Development of a Flexible Assembly System Using Industrial Robot with Machine Vision Guidance and Dexterous Multi-finger Gripper

Atul Mishra, I. A. Sainul, Sudipta Bhuyan, Sankha Deb, Debashis Sen and A. K. Deb

Abstract In today's era of mass customization, assembly automation systems should be designed with necessary production flexibility to cope with the growing product varieties to adapt to diverse customer requirements, yet the production costs should not be significantly different from those of comparable products made by mass production. In order to cope with this product variety-cost trade-off, robotics offers a flexible automation technology for turning assembly systems into efficient and flexible systems. Despite their great potential for high flexibility, there is a range of issues which must be addressed for its successful implementation. This chapter examines some of these key issues and challenges, reviews the results of previous research and describes our ongoing research on development of a flexible assembly system for mechanical products, using an industrial robot with machine vision guidance and dexterous multi-finger gripper. As part of the research work reported in this chapter, a Sexual Genetic Algorithm (SGA)-based approach for generation of optimal assembly sequence, a knowledge-based system for generating the robot task-level plan, a multi-finger robot gripper for flexible assembly based on a tendon-driven mechanism and an impedance control algorithm, and finally a strategy for implementation of robotic assembly under machine vision guidance have been presented.

A. Mishra · S. Deb (✉)
FMS and CIM Laboratory, Department of Mechanical Engineering, IIT Kharagpur, Kharagpur, India
e-mail: sankha.deb@mech.iitkgp.ernet.in

I. A. Sainul · S. Bhuyan
Advanced Technology Development Centre, IIT Kharagpur, Kharagpur, India

D. Sen
Department of Electronics & Electrical Communication Engineering, IIT Kharagpur, Kharagpur, India

A. K. Deb
Department of Electrical Engineering, IIT Kharagpur, Kharagpur, India

© Springer Nature Singapore Pte Ltd. 2018

S. S. Pande and U. S. Dixit (eds.), *Precision Product-Process Design and Optimization*, Lecture Notes on Multidisciplinary Industrial Engineering,
https://doi.org/10.1007/978-981-10-8767-7_2

Keywords Flexible assembly system · Assembly sequence optimization
Robot task-level planning · Multi-finger gripper for flexible assembly
Vision-guided robotic assembly

1 Introduction

In the past, automation of assembly operations in industries had been successfully accomplished by employing fixed automation equipment for mass production, resulting in lower prices of products due to economies of scale. The reasons for its success can be attributed to primarily stable business environments, in which customers did not demand product differentiation. However, in today's era of mass customization, mass-produced products do not satisfy customers, which are designed largely to fulfil the average needs of customers. Therefore, flexibility is needed in assembly production systems with the necessary agility and adaptability to diverse customer requirements and capability of mass customization. Yet, the production costs of mass customized products should be low enough so that their prices are not significantly different from those of comparable products made by mass production. In order to cope with this product variety-cost trade-off, robotics offers a flexible automation technology for turning assembly systems into efficient and flexible systems (Deb and Deb 2010). Reconfiguring the robotic assembly system for new products involves a change in the software program (which is accomplished by simply reprogramming the same robot) as opposed to major hardware changes that may be required in case of hard automation. This would permit quick changeover of the assembly system in response to customer demands. Thus, several products can be made using a single robotic workcell, resulting in saving costs associated with multiple installations. Despite their great potential for high flexibility, there is a range of challenging issues which must be addressed for successful implementation of flexible assembly systems using robots. This chapter examines a number of these issues, reviews the results of previous research and describes our ongoing research on development of a flexible assembly system for mechanical products, using an industrial robot with machine vision guidance and dexterous multi-finger gripper.

2 Review of Previous Research

2.1 Assembly Planning and Optimization

Before programming the robot motions for carrying out assembly, it is first necessary to plan the order or the sequence in which the components need to be assembled. Further, it is important to identify the optimal assembly sequence to minimize the assembly time and cost. Various approaches have been developed as reported in the literature for automatic generation of optimal sequence. A few of them include application of mathematical algorithms, graph theoretic approaches, and more recently various evolutionary soft computing based optimization techniques which include GA, memetic algorithm, ant colony optimization, artificial immune systems, cuckoo search algorithm, flower pollination algorithm, harmony search algorithm, particle swarm optimization, and hybrid of the above. A brief review of these approaches is given below.

Chen et al. (2004) used various graph-based methods such as above graph, Assembly Precedent Diagram (APD), and relational model graph to model the relative positions, precedence constraints and relationships between different parts of an assembly. For the generation and evaluation of a feasible assembly sequence, two methods namely a mathematical model based on penalty index and Revised Minimum Spanning Table (RMST) methods were used. To solve the assembly sequence optimization problem, Bonneville et al. (1995) developed a Genetic Algorithm (GA)-based approach, starting from feasible assembly sequences provided by experienced assembly planners. To generate the offspring from the parent population, crossover and mutation operators were used followed by evaluation and selection of resulting offspring. The feasibility of operations was checked by the Liaisons and geometric constraints. Chen and Liu (2001) developed an adaptive GA (where rules were used to vary the genetic-operator probabilities accordingly) for obtaining the global best or near best assembly sequences efficiently. Marian et al. (2006) also used GA for assembly sequence planning optimization where guided search was used to generate the solutions taking both extrinsic and intrinsic precedence relations into consideration. Stochastic sampling selection, guided search with supplementary conditions type crossover and modified guided search mutation operators were used. Choi et al. (2009) solved the assembly sequence optimization problem by an approach based on GA in which precedence preservative type crossover and swap mutation had been used. Mishra and Deb (2016a) proposed a GA-based assembly sequence optimization approach in which a partially matched crossover and a swap mutation were used. Wang et al. (2005) proposed an Ant Colony Optimization (ACO) approach for optimization of assembly sequences based on minimizing the reorientations during the assembly processes. The concept of assembly by disassembly was adopted by the authors, and disassembly matrix was used to ensure the feasibility of sequences. However, the ACO algorithm has a lot of parameters which needs to be varied in order to obtain the best convergence rate of the algorithm. A binary-coded GA was proposed by Mishra and Deb (2016b) to optimize the parameters of the ACO algorithm. Cao and Xiao (2007) developed the Immune Optimization Algorithm (IOA) for generation of optimal assembly plan. It worked on the bionic principles of artificial immune systems. Attributes such as degree of feasibility, base component location in the sequence, and number of tool changes and reorientations required were used to evaluate the assembly sequences. Lv and Lu (2010) showed an application of discrete Particle Swarm Optimization (PSO) in assembly sequence optimization. Number of tool changes, number of reorientation required, number of changes in operation type and number of interferences during the product assembly were considered as optimization criteria. The particle positions and velocities in PSO were updated using subtraction, addition and multiplication operators. Gao et al. (2010) proposed a memetic algorithm based assembly sequence optimization approach on where assembly sequences are treated as chromosomes that consist of genes containing the component number and its assembly direction. Number of reorientation required during assembly and assembly sequence feasibility as the objective function were considered. Generation of new offspring was done using PMX-type crossover and swap mutation along with the local search operator. Li et al. (2016) proposed an Improved Harmony Search (IHS) for assembly sequence optimization. The paper demonstrated novel features like an initial Harmony Memory (HM) using the Opposition-Based Learning (OBL) method, a way to improvise a new harmony and a strategy for local search. Number of reorientations required during assembly, number of tool changes, and stability criteria in the objective function were considered. Mishra and Deb (2016c) developed a discrete Flower Pollination Algorithm (FPA)-based assembly sequence optimization approach for generating not only the global best assembly sequence but as many unique optimum solutions as possible. Mishra and Deb (2017) proposed a discrete cuckoo search algorithm to generate global best assembly sequences considering minimization of orientation changes, tool/gripper changes, assembly stability and base component location. It was integrated with an upstream assembly product database to extract information on assembly directions, contact details, assembly precedence constraints and tool/grippers. Zhou et al. (2011) used a hybrid bacterial chemotaxis and GA-based approach for assembly sequence optimization. The chromosomes were the assembly sequences, wherein gene of the chromosome was the bacterium. Length of longest sub-sequence, number of reorientations required and number of tool changes were considered in fitness function. Xing and Wang (2012) developed a combination of PSO and GA approach based on graph theory for compliant assembly sequence optimization. Liaison graph and adjacency matrix were employed to evaluate the geometry of the compliant assemblies. The string of components represented the assembly sequences, whose length is equal to number of assembly components. Evaluation of assembly sequences was based on assembly variation due to dimensional tolerance. Karthik and Deb (2017) proposed and implemented a methodology for assembly sequence optimization using a hybrid Cuckoo Search Genetic Algorithm (CSGA).

2.2 *Robot Task-level Planning*

After assembly sequence planning, a robot task-level plan containing a sequence of intermediate steps to achieve the robot task-level goals has to be drawn up from the assembly sequence. Different approaches have been discussed in the literature for automating this task. A few of them include use of agent-based techniques, object-oriented approaches, knowledge-based systems, etc. A brief review of these approaches is given below.

Ramos (1992) presented an intelligent robotic assembly system, incorporating a link with computer vision and an efficient task-level planning methodology based on a cooperative agent-based approach and the implementation of the intended task using a robot manipulator. The various agents are MODELS (representing the models of objects), VISION, world descriptor that converts numerical data given by VISION in symbolic relationships and constraints, task-level planner, task executor which is responsible for the execution and monitoring of the tasks. Osuna et al. (2003) developed an intelligent task planning system that was built upon a cooperative web-based environment to integrate the product and assembly system design processes. For intelligent task planning, object-oriented and machine learning techniques were integrated. Cambon et al. (2004) developed a planner, aSyMov. They considered symbolic and geometric constraints at each step of the planning process. The representations used by a symbolic task planner and the representations used by a realistic motion and manipulation planning library had been shown to be effectively linked. Cho et al. (2010) proposed the implementation methods of robotic task planning and execution based on the description logic knowledge base. Their implementation used a description logic knowledge base which had been expanded to encompass the description of task's goal and behaviours. A representation scheme for knowledge had been investigated and a simple algorithm was realized. A complete process chain was presented in Thomas and Wahl (2010) that starts with initial specification of assembly tasks using assembly sequence planning and finishes with task planning and execution. Their system demonstrated how to generate robot programs automatically from CAD data. Backhaus (2013) introduced and discussed an approach to simplify the use of task-oriented programming for assembly systems. The system in the output provided the device-specific code based on the input as description of assembly systems and devices (namely robot, gripper, conveyor belt, etc.). Recently, Alatartsev et al. (2015) surveyed several papers and presented their findings detailing robotic task sequencing problem considering collision-free path planning, production scheduling, multi-robot task planning, task-level planning, combination of task-level planning and path planning, online control-based planning, and manipulation planning.

2.3 Design of Robotic Grippers and Their Controller Design

In a flexible assembly system, dexterous robot grippers in the form of multi-finger anthropomorphic hands have a crucial role to play to support the grasping and manipulation of objects of diverse geometric shapes. A number of multi-finger robot gripper designs can be found in the literature. These grippers can be classified according to the nature of actuation mechanism employed, i.e. either fully actuated grippers (i.e. same number of actuators as the number of DOFs) or under-actuated grippers (i.e. lesser number of actuators than the number of DOFs). A growing trend in gripper designs is the use of under-actuated mechanisms based on linkages and gear trains (LiCheng et al. 2009; Tlegenov et al. 2014). However, the limitations of these mechanisms, especially the increase in overall size and weight, and also their lack of compliance, have prompted researchers to think of other alternative options including use of Tendon-Driven Mechanisms (TDMs), which have the potential to overcome the above drawbacks. Tendons, or more generally speaking, cables, had been widely used earlier in many mechanical devices since the nineteenth century. The use of tendons for robotic applications to develop grippers is a more recent development and has been studied since the early 1980s. Several tendon-actuated robot grippers have been developed all over the world, worth mentioning among them are the Stanford/JPL hand by Salisbury and Roth (1983) and Loucks et al. (1987), the Utah/MIT hand by Jacobsen et al. (1984, 1986), the multi-jointed finger hand by Okada (1986), the Barrett gripper by Townsend (2000), the Belgrade/USC hand by Bekey et al. (1990), DLR hand developments by Butterfass et al. (1998, 2001), Gao et al. (2003) and Liu et al. (2008), the LMS hand by Gazeau et al. (2001), the NASA Robonaut hand by Lovchik and Diftler (1999) and Ambrose et al. (2000) and JU hand by Pal et al. (2008). Despite the aforementioned developments, it has been generally found that the industrial applications of tendon-driven robot grippers are more limited as compared to other types of mechanical robotic grippers. This is so because there are some challenges in its implementation particularly when it comes to control of the system as it cannot be directly controlled by conventional control techniques. Ozawa et al. (2014) developed a tendon-driven robotic hand and implemented different types of controller. Compliant behaviour was incorporated by adding springs to the passive tendons in the fingers. Abdallah et al. (2012) developed two-tier architecture of position and force control in two feedback loops which improves the overall performances of using only position or force controller. Impedance control approach is preferred while system interacting with the environment, as it gives great flexibility by allowing regulation of both motion and contact force. Diftler et al. (2011) designed dual-priority based impedance controller for Robonaut hand.

