mathematics

MDPI

Article

Hybrid Assembly Path Planning for Complex Products by Reusing a Priori Data

Guodong Yi, Chuanyuan Zhou, Yanpeng Cao * and Hangjian Hu

State Key Laboratory of Fluid Power & Mechatronic Systems, Zhejiang University, Hangzhou 310027, China; ygd@zju.edu.cn (G.Y.); zhouchuanyuan@zju.edu.cn (C.Z.); 21825239@zju.edu.cn (H.H.)
* Correspondence: caoyp@zju.edu.cn; Tel.: +86-0571-8795-1273

Abstract: Assembly path planning (APP) for complex products is challenging due to the large number of parts and intricate coupling requirements. A hybrid assembly path planning method is proposed herein that reuses a priori paths to improve the efficiency and success ratio. The assembly path is initially segmented to improve its reusability. Subsequently, the planned assembly paths are employed as a priori paths to establish an a priori tree, which is expanded according to the bounding sphere of the part to create the a priori space for path searching. Three rapidly exploring random tree (RRT)-based algorithms are studied for path planning based on a priori path reuse. The RRT* algorithm establishes the new path exploration tree in the early planning stage when there is no a priori path to reuse. The static RRT* (S-RRT*) and dynamic RRT* (D-RRT*) algorithms form the connection between the exploration tree and the a priori tree with a pair of connection points after the extension of the exploration tree to a priori space. The difference between the two algorithms is that the S-RRT* algorithm directly reuses an a priori path and obtains a new path through static backtracking from the endpoint to the starting point. However, the D-RRT* algorithm further extends the exploration tree via the dynamic window approach to avoid collision between an a priori path and obstacles. The algorithm subsequently obtains a new path through dynamic and non-continuous backtracking from the endpoint to the starting point. A hybrid process combining the RRT*, S-RRT*, and D-RRT* algorithms is designed to plan the assembly path for complex products in several cases. The performances of these algorithms are compared, and simulations indicate that the S-RRT* and D-RRT* algorithms are significantly superior to the RRT* algorithm in terms of the efficiency and success ratio of APP. Therefore, hybrid path planning combining the three algorithms is helpful to improving the assembly path planning of complex products.

Keywords: assembly; path planning; reuse; a priori path; RRT* algorithm

check for updates

Citation: Yi, G.; Zhou, C.; Cao, Y.; Hu, H. Hybrid Assembly Path Planning for Complex Products by Reusing a Priori Data. *Mathematics* **2021**, *9*, 395. https://doi.org/10.3390/math9040395

Academic Editor: Giovanni Angiulli

Received: 25 December 2020
Accepted: 13 February 2021
Published: 17 February 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Assembly is a crucial step to realizing the full functions of a product by constructing a complete structure, which becomes increasingly difficult with an increasing number of parts that make up the product [1–5]. Digital modelling and virtual assembly have been widely studied to solve this problem [6–9]. Assembly planning is an integral part of virtual assembly [10–13], which primarily includes assembly sequence planning (ASP), assembly line balancing (ALB), and assembly path planning (APP) [14–19].

APP consumes considerable time when there is a significant number of parts. Thus, some methods have been researched to improve its efficiency [20]. APP finds collision-free short paths from the initial location to the final position [20,21], which is similar to path planning for robots, vehicles, and drones, among other technologies [22–27]. Hassan et al. proposed a novel near-optimum APP algorithm based on the particle swarm optimization approach to solve the obstacle-free APP process in a 3D haptic-assisted environment [28]. Wei et al. proposed a centroidal Voronoi tessellation (CVT) for the intelligent control of self-assembly path planning of swarm robots [29]. Masehian et al. presented a new simple

greedy heuristic method to solve the assembly sequence and path planning problem; the method tries to locally minimize geometric interference between parts being assembled along the main directions in each iteration, and it employs a sampling-based stochastic path planner to arrange short paths for parts while avoiding workspace obstacles [20]. Morato et al. presented a technique that combines motion planning and part interaction clusters based on multiple rapidly exploring random trees to improve the generation of assembly precedence constraints [30]. These studies are beneficial for improving the efficiency of path planning. However, most of the APP methods mentioned above involve a large amount of repeated computation, leading to low planning efficiency and high costs when generating the assembly path through all parts.

