Imperial College
London

Final report for MSc Project

Development of Trajectory-Based Control of Supernumerary Robotic Limbs for Assembling Tasks

Author name:

Chi Cheng

01901969

Supervisors:

Dorian Verdel & Etienne Burdet

Submitted in partial fulfilment of the requirements for the award of MSc Human and Biological Robotics in Biomedical Engineering from Imperial College London

September 2024

Word count: 5310

Abstract

Supernumerary robotic limbs (SRLs) are advanced robot devices that can be used to augment human motion capability when facing complex tasks. In this project, a control strategy based on trajectory generating is developed for an assembling task. This method is autonomous and could achieve high efficiency without the prerequisite of demonstration data or complex physical models. The task scenario is separated into 8 stages with representative static states and action states of human and robots. Two action states of robots are movement based on determined trajectory by planning and tracing methods. A comprehensive RRT-based trajectory planning method was proposed to generate a collision-free path during the task process to completely eliminate the requirement of human interference. The proposed position control combined with the given trajectory ensures the correctness of the robot's motion during the collaboration process. The results demonstrate that the trajectory-based control method is useful for assembling tasks.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Dr. Dorian Verdel and Dr. Etienne Burdet, for their unwavering support, patience, and guidance throughout this project. Their insightful feedback and continuous encouragement during our regular meetings have been invaluable. I am also sincerely grateful to the members of our lab for their professional advice and kind encouragement, which have greatly contributed to the completion of this work.

Content

1. Introduction

Supernumerary robot limbs (SRLs) are wearable robotic instruments that provide additional motion capabilities to the human body. In multiple working environments, they could offer assistance like a co-worker or even a guide to complete tasks that a single person cannot accomplish alone [1]. Thus, SRLs were developed in various scenarios [2] for human augmentations. Current research on controlling those SRLs is separated due to the involvement of human command. Some control strategies are based on human command, whereas some strategies aim to eliminate the role of explicit command. Unfortunately, most control methods are task-based which can not be generally used in other tasks [3].

Our project faces a complex task in architecture, which aims to install a tube on the wall. This task is initially implemented by two workers, but with the help of SRLs, one worker is sufficient to finish this job. The objective of this project is to develop a controller and motion planner for two SRLs to operate within a virtual environment. SRLs are designed to cooperate with a human avatar in tasks such as picking up, moving, and holding a tube, as illustrated in Figure 1. The project encompasses several key tasks, including creating a virtual environment with models of both the human and SRLs, developing a low-level controller to precisely manipulate SRLs, and designing a trajectory planning algorithm that enables the gripper of SRLs to reach the designated grab position. Additionally, a trajectory generation algorithm will be developed to synchronize with the motion of the human hand. Section II reviews related work on SRLs, covering various prototypes and existing control strategies. Section III provides a detailed description of the virtual working scenario and outlines the essential methods used for low-level and high-level control. Section IV presents the performance outcomes of the controller and motion planners.

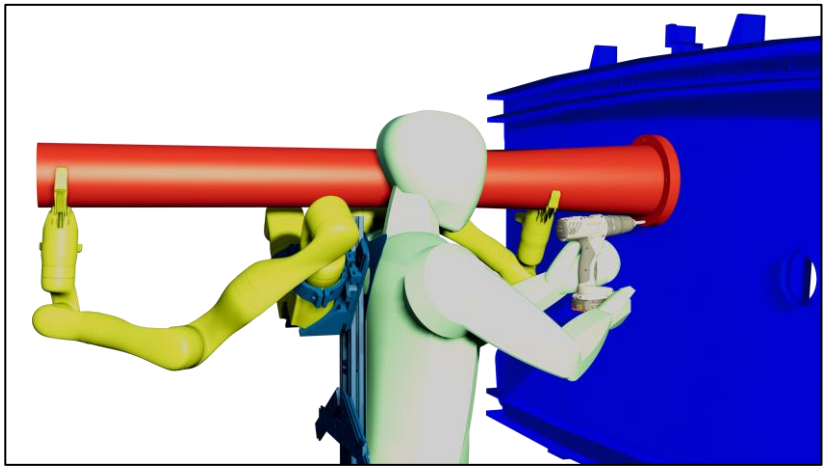


Figure 1: The worker can pick up, move and hold the tube with the help of SRLs.

2. Literature Review

2.1 Supernumerary Robot Limbs

SRLs, or Supernumerary Robotic Limbs, are wearable robotic devices designed to assist the human body by enhancing or augmenting its capabilities. These systems can be categorized based on their application (such as for auxiliary tasks or support), the materials used (rigid or soft robotics), and their structural design (such as arms, legs, or fingers) [4].

Recently, a wide range of innovative SRL prototypes have been developed for research purposes. Among the most prominent and classical areas of focus is SRL-supported assembly. The Asada team at the

Massachusetts Institute of Technology [5] pioneered the development of Supernumerary Robotic Arms (SRAs), initially designing a prototype aimed at assisting with auxiliary drilling tasks, as shown in Figure 2(a). Their SRAs consist of a pair of robotic arms attached to a wearable backplane positioned around the user's waist. These robotic arms help lift heavy objects, thereby reducing work-related strain. Subsequently, the team enhanced the SRAs to support overhead tasks [6]. In this iteration, the robotic arms were mounted on the user's shoulders via a backpack, as depicted in Figure 2(b). In 2021, Andrea et al. [7] proposed using SRAs to mitigate vibrations during drilling tasks, thereby reducing strain on the upper limb joints, as illustrated in Figure 2(c). Their prototype included a wearable gravity-compensating arm (Armor Man), a co-driven under-actuated manipulator (Soft-Hand), and a custom-damped wrist.

Other types of SRLs are also noteworthy. For example, Liang et al. [8] introduced a soft inflatable robotic arm made from TPU-coated fabric in daily work scenarios, shown in Figure 2(d). Unlike rigid robotic limbs, this arm features a bi-directionally curved torso and a tentacle-like gripper, offering a lightweight and foldable design that enhances the functionality of SRLs. Additionally, the supernumerary robotic legs introduced by Parietti et al. [9] in 2015 and the supernumerary robotic fingers developed by Guggenheim et al. [10] are remarkable SRL innovations as well. Moreover, some SRLs have been introduced into the arts, like Jizai Arms [11] for dancing, as shown in Figure 2 (g).

(a)

(b)

(c)

(d)

(e)

(f)

(g)

Figure 2: Several sorts of SRLs prototypes. (a) SRLs for auxiliary drilling [5]. (b) SRL aids the human worker in overhead [6]. (c) Drill holes with SRLs [7]. (d) Lift heavy objects with SRL [8]. (e) SRL for auxiliary support [9]. (f) Use the supernumerary fingers robot to open the door [10]. (g) Jizai Arms SRLs [11].

2.2 Control Strategies of SRLs

Once the prototypes of SRLs are designed, the most critical challenge becomes guiding their motion effectively. Depending on the origin of motion intent, existing SRL control methods are generally classified based on the autonomous level of the robots, which can be broadly categorized into two types: augmentation by transfer and autonomous behaviour control [1].

1) 'Augmentation by transfer' Control Methods

'Augmentation by transfer' for SRLs is the user-dominated method in which the robot has little or no feasibility of autonomous behaviour. These methods typically require explicit commands from the user and are aimed at extending or enhancing human capabilities. Due to the different ways to generate control signals, this strategy can be separated into manual control, bioelectric signal control, synergy-based control and action-based control. Hussain et al. developed a Vibrotactile ring for manual control in which stroke patients can directly control the robot arm to proceed with the pick-and-place motion [12]. Bioelectric signal control is the widely used method and mainly obtains EMG signals gathered from chest & abdominal muscle [13], frontal muscle [14] and auricle [15] as an initial control command. Synergy-based control and action-based control are similar, as their process SRLs' motion by mapping or mimicking robotic action from human action [16] [17].

2) 'Autonomous Behaviours' Control Methods

In contrast to direct control methods, 'autonomous behaviours' are more suited to this project, as they generate movements that respond to human actions and coordinate with them without requiring explicit user commands. These methods can be categorized based on their motion planning strategies into two types: demonstration-based control and balance model-based control.