2.4 Vision-Guided Robotic Assembly

Once the task-level plan for the required assembly is generated, a robot manipulator guided by a machine vision system can be deployed to automatically assemble the constituent parts. A few investigations in this regard are discussed below. Pena-Cabera (2005) proposed a method of object recognition and their localization for assembly components. Using a Fuzzy ARTMAP neural network model, assembly components were accurately recognized. Golnabi and Asdapour (2007) described the role and importance of the machine vision systems in industrial applications. System design methodology and a generic machine vision model were discussed. The justification for utilizing machine vision had been considered on the basis of economic and logistic considerations. Zhang et al. (2011) introduced a vision-guided alignment method that utilized a camera space manipulation control process and relied on the components' CAD model. To achieve the high accuracy alignment for the final assembly, a local calibration method was used. Kobari et al. (2013) presented an error resilient vision-based motion control method for parts assembly. They considered those assembly parts which get deformed during assembly. Visual information obtained from a camera reading the deformation of the parts is used to determine the force applied. Chen et al. (2015) reviewed papers on vision system working in combination with force control integrated assembly and demonstrated a system using a high accuracy assembly process. In a semi-structured environment, industrial robot can perform assembly more accurately by taking both vision and force control data. The vision system was not calibrated carefully as it provided only rough position data of parts. In addition to vision, to deal with errors, a local searching method was used which was based on force–torque control.

3 Proposed Methodology

The following sections discuss the methodologies that have been proposed in this work for (1) assembly sequence planning and optimization, (2) robot task-level plan generation, (3) design of a multi-finger robot gripper for flexible assembly, and finally (4) implementation of the robotic assembly system under the guidance of machine vision.

3.1 Assembly Sequence Planning and Optimization

In the following sections, the key issues and challenges in assembly sequence planning and optimization, a brief problem description and the methodology proposed in this work and the results and discussions will be presented.

3.1.1 Key Issues and Challenges

Before the robot is programmed for performing assembly of a product, it is necessary to decide the order or sequence in which the components need to be inserted, such that no component interferes with others, thereby implying that they must comply with certain precedence constraints. A significant amount of human expertise and experiential knowledge is needed to accomplish the above. Moreover, an assembly product may be assembled by following number of distinct sequences, out of which the optimal sequence must be identified. Further, as the part count (i.e. the number of components in the assembly) increases, the number of feasible assembly sequences also growths exponentially, which makes assembly sequence optimization arduous and time consuming, if it is done manually. Keeping the above in mind, in this work, a Sexual Genetic Algorithm (SGA)-based approach is proposed for automating the generation of optimum assembly sequence. First, a brief problem description is given below.

3.1.2 Brief Problem Description

The problem of assembly sequence optimization involves determining the best feasible assembly sequence based on one or more criteria such as minimization of number of direction changes and tool changes, maximizing the stability of the components/sub-assemblies while performing the assembly, etc. It is obvious that changing the direction of assembly of the consecutive components increases the handling time resulting in increase in overall cost of the assembly. Equation (1) shows how to compute the total number of direction changes (n_d).

$$n_\mathrm{d} = \sum_{i=1}^{n-1} (\mathrm{dir\_change}_{i,i+1}) \tag{1}$$

where

$$\mathrm{dir\_change}_{i,i+1} = \begin{cases} 0, & \text{if assembly\_dir}_i = \text{assembly\_dir}_{i+1} \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

$\mathrm{dir\_change}_{i,i+1}$ represents the change in direction of assembly for two successive assembly operations and n represents the number of components in the assembly, $\mathrm{assembly\_dir}_i$ is the direction of assembly for component number i (the direction of assembly for components may be one of the following namely, $\pm x$ or $\pm y$ or $\pm z$).

At times, several tools and grippers (in case of robotic assembly) are required for performing the assembly or manipulation. The change in tool/gripper is also a non-productive task as it consumes handling time leading to increase in overall assembly cost. Hence, minimizing the tool changes is also one of the important criteria to reduce the overall assembly time and cost. Equation (3) shows how to compute the number of tool changes (n_t).

$$n_\mathrm{t} = \sum_{i=1}^{n-1} (\mathrm{tool\_change}_{i,i+1}) \tag{3}$$

where

$$\mathrm{tool\_change}_{i,i+1} = \begin{cases} 0, & \text{if tool\_number}_i = \text{tool\_number}_{i+1} \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

$\text{tool\_change}_{i,i+1}$ represents the change in assembly tool/gripper for two successive assembly operations and n represents the number of components in the assembly; tool_number_i represents the tool number required for handling/insertion of component number i (a unique tool number is given to each tool in the tool database).

The base component in an assembly sequence should be the first component in assembly order. This base component information is provided by the user. The location of the base component in the sequence is represented by B, whose value is calculated as follows.

$$\text{Location of base component}, B = \begin{cases} 1, & \text{if the base component is in first position in the sequence} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

The connections between the components can be categorized into stable connection (SC), conditional stable connection (CSC) and unstable connection (USC). As the name suggests, SC signifies inseparable contact between components. Sometimes between components, there is CSC, which signifies that the connections are not autonomously stable, but components may separate from each other spontaneously. For unstable connections, fixtures are necessary to hold the components in place and maintain contact. This classification scheme and its details can be found in the paper by Li et al. (2016). To calculate the stability index of an assembly sequence, a stability (connection) matrix is used, which is a square matrix, having rows and columns equal to number of assembly components. The value of each element of the stability matrix can be 0, 1 or 2 depending on whether the connection type is unstable, conditionally stable or stable, respectively. The stability index (SI) can be found out as follows.

$$\text{SI} = \sum_{i=2}^{n} S_i | 0 \leq \text{SI} \leq 2n - 2 \tag{6}$$

where S_i is the stability of the component "i" and n is the number of components in the assembly.

The feasibility violations (FV) are the number of precedence violations because of certain component number violating the precedence. This can be understood from the example explained below, say, an assembly has four components and one of the infeasible sequences for this assembly is [4, 1, 2, 3]. Suppose that the Precedence Matrix (PM) of the assembly is as follows.

PM =

Component Number	1	2	3	4
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	1	1	1	0

(7)

This PM conveys the information as to which component has precedence with which component. The dimension of this matrix is equal to the number of assembly components. Here, 0 implies that no precedence constraint exists between the components and 1 implies that a precedence constraint exists. Now, for calculating the number of feasibility violations, from an infeasible solution, the first component is selected and checked to determine with which components it has precedence, and whether it is following the precedence or not. In this example, component ‘4’ has precedence with components ‘1’, ‘2’ and ‘3’, since the value in the PM is 1 here for each of PM(4,1), PM(4,2) and PM(4,3). So because of location of component ‘4’ in the assembly sequence, it has a total of three feasibility violations. Next, the precedence is checked for the second component of the assembly sequence, i.e. component ‘1’. It reveals that it does not require any other component to be assembled before it, and hence it has no feasibility violation. In a similar manner, each component of the assembly sequence is checked to see if it satisfies the precedence criterion or not. For the solution [4, 1, 2, 3], components 2 and 3 also do not have any feasibility violations. Hence, it can be concluded that the number of feasibility violations for the infeasible solution [4, 1, 2, 3] is 3. It is obvious that for a feasible assembly sequence, the number of feasibility violations would be zero.

3.1.3 Proposed Approach Based on Sexual Genetic Algorithm

The Genetic Algorithm (GA) tries to mimic the various processes of natural selection involved in the intricate process of biological evolution. One such process is mate choice, or sexual selection that counteracts as well as enhances natural selection. Nowadays, much of the biological diversity and complexity is ascribed to this force. The Sexual Genetic algorithm (SGA) can successfully counter to a large extent the general tendency of a GA of slow convergence and getting stuck in a suboptimal solution. Hence, a SGA-based strategy has been adopted in this work. The entire population in SGA is divided into male and female chromosomes. Reproduction is done sexually to emulate male vigour and female choice; hence, in crossover, any female chromosome will mate only with a male chromosome having better average fitness (Goh et al. 2003). The rest of the process is similar to that of a GA.

The steps involved in the SGA are stated in Fig. 1. The genetic operators modified to suit the particularities of the assembly sequence planning problem are described here as well. A flowchart of the SGA is given in Fig. 1.

Selection

A selection operator called roulette wheel selection has been used that selects the assembly solutions to form the mating pool. All the solutions are evaluated using the fitness function, and the ranges for the solutions are created according to their fitnesses. Random numbers are generated multiple times, which equals the size of the population. Depending on the ranges within which these random numbers fall in, corresponding solutions are selected. The selected solutions are then put into the

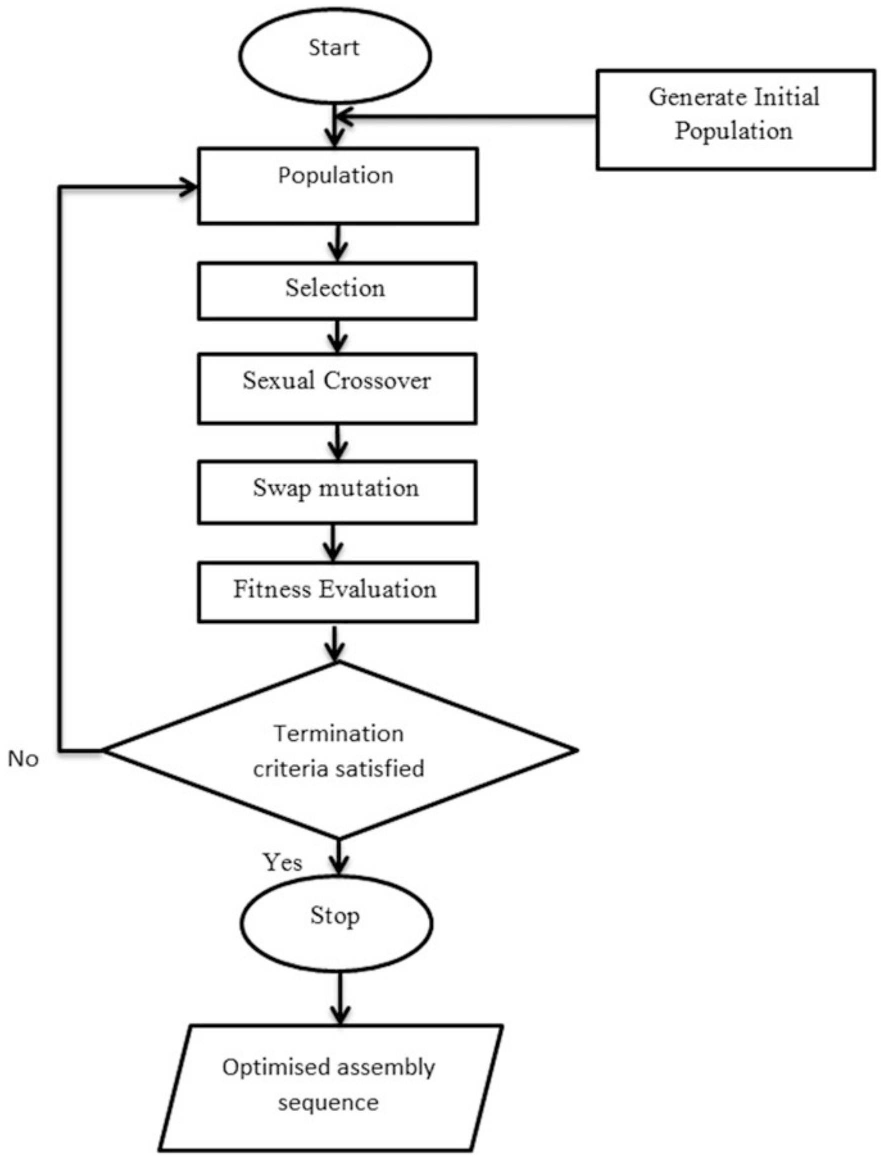

Start
Generate Initial Population
Population
Selection
Sexual Crossover
Swap mutation
Fitness Evaluation
Termination criteria satisfied
No
Yes
Stop
Optimised assembly sequence

Fig. 1 The SGA flowchart for finding the optimal assembly sequence

mating pool on which special crossover operation is carried out. The size of the mating pool is equal to population size of the GA.

Sexual Crossover

Sexual crossover is the main genetic operator that differentiates SGA from a generic GA. In the process, a female chromosome is selected to mate with a male partner of a better general fitness (best among 4–5 male chromosomes). It is ensured that the off-springs are of different genders so that the sex ratio balance is maintained. The constrained nature of assembly sequence planning and necessity to maintain feasibility at every step calls for a modified crossover technique. The steps involved are described as below:

(a) The population of chromosomes is first randomly divided into two: male population and female population, each containing equal number of chromosomes.
(b) Select each female chromosome in a consecutive order (without replacement), i.e. each female will only get to be selected to mate once.
(c) Select a male chromosome of a better general fitness (best among 4–5 male chromosomes).
(d) Apply the Partially Matched Crossover (PMX) operator as follows. Generate an offspring by selecting a part of a solution (i.e. number of successive genes) from one parent and conserving the order and the positions of as many genes as possible from the other parent. A two-point crossover is applied where the part of a solution serves as boundaries for the swapping operations. This crossover operator takes care of the problem of repetitions of part numbers in the assembly sequence.

Swap Mutation

The mutation operator called swap mutation is used in this work. The genes are swapped arbitrarily inside a solution in case of swap mutation.