In these studies, the rapidly exploring random tree (RRT) is a typical random sampling algorithm in path planning which explores free space by generating leaf nodes randomly and uniformly to expand a tree structure. A collision-free path is obtained by backtracking from the leaf node to the parent node in sequence when a leaf node reaches the target area [30,31]. The randomness of the sampling points makes it difficult to obtain the optimal path even if the number of sampling points is sufficient. Therefore, the rapidly exploring random tree star (RRT*) algorithm is one algorithm that has been proposed to improve the RRT algorithm, making the path cost lower by improving the nearest point selection strategy [23,32]. However, the RRT* algorithm cannot obtain a low-cost assembly path in a short time, or even a valid assembly path, when the number of sampling points is small.

In assembly planning, the final positions of the assembled parts are in close proximity, although their initial positions are different. Hence, most of the assembly paths overlap or partially overlap, meaning that parts of these paths are repeatedly planned. Therefore, finding and reusing the planned paths is a feasible way to reduce repeated planning and improve planning efficiency [33,34]. The difference in initial part positions in the assembly presents a challenge for the direct reuse of the planned paths as a whole. Accordingly, the assembly paths must be segmented for effective reuse. Therefore, a hybrid path planning method based on the improved RRT* algorithm is proposed in this paper to plan the motion segments as paths with different starting points and the same target point by reusing the planned paths.

The rest of this paper is organized as follows. The establishment of the configuration space for path planning is introduced in Section 2. In Section 3, the static RRT* (S-RRT*) and dynamic RRT* (D-RRT*) algorithms are proposed, and the hybrid path planning method based on the improved RRT* algorithm is illustrated in detail. A case study is provided in Section 4 to compare and analyze the RRT*, S-RRT*, and D-RRT* algorithms. Section 5 concludes the paper.

2. Establishment of the Configuration Space for Path Planning

2.1. Segmentation of the Assembly Path

A position near the product assembly is defined as the target position. All the parts are moved from their initial positions to the target position and then assembled in the final positions. Therefore, an assembly path is divided into two segments: a motion segment from the initial position to the target position and an assembly segment from the target position to the final position, as shown in Figure 1.

The motion segment is arranged by reusing the planned path, and the assembly segment is directly planned. Combining the planning results of the two segments reduces repeated planning of the same or similar paths and improves the success ratio.

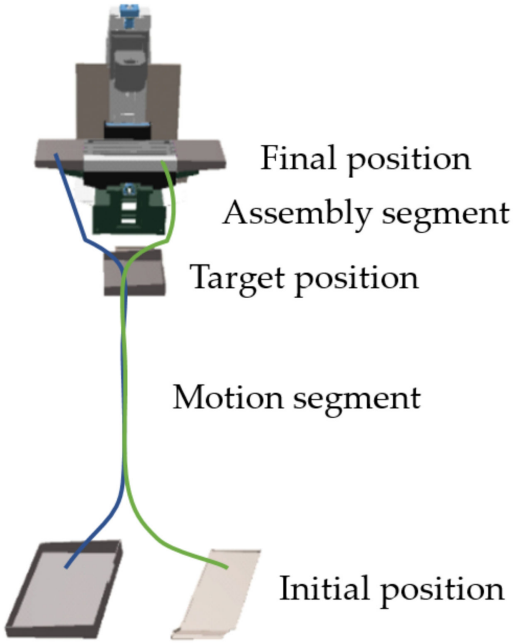


Figure 1. Segmentation of the assembly path.

2.2. Part Shrinkage and Obstacle Expansion

There are many parts and obstacles in the assembly space of a complex product, which leads to high-complexity collision detection. Consequently, the assembly space is typically mapped to a configuration space for unified modelling before path planning. However, a substantial number of components with different shapes and sizes makes path planning a sizeable task if a new configuration space is created for each part. Therefore, the configuration space must be constructed reasonably and efficiently to support reusable path planning.

In path planning based on collision detection, a part is typically described with a bounding volume. The circumscribed sphere of a part is selected as its bounding sphere to simplify the description and calculation. In the equivalent transformation from the assembly space to the configuration space, the part shrinks according to the radius of its bounding sphere. All obstacle outlines are correspondingly expanded with the same radius (see Figure 2). In this way, path planning for a part in the assembly space is transformed into finding a noninterference path in the configuration space.

