# Multi-objective grasp pose optimisation for robotic 3D pipe assembly manipulation

Zhang, Zebang; Saadat, Mozafar

*DOI:*
10.1016/j.rcim.2022.102326



*Document Version*
Publisher's PDF, also known as Version of record

*Citation for published version (Harvard):*
Zhang, Z & Saadat, M 2022, 'Multi-objective grasp pose optimisation for robotic 3D pipe assembly manipulation', *Robotics and Computer-Integrated Manufacturing*, vol. 76, 102326. https://doi.org/10.1016/j.rcim.2022.102326

Robotics and Computer–Integrated Manufacturing 76 (2022) 102326

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

# Multi-objective grasp pose optimisation for robotic 3D pipe assembly manipulation

Zebang Zhang *, Mozafar Saadat

*Department of Mechanical Engineering, School of Engineering, University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom*

ARTICLE INFO

*Keywords:*
3D pipe assembly
Multi-objective optimization
Grasp pose optimization
Robotic manipulation
Deformable object
Bees Algorithm

ABSTRACT

This paper considers the problem of grasp pose optimisation for manipulating 3D pipe assemblies during the manufacturing process. The method presented in this paper is specifically developed for manufacturing cryogenic pipe assemblies autonomously in a Factory-In-A-Box scenario (i.e., a compact factory built inside an industrial container). However, it also can be used for robotic manipulation of general frame structures. The problem is formulated as a constrained multi-objective optimisation problem. The optimisation algorithm searches for solutions that satisfy two constraints: (i) robot workspace reachability (i.e., feasible inverse kinematics solution for a given end-effector pose); and (ii) any possible collision when executing the post-grasp trajectory, and continuously improves them based on three objectives: (i) minimise robot joint motion for executing a specified pipe trajectory; (ii) minimise the sum of maximum deformation of pipe assembly along the trajectory; and (iii) minimise the sum of robot force manipulability along the trajectory. A special constraint handling method is used to decouple the constraint and objective evaluation process, allowing expensive objectives to be evaluated only when constraints are satisfied. The algorithm explicitly considers the possibility of multiple inverse kinematics solutions and uses a graph search algorithm (Dijkstra's Algorithm) to find the optimal trajectory amongst all feasible trajectories. The weighted sum approach is used to combine the three objectives with weights determined by the Analytical Hierarchy Process. The optimisation problem is solved using the Bees Algorithm with a proposed problem-specific local search strategy. Extensive benchmarks show that the proposed strategy achieves better overall results than the default strategy of the Bees Algorithm and other metaheuristics.

## 1. Introduction

Robotic manipulation including grasping has been an active research topic for decades. Most research on grasp synthesis focuses on finding a successful grasp configuration so that the object is firmly attached to the end-effector. Methods for grasp synthesis includes data-driven approaches and analytic approaches [1,2]. Recent advances in data-driven grasping approaches focus on using deep learning and vision sensors to generate the grasp pose [3]. However, the grasping motion alone is only one stage of the whole manipulation process. After successfully grasping the object, the robot manipulator is expected to move the object to the goal pose and perform subsequent tasks. To perform the required movements after grasping, motion planning is a necessary step. The robot motion planning problem is defined as finding a continuous trajectory from the start configuration of the robot to the goal configuration (or goal region) while avoiding any obstacles inside the environment [4]. Sampling-based planners like Rapidly-exploring Random Trees (RRT) [5] and Probabilistic Road Map (PRM) [6] are widely used for solving high dimensional motion planning problems such as planning motion for a robot manipulator.

Although the individual problem of grasping and motion planning have been studied for decades, the literature on the combined problem is still very limited due to the complexity of the individual problem. The whole manipulation process is treated as an optimal control problem in [7] so that the locally optimal grasp contact position, grasping force and robot arm trajectory can be obtained by solving a single optimisation problem. Grasp-RRT is developed in [8] to plan collision-free grasping motion for the robot given the start configuration of the robot and the object pose. The impact of choosing different grasp poses on the post-grasp manipulation process has been analysed in [9]. In their work, the optimal trajectory of the manipulated object is assumed to be found either by human knowledge or an optimal motion planner [10,11]. Then a few grasp candidates are generated by a grasp planner. Next, for each grasp pose, an inverse kinematics (IK) solver is used to generate robot

* Corresponding author.
*E-mail addresses:* zebang22@gmail.com (Z. Zhang), m.saadat@bham.ac.uk (M. Saadat).

https://doi.org/10.1016/j.rcim.2022.102326
Received 5 March 2021; Received in revised form 4 February 2022; Accepted 5 February 2022
Available online 11 February 2022

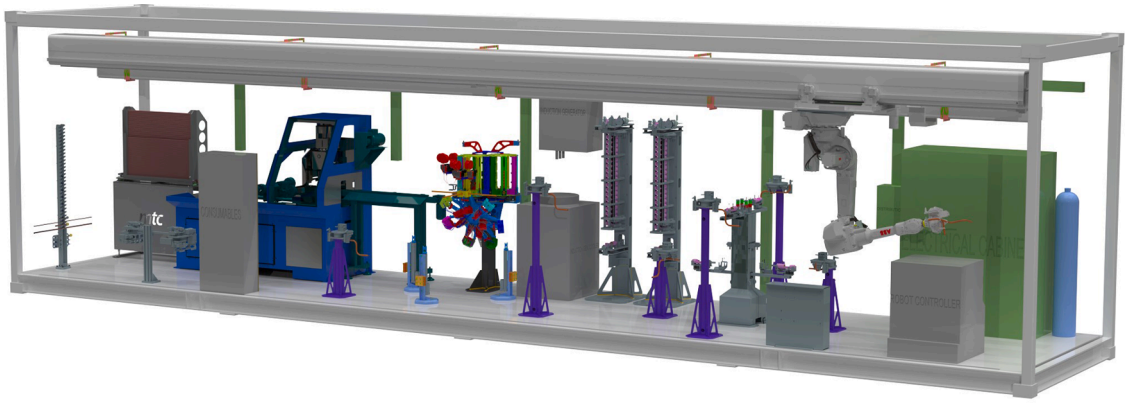

**Fig. 1.** Simulated factory-in-a-box layout Image Courtesy of Manufacturing Technology Centre (MTC), Coventry, UK.

motion and an objective function is defined to evaluate the quality of the grasp candidates. In this way, a more efficient grasp pose is found by considering the robot motion after grasping. The objective used in [9] is to maximise its distance from any collisions along the trajectory.

The work mentioned above mainly focuses on the grasp and motion planning of rigid objects, However, many objects in industrial and domestic environments are deformable. Manipulation of deformable objects has been reviewed in [12–14]. This paper specifically analyses the manipulation of 3D pipe (or in general, frame) structure. The 3D pipe is bent or assembled by joints from multiple 1D pipes. A 1D pipe can be viewed as a deformable linear object (DLO), i.e., they are much larger along one dimension than the other two. Manipulation planning strategies have been widely researched for DLOs [15–17]. In [15], A two-phase path planner together with a cable pose measurement approach is proposed for cable grasping. The method focuses on the pre-grasp stage and uses force directional manipulability to determine the optimal grasp pose. A method for automatic mating of a wire harness onto a car body by wire tracing operation is proposed in [16]. In [17], A manipulation framework is proposed to shape the cable by environmental contacts. However, common techniques used in the literature for DLO manipulation like structure reshaping and exploiting environmental contacts are not suitable for 3D pipe manipulation due to the complex structure and relatively large strain. Instead, the method presented in this paper aims at maintaining the geometry and reducing the deformation of the pipe during the manipulation process.

Researchers have investigated the problem of minimising deformation when handling compliant sheet metal parts. A trajectory optimisation approach is used in [18] to plan a minimal deformation trajectory while maintaining the same productivity. A response surface model is generated based on finite element analysis (FEA) of the handled part and end-effector, so that deformation can be quickly estimated during optimisation. Other approaches attempt to reduce the deformation of an object during handling by designing a part-specific end-effector layout. A simple methodology is proposed in [19] to determine the number of vacuum cups needed for handling a compliant sheet metal part as well as their locations. The method focuses on placing vacuum cups evenly based on the gravity distribution of the object. Although the deformation and holding force decrease significantly by using the proposed method, the deformation information is not directly used during the optimisation process, which means the positions of the end-effector could be further optimised. A methodology is proposed in [20] to optimise the design of end-effector and robot motion simultaneously in order to achieve less cycle time as well as less deformation for a multi-robot system. Although the deformation information is used directly, it also relies on a precomputed model to estimate the deformation during optimisation as in [18], which is not suitable for manipulating objects with different dimensions and geometries. To address the above limitations, the end-effector used in this paper is a general two-finger gripper so that the optimisation is performed at the actual manipulation stage rather than the design stage and no precomputed deformation model is required. Hence, the proposed method is more versatile to different pipe geometries.

The aim of this paper is to present a general method for grasp pose optimisation based on post-grasp trajectory objectives. This method is specifically developed for robot that perform repetitive 3D pipe assembly manipulation in a static environment. Therefore, once the optimal solution is computed offline, the robot can execute the trajectory repetitively with the optimal grasp to increase the production efficiency. Previous work on the similar problem like [9] only deals with a limited number of grasp candidates, thus no optimisation algorithm is used. This paper follows the same assumption in [9] that an optimal object trajectory is known beforehand. However, this paper considers deformation and other objectives that are not used in previous work and uses Analytical Hierarchy Process (AHP) to determine weights for each objective. The method also decouples the constraint and objective evaluation process, allowing expensive objectives (e.g., deformation) to be evaluated only when constraints are satisfied, thus significantly reducing the computation time. In addition, this paper presents a method effectively utilising multiple IK solutions (for the same end-effector pose) to further optimise the objective cost. A problem-specific local search strategy for optimisation is also proposed to achieve a high success rate in finding the optimal solution.