Fitness Evaluation of the Population

For the given problem of assembly sequence optimization, a random population of individuals (i.e. chromosomes representing assembly sequences) is generated which can be feasible or infeasible. This population is evaluated using a fitness function (FF) given in Eq. 8, consisting of number of tool changes (n_t) direction changes (n_d), base component location in assembly sequence (B) and stability index (SI). To ensure feasibility of final individuals, a feasibility criterion is incorporated that is measured in terms of number of feasibility violations (FV) due to components' locations in the sequence.

$$\mathrm{FF} = w_1 \frac{(n-1-n_\mathrm{d})}{(n-1)} + w_2 \frac{(n-1-n_\mathrm{t})}{(n-1)} + w_3 * B + w_4 * \frac{\mathrm{SI}}{(2n-2)} + \frac{1}{\left(\frac{FV}{n}+1\right)} \tag{8}$$

where n represents number of assembly components and w_1, w_2, w_3 and w_4 are weight coefficients for direction changes, tool changes, base component location and stability index, respectively, with their corresponding values as 0.25, 0.25, 0.1 and 0.4.

3.1.4 Results and Discussions

To demonstrate the working of the proposed system, a 14-component product assembly shown in Fig. 2 has been used. Figure 3a, b shows an extract of the Precedence Matrix (PM) and Stability Matrix (SM) of the above assembly, respectively. Table 1 gives the information on the assembly directions and tools/grippers required for the assembly.

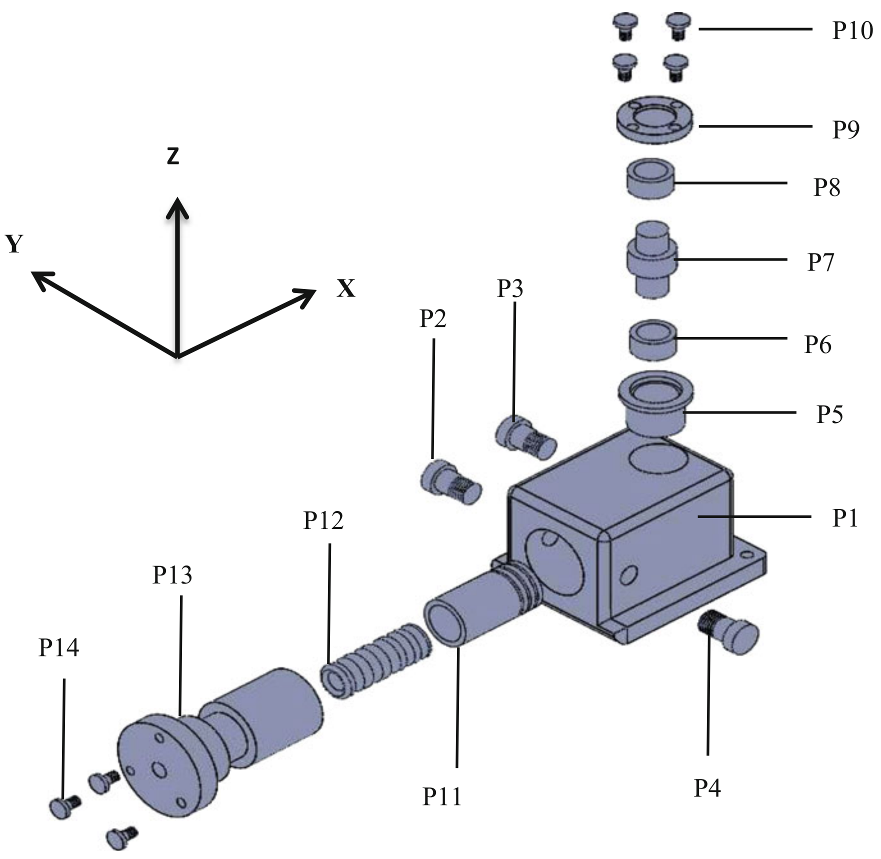
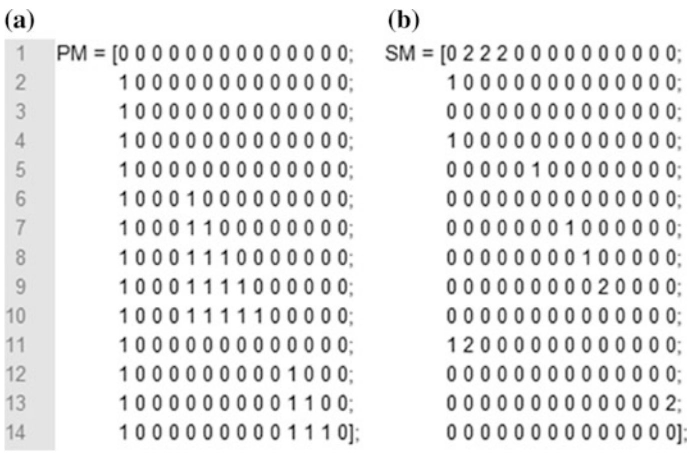


Fig. 2 14-component assembly


(a)

```
PM = [0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      1 0 0 0 0 0 0 0 0 0 0 0 0 0;
      1 0 0 0 0 0 0 0 0 0 0 0 0 0;
      1 0 0 0 0 0 0 0 0 0 0 0 0 0;
      1 0 0 0 0 0 0 0 0 0 0 0 0 0;
      1 0 0 0 1 0 0 0 0 0 0 0 0 0;
      1 0 0 0 1 1 0 0 0 0 0 0 0 0;
      1 0 0 0 1 1 1 0 0 0 0 0 0 0;
      1 0 0 0 1 1 1 1 0 0 0 0 0 0;
      1 0 0 0 1 1 1 1 1 0 0 0 0 0;
      1 0 0 0 0 0 0 0 0 0 0 0 0 0;
      1 0 0 0 0 0 0 0 0 0 1 0 0 0;
      1 0 0 0 0 0 0 0 0 0 1 1 0 0;
      1 0 0 0 0 0 0 0 0 0 1 1 1 0];
```

(b)

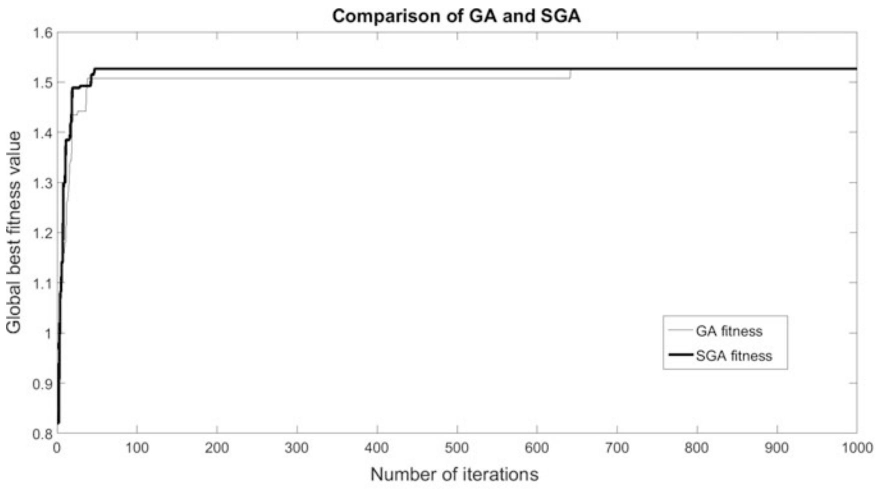
```
SM = [0 2 2 2 0 0 0 0 0 0 0 0 0 0;
      1 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      1 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 1 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 1 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 1 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 2 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      1 2 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 2;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0];
```

Fig. 3 **a** Precedence matrix and **b** stability matrix for the assembly shown in Fig. 2

Table 1 Information on assembly directions and tools/grippers required for the assembly shown in Fig. 2

Component number	Assembly direction	Tool/gripper number
1	*Z*−	1
2	*Y*−	2
3	*Y*−	2
4	*Y*+	2
5	*Z*−	3
6	*Z*−	3
7	*Z*−	4
8	*Z*−	5
9	*Z*−	5
10	*Z*−	6
11	*X*+	5
12	*X*+	5
13	*X*+	5
14	*X*+	6

The proposed SGA has been used to generate the best assembly sequence for the above 14-component product assembly. The main parameters affecting the performance of SGA algorithm are population size, crossover probability (p_c) and mutation probability (p_m) used in generating new eggs and number of iterations. To study their effects on performance of the SGA, a sensitivity analysis has been carried out and the optimum set of parameters found to be able to provide the optimal/near optimal assembly sequence in reasonable time is population size of 20, crossover probability of 0.90 and mutation probability of 0.05. After the SGA is run using the above-mentioned parameters, the optimal assembly sequence obtained is [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14] with fitness value of 1.5269, requiring seven tool changes, four reorientation changes and stability index of nine, having the base component at the first location of the assembly sequence, and most importantly the sequence was found to have no feasibility (precedence) violations. The efficiency of the proposed SGA is evaluated by comparing it with previously developed GA (Mishra and Deb 2016). The best sequence generated by the GA algorithm is also found to be [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14] with fitness value of 1.5269, requiring seven tool changes, four reorientation changes and stability index of nine, having base component at the first location of the assembly sequence and no feasibility violations. Figure 4 shows comparison between the



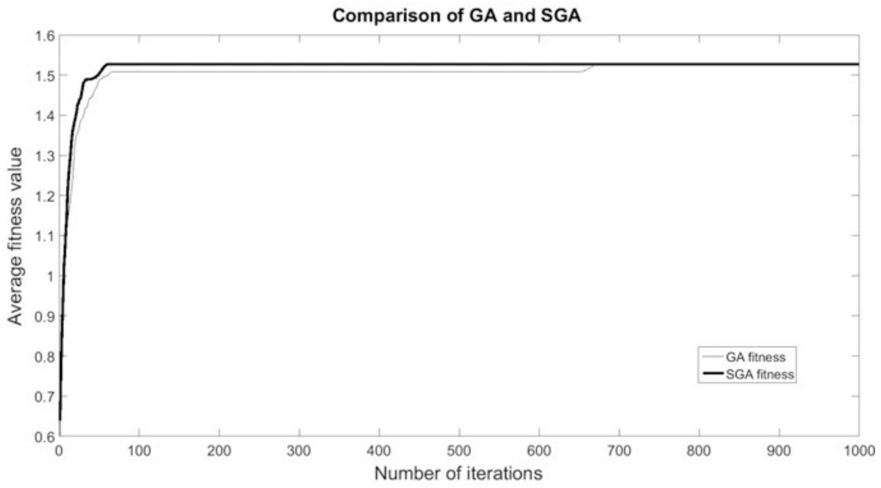


Fig. 4 Comparison between the convergence graphs of the proposed GA algorithm and SGA algorithm

Table 2 Comparison between the results of the GA and SGA

	GA	SGA
Best fitness value	1.5269	1.5269
Mean of the best fitness values for 10 runs	1.5015	1.5138
Number of runs out of 10 independent runs in which convergence to global best fitness was achieved	3	6
% of obtaining the optimal assembly sequence	30	60

convergence graphs of the GA and SGA algorithms from which it can be concluded that the proposed SGA clearly outperforms GA in terms of convergence speed as it is able to reach the global best fitness value in lesser number of iterations. The SGA reached the global best solution in 19 iterations, while the GA took 669 iterations to reach the global best solution. Table 2 gives a summary of the results of comparison between the GA and the SGA algorithms. It is found that in 6 times out of 10 runs, the SGA could find the optimal solution of 1.5269, which gives a 60% probability of finding the best solution or assembly sequence. In contrast, the probability of finding the best solution by GA is only 30%.

From the above, it is evident that the proposed SGA-based approach is able to generate the optimum solution. Furthermore, the algorithm is also more consistent than the previously developed GA-based approach. It should be noted that the optimal assembly sequence provided by the SGA requires the least number of orientation changes and tool/gripper changes, it is the most stable and most importantly it does not have any feasibility violations.

3.2 *Knowledge-Based System for the Generation of Task-level Assembly Plan*

The following sections present the key issues and challenges in generation of task-level assembly plan, architecture of the proposed knowledge-based system and the results and discussions.

3.2.1 Key Issues and Challenges

After assembly sequence planning, a robot task-level plan has to be drawn up from the assembly sequence. The task-level plan contains a sequence of intermediate steps to achieve the robot task-level goals such as, for example, for performing assembly of two parts *A* and *B*, obtain first the initial location coordinates of part *A* using a suitable sensor like an overhead vision camera, move the robot over part *A*, perform grasping of part *A* by closing the gripper, obtain the location coordinates of its destination on part *B* in the assembly jig by vision camera, move the part

A over part *B*, open gripper to insert part *A* inside part *B*, etc. For a product requiring large number of components to be assembled, the traditional method of manually generating the aforementioned task-level plan is very tedious and time consuming. Although this task is usually performed by experts, any inadvertent mistake can be very costly and result in loss of productivity. Hence, it will be beneficial if this task is automated, for which a knowledge-based expert system approach has been proposed in this work.

A knowledge-based system has been developed to generate the task-level plan for robotic assembly from a given feasible and optimum assembly sequence that is generated by the assembly sequence planner described in the previous section. The above knowledge-based system has been implemented using the expert system shell, CLIPS.

3.2.2 Architecture of the Proposed Knowledge-Based System

The 'C Language Integrated Production System', abbreviated as 'CLIPS', is an expert system shell that was first developed at NASA (Giarratano and Riley 2002). It basically consists of a set of knowledge-based rules, supported by facts in order to fire or activate a rule that are stored in a database. An executable CLIPS rule basically consists of two parts with an 'IF' and a 'THEN' statement. In order for the rule to fire successfully to give a desired output, the syntax of its 'IF' part must match with the syntax of the facts provided in the database. If this condition is satisfied, the 'THEN' part of the rule comes into action. Both the facts and the IF-THEN rules must be defined first, before being used. The CLIPS system consists of mainly three components namely, a database, a knowledge base and an inference engine. In addition, the CLIPS system also has a user interactive interface. The architecture of the developed knowledge-based system using CLIPS has been shown in Fig. 5.

