Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 5240–5247

Multi-arm motion-generating method based on each robot movement vector in time slice

N. Hayashi*, N. Nakasu**

* Hitachi, Ltd., 292, Yoshida-cho, Totsuka-ku, Yokohama-shi, Kanagawa-ken, Japan (naohiro.hayashi.tg@hitachi.com).

** Hitachi, Ltd., 292, Yoshida-cho, Totsuka-ku, Yokohama-shi, Kanagawa-ken, Japan (nobuaki.nakasu.uy@hitachi.com).

Abstract: This paper proposes a multi-arm-motion-generation method. Arm robots are expected to substitute workers, especially since multi-arm flexibly is adaptable to diverse product models. However, due to the complexity of the multi-arm motion, many hours of manual motion teaching are required. Therefore, we developed a multi-arm motion-generating method for automatically generating collision-free multi-arm motions with an equal or faster time than manually teaching motion. Our proposed method is a collision-circumvent and stop-avoidance method that takes into account each robot movement vector in time slices. Experiments were conducted to evaluate the proposed method, and the results indicate that the method can generate a short circumvent avoidance trajectory compared with a conventional collision-avoidance method for avoiding all passing areas of robots.

Keywords: Industrial robots, motion planning, robot cell, path planning, multi-arm, assembly

1. INTRODUCTION

Product life cycles have become shorter, and mixed-flow production has increased due to fluctuating demand. Therefore, it is necessary to rapidly change production-line capacity to cope with such fluctuations. However, due to the shortage in manufacturing workers and increasing labor costs, it has become difficult to increase the number of workers to cope with capacity changes or maintain the current capacity. There is an increasing demand for robots to replace human workers. Arm robots are particularly useful because they have a high degree of freedom of movement, so they can replace complex human operations such as assembly work. Compared with a single arm robot, multiple single-arm robots require less supplemental tasks, such as inter-robot workpiece transfer and hand-tool exchange, and reduce operation time due to the parallelization of tasks. Because of their versatility and capability to execute complex operation tasks, using multiple single-arm robots together is attracting attention as an alternative resource to humans, as in the case of verifying aircraft assembly in the study by Solana *et al*. Due to the complexity of multi-arm movements, many hours of manual motion teaching are required than for single-arm robots. To execute the tasks assigned to a single-arm robot, the robot-motion teacher only needs to teach movements that do not collide with environmental objects, such as a table. However, there are two issues regarding motion teaching for multiple single-arm robots. It is first necessary to plan which arms will be assigned to each task and in what order the arms will be moved sequentially. The second issue is preventing collision with environmental objects as well as with other arms. These issues increase the hours of manual motion teaching and make it difficult to start up the line or change the line configuration.

The first issue is addressed through designing an optimal plan for the task allocation and motion sequence of each arm from many combinations. Task allocation and sequence are not uniquely determined. For example, when fastening a part with several screws, there is a pattern of combinations as to which arm tightens the screws through which screw holes. Thus, it is necessary to select and plan sequences that minimize operation time. For automating this motion sequence design, Wang *et al*. proposed a parts-allocation and assembly-sequence design method for assembly work with multiple single-arm robots, and Touzani *et al*. proposed a welding-point-allocation and welding-sequence-design method for welding work with multiple single-arm robots.

The second issue is addressed through generating multi-arm motion that prevents collisions with the other arms. It is not enough to simply move the hand from the starting position to the target position. It is necessary to make movements to prevent collisions with environmental objects, such as tables, or with the other arms. This increases the hours of manual motion teaching. To address this, we propose a method of automatically generating multi-arm collision-avoidance motion with a shorter motion time than manually created motion.

The rest of this paper is organized as follows. In Section 2, we give an overview of related research. In Section 3, we present our proposed method. In Section 4, we present the evaluation of the proposed method. Finally, we conclude the paper and mention future work in Section 5.

2. RELATED RESEARCH

2.1 Review stage

This section discusses related research on trajectory-planning methods for generating the arm movements mentioned in the previous section. Methods for automatically generating trajectories between the current and target positions have become common. In tasks such as pick and place, which is a typical task performed by a single-arm robot, the robot needs

10.1016/j.ifacol.2023.10.122

to move its hand to the target position of the object. To perform this task, the trajectory-planning function generates a feasible trajectory between the current and target positions. It is important to find a trajectory that avoids collisions with objects in the environment. This function also optimizes the generated trajectory to the smoothest and shortest trajectory. The robot motion that executes an actual task consists of several trajectories. Therefore, a user inputs each hand-target position and posture to this trajectory-planning function in sequence. The function then generates the robot motion by calculating each corresponding trajectory.