Demonstration-based control relies on a data-driven approach, where control algorithms are derived from data recorded during human activities. For example, an auto model named ARMAX [5] was designed for iteratively refining a correlation function to model human skills and motion based on demonstration data like leading hand force or moving distance. Nevertheless, the ARMAX model struggles when correlations of data are weak. To handle system uncertainties and predict outcomes, Bonilla et al. used Colored Petri Nets (CPN) [18] to model tasks, allowing coordination between SRAs and users by detecting worker actions. Bonilla et al. further developed the CPN method by introducing the PLSR [6] for predictive analysis of the human demonstration data.

On the other hand, balance model-based control uses models of human-SRL interactions to develop strategies that maintain system stability and mitigate disturbances caused by human movements. For instance, Parietti et al. [9] proposed gait control strategies of supernumerary robotic legs for balance assistance. As a result, those robotic legs could assist the walking balance without disrupting the user's natural motion.

2.3 Motion Planning of Dual-arm Robot

While existing SRL control strategies are well-researched, our task scenario requires a more comprehensive approach, involving automated planning to avoid collisions with the human avatar. Therefore, inspired by the collaboration control strategy developed by Tu et al. [19], trajectory planning methods of traditional dual-arm robot systems would be considered.

Dual-arm manipulator motion planning involves generating a path or trajectory that guides the arms to the target location while avoiding obstacles and self-collisions, considering the manipulators' physical and kinematic constraints, including redundancy, singularity, joint limits, and velocity boundaries [20].

Song and Kumar [21] designed a controller and motion planner based on *artificial potential field* (APF) to transfer objects to the target. This method has been improved by Byrne et al. [22] which avoids local minima after adding sub-goal selection and goal configuration sampling. *Rapidly-exploring Random Tree*, also known as RRT, is another widely deployed trajectory planning method. After raised by LaValle in [23], various RRT-based method has been developed to optimize it performance in searching a feasible path. Basic RRT methods have been introduced for dual-arm robot trajectory planning firstly, such as RRT [24], RRT* [25] or RRT*Start[26]. Then, advanced RRT methods like Improved informed-RRT* [27] or P_RRT* [28] were adopted to gain global approximate optimal solutions and some research introduced cutting-edge RRT-based methods, like GA-RRT [29] or RRT-Connect[30] [31] to rapidly reduce the time for searching. With the ongoing development of RRT, increasingly effective RRT-based approaches for dual-arm control will continue to emerge. Völz and Graichen [32] treated motion planning as an optimization problem. Compared to RRT methods, this optimization method reduces the joint motion in an obvious manner. In further development, genetic algorithm [33] and sequential convex programming [34] were employed to find shorter paths & time or minimize the calculation time for a given path.

Furthermore, thanks to the recent development of machine learning, an AI-based motion planner was developed for dual-arm robots. For example, a QP-RNN [35] method was developed by Liang et al. to avoid the self-collision between robots. A reinforcement learning model like soft actor−critic (SAC) was proposed by Prianto et al. [36] to create a collision-free trajectory of dual-arm robots.

3. Methods

3.1 Working Scenario

1) SRLs model and Humanoid

SRLs in this project are constructed by a wearable bag and two 7Dof Kinova Gen3 robots. The frame definitions and dimensions of this robot are shown in Figure 3. Moreover, the joint position limits are listed in Table 1, corresponding to the rotation capability of each actuator. In the virtual environment depicted in Figure 4 (a), two KINOVA Gen3 robots are mounted on the back of a human avatar to ensure synchronised movement between the robots and the avatar. The robots are oriented at a 45-degree angle in the transverse plane relative to the z-axis, optimising their working space. Each robot is equipped with 8 control channels: the first 7 channels transmit torque values to the corresponding motors, which actuate the robot's 7 joints. The input torque limits are set according to KINOVA's official guidelines. The final channel transmits position signals to control the opening and closing of the gripper.

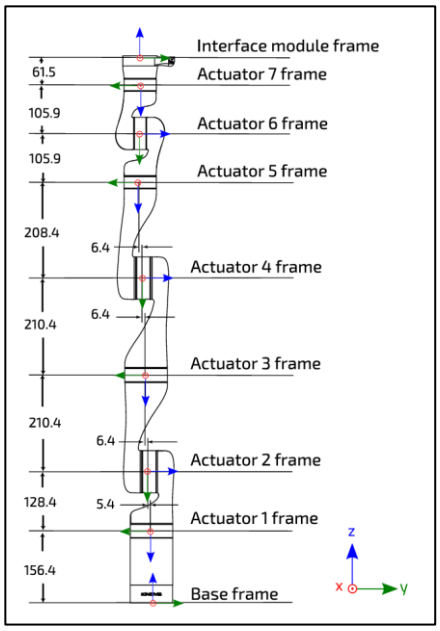
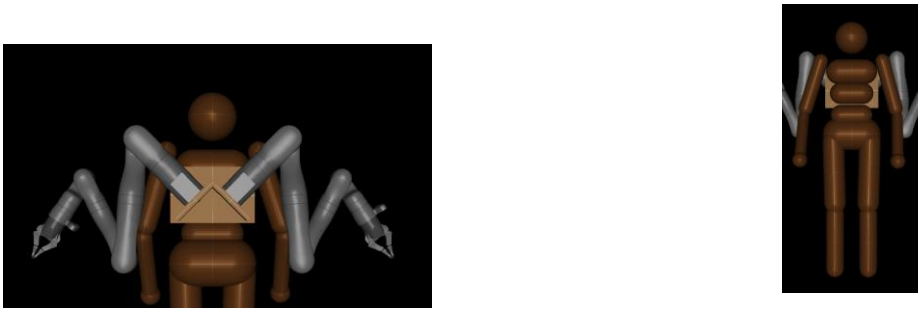


Figure 3: 7DoF robot frame definitions and dimensions. (all joints at 0 degrees, dimensions in mm) [37]

Actuator	Angular range	
	Lower limit	Upper limit
1	- ∞	+ ∞
2	- 128.9°	+ 128.9°
3	- ∞	+ ∞
4	- 147.8°	+ 147.8°
5	- ∞	+ ∞
6	- 120.3°	+ 120.3°
7	- ∞	+ ∞

Table 1: Gen3 7DoF spherical wrist robot joint position limits. [37]

Since developing an advanced human avatar is not the objective of this project, we provide a simplified avatar according to the initial humanoid model in Mujoco as shown in Figure 4 (b). This avatar has a channel transmitting a position signal to control the squat and stand behaviours. Each arm has two position signal channels. One channel takes the response to control the flexion & extension, while the other one controls the abduction & adduction. Furthermore, each forearm has a position signal channel to control the flexion & extension.



(a) (b)

Figure 4: (a) The virtual model of SRLs. (b) The ideal avatar of worker.

2) Assembling Stages & Task Description

The main assembling task in this letter is to lift a tube (diameter: 4mm, length: 2.5m) on the ground from one side (right side), and move it to the other side (left side) so that both robot arms can hold it. Besides the unpredictable bias of human behaviour, the entire process can be simplified as eight ideal stages.

- **Stage 1.** The worker squats down to reach the tube.
- **Stage 2.** The worker grasps the tube, while the robot arm on the same side simultaneously moves into position to also grasp the tube.
- **Stage 3.** The worker stands up, lifting the tube with the coordinated effort of the human arm and the robot arm.
- **Stage 4.** The worker rises the tube above head level, with the robot arm moving synchronously with the worker's hand.
- **Stage 5.** The tube is transferred to the other hand and the end effector of the robot arm shift to trace it.
- **Stage 6.** The tube is moved to the assembling position, which is closely up to the worker's shoulder.
- **Stage 7.** The other robot arm moves to grasp and hold the front part of the tube.
- **Stage 8.** The worker releases the tube and begins the installation process.