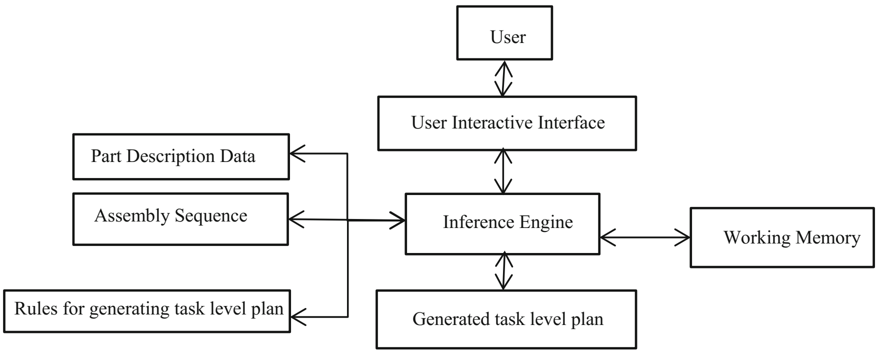

User
User Interactive Interface
Part Description Data
Assembly Sequence
Inference Engine
Working Memory
Rules for generating task level plan
Generated task level plan

Fig. 5 Architecture of the developed knowledge-based system

Database

The database consists of a list of facts about the problem being solved, which forms the input to the expert system and must be entered by the user. The facts are represented in the CLIPS expert system shell using the concept of template. A template is an organized structure of input data that is used to define and store values for a fact description in a field called, the ‘slot’. For the given problem which is to generate a task plan for the robot, the input data required are detailed information about the assembly components, such as component number, its name, type (i.e. functional part or fastener), part count, components with which it has contact, assembly direction, any special assembly requirement, etc., which are to be provided by the user. All these data are stored in the database. An operational template in CLIPS format is presented in Fig. 6a, along with a sample of input database facts given in Fig. 6b.

Knowledge Base

The knowledge base contains domain knowledge encoded in the form of rules, to be used by the expert system. Sample rules for the knowledge base are presented

(a)

```
; DEFTEMPLATE 1
            (deftemplate MAIN::component
            (slot number (type INTEGER)(default ?NONE))
            (slot name (type SYMBOL))
            (slot component_type(type SYMBOL)(allowed-symbols functional_part fastener))
            (slot part_count(type NUMBER)(default 1))
            (multislot contact_with(type NUMBER)(default 0))
            (slot thread_present(type SYMBOL)(allowed-symbols yes no)(default no))
            (slot tool_required(type SYMBOL)(allowed-symbols screwing_machine …….))
            (slot assembly_direction (type SYMBOL)(allowed-symbols x+ y+ z+ x- y- z-))
            (slot assembly_requirement (type SYMBOL)(allowed-symbols pre_requirement  ……)))

; DEFTEMPLATE 2
            (deftemplate MAIN::assembly_method
            (slot assembly_method (type SYMBOL)(allowed-symbols manual robotic hybrid)))

;DEFTEMPLATE 3
            (deftemplate MAIN::sequence
            (multislot in_order_of (type INTEGER)))
```

(b)

```
(deffacts MAIN::components_data
            (component(number 1)(name base)(component_type functional_part)(part_count 1)(contact_with 2 3 4 5 7 9
            10 11 13 14)(thread_present no)(assembly_direction z-))
            ………………………))

(deffacts MAIN:assembly_method
                        (assembly_method (assembly_method robotic)))

(deffacts MAIN::seqeunce_data
                        (sequence (in_order_of 1 2 3 4 5 6 7 8 9 10 11 12 13 14)))
```

Fig. 6 **a** Templates for entering the assembly information in databases, **b** assembly information facts stored in the database, **c** a rule for generating task-level plan

(c)

```
; DEFRULE 1a
            (defrule MAIN::rule_1a
            (component(number ?nu1)(name ?na1)(component_type functional_part)(part_count 1))
            (sequence (in_order_of ?nu1 $?))
            (assembly_method (assembly_method robotic))
            =>
            (open "assembly_plan.txt" assembly_plan "w")
            (format assembly_plan "obtain coordinates of %s(%d) by vision%n" ?na1 ?nu1)
            (format assembly_plan "MOVE robot in joint mode above %s(%d)%n" ?na1 ?nu1)
            (printout assembly_plan "open gripper" crlf)
            (format assembly_plan "MOVE robot in linear mode to %s(%d)%n" ?na1 ?nu1)
            (printout assembly_plan "close gripper" crlf)
            (printout assembly_plan "obtain coordinates  of assembly fixture" crlf)
            ……………………….
            ……………………….
            (close))

; DEFRULE 3a
            (defrule MAIN::rule_3a
            (sequence (in_order_of $? ?nu1 ?nu2 $?))
            (component (number ?nu1)(name ?na1)(component_type functional_part)(assembly_direction ?ad1))
            (component (number ?nu2)(name ?na2)(component_type functional_part)(part_count ?pc2)
            ……………………….))
            (test (eq ?ad1 ?ad2))
            =>
            (open "assembly_plan.txt" assembly_plan "a")
            (printout assembly_plan "------------------------------" crlf)
            (format assembly_plan "obtain coordinates of %s(%d) by vision%n" ?na2 ?nu2)
            (format assembly_plan "MOVE robot in joint mode above %s(%d)%n" ?na2 ?nu2)
            (format assembly_plan "MOVE robot in linear mode to component %s(%d)%n" ?na2 ?nu2)
            (printout assembly_plan "close gripper" crlf)
            (printout assembly_plan "obtain coordinates  of assembly fixture" crlf)
            (printout assembly_plan "MOVE robot in joint mode above fixture" crlf)
            (printout assembly_plan "MOVE robot in linear mode to fixture" crlf)
            (format assembly_plan "open gripper and insert the component %s(%d)%n" ?na2 ?nu2)
            (printout assembly_plan "MOVE robot in linear mode above fixture" crlf)
            (printout assembly_plan "MOVE robot in joint mode to safe point" crlf)
            (format assembly_plan "REPEAT steps of block %d times%n" ?pc2)
            (close))
```

Fig. 6 (continued)

later in this section for illustration. For the given problem of generating the task-level plan, a set of production rules have been developed using domain knowledge for task-level planning to efficiently program a robotic assembly system. These rules are stored in the knowledge base of the expert system. An extract from knowledge base rules and database facts in CLIPS format are shown in Fig. 6b, c, respectively.

Inference Engine
The inference engine in CLIPS serves as the brain of the knowledge-based system. It is a computer program, used for making inferences from the given data facts

about a problem, through a process of reasoning. In CLIPS, the knowledge being stored in the form of rules, the inference engine makes inference by deciding, as to which of the rules from the knowledge base are satisfied by the database facts, and then it prioritizes the order of firing the satisfied rules.

User Interface
The user interactive interface in CLIPS consists of a display mechanism, to monitor each and every processing step during the execution of a program. It can be customized, by selecting or deselecting the options for viewing the status of database, rule activations, etc. during execution of the program in real time, which is sometimes useful for debugging purpose.

3.2.3 Results and Discussions

Example of an eight-component assembly as shown in Fig. 7 is given to demonstrate the results of the proposed knowledge-based system for generation of the task-level plans. Figure 8 shows the corresponding task-level plan generated by the knowledge-based system.

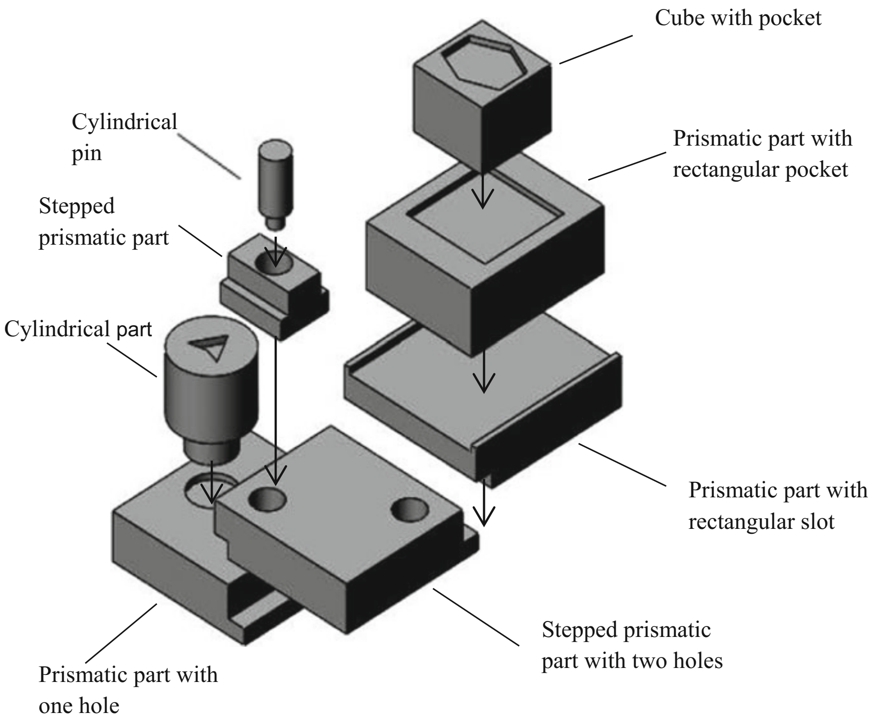

Cube with pocket
Cylindrical pin
Prismatic part with rectangular pocket
Stepped prismatic part
Cylindrical part
Prismatic part with rectangular slot
Stepped prismatic part with two holes
Prismatic part with one hole

Fig. 7 An eight-component assembly

```
obtain coordinates of prismatic_part_with_one_hole(1) by vision
MOVE robot in joint mode above prismatic_part_with_one_hole (1)
open gripper
MOVE robot in linear mode to prismatic_part_with_one_hole (1)
close gripper
check if assembly jig is empty
obtain coordinates  on assembly jig where prismatic_part_with_one_hole is to be placed
MOVE robot in joint mode above jig
MOVE robot in linear mode to jig
open gripper
MOVE robot in linear mode above jig

obtain coordinates of stepped_prismatic_part_with_two_holes(2) by vision
MOVE robot in joint mode above stepped_prismatic_part_with_two_holes (2)
MOVE robot in linear mode to component stepped_prismatic_part_with_two_holes (2)
close gripper
obtain the coordinates on assembly jig where stepped_prismatic_part_with_two_holes is to be placed
MOVE robot in joint mode above jig
MOVE robot in linear mode to jig
open gripper and insert the component stepped_prismatic_part_with_two_holes (2)
MOVE robot in linear mode above jig
MOVE robot in joint mode to safe point
REPEAT steps of block 1 times

obtain coordinates of prismatic_part_with_rectangular_slot(3) by vision
MOVE robot in joint mode above prismatic_part_with_rectangular_slot (3)
MOVE robot in linear mode to component prismatic_part_with_rectangular_slot(3)
close gripper
obtain the coordinates on assembly jig where prismatic_part_with_rectangular_slot is to be placed
MOVE robot in joint mode above jig
MOVE robot in linear mode to jig
open gripper and insert the component prismatic_part_with_rectangular_slot (3)
MOVE robot in linear mode above jig
MOVE robot in joint mode to safe point
REPEAT steps of block 1 times
 …………………………..
 …………………………..
```

Fig. 8 Extract of the task-level plan for the eight-component assembly

3.3 Proposed Multi-finger Robot Gripper for Flexible Assembly

In the following sections, the key issues and challenges in design of multi-finger robot gripper for flexible assembly, the proposed gripper design and the results and discussions will be presented.

3.3.1 Key Issues and Challenges

As manufacturing is increasingly shifting away from high-volume, low-mix production to high-mix, low-volume production, flexible assembly cells are steadily gaining importance. Such assembly cells commonly require parts of diverse geometric shapes and sizes to be handled to perform various tasks like picking, fixing, placing, mating, etc. The mechanical grippers routinely used in industrial robots to perform above assembly tasks are simple two-finger grippers or parallel jaw-type grippers, employing actuation mechanisms based on various types of linkages. They must be, however, custom engineered according to specific application requirements of grasping. For example, to hold objects whose basic shape is cuboidal having flat faces, a gripper design with flat fingertips is preferred. On the other hand, in case of objects whose basic shape is cylindrical or spherical, a different gripper design with fingertips having V-grooves must be employed to provide a larger area of contact with the curved surfaces of the object. Thus, these two-finger or parallel jaw-type grippers are very effective for handling operations in fixed assembly cells, where part shapes are either only cuboidal or only cylindrical. But they lack the grasping flexibility like human hands to adapt to objects of diverse geometric shapes and therefore if such type of two-finger grippers were to be employed for flexible assembly, multiple grippers will be needed. Additionally, frequent gripper changes in the robot will be necessary, resulting in loss of productivity. An alternative solution could be to design a single multi-finger robot gripper, inspired by a human being's hand that will be capable of performing grasping and manipulation of variety of different object shapes without the need for changing the gripper. However, such type of gripper design may be too complex to engineer in practice and not economically viable for industrial applications. Thus, it is necessary to develop simple multi-finger robotic gripper designs that will be able to overcome the limitations of existing two-finger gripper designs and will have necessary flexibility like that of a human hand to adapt to different geometric shapes of objects.