Typical trajectory-planning methods have been proposed for avoiding collisions with objects in the environment, i.e., graph search methods in discretized three-dimensional (3D) space (e.g., the Dijkstra method by Dijkstra *et al*. and A* method by Hart *et al*.), random sampling methods in 3D space or joint angle space (e.g., PRM by Kavraki *et al*. RRT by lavalle, and RRT-CONNECT by Kuner *et al*.), optimization methods of the initial trajectory by connecting the initial position and target position with a straight line (CHOMP by Ratliff *et al*,, STOMP by Kalakrishnan *et al*., and TRAJOPT by Schulman *et al*.). Within the field of trajectory planning, methods have been proposed for generating avoidable trajectories for dynamic obstacles, which are objects moving in the environment. With multiple single-arm robots, each arm can be treated as a dynamic interferer, so such methods can be applied to avoid dynamic obstacles for multiple single-arm robots. There are two main avoidance-trajectory-planning methods for dynamic obstacles. The first is a method with which the trajectory of a dynamic obstacle is unknown and the trajectory is re-planned sequentially while the robot is in motion. The second is a non-re-planning method with which the trajectory of the dynamic obstacle is given and the trajectory is generated without re-planning. With the re-planning method, the trajectory to the target is first generated on the basis of the static and dynamic environmental objects observed at the initiation. During actual operation, the environmental conditions are then monitored using external sensors and collisions are continuously predicted. If a collision is predicted, the robot re-plans its trajectory from its current position and keeps moving on that trajectory. Bruce *et al*. proposed a computationally efficient RRT-based trajectory re-planning method for mobile robots, and Yoshida *et al*. proposed an RRT-based re-planning method for arm robots. Sakahara *et al*. proposed a trajectory-planning method for mobile robots that uses RRT to search a 3D space with x-, y-, and time axes in 2D space. The advantage of this re-planning method is that even if an unknown moving object such as a worker appears, collision avoidance can be executed in real time by detecting it with an external sensor and re-planning the trajectory. Therefore, it is effective for cooperative work, such as when a human and robot work together in the same workspace. The disadvantage is that the method executes re-planning before the actual collision. Therefore, a sudden change in move direction may cause deceleration or a pause due to the lack of calculation time for re-planning, which slows the robot's operation.

Next, we discuss the non-replanning method. The non-replanning method is a trajectory-planning method that extends the search space in the time dimension, as proposed by Sintov *et al*. This method searches a space including the time axis. Therefore, it can only solve problems with a small number of dimensions and is difficult to apply to arm robots with multi-axis joints and a high-dimensional search space.

One method to ensure collision avoidance is to generate a trajectory that avoids all passing areas of dynamic obstacles. Alternatively, there is a method of pausing before the passing area until the dynamic obstacle has passed through the area. This method is useful for manual teaching. However, it is slow operation time because the trajectory takes a long path to avoid all the passing areas. The pausing time also becomes longer. When automatically generating trajectories by this method, trajectories are determined that cannot be generated in situations where the passing area overlaps the robot's target position. In such a situation, the robot can move to the target position if it pauses in front of the target position before the other robots leave there.

As discussed above, conventional methods have been proposed for generating avoidable trajectories for static as well as dynamic obstacles. However, when these methods are applied to multiple single-arm robots , the timing of the multi-arm operation will be limited. If sequential re-planning methods are used, deceleration and pauses occur during re-planning, resulting in slower motion. Non-re-planning methods are computationally infeasible for multiple single-arm robots, which have a high-dimensional search space. The non-re-planning method for avoiding all passing area of dynamic obstacles slows motion due to long detour trajectories, and there are situations in which trajectory generation is not possible. From the technical limitations of each of these conventional methods, it was found that circumvent avoidance increases operation time due to deceleration and long detour trajectories. It was considered that a collision-stop-avoidance method is necessary in situations where a circumvent-avoidance method is not suitable.

Therefore, the proposed method addresses the issues of slow circumvent avoidance and situations in which circumvent is not suitable by taking into account collision avoidance for each robot at each time-sliced sampling time.

3. MULTI-ARM-MOTION GENERATION METHOD

3.1 Approach

When the conventional methods described in the previous section are applied to generate multi-arm movements, there are situations in which the movement time increases due to long circumvent avoidance or movements cannot be generated. We take the new viewpoint of considering collision avoidance of each arm in the time slice. The proposed method combines circumventing and stopping in the time-slice space. We hypothesized that the proposed method prevents circumventing from going too far to generate motion even in situations where circumventing is not possible. This method does not follow the policy of re-planning methods or the method of avoiding all passing areas of robots. The proposed method generates a trajectory that shortens the operation time by determining positions that avoid collision or block each robot's movement direction at each time slice. Thus, the

proposed method generates trajectories not in the joint space but by searching in a uniformly discretized graph of 3D space, as shown in Fig. 1, which can easily represent the physical movement direction of the robot. When generating trajectories, the process of searching for trajectories in a 3D discretized graph is complex. This is because the proposed method determines the position of the other robot by the path of the graph and the time at which the hand reaches the grid point when searching within the graph grid from the start position to the target position of the hand. Therefore, the current time and positions of other robots must be recalculated sequentially while moving through graph space. When a robot detects a possible collision with another robot at a certain position and time, the robot must avoid blocking the direction of moving the another robot. In some cases, the direction to be avoided may be a large change in direction requiring deceleration. In such situations, the method must calculate complex planning for making the robot move in a direction to be avoided in advance. Therefore, the proposed method represents the robot motion of a time slice as a movement-direction vector in a 3D discretized graph space. Using this vector, the policy is to search for a point via the vector in a direction that is close to the target direction and can avoid collision with other robots.