In order to define the avatar and robotic state and action, five static states are listed:

$$S = \{\, body\ up, body\ down, exchanging, hold, installation \,\}$$

The action states of the worker avatar are shown as:

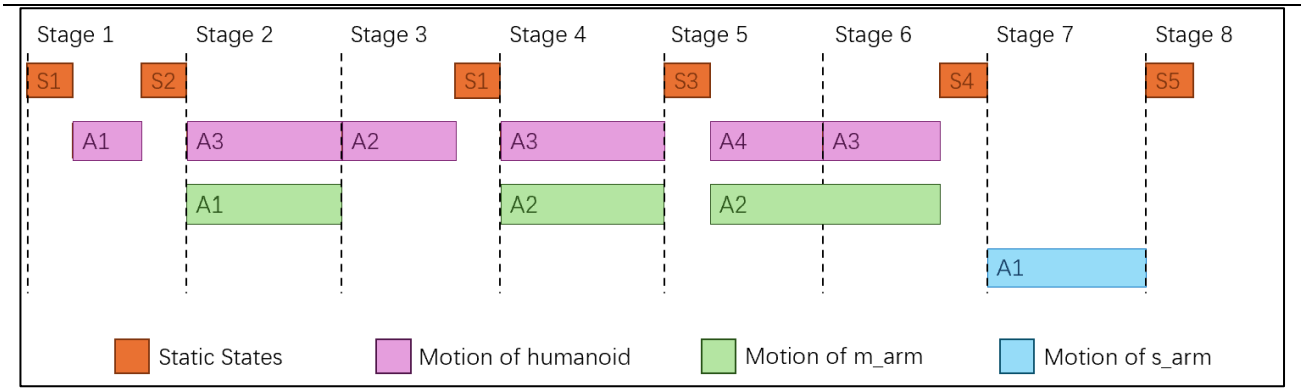
$$A_{avatar} = \{squat, stand, arm\ motion, hand\ change\}$$

There are only three motion models for SRLs, but different limb has its exclusive sequences in the entire process. Hence, the robot arm closer to the initial position of the tube is denoted as the main arm, whilst the other arm is denoted as the secondary arm. The motion models of SRLs are shown as:

$$A_{m\_arm} = \{motion\ planning, trajectory\ tracing\}$$

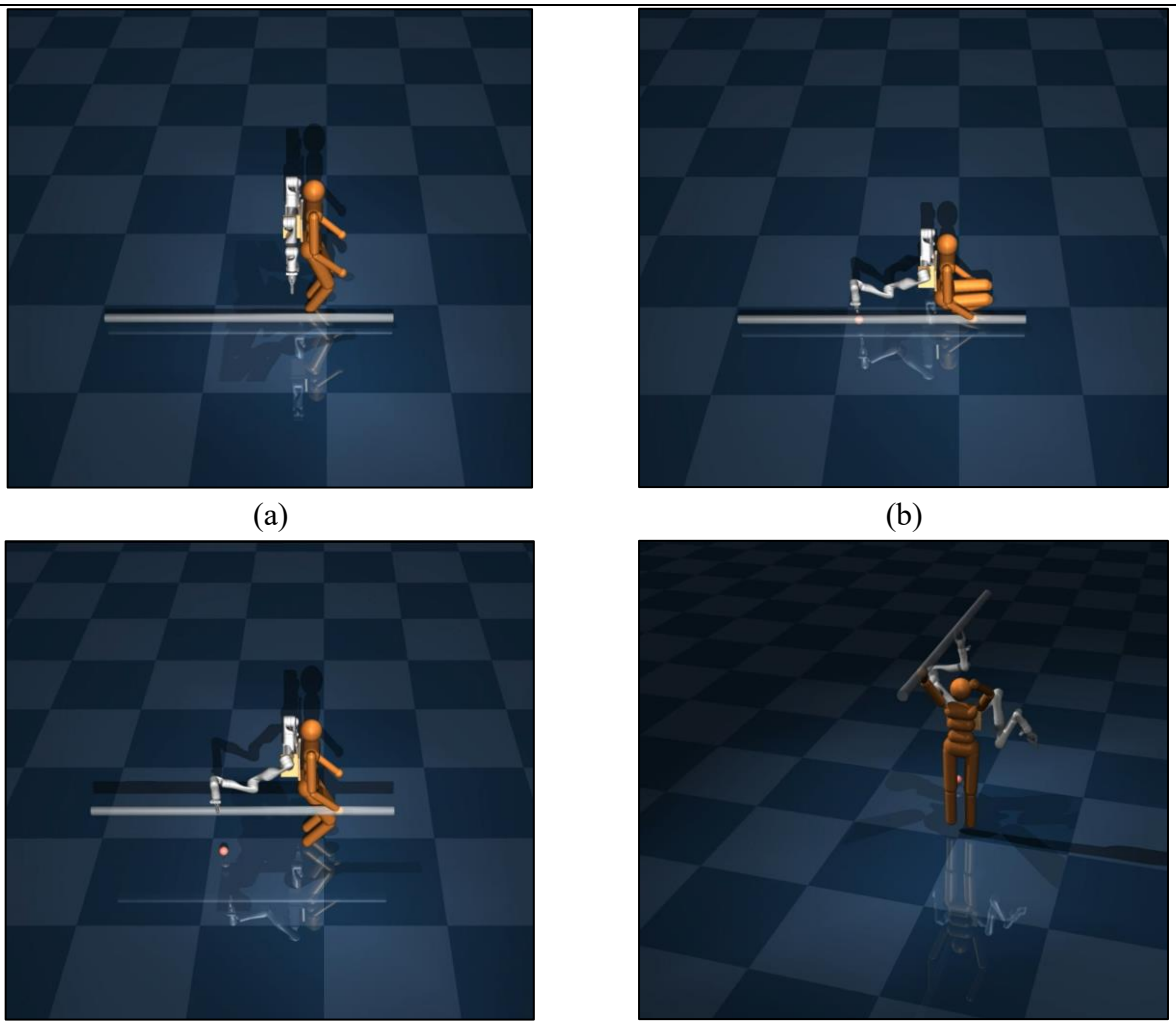
$$A_{s\_arm} = \{motion\_planning\}$$

In summary, the whole scenario can be defined by the state-action model as shown in Figure 5 and Figure 6.



(a)

Figure 5: The stages, states and motion in working scenario.



(a) (b)

(c) (d)