Further, the grippers actuated using linkages lack the compliance required for parts mating and insertion operations during assembly. This drawback can be handled by introducing a compliant behaviour into the gripper. There are two ways of ensuring compliant behaviour: either by a passive compliance or by an active compliance. Passive compliant behaviour can be achieved using a tendon-driven mechanism. The active compliance of the gripper can be ensured by a purposely designed control system, e.g. impedance control, which is an indirect method of force control via motion control without explicit force feedback.

Keeping the above in mind, a tendon-driven, multi-finger gripper is being proposed in this work that will have necessary flexibility like that of a human hand to adapt to different geometric shapes of objects and a control algorithm with necessary compliance to perform parts mating and insertion.

3.3.2 Proposed Multi-finger Gripper

Figure 9 shows the proposed model of a three-finger gripper. Ulrich et al. (1988) presented a more detailed description of the gripper mechanism. Sainul et al. (2016) discussed finger actuation mechanism for a three-fingered hand. The gripper consists of three identical fingers with each finger having three links namely knuckle, middle and distal links. Base of the thumb finger is fixed on the palm of the gripper. The other two opposing fingers have revolute joints at the base which helps fingers to spread sideways. Further, knuckle-middle and middle-distal links are connected using two more revolute joints. These two revolute joints help finger to accomplish closing and opening motion during grasping.

Proposed Actuation Mechanism for the Gripper

The articulated fingers of the gripper are actuated by a tendon-driven mechanism where remotely placed actuators at the palm generate grasping force to the gripper. Pulleys are used to join the knuckle-middle links and the middle-distal links. Non-stretchable tendons run over the pulleys and routing points which are used to guide the tendons as shown in Fig. 10. Spring-loaded tendons or spiral/torsional springs at the joints are used for extension motion of the fingers. The spreading of the opposing fingers is achieved by a DC motor and worm gear system placed inside the palm. One DC motor for each finger is used to drive the tendon–pulley system. A total of four DC brushless motors are used to actuate all the eight joints of the gripper.

An end of a tendon is attached to the DC motor and the other end is fixed at the distal link. The tendon force generates torque at the middle and distal joints. Pulling tendons can only generate force in one direction (tendons only can be pulled, not push), so two tendons are required to control a single joint, one tendon for each

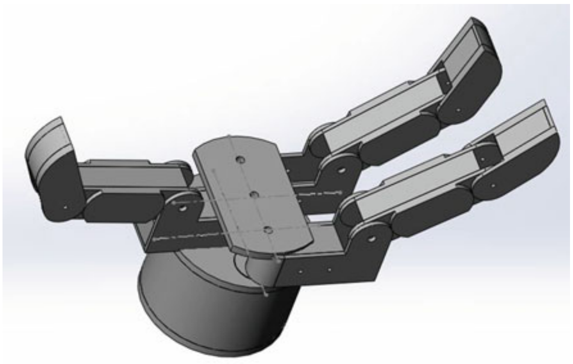


Fig. 9 Model of the robot gripper

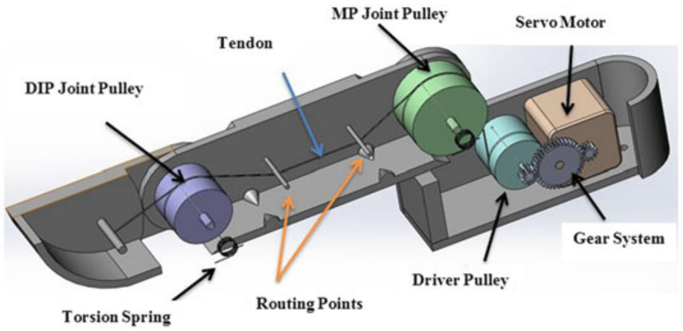


Fig. 10 Model of the tendon-driven finger showing the tendon and pulley arrangement

direction, i.e. a finger having n joints requires a total of $2n$ number of tendons for controlling all the joints independently. Although a minimum of $n+1$ tendons can independently control the joints with limited control capabilities which is implying that a finger having two joints needs at least three tendons, here, one tendon is used for finger flexion and two spring-loaded tendons are used for finger extension and they are called active and passive tendons, respectively.

Let n is the total number of joints, m is the total number of active and passive tendons, where $m = n + 1$, $l \in R^m$ be the tendon displacements, and $q \in R^n$ be the joint displacements. Then, the relation between the two displacements is as follows:

$$\boldsymbol{l} = \boldsymbol{J_j q} + \boldsymbol{l_0} \tag{9}$$

where $J_j \in R^{m \times n}$ is the Jacobian matrix and $l_0 \in R^m$ be the initial tendon displacements vector.

Let $\tau \in R^n$ be the joint torques and $f_t \in R^m$ be the tendon forces. Then, the relation between them is as follows.

$$\tau = J_j^T f_t \tag{10}$$

Let r_1 and r_2 be the pulley radii. Then, the Jacobian matrix for tendon arrangement as shown in Fig. 11 can be written as follows:

$$\boldsymbol{J}_j = \begin{bmatrix} r_1 & r_2 \\ r_1 & -r_2 \\ -r_1 & 0 \end{bmatrix} \tag{11}$$

DC motors control the tendon displacements or the tendon tension forces of the active tendons. The other two passive tendons are not actively controlled. Use of only one active tendon to control two joints of a finger makes the system under-actuated. Here, the use of passive tendons ensures that the joint displacements are uniquely determined for given tendon displacements and makes the

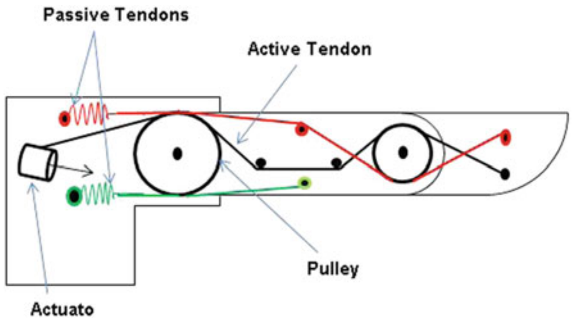


Fig. 11 Internal view of tendons run over the pulleys and routing points

system controllable in tendon space, i.e. tendon displacements and tendon forces. As the force in the passive tendons cannot be actively controlled, a limited set of joint displacements or torques can be achieved.

Control System of the Gripper

In recent times, the control of under-actuated system has drawn a lot of attention in the robotics field due to various advantages of under-actuated system. Under-actuated system requires less number of actuators compared to the full-actuated system which reduces the overall weight. Although the control system for under-actuated system is lot more challenging than the conventional full-actuated system, object grasping task is divided into two stages, pre-shaping of the fingers before grasping and gripper closure. In pre-shaping stage, the gripper adjusts its finger position by spreading the opposing fingers according to the various types of grasp which depend on the shape of object. During the pre-shaping, fingers make no interaction with the object. Gripper closure involves under-actuated control of the last two joints. The dynamic equation of a finger with three joints is as follows:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{H}(\dot{\boldsymbol{q}}, \boldsymbol{q}) + \boldsymbol{G}(\boldsymbol{q}) = \tau \tag{12}$$

where $M \in R^{3\times3}$ be the inertia matrix, $H \in R^3$ is the joint velocity and position dependent Coriolis term, and $G \in R^3$ is the nonlinear gravity term.

A subsystem of the dynamic Eq. (12) with last two joints is considered in the subsequent section, as tendons are used only for the last two joints of a finger. Coefficients with bar line are used to differentiate with derived system from the original system. The joint torque and tendon force relation is as follows:

$$\bar{\tau} = \boldsymbol{J}_j^T \boldsymbol{f}_{\mathrm{t}} \tag{13}$$

However, force in the passive tendons cannot be controlled as they are spring loaded. The force vector f_{t} is partitioned to separate the active component f_{a} from the passive component f_{p}. Then, the control input is as follows:

$$\bar{\tau} = \boldsymbol{J}_j^T f_{\mathrm{a}} + \boldsymbol{J}_{\mathrm{p}}^T \boldsymbol{f}_{\mathrm{p}} \tag{14}$$

by, putting Eq. (14) in Eq. (12) and rearranging the passive component, the dynamic equation of motion is obtained as follows:

$$\bar{\boldsymbol{M}}\ddot{\bar{\boldsymbol{q}}} + \bar{\boldsymbol{H}}(\dot{\bar{\boldsymbol{q}}}, \bar{\boldsymbol{q}}) + \bar{\boldsymbol{G}}(\bar{\boldsymbol{q}}) - \boldsymbol{J}_{\mathrm{p}}^T \boldsymbol{f}_{\mathrm{p}} = \boldsymbol{J}_j^T f_{\mathrm{a}} \tag{15}$$

where $\overline{M}$ be the inertia matrix, $\overline{H}$ be the Coriolis force, $\overline{G}$, be the gravity force, $\bar{q}$ and $\bar{\tau}$ are joint displacement and joint torque of the last two joints, respectively. The relation between the tendon forces and tendon displacements for the spring-loaded tendons is as follows:

$$\boldsymbol{f}_{\mathrm{p}} = \boldsymbol{K}\boldsymbol{l} = \boldsymbol{K}\boldsymbol{J}_{\mathrm{p}}\bar{\boldsymbol{q}} \tag{16}$$

where K be the spring stiffness matrix of the spring-loaded tendons.

Impedance Control of the Gripper

Diameter of the pulleys is chosen in such a way that generated torque at the middle joint is greater than the distal joint, i.e. $\tau_2 > \tau_3$. Such design makes sure that the applied tendon force moves middle joint first, once it is stopped by touching object or joint limit, then only distal joint starts to move. Figure 12 shows finger closure operation, where initially only first link moves, and then second link starts to move after first link touches the object.

During the grasping tasks, fingers make physical interaction with the environment. The generated contact forces due to interaction with the object cause a deviation from the desired finger trajectory. Impedance control is preferred for tasks involving interaction with the environment which regulates motion and contact forces without explicit force feedback. It is not possible to control all the joints independently due to the under-actuation mechanism (i.e. less number of control inputs than the total number of joints in a finger), so inspired by the work of Arai

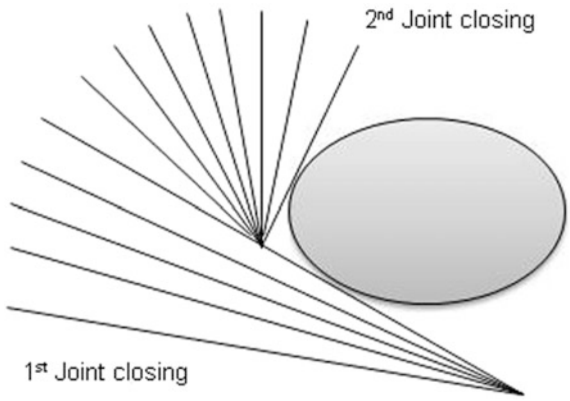


Fig. 12 Finger closure operation

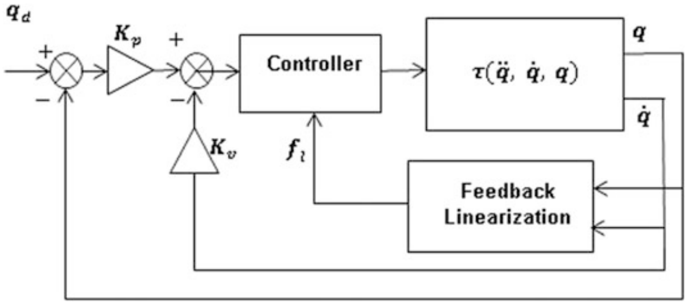


Fig. 13 Impedance-based grasp controller in joint space

and Tachi (1991); a two-phase control approach is proposed for controlling the joint displacements as well as the contact forces exerted by the links in joint space. In the first phase, tendon force controls only the middle joint. Then, in next phase, tendon force controls the distal joint.

Figure 13 shows the impedance-based grasp controller in joint space. In the presence of contact force, the dynamic equation of motion is as follows:

$$\overline{M}\,\ddot{\overline{q}} + \overline{H}(\dot{\overline{q}}, \overline{q}) + \overline{G}(\overline{q}) - J_{\mathrm{p}}^{T} K J_{\mathrm{p}} \overline{q} + \tau_{\mathrm{c}} = J_{j}^{T} f_{\mathrm{a}} \tag{17}$$

where τ_{c} is the generated torque at the joints due to the contact force.

From Eq. (17), the dynamic equation can be rewritten as

$$m_{22}\ddot{q}_2 + m_{23}\ddot{q}_3 + v_2 - t_2 + \tau_{\mathrm{c}}^{1} = r_1 f_{\mathrm{a}} \tag{18}$$

$$m_{32}\ddot{q}_2 + m_{33}\ddot{q}_3 + v_3 - t_3 + \tau_{\mathrm{c}}^{2} = r_2 f_{\mathrm{a}} \tag{19}$$

where $t = J_{\mathrm{p}}^{T} K J_{\mathrm{p}} \overline{q}$ and $v = \overline{H}\left(\dot{\overline{q}}, \overline{q}\right) + \overline{G}(\overline{q})$.

The following are the impedance control laws with feedback linearization.