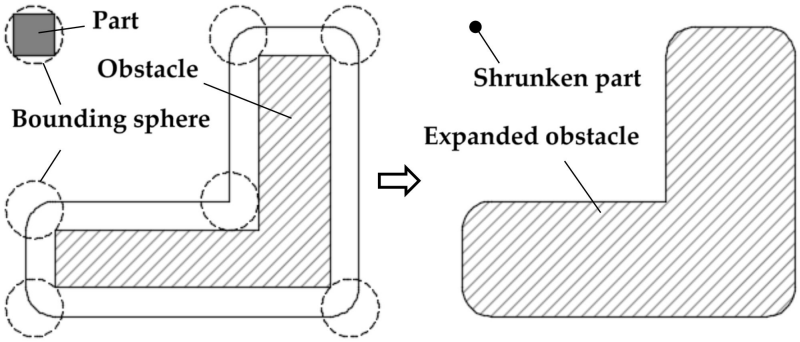


Figure 2. A part before and after shrinking, and an obstacle before and after expanding.

2.3. Establishment of the a Priori Path and a Priori Tree

An a priori path P_{prior} is a planned assembly path for a part, which is expressed as a linked list as shown in Equation (1):

$$P_{prior} = \{p_1, p_2, \ldots, p_i, \ldots, p_n\}, \quad (1)$$

where each p_i is a node of the a priori path, and p_1 and p_n are the start and end nodes corresponding to the part's initial position and target position, respectively.

A priori paths with different start nodes and the same end node form an a priori tree, where the leaf node is the start node of each path, the root node is the end node of all paths, and a branch from the leaf node to the root node is an a priori path, as shown in Figure 3. The reuse of an a priori path can be transformed into the search for an appropriate node of the a priori tree.

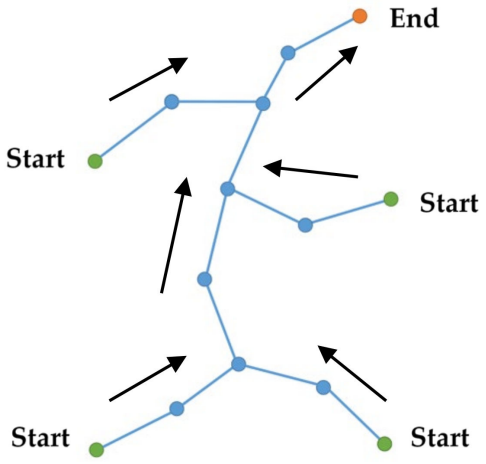


Figure 3. A priori paths with different start nodes.

A node on the a priori tree may belong to different paths, and the number of such paths reflects its "importance", which is defined as the weight of the node. The weight increases as the number of paths increases. The weight w_i of node p_i is defined as

$$w_i = n_i/n, \quad (2)$$

where n_i is the number of a priori paths crossing node p_i, and n is the total number of a priori paths. As shown in Equation (2), when the number of a priori paths crossing node p_i becomes bigger, the weight of node p_i also becomes bigger, which means that node p_i becomes more "important" among all the a priori paths.

2.4. Expansion of the a Priori Space

The path planning algorithm based on random sampling extends the path by generating random points in the configuration space. If the a priori tree is only described using its nodes, the probability that the sampling points generated randomly coincide with the nodes is notably small, which makes finding and reusing the planned paths challenging. To solve this problem, the a priori tree is converted to an a priori space by expanding its nodes and branches, improving the reusability of the a priori paths.

Since the weight of a node expresses its "importance", the expansion radius r_i of node p_i is positively related to its weight; that is,

$$r_i = d \cdot w_i, \quad (3)$$

where d is the minimum distance between the nodes of the a priori path. According to Equation (3), when the weight of a node increases, this means that the number of a priori

paths crossing node p_i increases, the radius of node p_i is greater, and the part of the a priori space of this node is larger.

Node p_i is expanded by constructing a circle with p_i as the center and r_i as the radius. External common tangents of every two adjacent circles are connected in turn. Figure 4 displays the space composed of all circles and common external tangents, which is the a priori space expanded from the a priori tree.

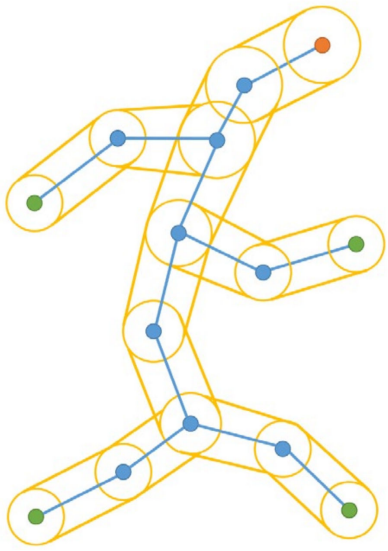


Figure 4. The a priori space expanded from the a priori tree.