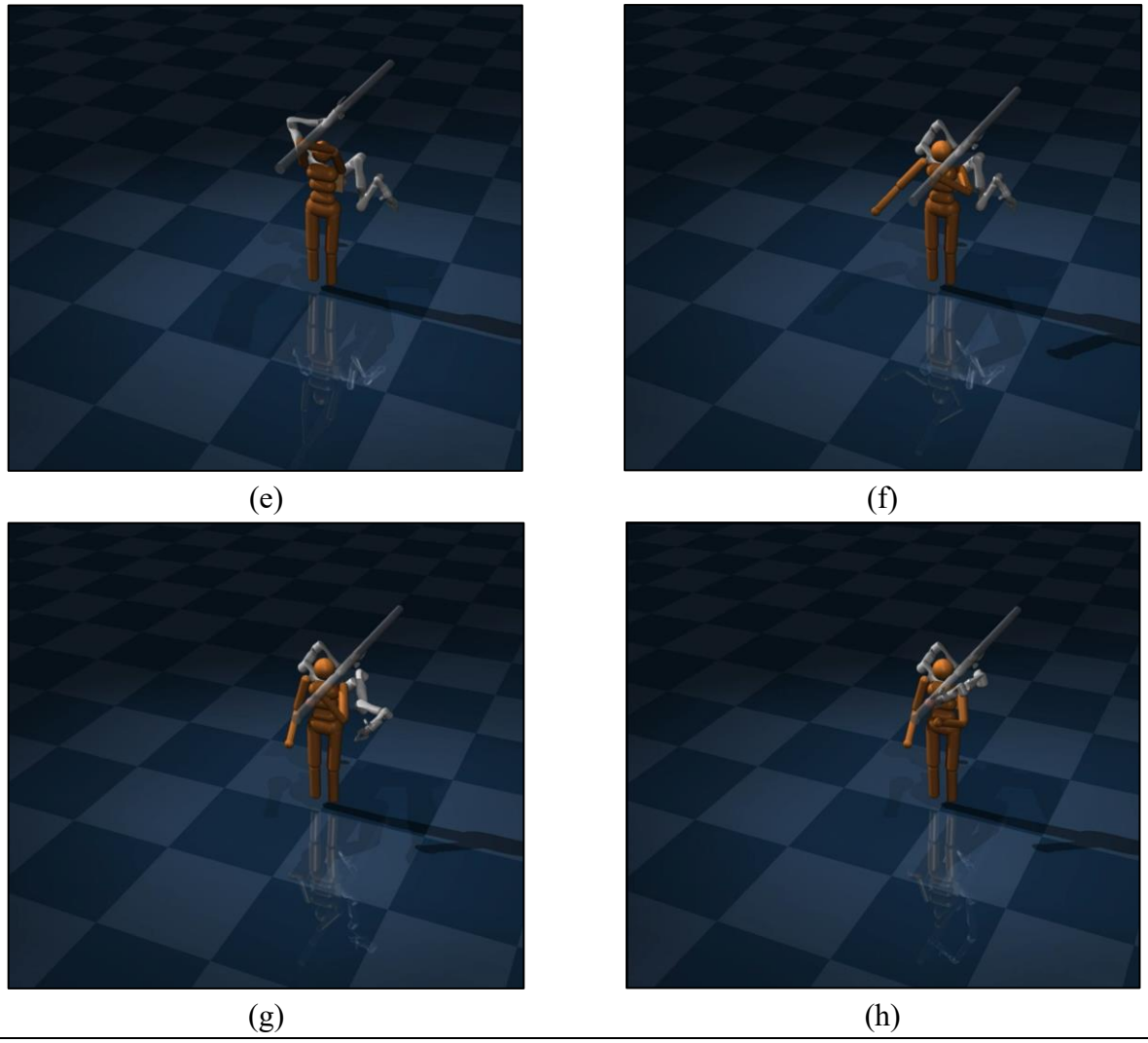


Figure 6: The snapshots of each stage. (a)-(h): The snapshots of Stage 1-8 of the assembling task.

3.2 Motion Strategies

1) Motion Planning

Motion planning for SRLs is conducted in a high-dimensional space, besides the human body and objects in the workspace of robots, the two robotic arms also act as dynamic obstacles to each other. Consequently, this study employs a random sampling-based motion planning approach and develops the motion planning analysis using the RRT algorithm. In this project, an improved RRT-Connect method is applied to create a collision-free trajectory. This trajectory would lead the gripper of SRLs to reach the grab point on the tube. The design of this method is inspired by the RRT-Connect [31] and the goal-directional RRT method [38]. The algorithm structure is developed by referring to a published "RRT-Connect" algorithm [39], which banned the swap function. Thus, their method is more likely a hybrid method of RRT-Connect and bi-directional RRT method. We redesigned its structure to process the real RRT-Connect method.

Compared to initial RRT method, RRT-Connect method adopts a technique to rapidly reduce the simulation time, which is establishes two search trees from start point q_{start} and goal point q_{goal}, simultaneously, as shown in Figure 7. The pseudo-code of RRT-Connect is shown in Algorithm 1.

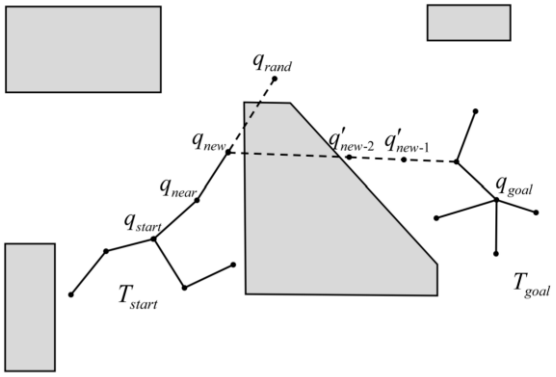


Figure 7: Node expansion of RRT-Connect algorithm [31].

Algorithm 1. RRT-Connect

1. $T_{start} \leftarrow q_{start}$; $T_{goal} \leftarrow q_{goal}$
2. **while** t < T **do**
3. $q_{rand} \leftarrow Sample()$; $q_{near} \leftarrow Nearest(T_{start}, q_{rand})$
4. $q_{new} \leftarrow Steer(q_{rand}, q_{near}, d_m)$
5. **if not** $Obstacle(q_{new})$ **then**
6. $T_{start} \leftarrow q_{new}$; $q'_{near} \leftarrow Nearest(T_{goal}, q_{new})$
7. $q'_{new} \leftarrow Steer(q_{new}, q'_{near}, d_m)$
8. **if not** $Obstacle(q'_{new})$ **then**
9. $T_{goal} \leftarrow q'_{new}$
10. **if** $Connect(q_{new}, q'_{new}, d_m)$ **then**
11. $T_{path} \leftarrow Node(T_{start}, T_{goal})$
12. **return** T_{path}
13. **end if**
14. **else**
15. $Swap(T_{start}, T_{goal})$
16. **end if**
17. **else**
18. $Swap(T_{start}, T_{goal})$
19. **end if**
20. **end while**

Within the exploring period T, the RRT-Connect algorithm starts by expanding the start tree T_{start}. A random point q_{rand} is first created in the workspace. Then, a tree node q_{near} among the start tree with the shortest distance from q_{rand} is selected, and a new dissociative node q_{new} is gained to represent the extension

from q_{near} and q_{rand} in a step size d_m. If q_{new} was not overlapped with obstacles, it would be added into the start tree T_{start} as a tree node. Subsequently, a similar procedure repeats to steer the goal tree T_{goal}. The only difference is that T_{goal} expands towards the tree node q_{new} instead of the random point q_{rand} (the strategy that bi-directional RRT adopted). During each round of extension, the distance between the new tree nodes in the start tree (q_{new}) and goal tree (q'_{new}) will be calculated at the final step. If the gap was shorter than a step size d_m, two trees were connected. Hence, the completed path T_{path} is determined by linking the tree nodes among T_{start} and T_{goal} according to the extension sequence. Furthermore, RRT-Connect would swap the roles of T_{start} and T_{goal} if q_{new} or q'_{new} overlapped with the obstacles. Compared to other double tree RRT methods, like the bi-directional RRT method, this approach avoids random searches toward collision-free directions, thereby expanding the search area and determining a more optimal trajectory.

Although the RRT-Connect method is an appropriate method to generate a trajectory, it possesses a common disadvantage of most RRT-involved methods: they easily explore irrelevant spaces of the task (lack of goal-directionality). For example, the entire workspace of the robot arm could be large, but in most cases, both the initial position and the target point are distributed in a limited area of the workspace, which causes other areas to become "useless". Therefore, several researchers developed informed RRT methods to limit the search in a useful space. RRT-GD is one of those methods. This method creates a 'sample space' that encompasses the target point and extends the random true towards the random point inside the sample space. This method dramatically reduces the path length and the search period compared to the original RRT method (the time ratio can reach nearly 1:300 in some cases). Inspired by the RRT-GD, our improved RRT-Connect method replaces the random sampling function with the 'sample area' to obtain a better trajectory, as shown in Figure 8 (a). In this case, the random point, generated as the direction for extension, is limited in the green 'sample area', which gives goal-bias direction. Algorithm 2 shows our sampling function corresponding to Sample() in Algorithm 1 Line 3. The if function in Algorithm 2 checks if the trees are swapped.

Nevertheless, it is apparent that only one sample area is in our algorithm, which means if T_{start} met an obstacle, and the two trees were swapped, the T_{goal} would simply extend towards a true random direction. The reason of this design is to avoid a death-end situation as shown in Figure 8 (b). In this situation, when two trees grow too close to the obstacle, the goal-bias extension function leads both trees to collide with the obstacles. Different from the original RRT method, which can avoid the collision by directly growing the tree in a free direction, RRT-Connect can only swap trees to find an available path, thus, it will drop into a death end and continuously swap without any new nodes be generated.