First phase: Control law is

$$f_{\mathrm{a}} = K_{\mathrm{p}}\left(q_2 - q_2^{d}\right) - K_v \dot{q}_2 + \frac{1}{r_1}\left((v_2 - t_2) + \frac{m_{23}}{m_{33}}(t_3 - v_3)\right) \tag{20}$$

At equilibrium, generated torque at the middle joint due to contact force is as follows:

$$\tau_{\mathrm{c}}^{1} = r_1 K_{\mathrm{p}}\left(q_2 - q_2^{d}\right) \tag{21}$$

Second phase: Control law is

$$f_{\mathrm{a}} = K_{\mathrm{p}}\left(q_3 - q_3^{d}\right) - K_{\mathrm{v}} \dot{q}_3 + v_3 - t_3 \tag{22}$$

At equilibrium, generated torque at the distal joint due to contact force is as follows:

$$\tau_c^2 = r_2 K_p \left(q_3 - q_3^d\right) \tag{23}$$

3.3.3 Results and Discussions

The impedance control has been implemented on the proposed three-finger gripper. Table 3 shows all the parameters of each finger of the gripper, where (l_1, l_2, l_3) and (m_1, m_2, m_3) are the link lengths and masses of the knuckle, middle and distal links, respectively.

Figure 14 shows the joint displacement responses for implementation of impedance control laws as discussed in previous section. A time step of 10^{-3} s, proportional gain of $K_p = 10$ and derivative gain of $K_v = 0.1$ are used for the

Table 3 Finger parameters of the gripper

l_1 (mm)	l_2 (mm)	l_3 (mm)	m_1 (g)	m_2 (g)	m_3 (g)	r_1 (mm)	r_2 (mm)
60	60	40	50	50	40	10	8

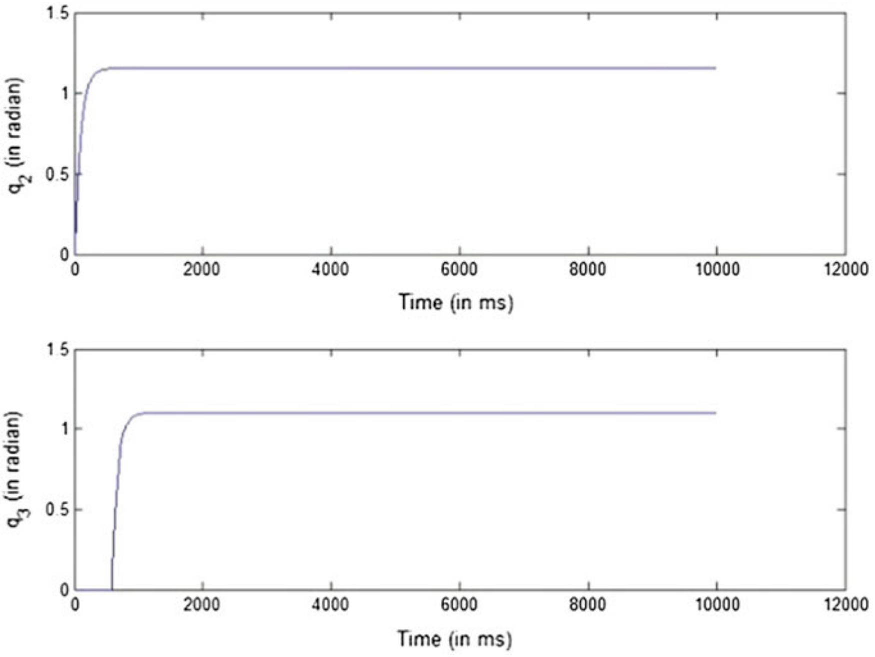


Fig. 14 Results of impedance control

implementation. From the figures, it is clearly seen that the middle link starts to move first and once the desired joint values are reached, then only the distal link starts to move.

Figures 16 and 17 show the grasping poses for cylindrical and cuboidal objects. The snapshots show the intermediate stages of pre-shaping and fingers closure. Figure 15a shows the pre-shaping stage where the two opposing fingers spread sideways. Figure 16b, c shows the implementation of the proposed two-phase control approach for gripper closure, Fig. 15b shows the closure of middle link and Fig. 15c shows the distal link closure, once the middle link is stopped by touching the object. Pre-shaping operation is not required for grasping of object shown in Fig. 16.

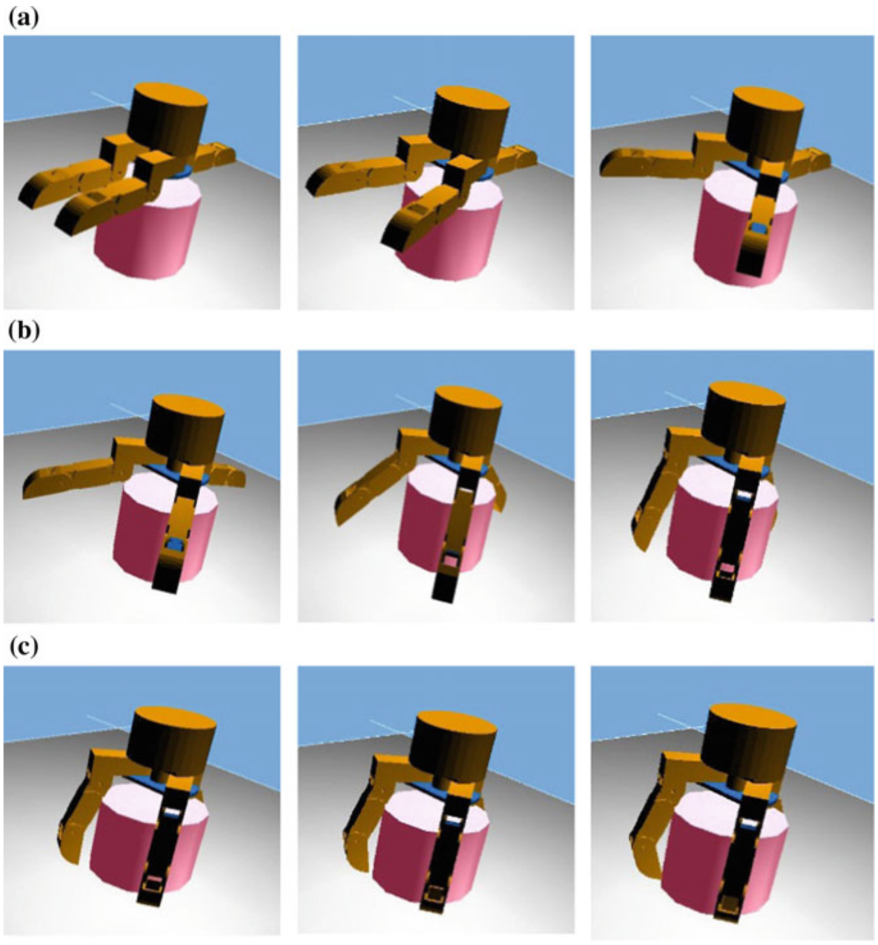


Fig. 15 Finger grasping poses for a cylindrical object

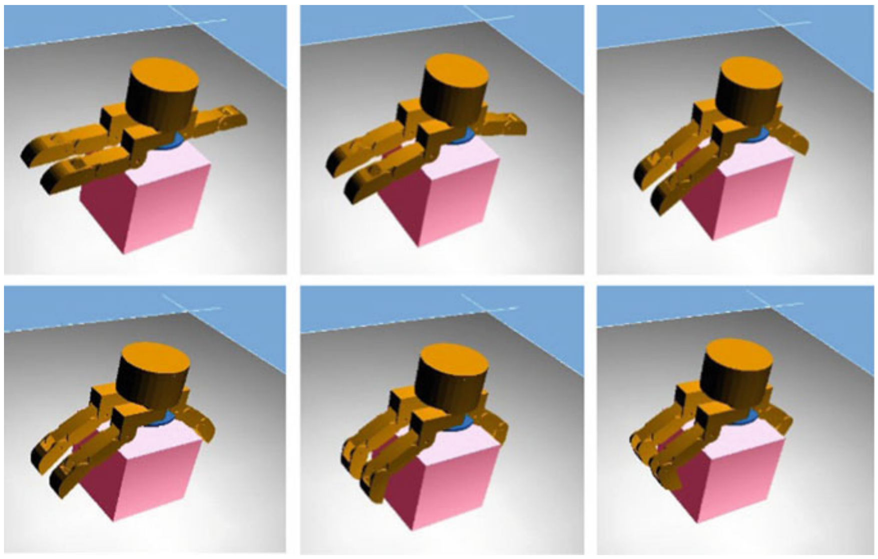


Fig. 16 Finger grasping poses for a cuboidal object

3.4 Vision-Assisted Robotic Assembly

In the following sections, the key issues and challenges in implementation of the assembly task-level plan using a vision-guided robotic system will be presented.

3.4.1 Key Issues and Challenges

In accordance with the task-level plan, the robot needs to be programmed to perform the assembly operations. The conventional online programming requires, first, the desired motion cycle to be taught to the robot by manual lead-through technique or powered lead-through technique with the help of a teach pendant to carefully define robot end-effector positions and orientations during and at the end of the motion. Then, it requires the programmer to prepare textual commands, describing the desired motion of the manipulator, and finally the program is entered into the controller memory for playback. Obviously, the online programming method is arduous and time consuming. The quality of motions and the end-effector positions taught relies on the skill level of the programmer. This exercise may have to be repeated again for robotic assembly of a new product even with slight variations in design, and thus it lacks the flexibility and reusability. To overcome the above drawbacks, a strategy for automatic programming of industrial robots to perform assembly operations by machine vision guidance has been proposed in this work. Furthermore, as inspection is important in any assembly process to rule out the possibility of erroneous robotic assembly, a strategy for automated assembly inspection based on machine vision has been also proposed. The aforementioned proposed system has been demonstrated by performing a mechanical assembly using a vision-assisted robotic assembly system, comprising of an industrial robot manipulator, an overhead CCD camera and the National Instrument (NI)'s LabVIEW graphical programming environment.

3.4.2 Implementation of Assembly Task Plan by Vision-Guided Robot

After the task-level plan is generated using knowledge-based system described in Sect. 3.2, it is implemented by a vision-guided robotic system that has been developed. Figure 17 shows the system setup. It consists of a robotic system comprising of a six-axis industrial robot manipulator (model Yaskawa Motoman MH5) equipped with a two-finger pneumatic gripper and a robot controller (model FS100) with open software architecture and a robot control interface (Digimetrix), a vision system comprising of an overhead-mounted CCD camera and NI vision assistant utility for image processing, and NI LabVIEW graphical programming environment for implementing the overall system. The storage bin containing the parts that are to be assembled and the assembly jig on which the assembly is to be built up are both placed within the field of view of the overhead camera and within the robot work envelope.

Recognition of Parts, and Estimation of Position and Orientation by Machine Vision

To carry out assembly using the robot, it is necessary to first of all recognize all the parts involved from the storage bin and determine their initial locations and orientations, which are accomplished by the machine vision system. The vision system

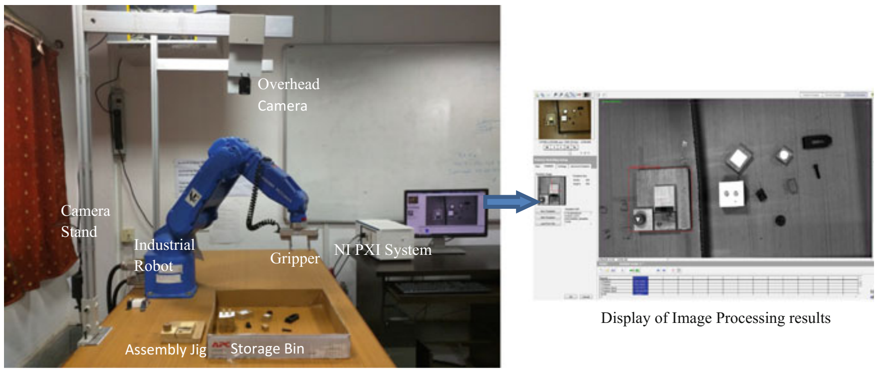


Fig. 17 Setup for vision-guided robotic system for assembly

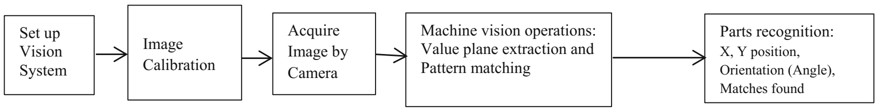

Set up Vision System
Image Calibration
Acquire Image by Camera
Machine vision operations: Value plane extraction and Pattern matching
Parts recognition: X, Y position, Orientation (Angle), Matches found

Fig. 18 Flowchart of operations performed by machine vision system for part recognition and determining position and orientation

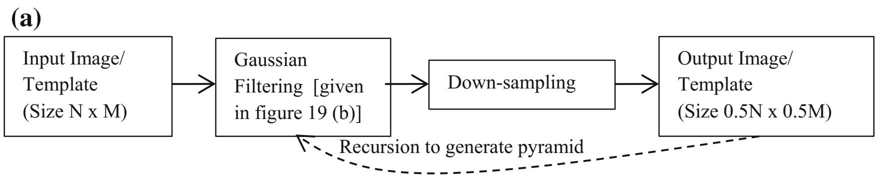
is used to further determine the target hole (cavity) locations and their orientations on the base part of the assembly or sub-assembly into which each part has to be inserted. The vision assistant utility of LabVIEW has been used to accomplish these tasks. The vision system hardware consists of an overhead camera used to acquire the image of the parts placed inside the robot work envelope and a PC with data acquisition board. The camera used is Basler acA1300-22gc GigE camera with Sony ICX445 CCD sensor of 1.3 MP resolution, 22 fps and with Edmund Optics lens of 6 mm focal length.