The remainder of the paper is structured as follows: Section 2 presents an industrial case study. In Section 3, the problem is introduced and then optimisation variables, objectives, and constraints are modelled. Section 4 describes the details of the objective and constraint evaluation method. Section 5 introduces the optimisation algorithm used in this paper and proposes problem-specific search strategies. Section 6 presents the experiments, results, and analyses. Section 7 concludes the paper.

## 2. Industrial case study

Deploying a flexible robotic workcell is becoming an increasingly popular choice with the trends of mass customisation [21], especially for small and mid-size enterprises. A good example of deploying such a flexible robotic workcell is the Factory-In-A-Box (FIAB) project. FIAB is a high-profile project funded by Innovate UK as part of the Thermal Energy Research Accelerator (T-ERA) carried out at the Manufacturing Technology Centre (MTC), UK, with the support of the University of Birmingham. A simulated FIAB environment is shown in Fig. 1. FIAB is a successful demonstrator of a high technology, compact and autonomous factory inside a container to manufacture cryogenic pipe assemblies with the focus on flexibility (the factory can adapt quickly to manufacture pipes of different structures at different quantities according to customer needs) and mobility (the factory can be rapidly deployed

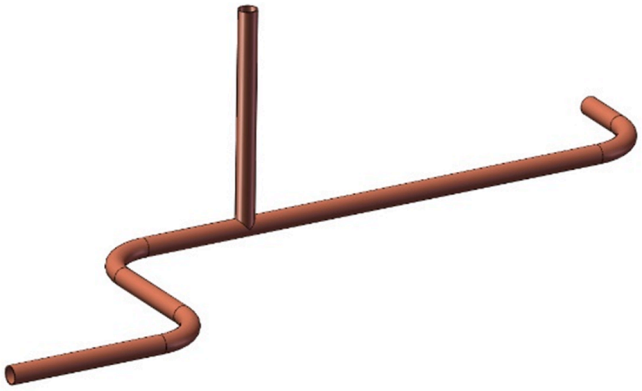

**Fig. 2.** A 3D pipe assembly.

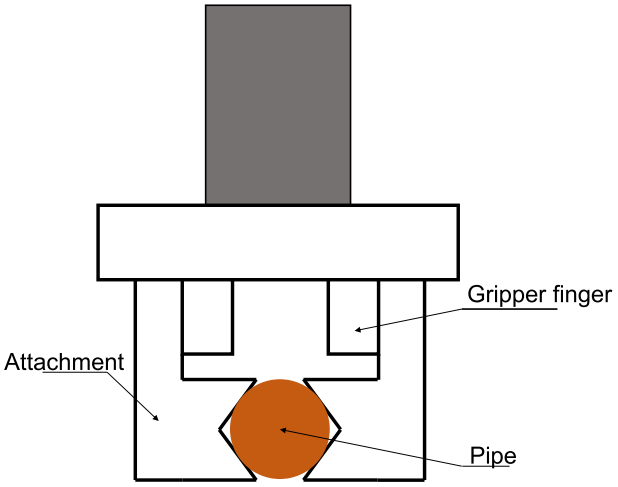


**Fig. 3.** Attachment design for securely pipe grasping.

anywhere) [22]. Cryogenic piping systems are used for applications which require extremely low temperature, typically lower than -150 °C. FIAB accepts orders from customers and performs simulations to check if the required dimensions of the pipes can be manufactured by the facility. The manufacturing process usually involves cutting, bending, brazing, and pressure testing. An industrial robot is mounted on an overhead rail to grasp and transfer pipes between different stations inside FIAB. Currently, the solution for grasping pipe assemblies is by designing a part-specific end-effector. While this solution provides a robust grasp and reduces excessive deformation during manipulation, it also increases the weight of the end-effector which makes the process less energy efficient. Besides, as the custom-designed end-effector is larger and more complex in terms of its geometry, the possibility of collision with the end-effector increases. Therefore, it may be infeasible for some pipes to be manufactured. More importantly, the custom-designed gripper is much more expensive than a general gripper.

Therefore, the method proposed in this paper is based on using a general two-finger gripper. As shown in Fig. 2, the pipe assembly consists of several sections. The simple cylindrical shape of each section makes it possible for the assembly to be grasped by a low-cost parallel gripper robustly with a properly designed attachment (as shown in Fig. 3). Although the grasping motion alone is easy, other issues may arise when the robot motion after grasping is considered. For example, some grasp poses may lead to collisions between the robot (or grasped object) and the environment. This is very common inside FIAB since the space is relatively compact. Other grasp poses may lead to redundant motion of the robot, proximity to robot singularity and excessive deformation of the pipe assemblies. Therefore, an optimisation process needs to be implemented to find a suitable and possibly optimal grasp pose for the pipe assembly.

## 3. Problem formulation

### 3.1. Problem description

The simulated environment is shown in Fig. 4. The pipe is placed on a fixture for the robot to grasp. The initial pose of the pipe is denoted as $\boldsymbol{A}_{p,0}$. $\boldsymbol{A}_{p,0}$ is a homogeneous transformation matrix with respect to world

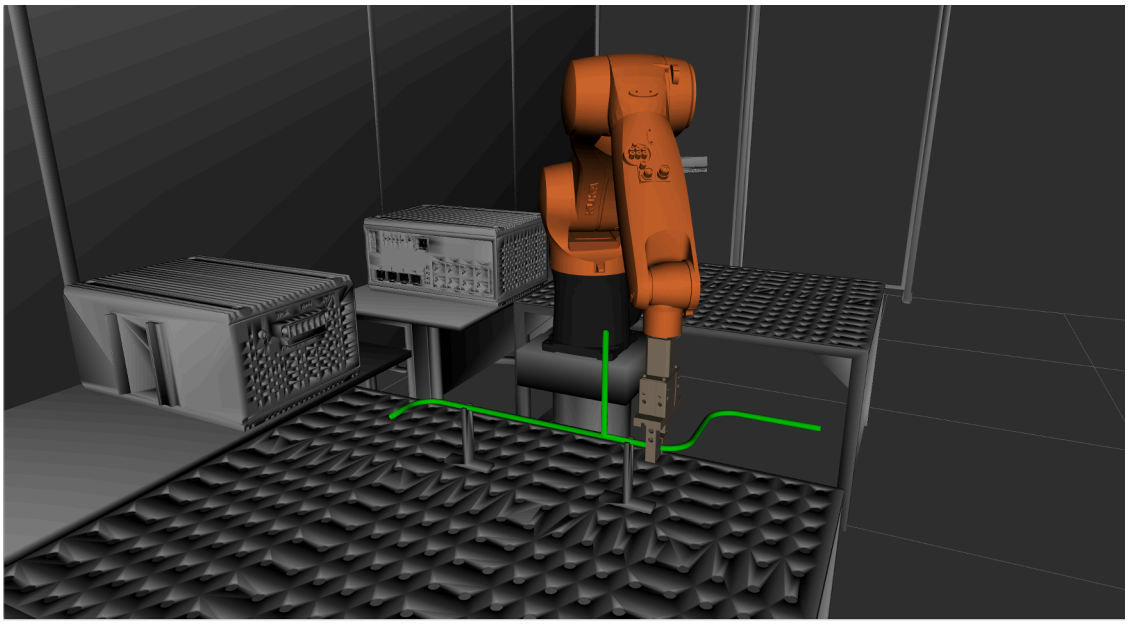

**Fig. 4.** Robot picks up a pipe assembly and transfer it to desired pose.

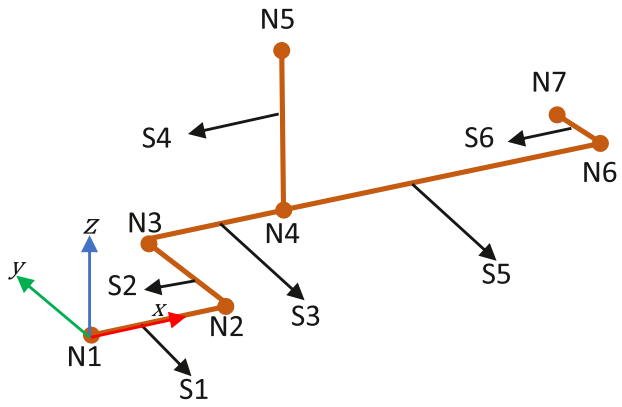


**Fig. 5.** Abstracted pipe assembly in its local frame.

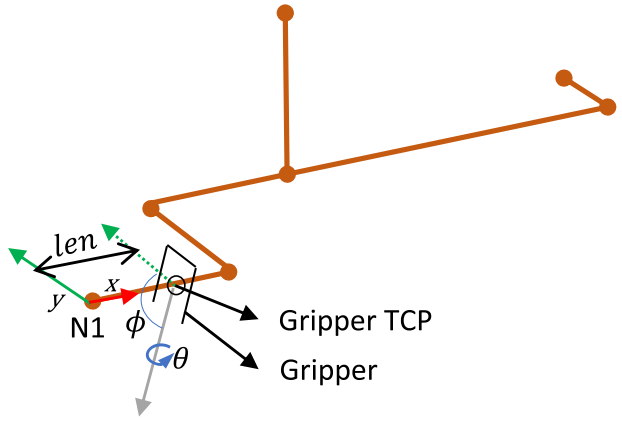


**Fig. 6.** Variables that define a grasp.

frame. After successfully grasping the pipe, the robot will transfer the pipe to a desired pose $\boldsymbol{A}_{p,n}$. The trajectory of the pipe is denoted by a series of waypoints: $\boldsymbol{A}_{p,1}, \boldsymbol{A}_{p,2}, \ldots, \boldsymbol{A}_{p,n}$.