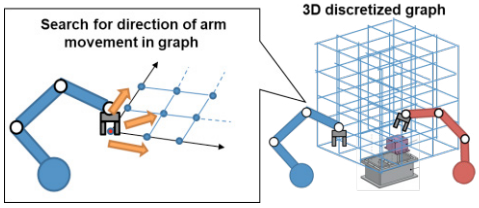


Fig.1 Trajectory search in 3D discretized graphs

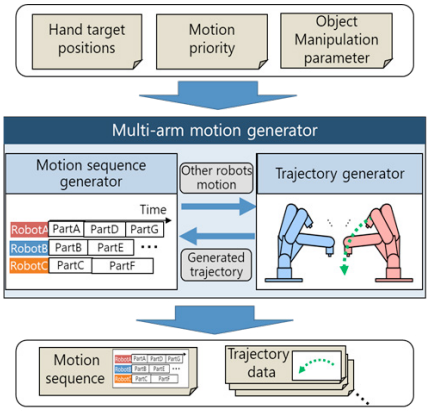


Fig. 2 System configuration of proposed multi-arm-motion generation method

3.2 System configuration

Figure 2 shows a diagram of the system configuration of the proposed method. The system generates motion using the following three types of input data shown in the upper part of the figure. The hand-target-position data are the target positions of the hand, such as the part-supply position, assembly position, and replacement-hand position. The motion-priority data describe which motion should be prioritized without avoiding motion during work, and the object-manipulation-parameter data include parameters related to hand controlling such as hand linear travel distance and hand open/close waiting time during part grasping and insertion. The multi-arm-motion planning generator consists of the motion-sequence generator and trajectory generator, as shown in the centre of the figure, and repeating the process between these two generators generates motion. The trajectory generator functions in sequence in accordance with the motion priority of the input data. Figure 3 gives an overview of the processing between the motion-sequence generator and trajectory generator on the basis of this motion priority. The proposed method of this paper is in the trajectory generator.

Step 1: The trajectory generator obtains the target position of the current planned motion from the hand-target-position data.

Step 2: From the motion-sequence generator, the trajectory generator obtains the trajectory data of other robots that have already been generated because they have a higher motion priority than the current planned motion.

Step 3: The trajectory generator generates trajectories using circumvent- and stop-avoidance methods on the basis of the trajectory data of the other robots and selects the trajectory with the shortest operation time among the two trajectories as the generated trajectory.

Step 4: The trajectory generator sends the generated trajectory data and trajectory operation time of the trajectory to the motion-sequence generator.

Step 5: The motion-sequence generator updates the motion sequences, such as which trajectory and hand operation of each robots executes at which time, using the acquired trajectory operation time and the hand control waiting time from the object-manipulation-parameter data.

Step 6: If all trajectories in the motion-priority data have been generated, the system ends multi-arm motion generation. If not, it returns to Step 1 to generate the next motion.

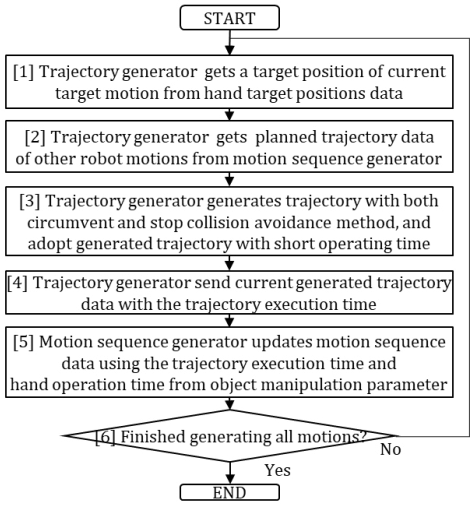


Fig. 3 system flow

The multi-arm-motion generator generates all trajectories by repeating this process between the motion-sequence generator and trajectory generator. The trajectory generator generates a 3D discretized graph and trajectory planning of the circumvent- and stop-avoidance methods using the graph. The

trajectory generator uses these two avoidances methods and selects the trajectory with the shortest operation time among the two trajectories as the generated trajectory.

3.3 Generation of 3D discretized graphs

The proposed method plans a trajectory to the target hand position in 3D space. At this point, the method searches for a point via the hand, which is a position where the hand can move, within a certain range from the current hand position toward the goal direction. Within this fixed range of space, there are countless positions where the hand is a candidate for movement. Therefore, the amount of computation increases when the method examines constraints such as whether the hand can reach each position, whether it collides with fixed objects such as the robot base, and whether it collides with other moving robots. Therefore, the method reduces the number of search points by approximating the 3D space as a discretized space in the form of a grid with fixed intervals. Figure 4 shows an example of a 3D discretized graph and the paths explored in the graph. Although the figure shows only a graph of a part of the local space, the proposed method generates the graph within the movable range of robots.

