result is *0* for both

M	P_1	P_2	P_3	P_4	P_5	P_6
P_0	1	-1	-1	-1	-1	-1
P_1	-9	1	1	-1	-1	1
P_2	-9	-9	0	1	-1	0
P_3	-9	-9	-9	1	-1	0
P_4	-9	-9	-9	-9	1	0
P_5	-9	-9	-9	-9	-9	0

Table 1: The Deduction Matrix of Die-Set

of them. For the elements in the upper-right region, symbolic reasoning takes place first. For example, the value of $m(0,1)$ becomes *-1* simply because $m(0,1)$ and $m(1,1)$ are already labeled with *1*. In the case of a *0*-labeled element in the inference, a check with path generation is unavoidable. This is how $m(1,2)$ obtains its value as $m(2,2)$ has a *0* value.

Sequence generation begins to construct an assembly tree right after constraints deduction completes the deduction matrix in Table 1. The top two levels of the tree are a graph description of the first row of the matrix that links P_0 with all the objects with a *1*-labeled element in this row. Therefore, element $m(0,0)$ results in a link from P_0 to P_1 in the tree. The tree then expands node P_1 by adding its child nodes P_2, P_3, and P_6. The next level then expands the three nodes with their brothers as the algorithm forbids P_4 becoming a child because it has two parents P_2 and P_3 who happen to be brothers. The same rules apply on the rest of the tree, and the final product is a tree in Fig. 5.

Fig. 5 shows that there are ten possible sequences to assemble the die-set, including the one defined by the operator. Although all sequences in the tree result in the same product, the moving paths of objects may differ due to the change of assembly order. The in-assembly positioning of objects, therefore, always rely on the neural network to generate collision-free paths for robotic manipulators to accomplish the assembly tasks. For the first sequence in the tree, as an example, the order and operation of assembly tasks are as below.

$$Tab \xrightarrow{\tau_1} P_1 \xrightarrow{\tau_2} P_2 \xrightarrow{\tau_3} P_3 \xrightarrow{\tau_6} P_6 \xrightarrow{\tau_4} P_4 \xrightarrow{\tau_5} P_5 \qquad (6)$$

5.2 Path generation with the neural network

The biologically inspired neural network works with multi-dimensional applications. For the purpose of illustration, this group of experiments uses a projected layout of the die-set assembly workcell onto a two-dimensional plane, as in Fig. 6. The assembly task is to move the guidepost at the top-left area to its resting place at the center-left region along an optimal path by a particular robotic manipulator. The goal, however, is to evaluate the necessary robotic operations when provided with different robotic manipulators.

For a robotic manipulator of two-degrees of freedom, the neural network has two dimensions. A neural network with 40×40 topographically ordered neurons is constructed to characterize the workspace at a size of 40×40. The initial values of all neural activities in the shunting equation of (3) are zero. The parameters are chosen as $A = 10$ and $B = D = 1$ for the passive decay rate and the upper and lower bounds; $\mu = 1$, $c = 0.9$, $s = -0.7$ and $r_0 = 2$ for the neighborhood connections; and $V = 100$ for the external inputs. The original configuration of the robotic manipulator locates its end-effort at *(5,20)* to pick up the guidepost, and the final configuration is at *(12,14)* to place the object.

Fig. 6.(a) depicts the collision-free path to complete this task. Due to the inability of this robotic manipulator to rotate objects, the guidepost under manipulation has to go around from the top and then to the bottom of the punch holder before it reaches to its destination. Fig. 7.(a) shows the traveling distance along the x and y axis, which took 61 steps to complete. In comparison, Fig. 7.(b) shows that only 17 steps is necessary for a robotic manipulator of three-degrees of freedom to complete this task. The task, however, requires an addi-