As mentioned above, the end-effector can grasp the pipe at almost any position with a properly designed attachment. However, not every pose is reachable for the robot to perform grasping, or even if the grasp pose is reachable, the predefined pipe trajectory cannot be followed exactly. In order to achieve fast and efficient production as well as maintain the quality of the pipe, there are several issues that need to be considered when the robot chooses a pose to grasp the pipe:

- the trajectory of the pipe can be followed exactly.
- there is no collision between the robot (with the grasped pipe) and the environment or self-collision.
- minimise the robot joint motion distance.
- minimise the deformation of pipe along the trajectory.
- minimise force manipulability of robot along the trajectory.

The first two issues are modelled as constraints and the latter three are modelled as objectives in Section 3.3.

### 3.2. Abstracted pipe geometry

The pipe assembly is abstracted to simplify the process of generating grasp poses and computing the deformation after being grasped by the robot. The abstraction of the 3D pipe assembly in Fig. 2 is shown in Fig. 5. The assembly consists of several sections $(S_1, S_2, S_3, \ldots)$. Each section is defined by two nodes. Each node stores the 3D position with respect to its local coordinate system. Each section stores properties like diameter, wall thickness and material density. In this way, theoretically, the pipe can consist of sections with different materials (PVC and copper) and diameters. The geometry of the pipe assembly in Fig. 5 can be defined by a 6 by 2 matrix $\begin{bmatrix} 1 & 2 & 3 & 4 & 4 & 6 \\ 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix}^T$, where each entry of the matrix is the index of the node and each row defines a section.

### 3.3. Mathematical modelling

In this section, the optimisation variables, constraints, and objectives of the problem are modelled based on the issues discussed in Section 3.1.

#### 3.3.1. Optimisation variables

The problem is aimed at finding the optimal grasp pose. A grasp pose is essentially a rigid body transformation and can be represented by a homogeneous transformation matrix. In our problem, the grasp pose must satisfy certain constraints to enable feasible grasping. Specifically, the position vector must be on the centreline of the pipe and the orientation vector has to be perpendicular to the pipe centreline. Given the geometry of the pipe assembly, a grasp pose can then be defined with 4 variables as shown in Fig. 5 and Fig. 6. The definitions of the variables are given as follows:

- grasp section ($S$): this variable specifies which section of the pipe the robot will grasp.
- grasp position ($len$): this is the length between the starting node of a section (e.g., the starting node of S1 is N1) and the grasp location (where gripper TCP is placed).
- grasp angle ($\phi$): this is the angle between the y-axis and the gripper approaching direction (grey arrow).
- flip angle ($\theta$): this angle is around the gripper approaching direction (grey arrow) and can only be either 0 or 180 due to geometric constraint of pipe and the design of parallel gripper.

Given the geometry of pipe P, grasp $g = \{S, len, \phi, \theta\}$ can then be generated. Knowing the grasp parameter, it becomes convenient to calculate the end-effector grasp pose $\boldsymbol{B}_e^P$ in the matrix form with respect to the pipe local coordinate system.

#### 3.3.2. Optimisation constraints

*3.3.2.1. Reachability constraint (C1).* The first constraint that needs to be considered is the reachability of the robot. All the end-effector poses for completing the trajectory must be inside the workspace of the robot. Given the trajectory of the pipe and the generated local grasp pose $\boldsymbol{B}_e^P$, the robot end-effector pose in the world frame can be computed as follows:

$$\boldsymbol{T}_{e,i} = \boldsymbol{A}_{p,i}\boldsymbol{B}_e^P(g) \tag{1}$$

Therefore, the robot joint trajectory can be computed by solving the inverse kinematics problem:

$$q_i = \mathrm{IK}\left(T_{e,i}\right) \tag{2}$$

$\boldsymbol{q}_i$ is a vector that describes the robot configuration. The dimension of $\boldsymbol{q}_i$ is the degrees of freedom of the robot. If for i=0..n, , Eq. (2) is solvable and the solution satisfy the robot joint range constraint, then this grasp pose satisfies the reachability constraint.

*3.3.2.2. Collision constraint (C2).* The second constraint that needs to be satisfied is that the whole manipulation process must be collision-free. To satisfy this constraint, collision checking is performed for all the robot configurations $(\boldsymbol{q}_0, \boldsymbol{q}_1, \ldots, \boldsymbol{q}_n)$ and intermediate states during the manipulation process. When performing collision checking, a safe distance is implemented to ensure that no collision happens in case of any uncertainties e.g., geometric modelling errors, robot motion inaccuracy

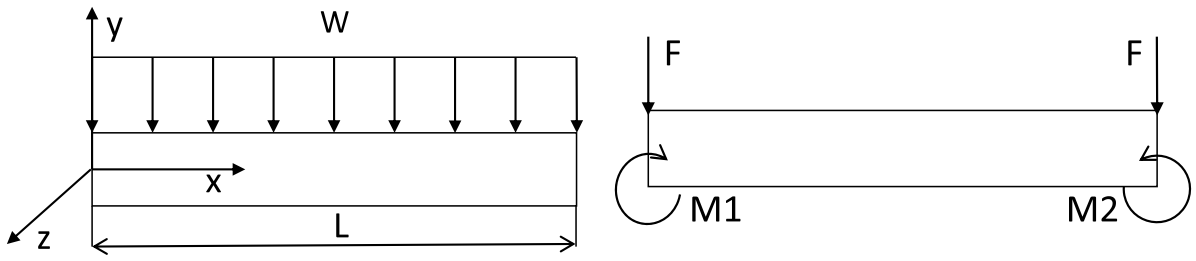


**Fig. 7.** (a) Loading condition of pipe element under gravity (b) Equivalent loading condition.

or part deformation. The collision checking is performed by an open source library FCL [23] which supports both collision detection and distance queries.

### 3.3.3. Optimisation objectives

*3.3.3.1. Joint motion (O1).* Objective O1 is defined as the sum of squared displacements between two consecutive waypoints to encourage minimum robot joint motion. Since the robot configurations required to follow the given pipe trajectory have been calculated already using IK solver in the constraint checking process, the computation of O1 is straightforward:

$$Cost(\mathrm{O1}) = \sum_{i=1}^{n} \| \boldsymbol{q}_i - \boldsymbol{q}_{i-1} \|^2 \tag{3}$$

where $n$ is the number of waypoints of the trajectory as mentioned above.

*3.3.3.2. Deformation of the pipe (O2).* This objective intends to minimise the sum of maximum deformation that happens at each waypoint along the trajectory. In this way, the geometric shape of the pipe can be maintained during the manipulation process. The deformation depends on the grasp position with respect to the pipe frame as well as the orientation of the pipe with respect to the world frame. The pipe is modelled as a frame structure that consists of arbitrarily orientated beam members which are connected rigidly. The beam members support bending, shearing as well as axial loads. A custom FEA program is implemented by using the matrix method described in [24]. The key step here is to reconstruct the boundary condition when evaluating different grasp poses. After grasping, the pipe is assumed to be rigidly supported at the grasping location. The initial pipe structure is created before the optimisation process starts as in Section 3.2. Once the current grasp pose is determined, a new node is created at the grasp location and the original grasp section is broken into two sections. The matrix used to store the geometry of the pipe assembly is updated accordingly. After creating the new geometry, the stiffness matrix of the pipe can be

$$K = \begin{bmatrix}
\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} & 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} \\
 & & \frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 & 0 & 0 & -\frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 \\
 & & & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 \\
 & S & & & \frac{4EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 \\
 & & Y & & & \frac{4EI_z}{L} & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} \\
 & & & M & & & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
 & & & & M & & & \frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} \\
 & & & & & E & & & \frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 \\
 & & & & & & T & & & \frac{GJ}{L} & 0 & 0 \\
 & & & & & & & R & & & \frac{4EI_y}{L} & 0 \\
 & & & & & & & & Y & & & \frac{4EI_z}{L}
\end{bmatrix} \tag{4}$$

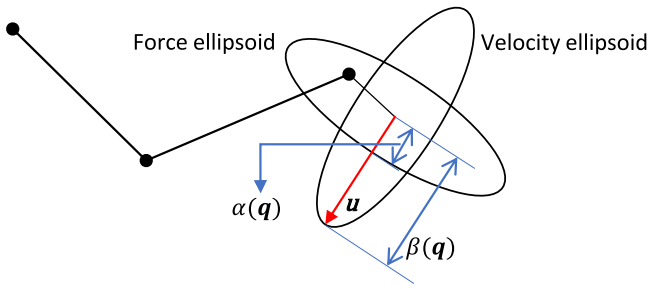


**Fig. 8.** Force and velocity ellipsoid for a 3-link planar robot.

constructed. The general stiffness matrix for a 1D pipe section in the local frame is given by Eq. (4):

where E and G are the Young's modulus and the shear modulus. $I_y$ and $I_z$ are the second moment of area. $J$ is the torsional constant (same as $I_x$ in the circular case) and $L$ is the length of the pipe section. Since each node has 6 degrees of freedom, assuming the structure has N nodes, the stiffness matrix for the whole structure is a 6N by 6N matrix. The stiffness matrix needs to be transformed from local to the global coordinate system by:

$$\boldsymbol{K}_{global} = \boldsymbol{R}^T \boldsymbol{K} \boldsymbol{R} \tag{5}$$

where $\boldsymbol{R}$ is the rotation matrix which represents how each pipe section is orientated from the global coordinate system. Details about constructing the stiffness matrix for a structure can be found in [24].