Figure 18 shows the operations performed by the machine vision system.

Following are the steps for recognition of parts and determining their locations and orientations. First, the camera is to be calibrated to map the pixels of the image to real world coordinates of the robot. The calibration is done by the point coordinates calibration routine of LabVIEW by feeding the real world coordinates of any four different pixels from an image taken by the camera in the given workspace. After the calibration, the setup is kept fixed and images of the workspace, containing the parts to be assembled and the assembly jig, are captured using the camera for use in robotic manipulation. The captured images are colour images containing the (R)ed, (G)reen and (B)lue channels at every pixel, and hence, every pixel can be represented in the RGB space. As changes in the brightness of an image are more distinguishable than that only in colour (Wang et al. 2001), a perceived brightness measure from the image simply called the value has been considered. This is obtained by converting a pixel in an RGB space to that in HSV (Hue–Saturation–Value) space, where H and S are the chromatic components and V is the value. V is simply computed as (Solomon and Breckon 2011)

$$V = \max\left(\frac{R}{255}, \frac{G}{255}, \frac{B}{255}\right), \quad R, G, B \in [0, 1, 2, \ldots, 254, 255] \tag{24}$$

In the next step, a pattern matching approach is applied on the value plane of the image, to detect the presence of a particular part. This system matches available templates of parts to the various regions of the image to identify a part in it. The well-known normalized 2D cross-correlation value (Haralick and Shapiro 1992) between templates and similar-sized local regions in the captured image is maximized (best fit above a threshold) to find a part. To make the matching rotation invariant, finite number of rotations (between 0° and 360°) of a given template is considered. Scale invariance of the pattern matching is also important and hence multi-scale processing is performed during the pattern matching. Multi-level


(a)
Input Image/ Template (Size N x M)
Gaussian Filtering [given in figure 19 (b)]
Down-sampling
Output Image/ Template (Size 0.5N x 0.5M)
Recursion to generate pyramid

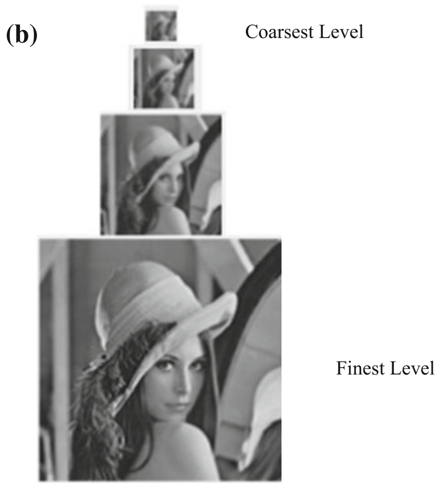


Fig. 19 **a** Multi-level Gaussian pyramid decomposition, **b** the pyramid structure. *Source* The USC-SIPI image database (http://sipi.usc.edu/database)

Gaussian pyramid decomposition (Gonzalez and Woods 2008), shown in Fig. 19, of both the template and the image at hand, is considered.

The matching is done across all the scales provided by the pyramid decomposition of both the template and the image. The number of levels in the pyramid has been considered as 4. The matching operation simply considers the real values from the value planes of the images (and templates) and works fine even in the presence of intricate textures with dense edges. It is found that the NI vision assistant utility's pattern matching described above is able to accurately locate objects that vary in size and orientation (0°–360°). To reduce complexity of the processing, a coarse-to-fine matching is considered. This helps immensely as all possible rotations and locations are considered only at the coarsest level, and then for the finer level, only a subset of possible rotations and locations are considered based on the best match orientation in the coarsest level. This avoids exhaustive search at every level of the pyramid, and actually makes the processing faster than that without pyramid decomposition, and this is another advantage of using the decomposition other than the intended scale invariance through multi-scale processing.

Once the pattern matching operation is completed, we have the estimated locations and orientations of all the parts identified. The *X*, *Y* position is then easily mapped to the world coordinates as per the initial calibration.

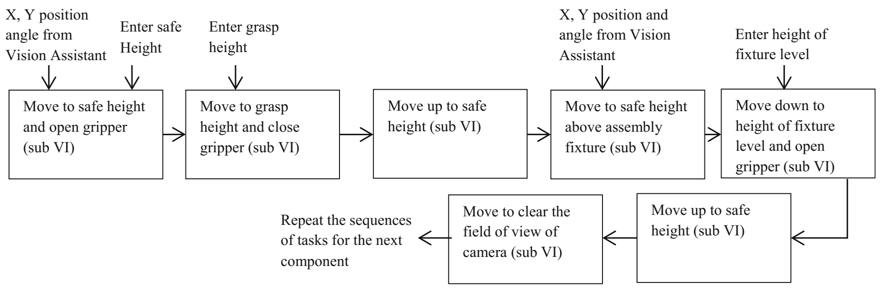

X, Y position angle from Vision Assistant
Enter safe Height
Enter grasp height
X, Y position and angle from Vision Assistant
Enter height of fixture level
Move to safe height and open gripper (sub VI)
Move to grasp height and close gripper (sub VI)
Move up to safe height (sub VI)
Move to safe height above assembly fixture (sub VI)
Move down to height of fixture level and open gripper (sub VI)
Move up to safe height (sub VI)
Move to clear the field of view of camera (sub VI)
Repeat the sequences of tasks for the next component

Fig. 20 Flowchart of the strategy for robotic assembly

Strategy for Robotic Assembly Under Machine Vision Guidance

After the assembly parts are recognized and their positions and orientations determined by machine vision system, the decisions regarding motion of the robot manipulator to perform the assembly operations are implemented in accordance with the assembly task plan generated earlier by the robot task-level planner. The robot task-level plan lists sequentially the handling tasks that are to be performed by the robot to pick up the parts one by one from their initial locations and orientations in the storage bin, and also the insertion tasks that are to be performed by the robot to assemble them into the respective positions and orientations of the holes (cavities) in the base part on the assembly jig. The destination coordinates on the assembly jig are also obtained from the vision system. The task-level plan also contains details of the robot motion sequences necessary for handling each part in a proper order as shown in Fig. 20, starting from moving the robot to a safe height above the recognized part before picking it up and ending with actuating the opening of the gripper fingers to release the part on the base part before moving up to a safe height. After these motion sequences, the manipulator needs to move away to clear the field of view of the camera and await further instructions for handling and insertion of the next part. For this, machine vision system is employed repeatedly as explained earlier. Each of the above-mentioned sequences has been implemented by developing a sub-VI in LabVIEW. If any part to be assembled is missing from the bin, the robot will automatically stop the assembly process after getting a “No match found” signal from the vision system.

Strategy for Assembly Inspection by Machine Vision

In any assembly process, inspection plays a vital role as there is always a possibility of erroneous robotic assembly due to missing parts. This defect may result from accidental slipping of a part from the gripper during handling before the robot reaches the assembly jig. In that case, the missing part from the assembly can be detected by first taking an image of the intermediate stage of assembly with the sub-assembled parts, each time after the robot completes an assembly operation,

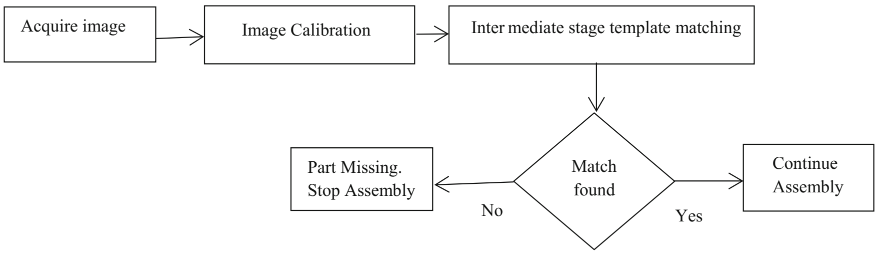
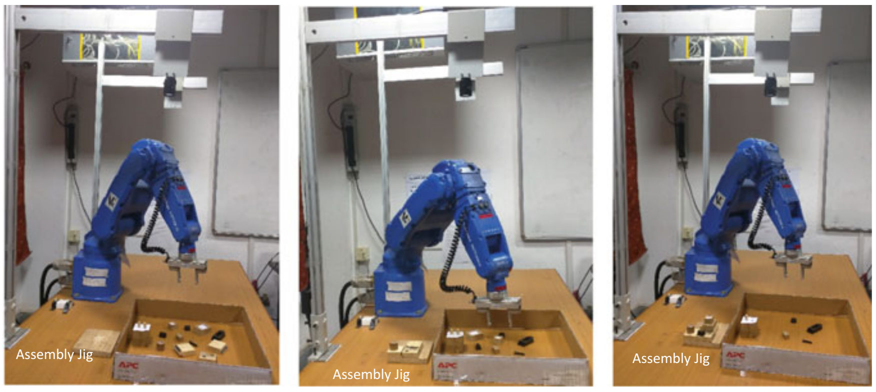

Acquire image
Image Calibration
Inter mediate stage template matching
Match found
No
Yes
Part Missing. Stop Assembly
Continue Assembly

Fig. 21 Flowchart of operations performed by machine vision for assembly inspection

and then performing template matching of that sub-assembly image with the previously stored correct intermediate stage image. If the match is found to be perfect, a signal ‘Continue assembly’ is sent to the robot controller to proceed for picking the next part. Otherwise, robot stops after receiving a ‘Part missing, stop assembly’ signal. This same process is continued at each stage of the assembly to check for missing parts. Flowchart of the inspection process is shown in Fig. 21.

3.4.3 Results and Discussions

The eight-component assembly shown in Fig. 7 is used to demonstrate the results of the developed vision-assisted robotic assembly system. Figure 23a–c, shows the initial, intermediate and the final states of the task environment, respectively, indicating the positions of the jig as well as other parts and the robot’s movement for performing the assembly. All the above parts were initially scattered randomly in the task environment of the robot but placed in a bin within the field of view of the overhead camera as shown by the initial state of the task environment in Fig. 22a. Table 4 summarizes the information about the estimated initial positions

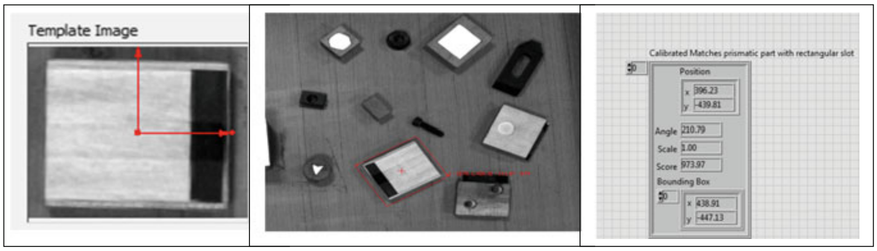


(a) Initial state (b) Intermediate state (c) Final state

Fig. 22 Initial, intermediate, final states of the task environment showing the robot’s movement for performing assembly

Table 4 Summary of information of estimated initial positions

S. No.	Assembly part	Real world coordinates (mm)		Angle in deg.	Score (out of 1000)	No. of matches	Destination coordinate on assembly fixture (*X*, *Y*, in mm)
		X	*Y*				
1	Prismatic part with one hole	455.876	−453.262	131.384	970.814	1	(82.967, −487.149)
2	Stepped prismatic part with two holes	366.85	−488.54	87.52	900.36	1	(136.215, −493.232)
3	Prismatic part with rectangular slot	396.238	−438.93	210.591	973.521	1	(136.914, −443.458)
4	Cylindrical part	311.134	−431.381	88.0231	787.212	1	(71.8167, −488.361)
5	Stepped prismatic part with one hole	367.907	−393.178	48.0247	916.051	1	(110.927, −496.193)
6	Cylindrical pin	229.033	−493.784	11.1204	947.648	1	(368.281, −388.704)
7	Prismatic part with rectangular pocket	398.583	−319.615	230.692	953.062	1	(125.933, −444.452)
8	Cube with pocket	303.292	−363.209	312.238	956.559	1	(121.456, −442.341)



```
Calibrated Matches prismatic part with rectangular slot
0
Position
x 396.23
y -439.81
Angle 210.79
Scale 1.00
Score 973.97
Bounding Box
0
x 438.91
y -447.13
```

(a) Stored template (b) Screenshot display of Image Processing result (c) Calibrated matches

Fig. 23 Output from the vision assistant utility of LabVIEW

of all assembly parts including the holes (cavities) on the jig that were extracted using the vision assistant utility of LabVIEW. Figure 23 presents an extract of a typical output from the vision assistant for one of the assembly parts, showing the stored template, recognized part, estimated initial *X*, *Y* positions, angle, score, etc. Finally, Fig. 23c shows the final state of the task environment after successful completion of the assembly by the robot.

4 Summary and Conclusions

In an attempt to support the ongoing efforts for developing flexible robotic assembly system with necessary agility and adaptability to satisfy diverse customer requirements and capability of mass customization, this chapter has examined some of the crucial issues and challenges that need to be addressed for its successful implementation. Some of the significant contributions of the reported research are as follows.