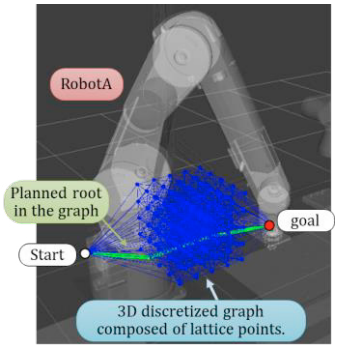


Fig. 4 3D discretized graph

3.4 Trajectory search based on circumvent avoidance

This section describes a method of searching in graph space to generate trajectories using circumvent avoidance. The flow of the circumvent-avoidance planning is shown in Fig. 5. Each grid point in the graph is described as a node.

Step 1: From the current position node n_{cur} representing the current hand position in the graph, update the vector toward the goal position n_{goal} to the local target-direction vector v_{local} representing the target movement direction. This algorithm searches in the graph toward v_{local}. The v_{local} may be updated in this flow to a vector in the direction of avoidance instead of the goal direction depending on collision avoidance. Figure 6 shows, for simplicity, the path in the goal direction described in 2D space. It gives an overview of steps 1 through step 5, which extend the path in the direction of the goal described in 2D space.

Step 2: Calculate the current direction of movement v_{cur} of the current robot (hand) with the following equation using n_{cur} and previous node n_{prev} as follows:

$$\mathrm{v}_{cur} = n_{cur} - n_{prev} \qquad (1)$$

Based on this current movement direction v_{cur}, the method stores only the nodes in the specified range adjacent to n_{cur} as the candidate node group $G_{cand}(n_{cur}, \mathrm{v}_{cur})$. Specifically, the method calculates the angle θ_{cur} between v_{cur} and the vector in the direction of adjacent node n_{adjoin} as follows:

$$\theta_{cur} = \left| \cos^{-1} \frac{\mathrm{v}_{cur} \cdot (n_{adjoin} - n_{cur})}{\|\mathrm{v}_{cur}\| \|n_{adjoin} - n_{cur}\|} \right|. \qquad (2)$$

If θ_{cur} is less than or equal to the threshold, the method adds n_{adjoin} to G_{cand}. The threshold is set to 45° to restrict the hand from moving in the direction more than 90° from v_{cur} and moving within the graph. This process prevents the robot hand from suddenly changing direction and causing the robot to slow down or pause.

Step 3: Among the n_{adjoin} in this G_{cand}, the method sets the node with the orientation closest to the current v_{local} set as the candidate node n_{cand}. The method then updates the state of the robot to the hand position and robot posture when the robot moves to n_{cand}. Specifically, the angle θ_{local} between v_{local} is calculated as follows:

$$\theta_{local} = \left| \cos^{-1} \frac{\mathrm{v}_{local} \cdot (n_{adjoin} - n_{cur})}{\|\mathrm{v}_{local}\| \|n_{adjoin} - n_{cur}\|} \right| \qquad (3)$$

The method then calculates the vector in the direction of n_{adjoin} and selects the n_{adjoin} with the smallest θ_{local} as n_{cand}.

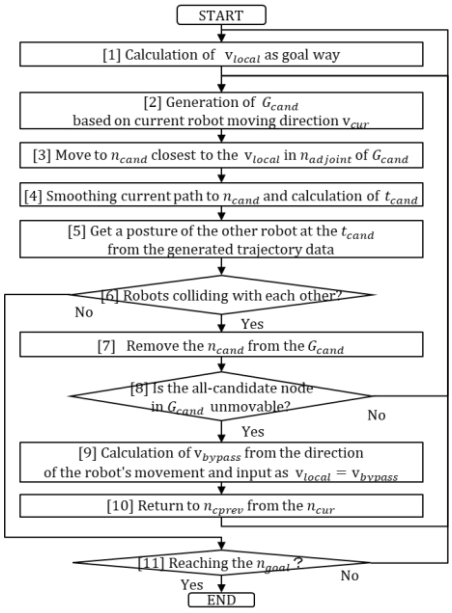


Fig. 5 Flow of circumvent-avoidance planning

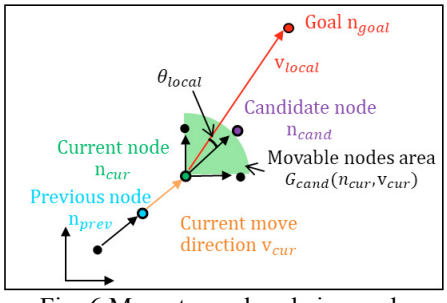


Fig. 6 Move to goal node in graph

Step 4: Smooth the discontinuous trajectory to n_{cand} by moving in the lattice space. The method calculates the arrival time t_{cand} to n_{cand} with this smoothed trajectory. The smoothing and arrival time is calculated using Time Optimal Trajectory Generation method (TOTG) of Kunz *et al.* TOTG

allows smoothing by adjusting each waypoint within a specified range and generates a smooth trajectory that does not deviate from the trajectory before smoothing.

Step 5: Obtain the posture of the other robot at t_{cand} from the generated trajectory data and update the other robot postures.