After obtaining the stiffness matrix, the forces acting on the pipe will then be determined. The weight of the pipe distributes evenly along its length as shown in Fig. 7(a). Therefore, the loading conditions is equivalent to having one force and one moment acting on each node of the section as shown in Fig. 7(b). Assuming $W$ is the weight per unit length, the equivalent loading can be obtained as follows:

$$F = -WL/2$$

$$M1 = WL^2/12 \tag{6}$$

$$M2 = -WL^2/12$$

The force vector can then be constructed for each node by using the above equations. Once the force vector is obtained, the displacement vector $\boldsymbol{U}$ can be computed by:

$$\boldsymbol{F} = \boldsymbol{K}\boldsymbol{U} \tag{7}$$

Both $\boldsymbol{F}$ and $\boldsymbol{U}$ are all column vectors of size 6N.

The maximum deformation (maxDeform) of the pipe at one specific pose can then be obtained by computing the norm of displacement vector $U$ for each node. The $Cost(\mathrm{O2})$ is defined to be the sum of the maximum deformation at each pose along the trajectory as follows:

$$Cost(\mathrm{O2}) = \sum_{i=1}^{n} maxDeform_i \tag{8}$$

Note here the deformation of the pipe at the first trajectory waypoint (i = 0) is not computed since the pipe is still placed on a fixture.

*3.3.3.3. Force manipulability (O3).* The third objective is to minimise the force manipulability along the end-effector moving direction. According to the force/velocity duality, minimising the force manipulability is equivalent to maximising its velocity manipulability. Therefore, for a given set of joint velocities, the end-effector can move faster with a large velocity manipulability.[1] In order to compute the manipulability along a specific trajectory, the manipulability ellipsoids are constructed as follows:

$$\boldsymbol{v}^T \left(\boldsymbol{J}_f(\boldsymbol{q}) \boldsymbol{J}_f{}^T(\boldsymbol{q})\right)^{-1} \boldsymbol{v} = 1 \tag{9}$$

$$\boldsymbol{\gamma}^T \left(\boldsymbol{J}_f(\boldsymbol{q}) \boldsymbol{J}_f{}^T(\boldsymbol{q})\right) \boldsymbol{\gamma} = 1 \tag{10}$$

where $\boldsymbol{J}_f(\boldsymbol{q})$ is the Jacobian matrix of joint configuration $\boldsymbol{q}$. $\boldsymbol{v}$ is the velocity vector of the end-effector and $\boldsymbol{\gamma}$ is the force (torque) vector of the end-effector. Eq. (9) defines the velocity manipulability ellipsoid and Eq. (10) defines the force manipulability ellipsoid. Both ellipsoids are shown in Fig. 8 for a simple 3-link planar robot manipulator.

Given a unit vector $\boldsymbol{u}$ represents the direction of movement of the end-effector, the velocity manipulability ($\beta(\boldsymbol{q})$) and the force manipulability ($\alpha(\boldsymbol{q})$) are defined to be the length of the vector from the centre of the ellipsoid along $\boldsymbol{u}$ to the surface of the respective ellipsoid. $\beta(\boldsymbol{q})$ and $\alpha(\boldsymbol{q})$ can be computed by rearranging Eq. (9) and (10) as follows:

$$\beta(\boldsymbol{q}) = \left(\boldsymbol{u}^T \left(\boldsymbol{J}_f(\boldsymbol{q}) \boldsymbol{J}_f{}^T(\boldsymbol{q})^{-1} \boldsymbol{u}\right)\right)^{-1/2} \tag{11}$$

$$\alpha(\boldsymbol{q}) = \left(\boldsymbol{u}^T \boldsymbol{J}_f(\boldsymbol{q}) \boldsymbol{J}_f{}^T(\boldsymbol{q}) \boldsymbol{u}\right)^{-1/2} \tag{12}$$

As shown in Fig. 8, a direction with small $\alpha(\boldsymbol{q})$ has a relatively large $\beta(\boldsymbol{q})$ which suggests that the end-effector is relatively easier to move along the given direction $\boldsymbol{u}$. The objective function is defined as follows:

$$Cost(\mathrm{O3}) = \sum_{i=1}^{n} \alpha(\boldsymbol{q}_i) \tag{13}$$

Note here again $i$ starts from 1 rather than 0, since $\boldsymbol{u}_i$ is defined to be the vector when robot attempts to move from pose $i-1$ to pose $i$. In this way, all three objectives are consistent in the sense that they are trajectory-based objectives since there have to be at least 2 waypoints on the trajectory.

## 4. Objective and constraint evaluation

### 4.1. Constraint handling method

This paper handles constraints by using the method reported in [25]. The method compares two solutions based on the following 3 criteria:

- A feasible solution is always better than an infeasible solution.
- Between two infeasible solutions, the one that violates the constraint less is considered to be better.
- Between two feasible solutions, the one with a better objective cost is better.

The advantages of this method are twofold. Firstly, it does not require an explicit penalty parameter to handle the constraints. Besides, it allows objectives to be evaluated only when all the constraints are satisfied, which significantly reduces the algorithm running time

[1] Although the intention is to maximise the velocity manipulability, the velocity manipulability is not directly used in the paper to avoid the scenario that 2 objectives need to be minimised while the other one needs to be maximised. This scenario often leads to negating the objective to be maximised. However, negative objective costs are not ideal since Dijkstra's Algorithm used later requires edge weights of the graph to be positive. Therefore, the force manipulability is used instead.

**Table 1**
Random indices from [27].

| $m$ | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|
| RI | 0.58 | 0.9 | 1.12 | 1.24 | 1.32 | 1.41 | 1.45 | 1.49 |

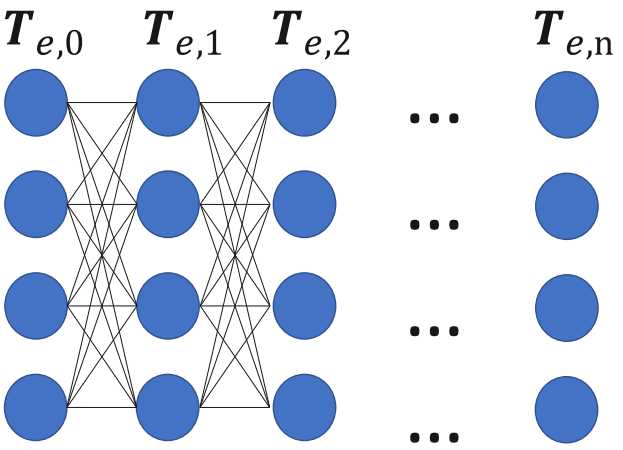


**Fig. 9.** Robot IK solution graph.

especially in the case of having an expensive objective function.

The optimisation problem has two constraints. The collision constraint is checked by the FCL library [23]. To check if the reachability constraint is satisfied, a third party IK solver (IKfast [26]) is used. The returned result of the IK solver is binary, either successful or not, which means it is impossible to compare two infeasible grasps which one violates the constraint more. To solve this problem, a combined constraint cost is defined for each waypoint along the trajectory as follows:

$$Cost(\mathrm{C_i}) = \begin{cases} 0, & \text{if both C1 and C2 are satisfied} \\ 1, & \text{if C1 is satisfied while C2 is not} \\ 2, & \text{if C1 is not satisfied} \end{cases} \tag{14}$$

And the cost for the whole trajectory is defined as follows:

$$Cost(\mathrm{C}) = \sum_{i=0}^{n} Cost(\mathrm{C_i}) \tag{15}$$

It should be noted that the constraint cost for the waypoint is set to 2 automatically when C1 is not satisfied. That is because the joint configuration ($\boldsymbol{q}$), which is required for checking C2, can only be obtained when C1 is satisfied.

### 4.2. Weights selection for combining multiple objectives

A weighted sum approach is used to handle multiple objectives:

$$Cost(\mathrm{O}) = w_1 Cost(\mathrm{O1}) + w_2 Cost(\mathrm{O2}) + w_3 Cost(\mathrm{O3}) \tag{16}$$

To systematically determine the weights for each objective, an Analytical Hierarchy Process [27] is used. The approach determines the relative importance amongst objectives by a series of pairwise comparisons. The results of the comparison are used to generate a comparison matrix as follows:

$$M = \begin{pmatrix} & \mathrm{O1} & \mathrm{O2} & \mathrm{O3} \\ \mathrm{O1} & 1 & 6 & 2 \\ \mathrm{O2} & 1/6 & 1 & 1/4 \\ \mathrm{O3} & 1/2 & 4 & 1 \end{pmatrix} \tag{17}$$

The entry of the matrix tells the preference level between the 2 objectives. For example, the entry at $(\mathrm{O1},\ \mathrm{O2})$ is 6, which means O1 is much preferred to O2. Then the consistency of the matrix is verified by computing the consistency ratio ($CR$) as follows:

**Function 1** GRASP-EVALUATION-MULTI-IK($g$)

```
Cost(C) = 0, Cost(O) = 0
for i = 0 : n,
    T_{e,i} = A_{p,i} B_e^P(g)
    Q_i = MULTI-IK(T_{e,i})   // Q_i is a vector of vector or 2d array
    if sizeof(Q_i) == 0 // IK solver returns 0 solution
        Cost(C) += 2
    else
        for j = 0 : sizeof(Q_i)
            if not COLLISION-FREE(Q_i[j])
                DELETE Q_i[j] from Q_i   // delete solution in collision
        if sizeof(Q_i) == 0   // no IK solution satisfies collision constraint
            Cost(C) += 1
if Cost(C) == 0, optimal_cost = Infinity
    G = CONSTRUCT-GRAPH (Q_0, Q)   // construct graph
    for j = 0 : sizeof(Q_0),
        current_cost = DIJKSTRA-SEARCH (G, j)   // search graph given the start vertex
        if current_cost < optimal_cost, optimal_cost = current_cost
    Cost(O) = optimal_cost + w_2 Cost(O2)
    g.cost = {0, Cost(O)}
else
    g.cost = {Cost(C), 0}
```

**Table 2**
BA parameters and definitions.

| Parameter | Definition |
|---|---|
| *ns* | Number of scout bees |
| *ne* | Number of elite sites |
| *nb* | Number of best sites |
| *nre* | Number of recruited bees for elite sites |
| *nrb* | Number of recruited bees for remaining best sites |
| *stlim* | Number of no improve iterations before site abandonment (stagnation limit) |

$$\begin{cases} CI = \dfrac{\lambda_{max} - m}{m-1} \\ CR = \dfrac{CI}{RI} \end{cases} \tag{18}$$

where $CI$ is the consistency index of the comparison matrix $M$, $RI$ is the average random index (given in Table 1), $CR$ is the random consistency ratio of the comparison matrix, $\lambda_{max}$ is the maximal eigenvalue, and $m$ is the order of the judgement matrix. If $CR$ is less than 10%, the matrix is considered to have an acceptable consistency. In our case, the $CR$ is 0.79%, which is acceptable.