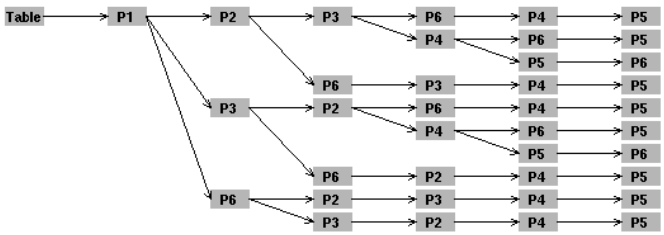
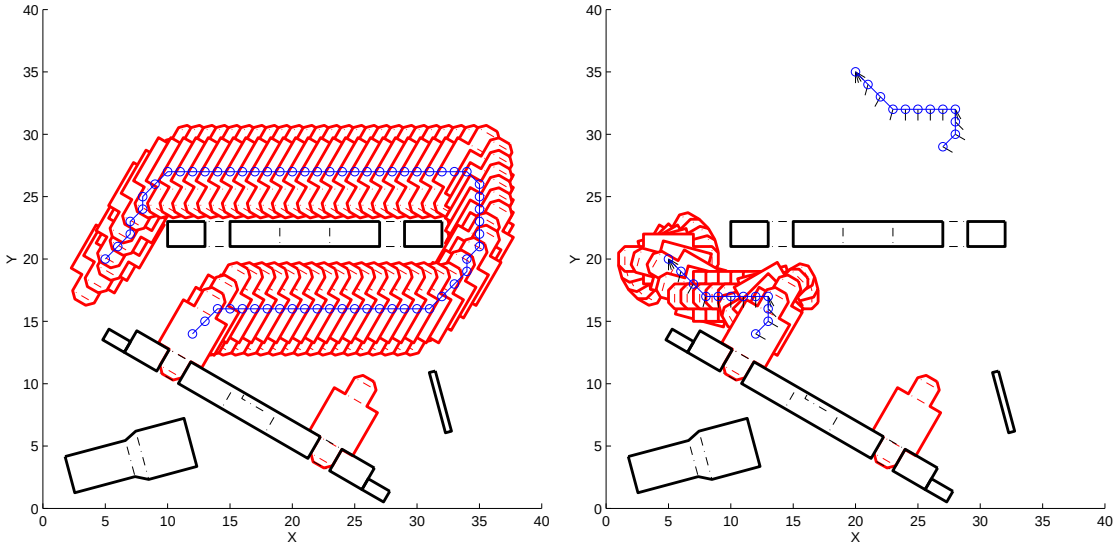


Figure 5: The Assembly Tree of the Die-Set



(a) without rotation (b) with rotation

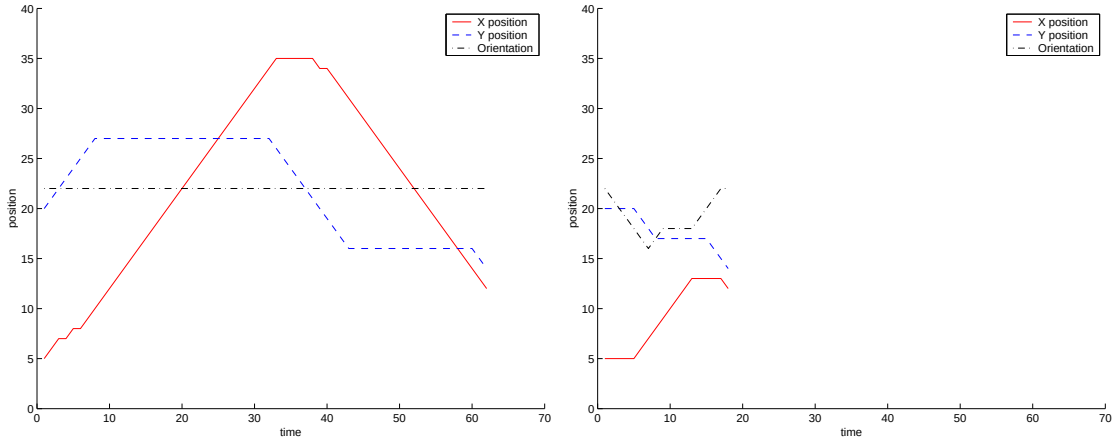
Figure 6: Collision-Free Moving Paths without/with Rotation

tional dimension of 24 neurons for path planning and another type of robotic operation for object rotation (Fig. 6.(b)).

5.3 Discussions

Experiments demonstrate the capability of virtual assembly to integrate different technologies into a seamless environment, and the advantages that it brings to product manufacturing. The process of virtual programming creates a scene of product with an assembly sequence, which in turn helps the interactive planner to decide the precedence constraints and generate alternative sequences. While the user-defined assembly sequence tells the pre-assembly and post-assembly positions, the biologically inspired neural network determines the in-assembly positioning with robot-level details.

The planning process of Fig. 4 includes constraints deduction and sequence generations. For the elements along the diagonal and in the upper-right region of a deduction matrix, there is a need of $n!$ checks to establish precedence relationships. Experiment results indicate that a large portion of the checks uses symbolic reasoning. The remaining checks use reference-alignment relationship first and path generation the last. As for tree construction, it is pure symbolic. Its complexity ranges from $O(n)$ to $O(n!n)$ depending on the product and its components. In comparison to completely autonomous planning with a high-dimensional configuration space, the presented approach of virtual assem-



(a) without rotation (b) with rotation

Figure 7: Translation Distances and Rotation Angles

bly is efficient and practical.