Step 6: Determine if there is any collision between robots at t_{cand}. If there is no collision, determine that the node is movable and update this n_{cand} to n_{cur}. Then, transition to step 11, and if n_{cur} has not reached n_{goal}, return to step 1. If there is collision, transition to step 7.

Step 7: The method excludes n_{cand} from G_{cand}. Then, return to Step 3, search for the next n_{adjoin} from the next G_{cand}.

Step 8: Repeat the checking of each n_{cand} in G_{cand} up to this point. If all n_{cand} are unreachable, the robot is in a position where it cannot avoid other robots from its current position. Therefore, transition to Step 9.

Step 9: Start circumvent avoidance retroactively from the node that has been passed in the past. Figure 7 shows the calculation of the avoidance direction on the basis of the movement directions of each robot when collision occurs.

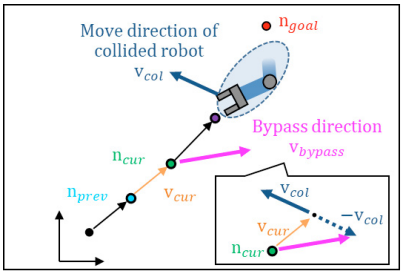


Fig.7 Calculation of circumvent direction

In step 9, the method calculates the vector v_{bypass}, which represents the direction to circumvent avoidance on the basis of v_{cur} and the opposite direction of the movement direction v_{col} of the collided robot as follows:

$$v_{bypass} = v_{cur} - v_{col} \qquad (4)$$

By updating v_{local} to v_{bypass}, the method avoids collision with the other robot.

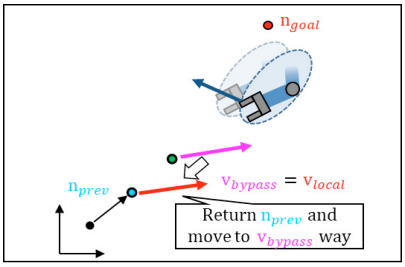


Fig. 8 Set circumvent direction from previous node

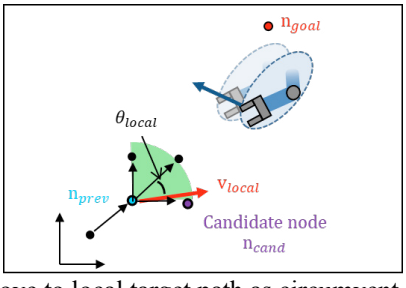


Fig. 9 Move to local target path as circumvent direction

Step 10: Return from n_{cur} to n_{prev} that has been traversed one previous time. Figure 8 shows setting v_{local}, which is circumvent avoidance, from n_{prev}.

The process then returns to step 2. As shown in Fig. 9, the method stores moving n_{adjoin} that is closer to v_{local} where this direction is to be avoided. If the method finds a new n_{cand} in G_{cand}, it is considered to have been circumvented. Therefore, the method updates v_{local} to the vector v_{goal} in the direction of the normal n_{goal}. It then returns to the regular search. If there are no n_{adjoin} that can be avoided by returning to n_{prev} that was passed one node earlier and starting avoidance, the method returns to one more previous node in steps 9 and 10. Then, re-searches the avoidance route from nodes that have been passed.

By using this flow to search in the graph, the hand can avoid collision with other robots by changing direction more than 45° from v_{cur}. Then, a slightly larger circumvent avoidance can be executed from the previous passed point position. However, when the target position is occupied by another robot, the generated trajectory is such that the hand turns and waits near the target position until the other robot leaves. This situation of waiting while turning can be determined because v_{bypass} is in the direction of returning to the start position. In such cases, the method ends trajectory generation. When the target position is occupied by another robot for a long time, the trajectory of the collision-stop-avoidance method described in the next section will be selected.

3.5 Trajectory search based on collision-stop avoidance

The collision-stop-avoidance method is a general stop-avoidance method. When this method predicts a collision with another robot, the method calculates the time it takes for other robots to leave from there. Until this time elapses, the robot pauses in front of the predicted collision position. We implement this collision-stop-avoidance method so that the robot can pause each time when it collides with other robots. If the pausing position blocks the movement direction of other robots, the method changes the stop position at a position further back in the trajectory. By combining this method with the trajectory generator, two types of collision-avoidance trajectories can be generated. If circumvent avoidance cannot be used to generate a trajectory, or if the circumvent has a longer operating time than the stop avoidance, the trajectory generator uses this collision-stop-avoidance method.

3.6 Motion-sequence generator

Based on the input motion-priority data, the motion-sequence generator has sequence data that indicate the parts allocation of each robot and motion priority of each robot and part pair. Based on the trajectory data sent from the trajectory generator and its trajectory operation time, the motion-sequence generator adds each robot trajectory and the accompanying hand-control start time to its motion-sequence data. For these hand control commands, users defines the necessary hand-control names and hand-control waiting times as object-manipulation-parameter data as described in Section 3.1. The motion-sequence generator then describes them in the operation sequence data. Therefore, this generator manages what type of motion is executed by each robot at each time

slice. This makes it possible to transmit the status of each robot to the trajectory generator.