For a consistent matrix, the weights are computed by normalising each column of the matrix and then calculating the average of each row. The resultant weight vector is $\boldsymbol{w} = [w_1\ w_2\ w_3]^T = [0.56\ 0.12\ 0.32]^T$

### 4.3. Considering multiple IK solutions

Eq. (2) assumes only one IK solution is available given an end-effector pose. However, there could be at most 16 different joint configurations for a 6 DoF robot [28]. Analytical IK solvers like IKFast can compute multiple IK solutions efficiently. In this section, the method used to handle multiple IK solutions is introduced. As shown in Fig. 9, each circle corresponds to a robot configuration (IK solution). All the circles in the same column have the same end-effector pose. A graph can then be constructed. The vertices of the graph are the robot configurations (blue circles in Fig. 9). The edges are created by connecting each vertex in one column to all the vertices in the next column. The cost of each edge is the weighted sum of joint motion distance (O1) and manipulability (O3). Deformation objective (O2) is not used to compute the cost because the deformation only depends on the orientation of the pipe and the grasp pose. Different arm configurations producing the same end-effector pose will not have an impact on the deformation cost. In this way, evaluating the cost of a single grasp is converted to a graph search problem. The problem is solved by running Dijkstra's algorithm for each start vertex (i.e., vertices in the first column). This process is similar to the Descartes Planner in ROS industrial project [29].

### 4.4. Overall evaluation process

The complete procedure for evaluating objective and constraint cost is presented in **Function 1**. The function starts by initialising both constraint cost and objective cost to be 0 and then for each waypoint on the trajectory of the pipe, the end-effector pose is computed. Given an end-effector pose, a subfunction MULTI-IK is called to get multiple IK solutions ($\boldsymbol{Q}_i$) from the IK solver. If the size of $\boldsymbol{Q}_i$ is zero, the end-effector pose is determined to be not reachable (i.e., C1 is not satisfied), thus the constraint cost is incremented by 2. If there is at least 1 feasible IK solution, the function will perform collision checking for all IK solutions. If there is no collision-free IK solution in $\boldsymbol{Q}_i$ (i.e., C2 is not satisfied), the constraint cost is incremented by 1. After checking whether constraints are satisfied or not, the objective cost will be evaluated for the constraint-free grasp. A weighted graph is created first (Line 14). Then, for each IK solution ($\boldsymbol{Q}_0[j]$) in $\boldsymbol{Q}_0$, Dijkstra's Algorithm is used to search for the shortest path from $\boldsymbol{Q}_0[j]$ to any IK solution in $\boldsymbol{Q}_n$, whose cost is assigned to *current_cost*. *optimal_cost* tracks the cost of the best IK solutions found so far in $\boldsymbol{Q}_0$. Then the cost for the grasp is set to be the sum of the optimal graph cost and the weighted deformation cost.

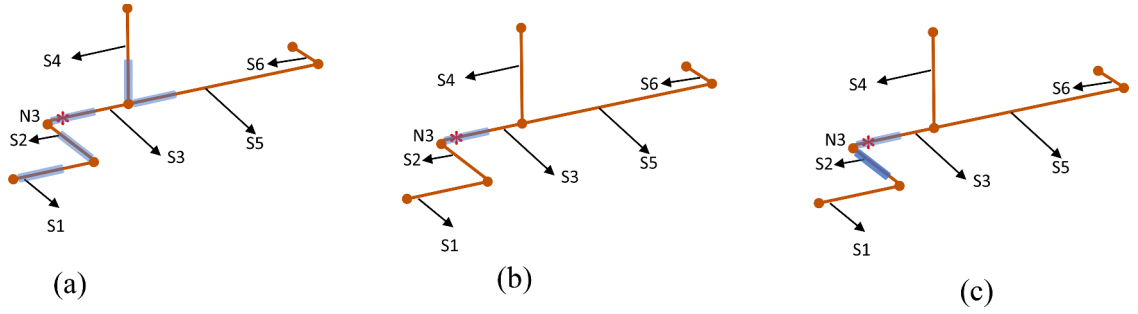


**Fig. 10.** Search neighbourhood for Str1, Str2 and Str3 respectively.

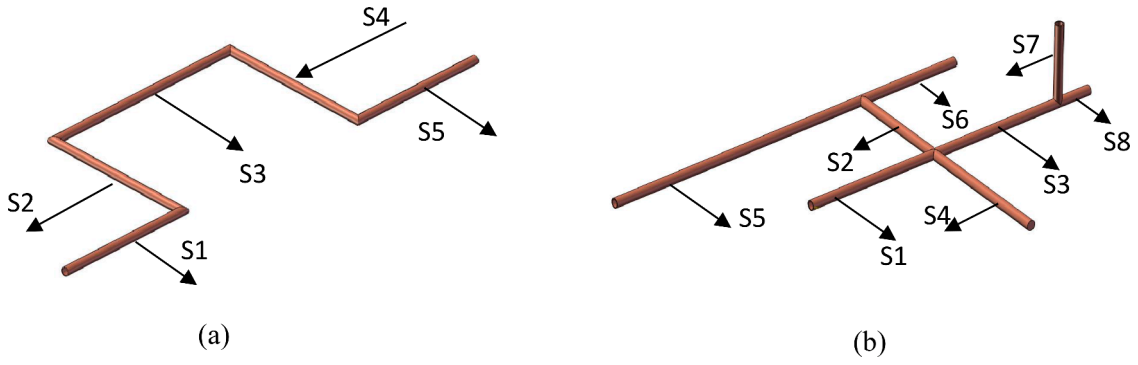


**Fig. 11.** (a) Pipe2 (b) Pipe3.

**Table 3**
Pipe assembly dimension.

| Section (mm) | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 |
|---|---|---|---|---|---|---|---|---|
| Pipe1 | 200 | 200 | 150 | 200 | 400 | 100 | \ | \ |
| Pipe2 | 200 | 200 | 300 | 200 | 200 | \ | \ | \ |
| Pipe3 | 200 | 150 | 200 | 200 | 400 | 150 | 100 | 50 |

**Table 4**
Pipe properties.

| Outer diameter (mm) | Inner diameter (mm) | Density (kg/m^3) | Young's modulus (GPa) | Shear modulus (GPa) |
|---|---|---|---|---|
| 12mm | 10mm | 8960 | 110 | 40 |

**Table 5**
Tested 9 sets of hyperparameters for the Bees algorithm.

| No. | ne | nre | nb | nrb | Random Scout | ns | Iteration |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 12 | 7 | 3 | 4 (10%) | 34 | 45 |
| 2 | 1 | 15 | 7 | 5 | 6 (10%) | 51 | 30 |
| 3 | 1 | 20 | 15 | 5 | 12 (10%) | 102 | 15 |
| 4 | 1 | 12 | 5 | 3 | 10 (30%) | 34 | 45 |
| 5 | 1 | 15 | 6 | 4 | 16 (30%) | 51 | 30 |
| 6 | 1 | 20 | 11 | 5 | 32 (30%) | 102 | 15 |
| 7 | 1 | 10 | 4 | 3 | 15 (45%) | 34 | 45 |
| 8 | 1 | 12 | 5 | 4 | 23 (45%) | 51 | 30 |
| 9 | 1 | 15 | 9 | 5 | 47 (45%) | 102 | 15 |

## 5. Optimisation algorithm

### 5.1. Original Bees algorithm

Bees algorithm (BA) [30], a population-based search algorithm which mimics the food foraging behaviour of honey bees, is used to solve the optimisation problem. The algorithm is shown in **Algorithm 1** and the definitions of hyperparameters are given in Table 2. The algorithm initialises *graspVec* by creating a colony of *ns* scout bees randomly in the search space. Each point in the search space (also known as a site in the Bees Algorithm literature) corresponds to a solution grasp *g*. After evaluating the cost of each site, all sites visited by scout bees are sorted and the best $nb < ns$ sites are selected for local search. amongst *nb* best sites, the scout bees at top *ne* sites recruit *nre* bees to perform local search in the neighbourhood. The bees at the remaining *nb-ne* best sites recruit *nrb* bees to perform local search ($nrb < nre$). If the result of local search does not improve and the site is searched again in the next iteration, the neighbourhood size is shrunk. The initial neighbourhood size is defined to be a proportion of the interval where the variable is defined. If the same site is searched for a predefined number of iterations (known as stagnation limit, *stlim*) without improving, the local minimum is considered to be reached and the scout bee at that site will perform random search again (site abandonment). After finishing local search, the remaining scout bees will be placed randomly in the search space to perform global search. Unlike the standard implementation of BA, the number of global searches in this work is set as $ns - ne \times nre - (nb - ne) \times nrb$ to keep the same number of function evaluations in the initialisation process and later iterations.