As for the neural network defined in (3), it is fundamentally a high-dimensional extension of the general shunting model of (2), whose neural activity is as stable as the original model. In the equation, x_i increases at a rate of $(B - x_i)S_i^+$, which is proportional to not only the excitatory input S_i^+ but also a gain control term $(B - x_i)$. When x_i is less than B, a positive excitatory contribution causes an increase in the neural activity. If x_i is equal to B, the excitatory term becomes zero and x_i no longer increases, no matter how strong the excitatory contribution is. When x_i exceeds B, $B - x_i$ becomes negative and the shunting term pulls x_i back to B, forcing x_i to stay below B.

A similar analysis applies to the last component of the general shunting model in which the inhibitory term forces the neural activity to stay above the lower bound $-D$. Once the neural activity goes into the range of $[-D, B]$, it stays in the range for any of the total excitatory and inhibitory inputs. A change to B and D will only change the range but not the relative value of neural activity. In comparison, parameters A and μ in (3) have a fundamental impact on the proposed model. While A influences the transient response to input signals, μ controls the propagation of neural activities among neurons.

The influence of parameters c and s on path generation is the clearance distance from obstacles. They determine the relative strength and the threshold of the negative neural connections respectively. When c is zero or s is higher than the neural activity bound, the generated path clips the corners of obstacles and runs down the edges of obstacles, which results in the so-called "too close" problems. On the other hand, if clearance from obstacles is too large, the generated path stays as far as possible from the obstacles when reaching the target, which results in the so-called "too far" problems.

Optimality refers to automatically generating smooth, continuous, and "comfortable" paths from the starting to the target configurations, without suffering from the two "too" problems. The term "real time" refers to the way that path generation reacts to changes in the environment. Moreover, this neural network does not suffer from local minimum, even in a complicated maze-type environment of many deadlock situations. Target configuration is the only neural activity source. The neural activity propagation from the target to the starting position always creates a feasible path with obstacle clearance.

6 Conclusion

This paper presents a novel approach of virtual assembly with biologically inspired intelligence. It develops a virtual assembly system to produce alternative assembly sequences from a single user-defined sequence, making it flexible for product engineers to choose the right design and a proper manufacturing process according to predetermined criteria without the need of physical realization. In addition, the biologically inspired neural network provides details of robot operations for quantified analysis.

As a pilot project of the vision on "manufacturing in the computer", virtual assembly helps to identify and resolve issues related to the construction of an integrated virtual manufacturing environment that could enhance all levels of manufacturing decision and control. Through the investigation of intelligent technology and its application in virtual assembly, this paper constitutes a preliminary work of this project. There are still plenty for improvements. Further research is under active investigation in the directions of new expansions and practical applications.

Acknowledgment

This work was funded by Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] K. Al-Sultan and D. Aliyu. A new potential field-based algorithm for path planning. *Journal of Intelligent and Robotic Systems*, 14(5):657–662, Oct 1996.

[2] T. Arai, T. Itoko, and H. Yago. A graphical robot language developed in Japan. *Robotica*, 15(1):99–103, Jan-Feb 1997.

[3] J. Bander. A heuristic-search algorithm for path determination with learning. *IEEE Transactions on Systems, Man, and Cybernetics, PART A: Systems and Humans*, 28(1):131–134, Jan 1998.

[4] M. Fowler and K. Scott. *UML Distilled: Applying the Standard Object Modeling Language*. Addison-Wesely, 1997.

[5] R. Gottipolu and K. Ghosh. An integrated approach to the generation of assembly sequences. *International Journal of Computer Applications in Technology*, 8(3-4):125–138, 1995.

[6] S. Grossberg. Nonlinear neural networks: Principles, mechanisms, and architecture. *Neural Networks*, 1:17–61, 1988.

[7] K. Gupta. The sequential framework for practical motion planning for manipulator arms: Algorithm and experiments. In K. Gupta and A. del Pobil, editors, *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, pages 9–31. John Wiley & Sons Ltd, 1998.

[8] A. Hodgkin and A. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology London*, 117:500–544, 1952.

[9] S. Jayaram, H. Connacher, and K. Lyons. Virtual assembly using virtual-reality techniques. *Computer-Aided Design*, 8(29):575–584, Aug 1997.

[10] S. Jayaram, U. Jayaram, Y. Wang, H. Tirumali, K. Lyons, and P. Hart. VADE: A virtual assembly design environment. *IEEE Computer Graphics and Applications*, 19(6):44–50, 1999.

[11] S. Kang and K. Ikeuchi. Toward automatic robot instruction from perception — mapping human grasps to manipulator grasps. *IEEE Transactions on Robotics and Automation*, 13(1):81–95, Feb 1997.