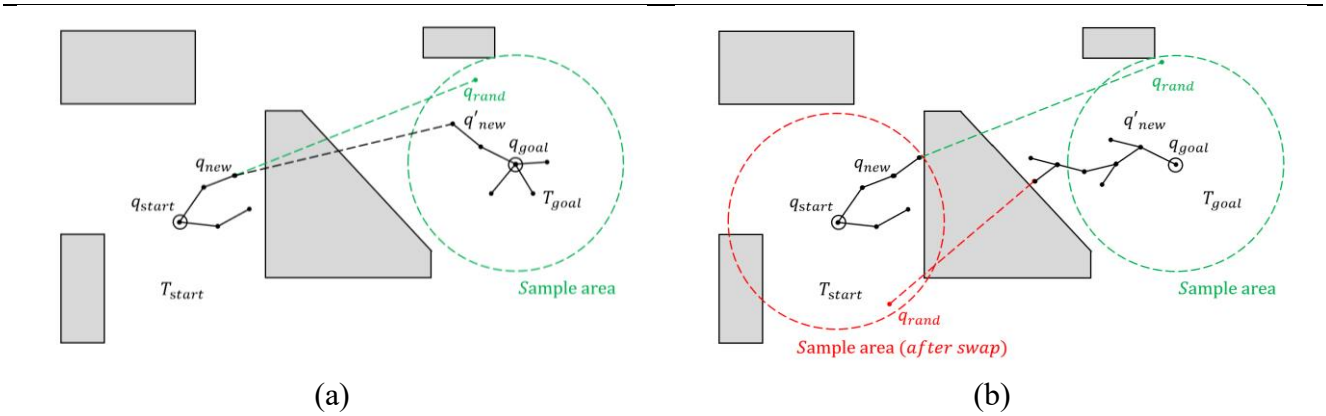


Figure 8: Node expansion of the goal-directional RRT method. (a) Only one sample area for expansion. (b) Two sample areas surround the start point and the goal point, respectively (death-end case).

Algorithm 2. Sampling Function

if q_{start} **is in** T_{start}
  $q_{rand} \leftarrow Sample(A_{goal})$
else
  $q_{rand} \leftarrow Sample()$
end if

Except for planning the path of the end-effector, the nodes on the path need to be recorded. In our algorithm, two nodes from T_{start} and T_{goal} are generated at one iteration and each node has three parameters: parent, pose and link information. The parent represents the previous node that extends to it. The function Node () could attain a completed path by iteratively collecting the parent node started from the connected node of T_{start} and T_{goal}. The pose contains sample value in joint space which represents the angle of each joint:

$$Pose = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7\}$$

In the process of generating the path, the pose of the start point is the initial angle of the robot, while the pose of the endpoint is determined by the numerical inverse kinematics solver IKPY [40] which converts the target point in Cartesian space into angles in joint space. After obtaining the start pose and the goal pose, another node could be created by trajectory planning. Finally, the link information is a group of data which represents the length, centre and radius of each link. With the link information, we applied the common library FCL (Flexible Collision Library) [41] to test if any link has a collision with obstacles.

RRT-based methods have excellent capability in global searching, but the paths they generate always face terrible smoothness. A typical method is using smoothing post-processing to curve the zigzag path. In our project, we employ Bezier curve which is the most used curve approximation function, to curve the path in joint space.

2) Trajectory Tracing

In order to support the worker to move the tube, it is necessary for the robot to trace the carrying hand to keep balance in the horizontal plane. Supposing that the human facing direction is the x-axis, it means that the position bias between the carrying hand and the end-effector of the robot should be as small as possible. Therefore, a trajectory tracing strategy is adopted in this case.

Sampling the motion path of the hand is the first step of tracing. The sampling result is a bunch of points in Cartesian space. After that, the trajectory of the robot could be determined. As mentioned before, if the facing direction was the x-axis, the gripper should reach the corresponding position which has no bias in y and z axes. Hence, the robot trajectory consists of corresponding points concerning the sampling result. The distance in x-axis, which is the gap between the gripper and hand, is determined as 1 meter.

Since the trajectory is gained, the next step is to iteratively compute the robot pose for each point. As mentioned earlier, the inverse kinematics solver used in this project is IKPY. This library has an excellent reputation in some papers due to its high generality and accuracy. In detail, its function 'inverse kinematics frame' can numerically compute the most appropriate pose of the robot with three inputs: initial angles of joints, goal position in Cartesian space and goal orientation of the end-effector in the matrix. In our scenario, by treating each point in the trajectory as the goal positions and orientations, the initial angles of each point could be determined and utilized by IKPY iteratively. Furthermore, since the robot should turn its end-

effector from a ‘grasping pose’ (z-axis of the end-effector is oriented downward) to a ‘hold pose’ (z-axis of the end-effector is oriented upward), the goal orientation at each point requires carefully determining to generate correct matrix. This is achieved by using Spherical Linear Interpolation (Slerp) from the scipy library, which interpolate smoothly between start and goal orientation.

3.3 Controller

Position/force control is one of the most straightforward control theories in the low-level control of robots. Since our project has no motion datasets for developing complex control strategies, and the advanced avatar is still in designing, position control is an appropriate method to manage the locomotion of the robot arms. In this case, we adopt a cascade PID controller [42] which utilizing the position error and theoretical instantaneous speed to generate the control signal.

The structure of our cascade controller is shown in Figure 9. Each joint in the robots has a sensor to monitor its rotation angle as a real position (P_{real}). It is the feedback signal for the position loop. However, since the simulation engine does not provide the velocity sensor, the feedback signal for the velocity loop is calculated by function e(x) as shown in Equation 1. It means that once the position value is returned from the sensor, a real angular velocity will be calculated as the feedback, where Δt is the unit timestep in the environment and $P_{real}(t-1)$ is the recorded position feedback in the last time. Then, the current real position feedback will be saved as a last value for the next computation. Simultaneously, a desired position value ($P_{desired}$) is determined by trajectory-generating functions (chapter 3.2) which denotes the expected angle each joint should achieve. Then, as shown in Equation 2, the bias between the expected angle and real angle is determined. Later, the position error serves as the input to the position PID controller, with its output being passed as the input signal (V_{error}) to the velocity PID controller, as described by Equation 3.

$$V_{real}(t) = (P_{real}(t) - P_{real}(t-1))/\Delta t \quad \text{Equation 1}$$

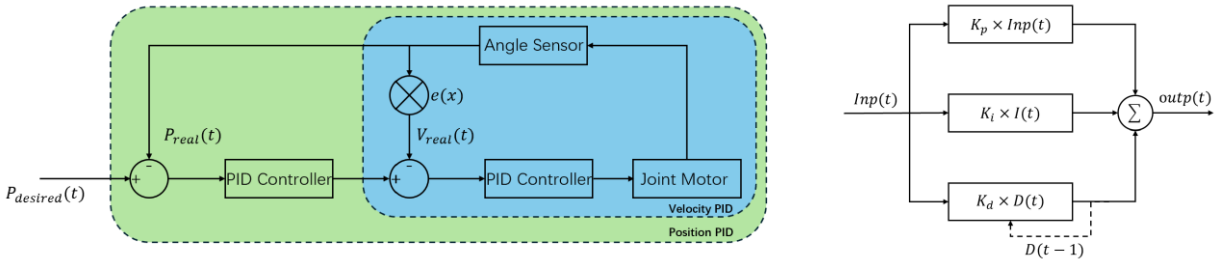
$$P_{error} = P_{desired}(t) - P_{real}(t) \quad \text{Equation 2}$$

$$V_{error} = P_{output} - V_{real}(t) \quad \text{Equation 3}$$

The structure of the PID controller is shown in Figure 9 (b). K_p, K_i and K_d are proportional, integral and derivative coefficients, respectively. Since the simulation in the virtual environment is discrete, the integral parameter I(t) is calculated by summing all input signal times the unit timestep, as shown in Equation 4. To enhance the smoothness of the derivative term D(t), we implement a low-pass filter (Equation 5) rather than calculating D(t) directly. The parameter K_f is the filter coefficient. This approach smooths the differential of the input signal, thereby reducing the impact of noise on the controller.

$$I(t) = \sum_{T=0}^{t} Inp(T) \cdot \Delta t \quad \text{Equation 4}$$

$$D(t) = \frac{K_f \times (Inp(t) - Inp(t-1)) + D(t-1)}{K_f \times \Delta t + 1} \quad \text{Equation 5}$$