(*continued on next column*)

(*continued*)

**Algorithm 1** BA

1. Initialise *graspVec*
2. **for** each grasp *g* in *graspVec*
3. GRASP-EVALUATION-MULTI-IK(*g*)
4. **while** stopping condition not true
5. Locate elite and non-elite best sites
6. Local search
7. Site abandonment and neighbourhood shrinking
8. Global Search

### 5.2. Local search strategy

The local search strategy of the standard Bees Algorithm is straightforward. Given a solution site, The neighbourhood of the site is defined as a hyperrectangle. Recruited bees are randomly placed in the hyperrectangle to generate new grasps. The size of the hyperrectangle is shrunk when the same site is searched multiple times without improving. However, the optimisation variables used in this paper does not ensure newly generated grasps inside hyperrectangular are close to each other in terms of their Cartesian coordinate (as shown in Fig. 10 (a)). Since only variable *S* and *len* determine the Cartesian coordinates of a grasp, three different neighbourhood generation strategies for *S* and *len* are presented as follows:

Str1: This strategy performs the default behaviour of the Bees Algorithm i.e., treating *S* and *len* independently. For example, if the solution site is on S3 and the neighbourhood size for *S* is 4, the newly generated grasps can be on S1 – 5. If the solution site is close to one end of the section, it is likely that the newly generated grasp position *len* is out of the range. In this case, the *len* will be set as the limit.

Str2: This strategy is similar to Str1 except that the neighbourhood size for *S* is always 0. This ensures the newly generated grasps are always close to the solution site being searched since they are constrained to be on the same section. However, this strategy is very conservative and may lead to early convergence.

Str3 (proposed): Like Str2, Str3 does not allow the section to be changed in the usual condition. However, if the newly generated grasp position *len* is out of the range, unlike Str1 and Str2, Str3 will explicitly find if there is any other section connected to the section of the solution site and select randomly from all connected sections to locate the new grasp. For example, the feasible range for *len* on S3 is [0, 150]. If the generated *len* is -10 on S3, the grasp will be located on S2 with *len* set as $max\_length(S2) - 10$. In this way, Str3 ensures the newly generated grasps are close to the solution site and also allow the change of section during local search to avoid early convergence.

The neighbourhoods of Str1, Str2 and Str3 are shown as the blue area in Fig. 10 (a), (b) and (c). The red star is the solution site.

## 6. Experiment and results

### 6.1. Experiment setup

Three different copper pipe assemblies are tested in a simulated environment using the proposed method. The geometry of Pipe1 is shown in Fig. 2. Pipe2 and Pipe3 are shown in Fig. 11(a) and (b) respectively. The dimension of each pipe section is given in Table 3. Other properties of the pipe are listed in Table 4. The program is run on a Linux machine with an Intel Core i7–4712MQ CPU @ 2.30 GHz and 8GB RAM. The IK solver and collision checking library are accessed through

**Table 6**
Optimal solutions found for each pipe.

| | $S$ | *len* | $\Phi$ | $\theta$ | Objective Cost |
|---|---|---|---|---|---|
| Pipe1 | S4 | 79 | -155 | 180/0 | 4.7638 |
| Pipe2 | S3 | 151 | -100 | 180/0 | 4.1487 |
| Pipe3 | S2 | 96/97 | 23 | 180/0 | 5.7905 |

ROS MoveIt/Descartes API.

To avoid the potential impact of hyperparameters on the performance of the algorithm, each strategy is tested with 9 different sets of hyperparameters (listed in Table 5) for 50 times. In the experiment, three sets of hyperparameters are tested in parallel to accelerate the computation. The percentage 10%, 30% and 45% under the "Random Scout" column indicate the approximate proportion of global search in population. The optimal solution is determined to be found if the objective cost is within $\pm 0.1\%$ range of the known optimal cost. The optimisation process stops if the optimal solution is found, or the maximum iteration is reached. For all 9 sets of hyperparameters, their maximum number of grasp evaluations are the same ($ns \times iteration = 1530$).

Genetic Algorithm (GA) [31] and Particle Swarm Optimisation (PSO) [32], two widely used population-based metaheuristics, are also implemented to solve the proposed problem for comparison. To ensure a relatively unbiased comparison, 9 sets of hyperparameters are tested for both GA and PSO and the stopping condition is the same as BA. For GA, 9 sets of hyperparameters are generated from 3 population sizes (34, 51 and 102) and 3 mutation rates (0.2, 0.3 and 0.4). Binary tournament selection is used in the parents selection process of GA. The solutions are real value encoded. Single point crossover is used with crossover rate 1. The mutation operator for the binary variable $\theta$ is simply bit-flip. For $S$, *len* and $\phi$, the mutation operator samples uniformly within a given mutation range. For $S$, the mutation range is $[S_1, S_{max}]$, where $S_{max}$ is the maximum section number of the current pipe. For *len* and $\phi$, if the current solution value is $a$, the mutation range is $[a - 100, a + 100]$, while satisfying the usual lower and upper limits of the variable.

For PSO, 9 sets of hyperparameters are generated from 3 population sizes (34, 51 and 102) and 3 connectivity levels (10%, 50%, and 100%). For example, if the population size is 34, a 10% connectivity level means each particle is connected to 3 closest particles (fractional part is truncated). Inertia weight is set to 0.7 and both acceleration coefficients (c1 and c2) are set to 2. The velocity update is not implemented for the grasp section $S$ and the flip angle $\theta$ since they are intrinsically discrete (binary). These two variables are updated to be the same as their personal best, global best or keep their original value with probability 0.3, 0.4 and 0.3, respectively.

An additional experiment is performed to compare the performance of the multi IK evaluation method (Function 1) and single IK counterpart (implemented by removing the graph search step from Function 1). Both methods use the same set of hyperparameters (No. 7 in Table 5, except the number of iterations). The same 3 pipes in Table 3 are tested in this experiment. However, the stopping condition is different since it is difficult to determine the optimal solution for the single IK method. Theoretically, the optimal solution should be the same as the multi IK method. However, in practice, it almost never finds the same optimal solution since 1). the single IK method uses a numerical IK solver which usually only finds the solution closer to the initially provided solution; 2). There are too many possible arm motions for a predefined workspace object trajectory (for example, imagine there are 10 waypoints, and each waypoint has 2–4 possible IK solutions, the probability for the IK solver to generate the exact optimal combination is between $1/2^{10}$ and $1/4^{10}$). Therefore, the stopping condition for both methods is set to be the completion of 30 search iterations (i.e., 1020 grasp evaluations).

### 6.2. Results and discussion

#### 6.2.1. Optimal grasp and robot trajectory

The optimal grasp parameters found for each pipe to complete the given pipe trajectory are listed in Table 6. The trajectory of the robot for manipulating Pipe1–3 is shown in Fig. 12. It can be found in Table 6 that the flip angle $\theta$ can choose either 0 or 180 for all 3 pipes. This is because the gripper that used in this paper is symmetric and mounted to be aligned with the rotation axis of the last joint of the robot. Therefore, $\theta$ can be chosen as either 0 or 180 without affecting the following motion of the robot.

#### 6.2.2. Comparison amongst different algorithms

For each algorithm (strategy), the best optimisation results for solving 3 problems are listed in Table 7. It is worth noting that the best set of hyperparameters is different for different problems. Therefore, each column in Table 7 is not for a single set of hyperparameters but the combination of best results from different sets of hyperparameters on

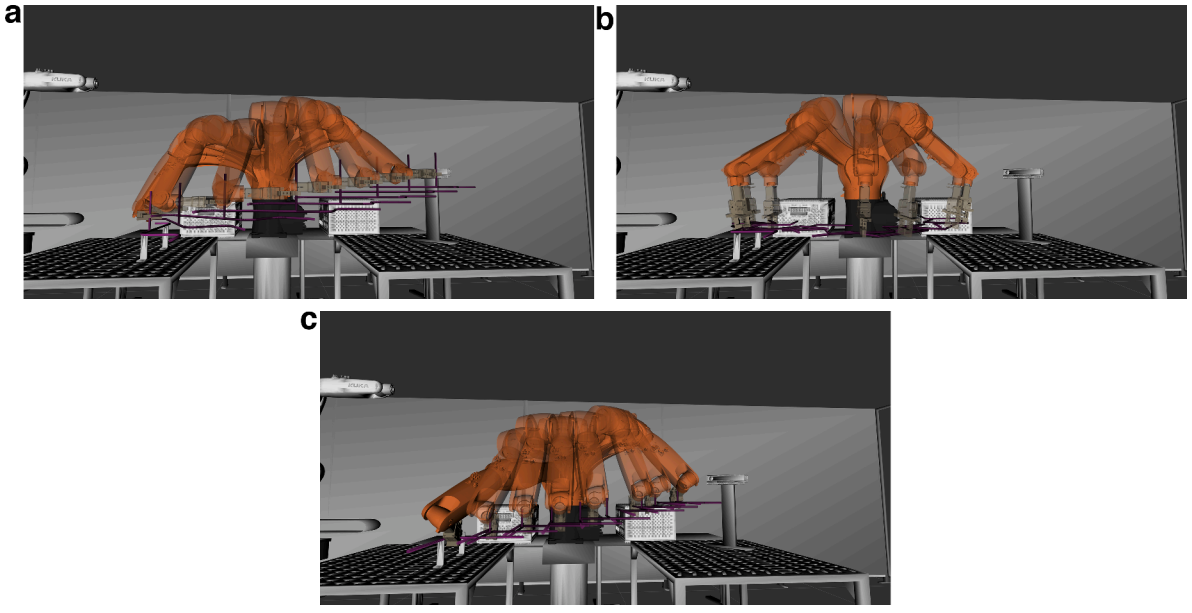


**Fig. 12.** Robot trajectory for manipulating Pipe1–3.

**Table 7**
Optimisation results of the best performed set of hyperparameters for each strategy.

| | Str1(BA) | Str2(BA) | Str3(BA) | GA | PSO |
|---|---|---|---|---|---|
| Grasp Evaluations | 1100 | 845 | 832 | 2172 | 1205 |
| Time (s) | 31.4 | 40.1 | 37.5 | 76.49 | 49.7 |
| Success Rate | 100% | 100% | 100% | 93.3% | 98% |