4. EXPERIMENT

4.1 Preliminary evaluation of circumvent-avoidance and collision-stop-avoidance methods

We first verified whether the trajectories generated with the proposed method could be executed without collisions between arms. As shown in Fig. 10, we created a situation in which Robot B crosses the path of Robot A (red arrow) in the movement direction of the blue arrow. We verified that the proposed method was able to generate a trajectory for robot A to move to the target position using circumvent avoidance. The motion timing of Robot B was set so that it would collide with Robot A when Robot A was moving straight toward the goal.

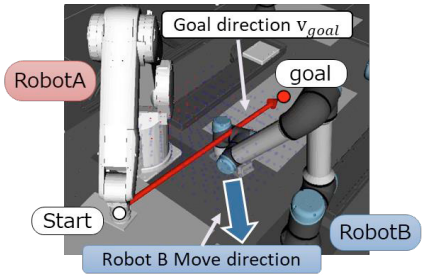
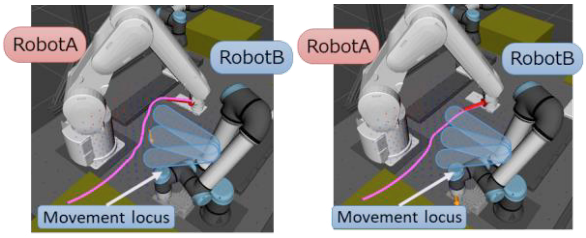
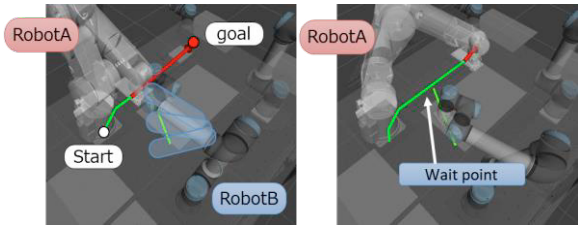


Fig. 10 Environment of circumvent avoidance



(a) Conventional method (b) Proposed method

Fig. 11 Trajectory paths with proposed and conventional methods

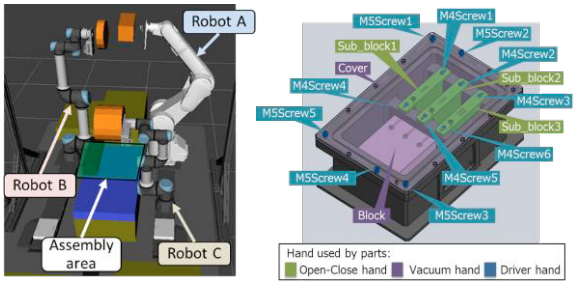


(a) Environment (b) Result

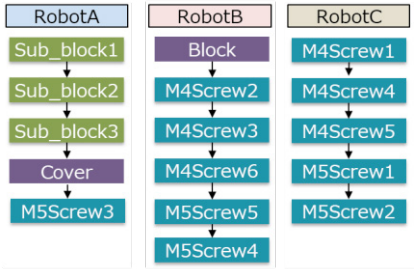
Fig. 12 trajectory paths with collision-stop-avoidance method

Figure 11(a) shows the results of generating the trajectory of Robot A with a conventional circumvent method, which circumvents all of Robot B's passing areas. This method generated a large circumvent trajectory because it avoids all passing areas, which are shown in the blue area. In contrast, Fig. 11(b) shows the trajectory of Robot A generated with the proposed method. The proposed method was able to generate a small detour trajectory for Robot A in accordance with the position of the robot obtained at each time slice. A comparison of the operation times of the two methods shows that the trajectory generated with the proposed method is more compact. Therefore, the operating time of the proposed method was about 29% shorter than that of the conventional method. These verification results indicate that the proposed method can generate trajectories with shorter operating time, although we expect the reduction in operating time to vary depending on the positional relationship of each arm and other factors.

Next, we verified the collision-stop-avoidance method. As shown in Fig. 12(a), the method generated a trajectory in which Robot A stopped and moved while avoiding Robot B. At this time, Robot B moved while occupying the area shown in blue; Figure 12(b) shows the generated trajectory of Robot A. Robot A moved on a trajectory to the target position while pausing before the collision point with Robot B. This method is a common method of pausing until the direction of movement becomes open after the method predicts the collision. However, by combining this stop method and the proposed method with hybrid to the multi-arm-motion-generation function, it is possible to generate trajectories with the collision-stop-avoidance method even in situations where it is impossible to generate trajectories with the proposed method.



(a) 3 single-arm robots (b) 16 parts of inverter product



(c) Assembly order of 16-part inverter product

Fig. 13 Overview of cell of 3 single-arm robots

4.2 Verification results of assembly operation

The trajectory planning of 3 single-arm robots in a component assembly operation was verified. In the verification, a sample product consisting of 16 parts was assembled. Figure 13(a) shows the simulation environment with the 3 single-arm robots used in the experiment. Assembly operations were verified using a sample inverter product consisting of the 16 parts shown in Fig. 13(b). Figure 13(c) shows the pre-set of parts allocation by each robot and their assembly sequence. During the sequence, Robot A executes two hand changing operations and Robot B executes one hand changing operation. Robot C does not change hands because it

is dedicated to screw tightening. Using this setup, the proposed method generated multi-arm motions of assembly operations, and the computation time was 8.8 minutes.