- A Sexual Genetic Algorithm (SGA)-based approach for assembly sequence planning and optimization has been developed and implemented. The information on assembly directions of the parts, precedence relationships, various tools/grippers needed and location of the base component in the assembly is inputted. It is capable of automatically generating the feasible and optimal sequences based on minimizing the number of reorientations and tool changes, and maximizing the stability of the components/sub-assemblies.
- Further, a robot task-level planning system has been proposed and its implementation shown using a vision-guided industrial robot manipulator for performing mechanical assemblies. To accomplish task-level planning, a knowledge-based system is developed and executed using the expert system shell CLIPS, which is capable of automatically generating all the assembly tasks that are to be performed by the robot from a given optimal assembly sequence generated by the above sequence planner.

- Moreover, a multi-finger, tendon-driven robotic gripper for flexible assembly has been proposed that has necessary flexibility like that of a human hand to adapt to different geometric shapes of objects and an impedance control algorithm with necessary compliance needed to perform parts mating and insertion during assembly.
- Finally, a strategy for robotic assembly under machine vision guidance has been also developed, in which the vision system comprising an overhead-mounted CCD camera and NI vision assistant utility for image processing has been used to guide a Motoman industrial robot in performing mechanical assembly operations in a task environment, where the parts are initially scattered randomly. Further, a strategy for automated assembly inspection based on machine vision has also been proposed to rule out any possibility of erroneous robotic assembly. The above system is found to be successfully working with minimal human intervention. Research work is presently ongoing for use of multiple sensors in conjunction with machine vision to guide the robot in performing more complex assembly tasks.

Acknowledgements The authors gratefully acknowledge the funding support received from Indian Institute of Technology Kharagpur under the Centre for Excellence in Robotics grants.

References

Abdallah, M.E., R. Platt, B. Hargrave, and F. Permenter. 2012. Position control of tendon-driven fingers with position controlled actuators. In *IEEE international conference on robotics and automation*, RiverCentre, Saint Paul, Minnesota, USA, 14–18 May 2012.

Alatartsev, S., S. Stellmacher, and F. Ortmeier. 2015. Robotic task sequencing problem: A survey. *Journal of Intelligent and Robotic Systems: Theory and Applications* 80 (2): 279–298.

Ambrose, R.O., H. Aldridge, R.S. Askew, R.R. Burridge, W. Bluethmann, M. Diftler, C. Lovchik, D. Magruder, and F. Rehnmark. 2000. Robonaut: NASA's space humanoid. *IEEE Intelligent System*, 57–63, July/Aug 2000.

Arai, H., and S. Tachi. 1991. Position control of a manipulator with passive joints using dynamic coupling. *IEEE Transactions on Robotics and Automation* 7 (4), Aug 1991.

Backhaus, J., and G. Reinhart. 2013. Efficient application of Task oriented programming for assembly systems. In *IEEE/ASME international conference on advanced intelligent mechatronics (AIM)*, Australia, 750–755.

Bekey, G.A., R. Tomovic, and I. Zeljkovic. 1990. *Control architecture for the Belgrade/USC hand*. New York: Springer.

Bonneville, F., C. Perrard, and J.M. Henrioud. 1995. A genetic algorithm to generate and evaluate assembly plans. In *Proceedings of INRIA/IEEE symposium on emerging technologies and factory automation*, 231–239, Paris, France.

Butterfass, J., G. Hirzinger, S. Knoch, and H. Liu. 1998. DLR's multisensory articulated part I: Hard- and software architecture. *IEEE International Conference on Robotics and Automation*.

Butterfass, J., M. Grebenstein, H. Liu, and G. Hirzinger. 2001. DLR-hand II: Next generation of a dextrous robot hand. In *IEEE international conference on robotics and automation*, vol. 1, 109–114.

Cambon, S., F. Gravot, and R. Alami. 2004. A robot task planner that merges symbolic and geometric reasoning. In *European conference on artificial intelligence (ECAI)*, 1–5, Apr 2004.

Cao, P.B., and R.B. Xiao. 2007. Assembly planning using a novel immune approach. *International Journal of Advanced Manufacturing Technology* 31 (7): 770–782.

Chen, S., and Y.J. Liu. 2001. An adaptive genetic assembly sequence planner. *International Journal of Computer Integrated Manufacturing* 14 (5): 489–500.

Chen, R.S., K.Y. Lu, and P.H. Tai. 2004. Optimizing assembly planning through a three-stage integrated approach. *International Journal of Production Economics* 88 (3): 243–256.

Chen, H., B. Zhang, and G. Zhang. 2015. Robotic assembly. In *Handbook of manufacturing engineering and technology*, 2347–2400.

Cho, J., H. Kim, and J. Sohn. 2010. Implementing automated robot task planning and execution based on description logic KB. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6472, 217–228.

Choi, Y.K., D.M. Lee, and Y.B. Cho. 2009. An approach to multi-criteria assembly sequence planning using genetic algorithms. *International Journal of Advanced Manufacturing Technology* 42: 180–188.

Deb, S.R., and S. Deb. 2010. *Robotics technology and flexible automation*. New Delhi: Tata McGraw Hill.

Diftler, M.A., J.S. Mehling, M.E. Abdallah, N.A. Radford, L.B. Bridgwater, A.M. Sanders, R.S. Askew, D.M. Linn, J.D. Yamokoski, F.A. Permenter, B.K. Hargrave, and R. Platt. 2011. Robonaut 2—the first humanoid robot in space. In *2011 IEEE international conference on robotics and automation*, Shanghai, China, 9–13 May 2011.

Gao, X.H., M.H. Jin, L. Jiang, Z.W. Xie, P. He, L. Yang, Y.W. Liu, R. Wei, H.G. Cai, H. Liua, J. Butterfass, M. Grebenstein, N. Seitza, and G. Hirzinger. 2003. The HIT/DLR dexterous hand: Work in progress. In *IEEE international conference on robotics & automation* Taipei, Taiwan, 14–19 Sept 2003.

Gao, L., W.R. Qian, X.Y. Li, and J.F. Wang. 2010. Application of memetic algorithm in assembly sequence planning. *International Journal of Advanced Manufacturing Technology* 49 (9–12): 1175–1184.

Gazeau, J.P., S. Zeghloul, M. Arsicualt, J.P. Lallemand. 2001. The LMS hand: Force and position controls in the aim of fine manipulation of objects. In *IEEE International conference on robotics and automation*, 2642–2648.

Giarratano, J., and G. Riley. 2002. *Expert systems principles and programming*, 3rd ed. Thomson Learning and China Machine Press. ISBN 7-111-10844-2.

Goh, K.S., A. Lim, and B. Rodrigues. 2003. Sexual selection for genetic algorithms. *Artificial Intelligence Review* 19: 123–152.

Golnabi, H., and A. Asdapour. 2007. Design and application of industrial machine vision systems. *Robotics and Computer-Integrated Manufacturing* 23: 630–637.

Gonzalez, R.C., and R.E. Woods. 2008. *Digital image processing*, 3rd ed. Pearson.

Haralick, R.M., and L.G. Shapiro. 1992. *Computer and robot vision*, vol. II, Addison-Wesley.

Jacobsen, S.C., J.E. Wood, D.F. Knutti, and K.B. Biggers. 1984. The UTAH/M.I.T. dextrous hand: Work in progress. *The International Journal of Robotics Research*.

Jacobsen, S., E. Iversen, D. Knutti, R. Johnson, and K. Biggers. 1986. Design of the Utah/M.I.T. dextrous hand. In *1986 IEEE international conference on robotics and automation*, vol. 3, 1520–1532, Apr 1986.

Karthik, G.V.S.K., and S. Deb. 2017. A methodology for assembly sequence optimization by hybrid Cuckoo-Search Genetic Algorithm. *Journal of Advanced Manufacturing Systems* (in press).

Kobari, Y., T. Nammoto, J. Kinugawa, and K. Kosuge. 2013. Vision based compliant motion control for part assembly. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Japan, 293–298, Nov 2013.

LiCheng, W., G. Carbone, and M. Ceccarelli. 2009. Designing an underactuated mechanism for a 1 active DOF finger operation. *Mechanism and Machine Theory* 44: 336–348.

Li, X., K. Qin, B. Zeng, L. Gao, and J. Su. 2016. Assembly sequence planning based on an improved harmony search algorithm. *International Journal of Advanced Manufacturing Technology* 84: 2367–2380.

Liu, Hong, P. Meusel, G. Hirzinger, Minghe Jin, Yiwei Liu, and Zongwu Xie. 2008. The modular multisensory DLR-HIT-hand: Hardware and software architecture. *IEEE/ASME Transactions on Mechatronics* 13 (4).

Loucks, C., V. Johnson, P. Boissiere, G. Starr, and J. Steele. 1987. Modeling and control of the Stanford/JPL hand. In *International conference on robotics and automation*, vol. 4, 573–578, Mar 1987.

Lovchik, C.S., and M.A. Diftler. 1999. The robonaut hand: A dexterous robot hand for space. In *IEEE international conference on robotics and automation*, 907–912.

Lv, H.G., and C. Lu. 2010. An assembly sequence planning approach with a discrete particle swarm optimization algorithm. *International Journal of Advanced Manufacturing Technology* 50 (5–8): 761–770.

Marian, R.M., L.H.S. Luong, and K. Abhary. 2006. A genetic algorithm for the optimisation of assembly sequences. *Computers & Industrial Engineering* 50 (4): 503–527.

Mishra, A., and S. Deb. 2016a. An intelligent methodology for assembly tools selection and assembly sequence optimisation. In *Proceedings of CAD/CAM, robotics and factories of the Future*, India, 323–333, Jan 2016.

Mishra, A., and S. Deb. 2016b. A GA based parameter meta-optimization of ACO algorithm for solving assembly sequence optimization. In *Proceedings of international conference on computers & industrial engineering CIE 46*, Tianjin, China, 29–31 Oct 2016.

Mishra, A., and S. Deb. 2016c. Assembly sequence optimization using a Flower Pollination Algorithm-based approach. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-016-1261-7.

Mishra, A., and S. Deb. 2017. Robotic assembly sequence planning and optimization by cuckoo search algorithm. In *Proceedings of international conference on production research 24th ICPR*, Poznan, Poland, July 30–Aug 3 2017.

Okada, T. 1986. Computer control of multi-jointed finger system for precise object handling. *International Trends in Manufacturing Technology Robot Grippers*, 391–417.

Osuna, R.V., T. Tallinen, J.L.M. Lastra, and R. Tuokko. 2003. Assembly and task planning in a collaborative web-based environment based on assembly process modeling methodology. In *Proceedings of IEEE international symposium on assembly and task planning*, 79–84, Jan 2003.

Ozawa, R., K. Hashirii, Y. Yoshimura, M. Moriya, and H. Kobayashi. 2014. Design and control of a three-fingered tendon-driven robotic hand with active and passive tendons. *Autonomous Robots* 36: 67–78.

Pal, S., S. Chattopadhyay, and S.R. Deb. 2008. Design and development of a multi-degree of freedom dexterous instrumented Robot Gripper. *Sensors and Transducers Journal* 87 (1): 63–73.

Pena-Cabrera, M. 2005. Machine vision approach for robotic assembly. *Assembly Automation* 25 (3): 204–217.

Ramos, C., and E. Oliveira. 1992. Planning, execution and sensor-based reaction for assembly robotic tasks, Technical paper, Faculdade de Engenharia da Universidade do Porto, Portugal.

Sainul, I.A., S. Deb, and A.K. Deb. 2016. A three finger tendon driven robotic hand design and its kinematics model. In *International conference on CADCAM, robotics and factories of the future*, Kolaghat, India, Jan 2016.

Salisbury, K.S., and B. Roth. 1983. Kinematics and force analysis of articulated mechanical hands. *Journal of Mechanisms, Transmissions and Actuation in Design* 105: 35–41.

Solomon, C., and T. Breckon. 2011. *Fundamentals of digital image processing*. Wiley & Blackwell.

Tlegenov, Y., K. Telegenov, and A. Shintemirov. 2014. An open-source 3D printed underactuated robotic gripper. In *IEEE/ASME international conference on mechatronic and embedded systems and applications*, Senigallia, 10–12 Sept 2014.

Townsend, W.T. 2000. MCB—industrial robot feature article-barrett hand grasper. *Industrial Robot: An International Journal* 27 (3): 181–188.

Thomas, U., and F.M. Wahl. 2010. Assembly planning and task planning—two prerequisites for automated robot programming. *Robotic Systems for Handling and Assembly, STAR* 67: 333–354.

Ulrich, N., R. Paul, and R. Bajcsy. 1988. A medium-complexity complaint end effector. In: *IEEE international conference on robotics and automation*.

Wang, Y., J. Ostermann, and Y.Q. Zhang. 2001. *Video processing and communications*. Prentice Hall.

Wang, J.F., J.H. Liu, and Y.F. Zhong. 2005. A novel ant colony algorithm for assembly sequence planning. *International Journal of Advanced Manufacturing Technology* 25: 1137–1143.

Xing, Y.F., and Y.S. Wang. 2012. Assembly sequence planning based on a hybrid particle swarm optimisation and genetic algorithm. *International Journal of Production Research* 50 (24): 7303–7312.

Zhang, B., J. Wang, G. Rossano, C. Martinez, and S. Kock. 2011. Vision-guided robot alignment for scalable, flexible assembly automation. In *IEEE international conference on robotics and biomimetics*, Thailand, 944–951, Dec 2011.

Zhou, W., J.R. Zheng, J.J. Yan, and J.F. Wang. 2011. A novel hybrid algorithm for assembly sequence planning combining bacterial chemotaxis with genetic algorithm. *International Journal of Advanced Manufacturing Technology* 52 (5–8): 715–724.