**Table 8**
Average optimisation results over 9 sets of hyperparameters for each strategy.

| | Str1(BA) | Str2(BA) | Str3(BA) | GA | PSO |
|---|---|---|---|---|---|
| Grasp Evaluations | 1572 | 1192 | 1168 | 2699 | 1891 |
| Time (s) | 41 | 57.4 | 51.4 | 92.4 | 93.2 |
| Success rate | 98.9% | 98.5% | 99.6% | 76.8% | 81.3% |

**Table 9**
Number of times that the algorithm fails to find global optimal.

| | Str1(BA) | Str2(BA) | Str3(BA) | GA | PSO |
|---|---|---|---|---|---|
| Pipe1 | 0 | 0 | 2 | 46 | 154 |
| Pipe2 | 9 | 2 | 2 | 76 | 7 |
| Pipe3 | 5 | 18 | 1 | 191 | 91 |
| Total | 14 | 20 | 5 | 313 | 252 |

different problems. The results show that by using an appropriate set of hyperparameters, all 3 BA local search strategies can achieve a 100% success rate out of 50 tests in finding the optimal solution. However, the computation speed of each strategy is different. The row named "Grasp Evaluations" lists the total number of grasps evaluated for solving 3 problems by each strategy. It is clear that Str2 and Str3 require significantly fewer grasp evaluations than Str1, which suggests that Str2 and Str3 should converge much faster than Str1. However, in terms of the actual running time shown in the next row, Str1 is a lot faster than both Str2 and Str3. The inconsistency is due to the constraint handling method used in this work (see Section 4.1). Since Str1 is intrinsically more stochastic and does not focus on a single section during the local search, the search efficiency of Str1 is lower than Str2 and Str3, thus it requires generating a large number of grasps to find the optimal solution. However, the grasp evaluation process is not required to be fully performed if the generated grasp is in constraint. Therefore, although Str1 generates more grasps to be evaluated, most of them can be evaluated within a short time. On the other hand, Str2 and Str3 mainly generate grasps that are close to the feasible grasp, which makes the generated grasps more likely to be feasible and need to be fully evaluated. Since evaluating the deformation objective O2 is relatively a time-consuming operation, Str1 has the advantage in terms of time consumption even though it generates more grasps. The results of GA and PSO are listed in the last two columns of Table 7. By using the algorithm setup in Section 6.1, the GA and PSO generally perform not as good as BA as they cannot achieve a 100% success rate on all three problems and the time required to finish the optimisation is longer.

The first three columns of Table 8 show the average results of 3 strategies over 9 sets of hyperparameters. Still, Str1 evaluates more grasps with less time than both Str1 and Str2. In terms of overall success rate, Str3 achieves the highest success rate. Table 9 lists how many times each local search strategy fails to find the optimal solution on the individual problem out of 450 tests (9 configs $\times$ 50 tests/config). It can be found that Str1 is likely to fail on Pipe2 and Str2 is likely to fail on Pipe3, while Str3 performs more consistently over different problems. The inconsistency of the algorithm performance is due to the choice of hyperparameters. More analyses regarding the impact of hyperparameters on the performance of the algorithm are presented in the next section. The average results of GA and PSO over all sets of hyperparameters are listed in the last two columns of Table 8. It can be found that, by using the algorithm setup in Section 6.1, GA and PSO are more sensitive to the selection of hyperparameters than BA as the success rate drops significantly compared to the best results in Table 7. The last 2 columns of Table 9 also show that the implemented GA and PSO in this work perform worse than BA on all tested problems except Pipe2, where the result of PSO is comparable to BA.

Based on the above analyses, generally, it is recommended to use Str3 for consistently good performance over different problems. However, if the computation time is extremely critical, Str1 can be considered as well.

#### 6.2.3. *The impact of the hyperparameters on the performance of the algorithm*

The impact of hyperparameters (i.e., population size and the proportion of global search) on the performance of BA is analysed in this section.

The size of scout bees

As shown in Table 10, by using a large size of population, all 3 strategies take longer to converge in terms of both the number of function evaluations and actual running time. In terms of success rate, Str1 with population size 102 only achieves a 97.7% success rate, which is obviously worse than others. Generally speaking, a large size of the population means fewer iterations can be run given a finite number of grasp evaluations, thus the algorithm cannot update current best solutions timely and reallocate computation resources efficiently, which leads to longer running time and less success rate. However, if the strategy lacks stochastics, using a large population size may have some

**Table 10**
Optimisation results using different population size.

| | Str1 | | | Str2 | | | Str3 | | |
|---|---|---|---|---|---|---|---|---|---|
| Population size | 34 | 51 | 102 | 34 | 51 | 102 | 34 | 51 | 102 |
| Grasp Evaluations | 1179 | 1453 | 2085 | 976 | 1137 | 1462 | 975 | 1090 | 1441 |
| Time (s) | 35.0 | 39.5 | 48.4 | 52.4 | 57.5 | 62.3 | 48.9 | 49.8 | 55.8 |
| Success Rate | 99.5% | 99.5% | 97.7% | 98.4% | 98.4% | 98.7% | 99.5% | 99.8% | 99.5% |

**Table 11**
Optimisation results using different proportion of global search.

| | Str1 | | | Str2 | | | Str3 | | |
|---|---|---|---|---|---|---|---|---|---|
| Global Search Proportion | 10% | 30% | 45% | 10% | 30% | 45% | 10% | 30% | 45% |
| Grasp Evaluations | 1503 | 1546 | 1668 | 1180 | 1208 | 1186 | 1212 | 1112 | 1178 |
| Time (s) | 41.2 | 40.3 | 41.6 | 67.4 | 56.8 | 48.0 | 61.5 | 47.4 | 45.3 |
| Success Rate | 98.4% | 99.3% | 99.1% | 96.8% | 99.6% | 99.1% | 99.3% | 100% | 99.6% |

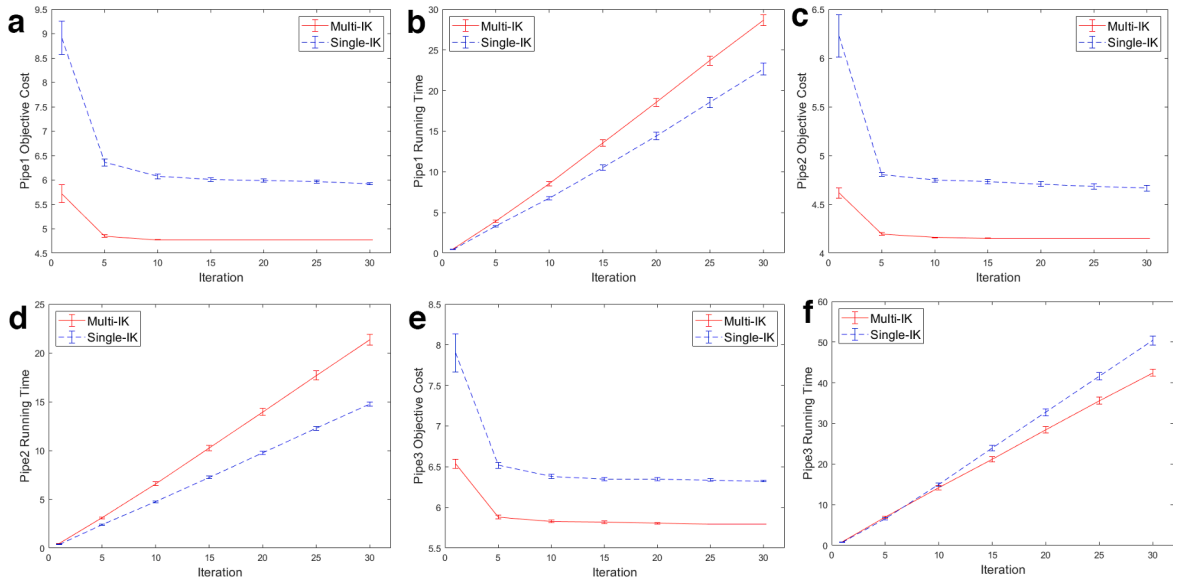


**Fig. 13.** (a) – (c) The average objective costs at each iteration for solving Pipe1 – 3 with multi IK and single IK method for 30 iterations. (d) – (e) The average time consumption at each iteration for solving Pipe1 – 3 with multi IK and single IK method for 30 iterations.

advantages. As in the case of Str2, using 102 achieves a 98.7% success rate, which is slightly higher than both 34 and 51. This is because a large population size ensures that initially the scout bees can cover the search space as much as possible and avoid early convergence.

The proportion of global search

As shown in Table 11, with the increase of global search proportion, the number of grasp evaluations for Str2 and Str3 almost stay at the same level, while Str1 increases steadily. This is because Str1 already has a lot of stochastics, continuing to increase the proportion of global search only makes the algorithm need more evaluations to converge. In terms of actual running time, Str1 is not impacted by the increase of global search proportion, while Str2 and Str3 run faster. This is due to the same reason why Str1 is faster than Str2 and Str3 as explained in Section 6.2.2. In terms of success rate, Str2 with 10% global search performs significantly worse than others. This is because Str2 does not allow any change of section during local search and relies heavily on global search to jump out of the local minimum. It is also interesting to see that the success rates of all 3 strategies go up when the global search proportion increase from 10% to 30%, and then drops when the proportion keeps increasing. This trend indicates that by keep adding more global searches to the algorithm a negative impact on the success rate may result.