[12] J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[13] R. Levary. Fundamentals of industrial robots. *International Journal of Computer Applications in Technology*, 7(1-2):55–66, 1994.

[14] L. Li and H. Ogmen. Visually guided motor control: Adaptive sensorimotor mapping with online visual-error correction. In *Proceedings of the World Congress on Neural Networks*, pages 127–134, 1994.

[15] F. Muniz, E. Zalama, P. Gaudiano, and J. Lopez-Coronado. Neural controller for a mobile robot in a nonstationary environment. In *Proceedings of 2nd IFAC Conference on Intelligent Autonomous Vehicles*, pages 279–284, 1995.

[16] H. Ogata and T. Takahashi. Robotic assembly operation teaching in a virtual environment. *IEEE Transactions on Robotics and Automation*, 10(3):391–399, June 1994.

[17] C. Ong and E. Gibert. Robot path planning with penetration growth distance. *Journal of Robotic Systems*, 15(2):57–74, 1998.

[18] P. Palamidese. A virtual reality interface for space planning tasks. *Journal of Visual Languages and Computing*, 10(2):99–115, April 1999.

[19] V. Rajan and S. Nof. Minimal precedence constraints for integrated assembly and execution planning. *IEEE Transactions on Robotics and Automation*, 12(2):175–186, April 1996.

[20] Y. Regev. The evolution of offline programming. *Industrial Robot*, 22(3):3–3, 1995.

[21] S. Russell and P. Norvig. *Artificial Intelligence: a modern approach*. Prentice-Hall, 1995.

[22] H. Sun, X. Yuan, G. Baciu, and Y. Gu. Direct virtual-hand interface in robot assembly programming. *Journal of Visual Languages and Computing*, 10(1):55–68, 1999.

[23] C. Tung and A. Kak. Integrating sensing, task planning, and execution for robotic assembly. *IEEE Transactions on Robotics and Automation*, 12(2):187–201, April 1996.

[24] P. Werkhoven and J. Groen. Manipulation performance in interactive virtual environments. *Human Factors*, 40(3):432–442, Sept 1998.

[25] D. Whitney. Research issues in manufacturing flexibility — an invited review paper for ICRA2000 symposium on flexibility. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 383–388, 2000.

[26] R. Wilson and J. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71(2):371–396, Dec 1994.

[27] C. Wu and N. Kim. Modeling of part-mating strategies for automating assembly operations for robots. *IEEE Transactions on Systems Man and Cybernetics*, 24(7):1065–1074, July 1994.

[28] L. Wyard-Scott and Q. Meng. A potential maze solving algorithm for a micromouse robot. In *Proc. of IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*, pages 614–618, May 1995.

[29] M. Yun, D. Cannon, A. Freivalds, and G. Thomas. An instrumented glove for grasp specification in virtual-reality-based point-and-direct telerobotics. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 27(5):835–846, Oct 1997.

Step 1. *Sequence Programming*:

For a given set of objects $\{P_1, ...P_n\}$,
  define an assembly sequence as in (4).

Step 2. *Constraints Deduction*:

Initialize an $n \times n$ deduction matrix with *-9*.
For each diagonal element $m(i,i)$, $i = 0 \cdots n-1$,
  check P_i and P_{i+1} for precedence constraint,
    set $m(i,i)$ to *1* if they are related;
    otherwise, set $m(i,i)$ to *0*.
For $i = 0$ to $n-2$ with $i++$,
  for $j = i+1$ to $n-1$ with $j++$,
    if $m(i,i+1)$ and $m(i+1,j)$ are either *1* or *-1*,
      set $m(i,j)$ to *-1*;
    else, check P_i and P_j for precedence constraint,
      set $m(i,j)$ to *1* if they are related;
      otherwise, set $m(i,j)$ to *0*.

Step 3. *Sequence Generation*:

Create a graph **G** with a level_0 node P_0.
Create an empty set $N_0(0)$.
Set both node index i and level index k to *0*.
Link_Node(i,k) {
  For $j = i$ to $n-1$ with $j++$,
    if $m(i,j) = 1$,
      if $m(l,j)! = 1$ for every l in $N_i(k)$,
        create a level_{k+1} node P_j;
        make a link from P_i at level_k to P_j;
        add j to $N_i(k)$;
  For every l in $N_i(k)$,
    create an empty set $N_l(k+1)$;
    add all the rest of $N_i(k)$ to $N_l(k+1)$;
    Link_Node$(l, k+1)$ }.

Step 4. *Interactive Evaluation*:

For every path connecting P_0 to a leaf of **G**,
  evaluate this sequence with measurements.
For every optimized sequence,
  play back tasks involved in each steps;
  redefine the task when necessary.

Figure 4: The Algorithm of an Interactive Assembly Planner