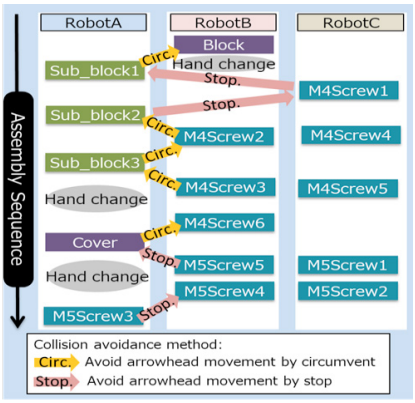


Fig. 14 Generated 3 single-arm robots motion sequence

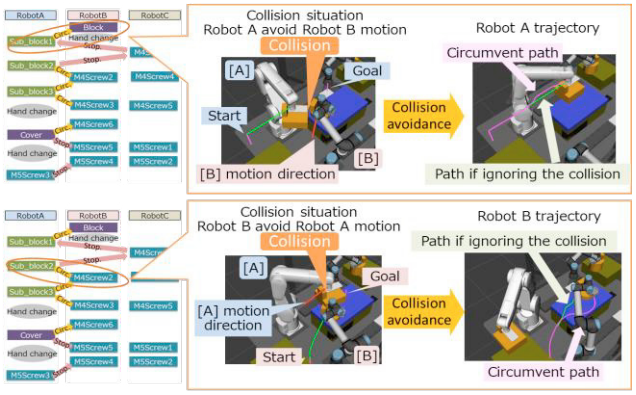


Fig.15 Example of generated circumvent avoidance trajectory

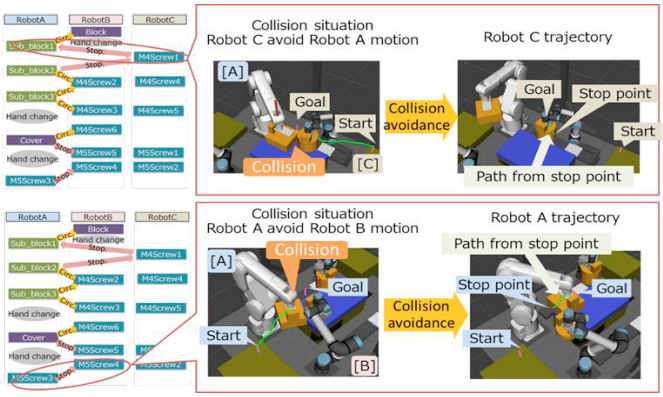


Fig.16 Example of generated stop-avoidance trajectory

Figure 14 shows the generated sequence of assembly operations and the points where collision avoidance was executed. The vertical axis represents the assembly sequence, and the horizontal axis represents each robot. As shown in the sequence, each robot executed assembly in accordance with the assembly sequence. Circumvent avoidance was executed between the movements of the two robots indicated with yellow arrows. Stop avoidance was executed between the movements of the two arms indicated with the red arrows. For example, at the top yellow arrow, Robot B has finished assembling the block component and is moving away from the assembly area. Robot A executes circumvent movement of Robot B and assembles the sub_block1 component. At the red arrow at the top, Robot A has finished assembling the sub block 1 part and is moving away from the assembly area. Robot C waits for this movement of Robot A by stop avoidance and assembles the M4Screw1 part.

Figure 15 shows an example of the generated circumvent trajectory. The orange oval of the sequence on the left side of Fig. 15 shows circumvent avoidance among the robots. The left side of the upper part of the figure shows the situation in which Robot A attempts to avoid Robot B, which is leaving the assembly area. The left side of the middle of the figure shows the situation in which Robot B attempts to avoid Robot A. The proposed method generates motion in order in accordance with the motion-priority data. Therefore, the robot motion with high motion priority has already been generated. Therefore, the pink trajectory on the right-side of the figure is generated to avoid the robot motion with high motion priority. Thus, the proposed method was able to execute circumvent avoidance.

An example of a trajectory in which stop avoidance is executed is shown in Fig. 16. The red oval of the sequence on the left of the figure shows stop avoidance among the robots. The left side of the upper row of the figure shows the situation in which Robot C attempts to move to the position of Robot A, which stays for a few seconds for hand motion in the assembly area, and the left side of the lower row shows the situation in which Robot A attempts to move to the position of Robot B. As shown in the right figure of each row, the robot on the avoiding side paused in front of the assembly area because it could not move to the target position by circumvent avoidance. This robot then started moving again and moved to the target position without collision. As a comparison of operation time, we asked an expert robot-motion instructor to teach the same assembly operation. The cycle time of the human-taught motion was 3.3 minutes. The motion generated with the proposed method took 2.2 minutes. Therefore, the cycle time was reduced by 33%. The reason for this reduction was that the human operator set a way point where the detour distance was longer and set a long pause time. These settings were made to allow a sufficient margin of close distance between robots for safe collision avoidance. In contrast, the proposed method was able to generate motion of circumvent or stop avoidance without increasing stop time or the length of the trajectory. The manual teaching required 11 days, but the proposed method only required 5 days: 2 days to prepare input data, 8.8 minutes for calculation, and 3 days for adjustment when the actual machine was used. This is a motion-time reduction of 55%.