(a) (b)

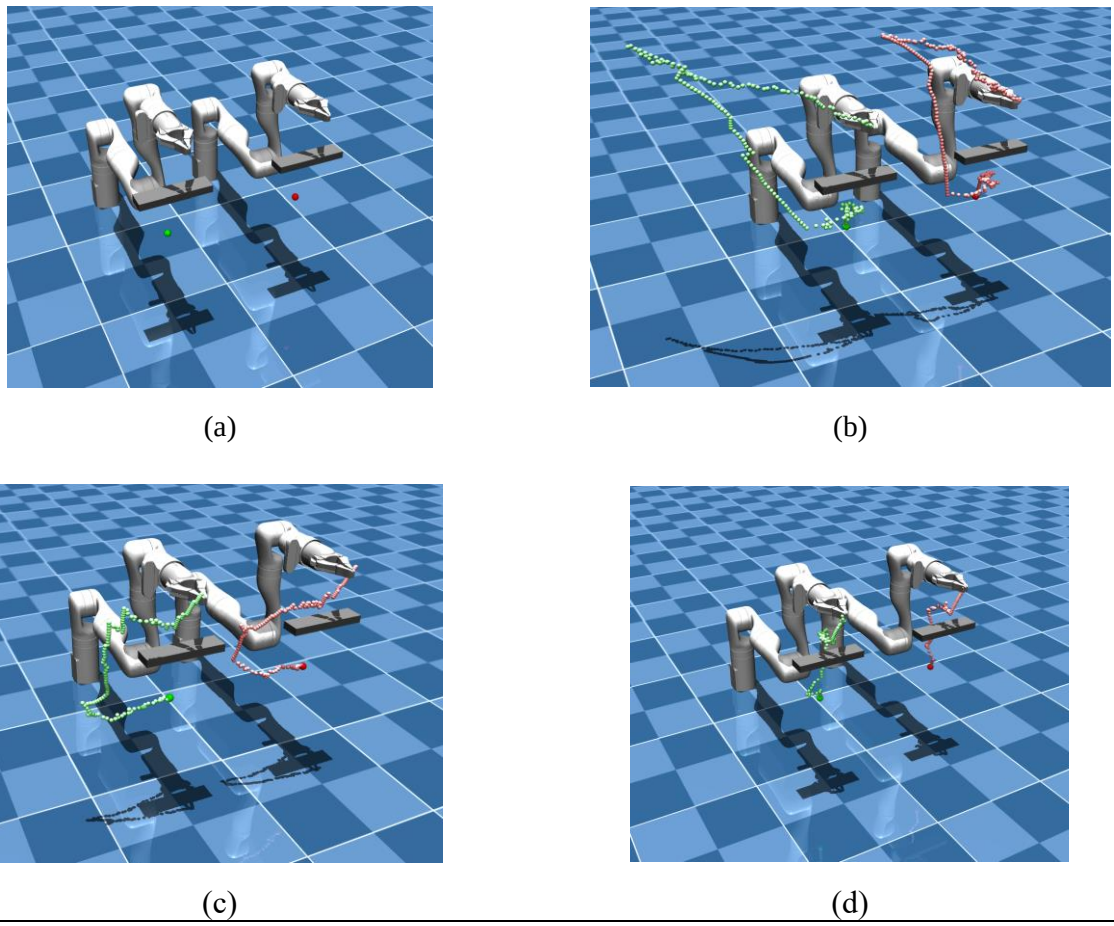
Figure 9: (a) The architecture of cascade PID controller. (b) The structure of PID controller

4. Results

SRLs are built based on two independent industrial robots 7-Dof Kinova Gen3 with end-effectors. The end-effector in this case is a L31K_Gripper. The base of SRLs is then mounted on the back of the humanoid via a package. Simulations were run on a single core of a standard laptop CPU (Intel Core i5, 1.10 GHz) through Windows S11. The virtual environment was based on Mujoco 3.1.5 and Python 3.10. All the Python algorithms, graphs and the demo video of the whole working scenario are uploaded to GitHub at https://github.com/CROBOT974/Development-of-Trajectory-Based-Control-of-Supernumerary-Robotic-Limbs-for-Assembling-Tasks.

4.1 Motion Strategies

In order to demonstrate the merit of the goal-directional RRT-Connect method. Hybrid RRT, RRT-Connect and our method are tested in a conventional trajectory planning task to compare their performance. As depicted in Figure 10, the dual arms need to cross the grey obstacles to reach the target. Figure 10 (a) illustrates that the red point is the target for the main arm, whilst the green point is the target for the secondary arm. Table 2 lists the average length, time spent and success rate on each method.



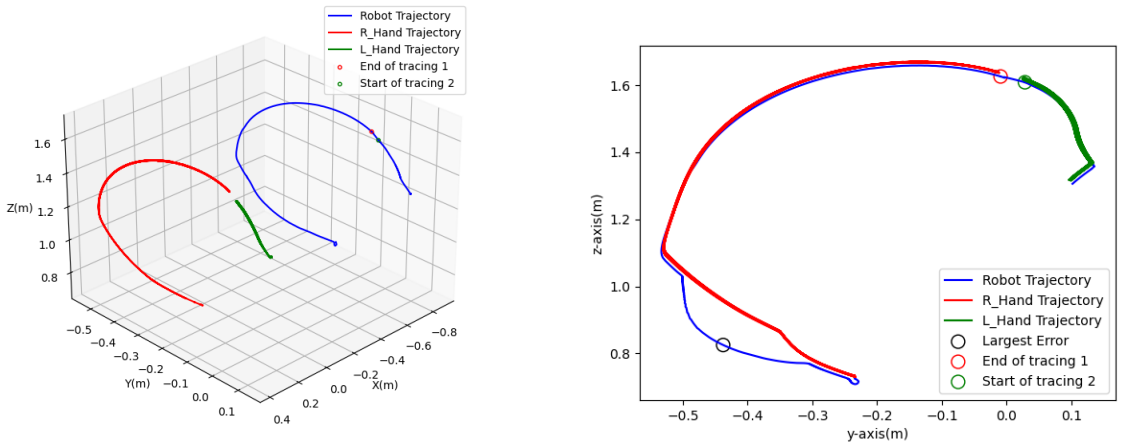
(a) (b) (c) (d)

Figure 10: Motion planning result comparison. (a) Environment for comparing the trajectory. (b) Hybrid RRT method. (c) RRT-Connect method. (d) goal-directional RRT-Connect method.

Methods	Average Time Spent (s)	Average Path Length (nodes)		The success rate in the 2000 iteration
		M_arm	S_arm	
Hybrid RRT	20.22	220	200	39.1%
RRT-Connect	9.57	133	143	100%
Our method	5.95	95	93	100%

Table 2: The average calculation time, path length and success rate of three RRT-based methods.

In order to detect the precision of the trajectory tracing function, the trajectory of the robot end-effector is compared with the path of the human hand in the z and y axes (the x-axis is the human-facing direction), as shown in Figure 11. The largest error in the y-z plane is 0.092 m.

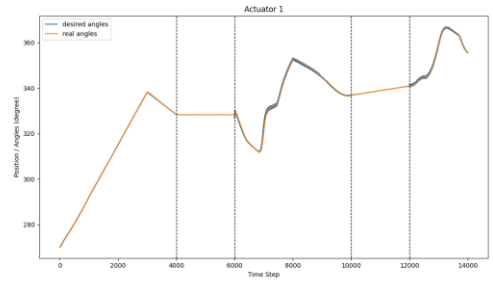


(a) (b)

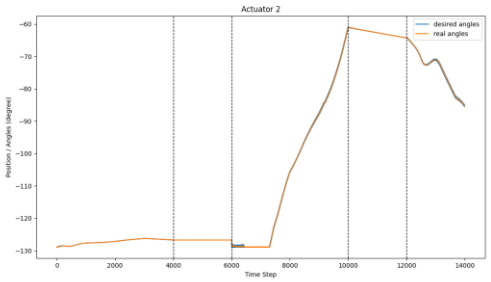
Figure 11: The trajectory of the robot end-effector compared with the trajectory of the worker's hand. (a) Trajectory in 3D space. (b) Trajectory in y-z plane

4.2 Precision of Controller

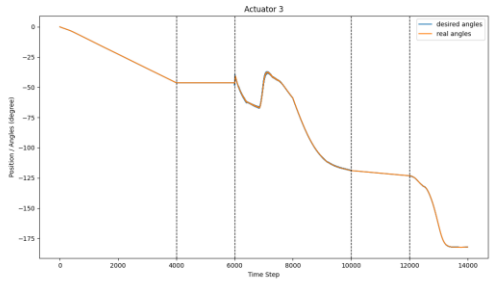
The precision of the position controller affects the overall performance of SRLs. To illustrate its precision, the angle movement of each joint in the right arm (main arm) is collected by position sensors and compared with the desired angles generated by trajectory planning and tracking. The result is depicted in Figure 12. The black dashed line separates the process into five parts, which correspond to stages 2 - 6. As the image shows, the position PID controller successfully drives the robot through the desired paths.