### *6.2.4. Comparison between multi IK and single IK method*

The final result presented is the comparison between the multi IK grasp evaluation method and the single IK counterpart. The single IK method has the advantage that it does not need an analytical solver to generate multiple IK solutions, which is preferable if the robot system is not standard and does not have available analytical solutions. Although the details of the single IK grasp evaluation method are not presented in the paper, it should be easy to implement by slightly modifying the multi IK method.

As shown in Fig. 13, for all 3 pipes, the multi IK method achieves significantly lower average objective cost (19.6%, 11.3% and 8.4% respectively). Besides, the standard errors are also smaller, which means the results are more consistent. In terms of the running time, multi IK requires more time to finish for solving Pipe1 and Pipe2 than the single IK. This is predictable as multi IK has an additional graph search process. However, it is interesting to see that single IK requires more time for solving Pipe3 than multi IK. A possible explanation is that the neighbourhood of the solution that single IK converges to may have many other feasible solutions, which requires a large amount of time to fully evaluate them. On the contrary, the neighbourhood of the true optimum that multi IK converges to has fewer feasible solutions, therefore the method skips the objective function evaluation process, which results in faster convergence. This result suggests that although multi IK requires an additional graph search process when evaluating a single grasp pose, it is not necessarily slower than the single IK method. Therefore, the multi IK method is preferable whenever a suitable IK solver is available.

## 7. Conclusion

In this paper, a methodology is developed to optimise the grasp pose for 3D pipe assembly manipulation. The method can effectively optimise the grasp pose based on three trajectory-based objectives (joint motion, deformation of the object and force manipulability) while satisfying reachability and collision constraint. The grasp evaluation process features a decoupled constraint handling method to reduce grasp evaluation time, an AHP method to select the weights for combining multiple objectives, and a Dijkstra's Algorithm to find optimal trajectory amongst all possible IK solutions. The Bees Algorithm is used to solve the constrained optimisation problem with a proposed problem-specific local search strategy. Extensive benchmarks have been performed to evaluate the performance of 3 local search strategies and 2 other metaheuristics (GA and PSO). It is found that BA with proposed Str3 is less sensitive to the hyperparameters and can achieve consistently good performance on different problems. Besides, the comparison between multi IK and single IK method proves that by considering multiple IK solutions, the objective cost can be improved significantly.

The method is intended for manufacturing pipe assemblies in the Factory-In-A Box (FIAB) scenario to address the limitation of specially designed end-effector, the compact environment of FIAB and the flexibility of pipe structure. However, it can also be generalised to manipulating other compliant objects that have many grasp candidates with alternative deformation estimation methods.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgement

The author would like to thank the Manufacturing Technology Centre (MTC), Coventry, UK, and Innovate UK for supporting this project through funding and case study.

## References

[1] J. Bohg, A. Morales, T. Asfour, D. Kragic, Data-driven grasp synthesis-A survey, IEEE Trans. Robot. (2014), https://doi.org/10.1109/TRO.2013.2289018.
[2] J.P.C. de Souza, L.F. Rocha, P.M. Oliveira, A.P. Moreira, J. Boaventura-Cunha, Robotic grasping: from wrench space heuristics to deep learning policies, Robot. Comput. Integr. Manuf. 71 (2021), 102176, https://doi.org/10.1016/j.rcim.2021.102176.
[3] A. ten Pas, M. Gualtieri, K. Saenko, R. Platt, Grasp pose detection in point clouds, Int. J. Rob. Res. (2017), https://doi.org/10.1177/0278364917735594.
[4] S.M. LaValle, Planning algorithms, 2006. https://doi.org/10.1017/CBO9780511546877.
[5] S.M. LaValle, J.J. Kuffner, Randomized kinodynamic planning, Int. J. Rob. Res. (2001), https://doi.org/10.1177/02783640122067453.
[6] L.E. Kavraki, P. Švestka, J.C. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robot. Autom. (1996), https://doi.org/10.1109/70.508439.
[7] M.B. Horowitz, J.W. Burdick, Combined grasp and manipulation planning as a trajectory optimization problem, in: Proc. - IEEE Int. Conf. Robot. Autom, 2012, https://doi.org/10.1109/ICRA.2012.6225104.
[8] N. Vahrenkamp, M. Do, T. Asfour, R. Dillmann, Integrated grasp and motion planning, in: Proc. - IEEE Int. Conf. Robot. Autom, 2010, https://doi.org/10.1109/ROBOT.2010.5509377.
[9] T. Pardi, R. Stolkin, A.M.E. Ghalamzan, Choosing Grasps to Enable Collision-Free Post-Grasp Manipulations, in: IEEE-RAS Int. Conf. Humanoid Robot, 2019, https://doi.org/10.1109/HUMANOIDS.2018.8625027.
[10] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. (2011).
[11] M. Zucker, N. Ratliff, A.D. Dragan, M. Pivtoraiko, M. Klingensmith, C.M. Dellin, J. A. Bagnell, S.S. Srinivasa, CHOMP: Covariant Hamiltonian optimization for motion planning, Int. J. Rob. Res. (2013), https://doi.org/10.1177/0278364913488805.
[12] M. Saadat, P. Nan, Industrial applications of automatic manipulation of flexible materials, Ind. Rob. (2002), https://doi.org/10.1108/01439910210440255.
[13] J. Sanchez, J.A. Corrales, B.C. Bouzgarrou, Y. Mezouar, Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey, Int. J. Rob. Res. (2018), https://doi.org/10.1177/0278364918779698.
[14] P. Jiménez, Survey on model-based manipulation planning of deformable objects, Robot. Comput. Integr. Manuf. (2012), https://doi.org/10.1016/j.rcim.2011.08.002.
[15] W. Wu, Y. Zhu, X. Zheng, Y. Guo, A novel cable-grasping planner for manipulator based on the operation surface, Robot. Comput. Integr. Manuf. 73 (2022), 102252, https://doi.org/10.1016/j.rcim.2021.102252.
[16] X. Jiang, Y. Nagaoka, K. Ishii, S. Abiko, T. Tsujita, M. Uchiyama, Robotized recognition of a wire harness utilizing tracing operation, Robot. Comput. Integr. Manuf. (2015), https://doi.org/10.1016/j.rcim.2014.12.002.
[17] J. Zhu, B. Navarro, R. Passama, P. Fraisse, A. Crosnier, A. Cherubini, Robotic manipulation planning for shaping deformable linear objects with environmental contacts, IEEE Robot. Autom. Lett. (2020), https://doi.org/10.1109/LRA.2019.2944304.
[18] E. Glorieux, P. Franciosa, D. Ceglarek, Quality and productivity driven trajectory optimisation for robotic handling of compliant sheet metal parts in multi-press stamping lines, Robot. Comput. Integr. Manuf. (2019), https://doi.org/10.1016/j.rcim.2018.10.004.
[19] H. Hoffmann, M. Kohnhäuser, Strategies to optimize the part transport in crossbar transfer presses, CIRP Ann. - Manuf. Technol. (2002), https://doi.org/10.1016/S0007-8506(07)61458-9.
[20] E. Glorieux, P. Franciosa, D. Ceglarek, End-effector design optimisation and multi-robot motion planning for handling compliant parts, Struct. Multidiscip. Optim. (2018), https://doi.org/10.1007/s00158-017-1798-x.
[21] B. Tipary, G. Erdős, Generic development methodology for flexible robotic pick-and-place workcells based on Digital Twin, Robot. Comput. Integr. Manuf. (2021) 71, https://doi.org/10.1016/j.rcim.2021.102140.
[22] M. Jackson, M. Wiktorsson, M. Bellgran, Factory-in-a-box — Demonstrating the next generation manufacturing provider, Manuf. Syst. Technol. New Front. (2008), https://doi.org/10.1007/978-1-84800-267-8_70.
[23] J. Pan, S. Chitta, D. Manocha, FCL: A general purpose library for collision and proximity queries, in: 2012 IEEE Int. Conf. Robot. Autom., IEEE, 2012, pp. 3859–3866, https://doi.org/10.1109/ICRA.2012.6225337.
[24] A.J.M. Ferreira, MATLAB codes for finite element analysis: Solids and structures, Solid Mech. Appl. (2009).
[25] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. (2000), https://doi.org/10.1016/S0045-7825(99)00389-8.
[26] R. Diankov, Automated construction of robotic manipulation programs, 2010. https://doi.org/isbn 9781124535470.
[27] T.L. Saaty, A scaling method for priorities in hierarchical structures, J. Math. Psychol. (1977), https://doi.org/10.1016/0022-2496(77)90033-5.
[28] R. Manseur, K.L. Doty, A robot manipulator with 16 real inverse kinematic solution sets, Int. J. Rob. Res. (1989), https://doi.org/10.1177/027836498900800507.
[29] J. Meyer, Descartes_Planner - ROS Wiki, (n.d.), 2022. http://wiki.ros.org/descartes_planner. accessed January 11.
[30] D.T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, M. Zaidi, The Bees algorithm - a novel tool for complex optimisation problems, in: Intell. Prod. Mach. Syst. - 2nd I*PROMS Virtual Int. Conf. 3-14, 2006, p. 2006, https://doi.org/10.1016/B978-008045157-2/50081-X. July.
[31] D. Goldberg, Genetic Algorithms in optimization, Search and Machine Learning, Addison Wesley, 1988.
[32] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proc. ICNN'95 - Int. Conf. Neural Networks, IEEE, 1995, pp. 1942–1948, https://doi.org/10.1109/ICNN.1995.488968.