The results indicate that the proposed method has the advantage of reducing motion time because it can generate short circumvent- and stop-avoidance trajectories compared with the conventional method used for human teaching for avoiding all passing areas of robots. The proposed method can avoid static simple-shaped obstacles such as a table and parts, but there are concerns regarding avoiding static obstacles when the target hand position is a more complex and narrow location. Therefore, we argue that the proposed method is

effective for generating dynamic collision-avoiding multi-arm motions in space of simple static obstacles. However, there is a need to improve the performance regarding more complex static-obstacle avoidance by changing the hand or elbow posture to various directions such as with joint-space search methods.

5. CONCLUSIONS

We proposed an automatic motion generation method for a multiple single-arm robots in response to the background that such robots are suitable as a worker replacement but requires man-hours to teach motion. Conventional methods have the technical problem of slow circumvent avoidance and, there are situations in which circumvent is not suitable. Therefore, on the basis of considering collision avoidance for each robot at each time-sliced sampling time, the proposed method generates time-sliced robot motions in 3D space and multi-arm motions that use both circumvent and stop avoidance in a hybrid manner. Based on the time-sliced robot motion, this method generates short circumvent-avoidance trajectories. By combining the proposed method with the collision-stop-avoidance method, the collision-stop-avoidance trajectory can be automatically generated for situations in which circumvent avoidance is not suitable. Experimental results indicate that the proposed method can reduce operation time by 33% compared with manual teaching. This comparative evaluation confirmed that the motion time can be reduced by 55%. For future work, a motion sequence optimization method will be investigated for optimizing parts allocation and the motion sequence of each arm to shorten operation time.

REFERENCES

Edsger W. Dijkstra, ”A note on two problems in connexion with graphs”, In Numerische Mathematik, vol.1, no.1, pp.269–271, 1959.

Peter E. Hart, Nils J. Nilsson, Bertram Raphael, ”A formal basis for the heuristic determination of minimum cost paths”, Transactions on Systems Science and Cybernetics, vol.4, no.2, pp.100–107, 1968.

L. Kavraki and J. Latombe: ”Probabilistic roadmaps for robot path planning”, Practical Motion Planning in Robotics: Current Approaches and Future Directions. pp.33-53, 1998.

M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor and S. Schaal,“STOMP: Stochastic Trajectory Optimization for Motion Planning”, IEEE Int. Conf. on Robotics and Automation, pp.4569–4574, 2011.

J.J. Kuner, S.M. LaValle, ”RRT-Connect: An Efficient Approach to Single-Query Path Planning”, IEEE Int. Conf. on Robotics and Automation, vol.2, pp.995–1001, 2000.

T. Kunz, M. Stilman, Time-optimal trajectory generation for path following with bounded acceleration and velocity. Robotics: Science and Systems VIII, pp.1-8, 2012.

S.M. lavalle, “Rapidly-exploring Random Trees”, A New Tool for Path Planning, TR98-11, Computer Science Dept., 1998

N. Ratliff, M. Zucker, J.A. Bagnell and S. Srinivasa: “CHOMP: Gradient Optimization Techniques for Efficient Motion Planning,” IEEE Int. Conf. on Robotics and Automation, pp.489–494, 2009.

H. Sakahara, Y. Masutani, , F. Miyazaki, . “Real time motion planning in dynamic environment containing moving obstacles using spatiotemporal RRT”. Transactions of the Society of Instrument and Control Engineers, 43(4), pp.277-284, 2007.

J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, P. Abbeel, P.Finding locally optimal,”collision-free trajectories with sequential convex optimization”, Robotics: science and systems, vol.9, no. 1, pp.1-10, 2013.

A. Sintov and A. Shapiro, "Time-based RRT algorithm for rendezvous planning of two dynamic systems," IEEE Int. Conf. on Robotics and Automation, pp. 6745-6750, 2014.

Y. Solana *et al*., “A case study of automated dual-arm manipulation in industrial applications,” IEEE Int. Conf. Emerg. Technol. Factory Automat., pp. 563–570, 2019.

H. Touzani, H. Hadj-Abdelkader, N. Séguy and S. Bouchafa, "Multi-Robot Task Sequencing & Automatic Path Planning for Cycle Time Optimization: Application for Car Production Line," in IEEE Robotics and Automation Letters, vol.6, no.2, pp. 1335-1342, 2021

Z. Wang, Y. Gan and X. Dai, "Assembly-Oriented Task Sequence Planning for a Dual-Arm Robot," in IEEE Robotics and Automation Letters, vol. 7, no. 3, pp.8455-8462, 2022

E. Yoshida, F. Kanehiro, K. Yokoi, P. Gergondet, “Online Motion Planning using Path Deformation and Replanning”, Journal of the Robotics Society of Japan, vol.29 ,no.8, pp.716-725, 2011.