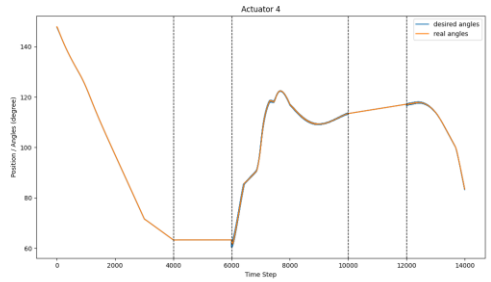
(a)



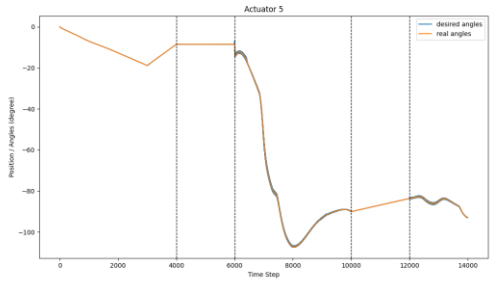
(b)



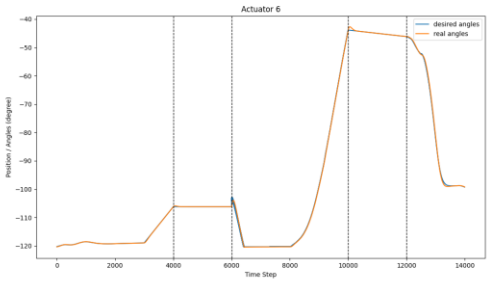
(c)



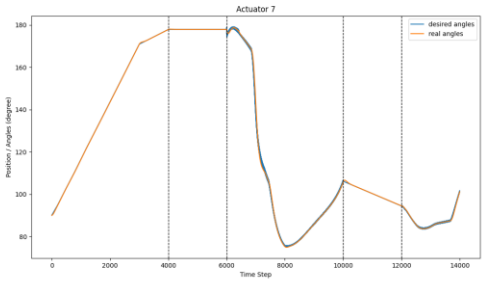
(d)



(e)



(f)



(g)

Figure 12: The angle movements of each joint of the main arm compared with the desired joint during the trajectory planning and tracing period.

5. Discussion & Conclusion

In this project, an effective and autonomous SRLs control method cooperating with a humanoid was proposed for a complex scenario. This paper explained the controller and control strategies for manipulating the robot arms and demonstrated their performance.

For the trajectory planning function, our method had best performance compared to the hybrid RRT and original RRT-Connect method. The reason that hybrid RRT took such a long period is it did not employ the swap function of the RRT-Connect method. This disadvantage brought about a much narrower search area than the RRT-Connect method. Therefore, it was less possible to meet a connection between two trees than the RRT-Connect method and was more easily to fail in a limited iteration. Compared to the original RRT-Connect method, our strategy had the merit of RRT-GD which leads the tree to approach the goal faster, while it did not lose the wideness of the tree like hybrid RRT.

The trajectory tracing has a large mismatch at the beginning because it met boundary issues. According to the angle limitations shown in Table 1 and Figure 12, joints 2 and 6 reached their lowest angle boundary. The numerical inverse kinematics function (IKPY) we used has a poor ability to handle such extreme situations, as it can only update the pose to approach the target in a given number of iterations, and probability fails to reach closely enough to the target when joints met limitation. For redundant robot arms, self-motion and smarter trajectory-generating functions may help to trade off the angles to avoid the boundary, but it requires an analytical inverse kinematics function.

Overall, for controlling SRLs, several key advantages were found in this project. Firstly, it builds a unique information communication channel between the worker and SRLs to deal with the task knowledge. The autonomous behaviour of the robot is purely based on the condition of the worker, which removes the requirement of active command or prerequisite of trail data. This design can effectively decrease the cognitive load of the worker, as none of the robot motion needs extra leading from the user. Secondly, the high level of autonomy ensures that, theoretically, this control model is able to approach the task with various task parameters, like the different initial or installing positions of the tube.

However, the project also has several limitations. One significant limitation is the reliance on position control rather than a more adaptive approach, such as impedance or admittance control, during the stage of moving the tube. Position control, while providing precise trajectory tracking, lacks the necessary compliance to effectively accommodate external forces and human interactions. This rigid approach may result in reduced efficiency and safety during cooperative tasks, as the robot cannot dynamically adjust its movements in response to unanticipated forces or changes in the environment. The other limitation is the lack of generalization. Similar as most research in this field, the task model is specifically designed and may not be compatible to other applications. Because the complexity of the task and working environment restrict the possibility of developing a unified control framework for SRLs with traditional control strategies.

Future work is to design an analytical inverse kinematics function in Python for the KINOVA Gen3 robot. After that, we can develop an advanced trajectory tracing method by considering the self-motion of the redundant robot arm to eliminate the boundary issue we met in our project. Furthermore, we can develop a compliant controller like admittance control to increase efficiency and safety during the moving of the tube. Then, it is possible to transfer and validate the control model to our real-world prototype. Real-world practice presents additional challenges such as sensor noise, system delays, and unpredictable external forces. Hence, successful deployment in real environments will not only confirm the model's feasibility but also allow for further optimization based on real-world feedback.

In conclusion, this project produces a complete robotic control strategy for installing the tube on the wall within a human-SRLs collaborative working scenario. The control framework integrates an advanced

trajectory planning method with an effective tracking approach, enabling the system to achieve rapid and precise motions through position control. Despite its effectiveness, the approach relies heavily on rigid control, which may limit flexibility in dynamic environments, thus, a further improvement of admittance control is considered. Detailed algorithmic explanations and a demonstration video are available on GitHub, offering valuable resources for further research.

6. References

1. J. Eden et al., “Principles of human movement augmentation and the challenges in making it a reality,” Nature Communications, vol. 13, no. 1, Mar. 2022.
2. M. A. Sada, M. Khamis, A. Kato, S. Sugano, T. Nakajima, and F. Alt, “Challenges and opportunities of supernumerary robotic limbs,” in Proc. CHI 2017 Workshop on Amplification and Augmentation of Human Perception, Denver, CO, USA, May 7, 2017.
3. S. -w. Leigh, H. Agrawal and P. Maes, "Robotic Symbionts: Interweaving Human and Machine Actions," in IEEE Pervasive Computing, vol. 17, no. 2, pp. 34-43, Apr.-Jun. 2018.
4. B. Yang, J. Huang, X. Chen, C. Xiong, and Y. Hasegawa, “Supernumerary robotic limbs: A review and future outlook,” IEEE Transactions on Medical Robotics and Bionics, vol. 3, no. 3, pp. 623–639, Aug. 2021.
5. B. Llorens - Bonilla, F. Parietti, and H. H. Asada, “Demonstration-based control of supernumerary robotic limbs,” 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2012.
6. B. L. Bonilla and H. H. Asada, “A robot on the shoulder: Coordinated human-wearable robot control using coloured petri nets and partial least squares predictions,” 2014 IEEE International Conference on Robotics and Automation (ICRA), May 2014.
7. A. S. Ciullo, M. G. Catalano, A. Bicchi, and A. Ajoudani, “A supernumerary soft robotic limb for reducing hand-arm vibration syndromes risks,” Frontiers in Robotics and AI, vol. 8, Aug. 2021.
8. Liang X, Hong K Y, Guo J, et al 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO) Macau Macao IEEE.
9. F. Parietti, K. C. Chan, B. Hunter, and H. H. Asada, “Design and control of supernumerary robotic limbs for balance augmentation,” 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 5010–5017, May 2015.
10. J. Guggenheim, R. Hoffman, H. Song, and H. H. Asada, “Leveraging the human operator in the design and control of supernumerary robotic limbs,” IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 2177–2184, Apr. 2020.
11. Tuvie, “Jizai Arms Supernumerary Robotic Limb System with Exchangeable Arms,” Jizai Arms. [Online]. Available: https://www.tuvie.com/jizai-arms-supernumerary-robotic-limb-system-with-exchangeable-arms/ (accessed Aug. 27, 2024).
12. I. Hussain, L. Meli, C. Pacchierotti, G. Salvietti, and D. Prattichizzo, “Vibrotactile haptic feedback for intuitive control of robotic extra fingers,” *2015 IEEE World Haptics Conference (WHC)*, pp. 394–399, June 2015.
13. F. Parietti and H. H. Asada, "Independent, voluntary control of extra robotic limbs," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, pp. 5954-5961, 2017.
14. I. Hussain, G. Spagnoletti, G. Salvietti, and D. Prattichizzo, “Toward wearable supernumerary robotic fingers to compensate missing grasping abilities in hemiparetic upper limb,” *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1414–1436, 2017.
15. N. S. Meraz, H. Shikida and Y. Hasegawa, "Auricularis muscles based control interface for robotic extra thumb," *2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, Nagoya, Japan, pp. 1-3, 2017.
16. F. Wu and H. Asada, “Bio-Artificial Synergies for Grasp Posture Control of Supernumerary Robotic Fingers”, 2014.
17. Y. Kondo, K. Takemura, J. Takamatsu, and T. Ogasawara, “Smooth humanrobot interaction by interruptible gesture planning,” *Proceedings 2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 213–218, July 2010.
18. B. Llorens-Bonilla and H. H. Asada, “Control and coordination of supernumerary robotic limbs based

on human motion detection and Task Petri Net Model," Volume 2: Control, Monitoring, and Energy Harvesting of Vibratory Systems etc. , Oct. 2013.
19. Z. Tu, Y. Fang, Y. Leng and C. Fu, "Task-Based Human-Robot Collaboration Control of Supernumerary Robotic Limbs for Overhead Tasks," in *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4505-4512, Aug 2023.
20. Abbas, M., Narayan, J. & Dwivedy, S.K. "A systematic review on cooperative dual-arm manipulators: modeling, planning, control, and vision strategies," *Int J Intell Robot Appl* **7**, pp. 683–707, 2023.
21. Peng Song and V. Kumar, "A potential field based approach to multi-robot manipulation," *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, Washington, DC, USA, vol.2, pp. 1217-1222, 2002.
22. Byrne, Steven et al. "Improved APF strategies for dual-arm local motion planning." *Transactions of the Institute of Measurement and Control*, vol. 37, pp. 73 – 90, 2015.
23. SM Lavalle, JJ Kuffner, "Rapidly-Exploring Random Trees: Progress and Prospects," Algorithmic & Computational Robotics New Directions, pp. 293–308, 2000.
24. Dong-Hyung Kim, Sung-Jin Lim, Duck-Hyun Lee, Ji Yeong Lee and Chang-Soo Han, "A RRT-based motion planning of dual-arm robot for (Dis)assembly tasks," *IEEE ISR 2013*, Seoul, Korea (South), pp. 1-6, 2013.
25. Wei, K.; Ren, B. "A Method on Dynamic Path Planning for Robotic Manipulator Autonomous Obstacle Avoidance Based on an Improved RRT Algorithm". *Sensors* 18, no. 2, pp. 571, 2018.
26. Noreen, Iram et al. "A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms," Int. J. Comput. Sci. Netw. Secur. IJCSNS, 2016.
27. Li, Q. et al. "Path Planning of Manipulator Based on Improved Informed-RRT* Algorithm," In Proceedings of the Intelligent Equipment, Robots, and Vehicles: 7th International Conference on Life System Modeling and Simulation, LSMS 2021 and 7th International Conference on Intelligent Computing for Sustainable Energy and Environment, ICSEE 2021, Hangzhou, China, 22–24 October 2021; Proceedings, Part III 7. Springer: Singapore, 2021; pp. 501–510.
28. Yi, J., Yuan, Q., Sun, R. et al. "Path planning of a manipulator based on an improved P_RRT* algorithm," *Complex Intell. Syst*, vol. 8, pp. 2227–2245, 2022.
29. W. Shi, et al. "Obstacle Avoidance Path Planning for the Dual-Arm Robot Based on an Improved RRT Algorithm," *Applied Sciences*, vol. 12, no. 8, pp. 4087,2022.
30. J. Shao, Y. Gan and X. Dai, "Autonomous Path Planning and Realization for Dual Robot Cooperation Based on ROS Framework," In Proceedings of the 2023 International Conference on Advanced Robotics and Mechatronics (ICARM), Sanya, China, pp. 1065–1070, 8–10 July 2023.
31. X. Chen, X. You, J. Jiang, J. Ye, and H. Wu, "Trajectory Planning of Dual-Robot Cooperative Assembly," Machines, vol. 10, no. 8, pp. 689–689, Aug. 2022.
32. A. Volz and K. Graichen, "An Optimization-Based Approach to Dual-Arm Motion Planning with Closed Kinematics," Oct. 2018.
33. L. Larsen and J. Kim, "Path planning of cooperating industrial robots using evolutionary algorithms," Robotics and Computer-Integrated Manufacturing, vol. 67, p. 102053, Feb. 2021.
34. P. Cao, Y. Gan, J. Duan, and X. Dai, "Time-optimal path tracking for coordinated dual-robot system using sequential convex programming," Jun. 2016.
35. J. Liang, Z. Xu, X. Zhou, S. Li, and G. Ye, "Recurrent Neural Networks-Based Collision-Free Motion Planning for Dual Manipulators Under Multiple Constraints," IEEE Access, vol. 8, pp. 54225–54236, Jan. 2020.
36. E. Prianto, M. Kim, J.-H. Park, J.-H. Bae, and J.-S. Kim, "Path Planning for Multi-Arm Manipulators Using Deep Reinforcement Learning: Soft Actor–Critic with Hindsight Experience Replay," Sensors, vol. 20, no. 20, p. 5911, Oct. 2020.
37. Kinova inc, "Gen3 user guider," KINOVA GEN3 ROBOT. [Online]. Available:

https://www.kinovarobotics.com/product/gen3-robots (accessed: Aug. 28, 2024).
38. J. Ge, F. Sun, and C. Liu, “RRT-GD: An efficient rapidly-exploring random tree approach with goal directionality for redundant manipulator path planning,” Dec. 2016.
39. FrankJIE09, “GitHub - FrankJIE09 / Dual_Robot_Motion_Planning_RRT_Connect: The algorithm is designed to plan the motions of two robots simultaneously, ensuring they do not collide with each other or with obstacles.,” GitHub, 2024. https://github.com/FrankJIE09/Dual_Robot_Motion_Planning_RRT_Connect (accessed Sep. 01, 2024).
40. P. Manceron, “IKPy,” GitHub, Mar. 28, 2024. https://github.com/Phylliade/ikpy (accessed Sep. 01, 2024).
41. BerkeleyAutomation, “GitHub - BerkeleyAutomation/python-fcl: Python binding of FCL library,” GitHub, Jul. 11, 2024. https://github.com/BerkeleyAutomation/python-fcl (accessed Sep. 01, 2024).
42. M. Li, Z. Liu, M. Wang, G. Pang, and H. Zhang, “Design of a Parallel Quadruped Robot Based on a Novel Intelligent Control System,” Applied Sciences, vol. 12, no. 9, pp. 4358–4358, Apr. 2022.