3. Algorithms for Path Reuse

The RRT algorithm is widely used in path planning; it randomly generates sampling points in the configuration space based on a tree structure and explores the configuration space according to the given steps. In theory, any point in the configuration space can be reached through the nodes of the tree in a specific sequence. Therefore, the RRT algorithm was used as the basis of the algorithms studied in this paper.

1. RRT* algorithm for new path planning

It is difficult for the RRT algorithm to find the optimal path due to the randomness of the sampling points. As a result, the RRT* algorithm was proposed to lower the path cost by improving the nearest point selection strategy [31] of the RRT algorithm. The basic process of the RRT* algorithm is consistent with that of the RRT algorithm, and the main improvement is the strategy of adding new nodes and branches to the exploration tree.

In the early stage of APP, the RRT* algorithm is adopted because there is no a priori path to be reused. The key steps of the RRT* algorithm are as follows:

Step 1: Point p_{rand} is randomly generated in the configuration space C.

Step 2: Node p_{near} closest to p_{rand} is selected from the exploration tree with a connecting line, and a point on this line with a distance s (extension step of the exploration tree) from p_{near} is subsequently taken as a new leaf node, p_{new}.

Step 3: The nearest node set ps_{near} is constructed with the nodes of the exploration tree in a certain range centered on p_{new}.

Step 4: Node p_{min} is found as the parent of p_{new} by traversing ps_{near}, through which p_{new} is connected to the exploration tree with the minimum path cost.

Step 5: p_{new} and p_{min} are added to the node set, and the branch connecting them is added to the branch set.

Compared to the RRT algorithm, it is similarly difficult for the RRT* algorithm to plan an assembly path with lower cost in a short time due to the excessive calculations.

2. S-RRT* algorithm for static reuse of a priori paths

In the assembly of a new part, if there is an assembled part whose bounding sphere is not smaller than that of the new part, its configuration space can be utilized for path planning of the new part without any modification. Therefore, the path of the part can be directly reused as an a priori path to improve the planning efficiency. An S-RRT* algorithm (an improved RRT* algorithm statically reusing the a priori paths) is proposed in this case.

The main improvement of the S-RRT* algorithm is reusing the a priori paths after the exploration tree $T_{explore}$ is extended to the a priori space C_{prior}. The S-RRT* algorithm aside from this reuse is consistent with the RRT* algorithm. The main steps of the S-RRT* algorithm are as follows:

Step 1: A new leaf node p_{new} is generated on the line that connects the random point p_{rand} and its nearest node p_{near}, with a distance of s (expansion step) from p_{near}.

Step 2: p_{new} is checked to confirm whether it is in the obstacle space $C_{obstacle}$. If so, the algorithm returns to Step 1; otherwise, the algorithm continues to the next step.

Step 3: p_{new} is checked to verify whether it is in C_{prior}. If so, the exploration tree has been extended to the a priori space, and the algorithm goes to the next step; otherwise, the algorithm returns to Step 1.

Step 4: Two nodes $p_{connect1}$ and $p_{connect2}$ are selected from the a priori tree T_{prior} and the exploration tree $T_{explore}$, respectively, according to p_{new} to connect the two trees.

Step 5: The new branch E_{new} connecting $p_{connect1}$ and $p_{connect2}$ is added to $T_{explore}$, and a new path p_{new} is planned by backtracking from the end node p_{end} to the start node p_{start} and added to the a priori path set PS_{prior}.

Step 6: After sampling N times, the cost of each new path in PS_{prior} is calculated, and the path with the lowest cost P_{min} is selected as the final path.

The pseudo-code of the S-RRT* algorithm is shown in Algorithm 1.

Algorithm 1 The static rapidly exploring random tree star (S-RRT*) algorithm.

Input: C_{plan}, $C_{obstacle}$, C_{prior}, p_{start}, p_{end}, S, N, T_{prior}, $T_{explore}$
Result: A path P_{min} with the minimum cost from p_{start} to p_{end}
$T_{explore}$.init ();
PS_{prior}.init ();
for n = 1 to N do
{
p_{rand}←Sample (C_{plan});
p_{near}←Near (p_{rand}, $T_{explore}$);
p_{new}←Steer (p_{rand}, p_{near}, S);
if CollisionFree($C_{obstacle}$, p_{new}) then
{
P_{near}←Near (x_{new}, $T_{explore}$);
p_{min}←SelectParentNode (P_{near}, p_{near}, p_{new});
$T_{explore}$.AddEdge (Edge (x_{min}, p_{new}));
$T_{explore}$.Rewire();
if InPriorSpace (C_{prior}, p_{new}) then
{
$p_{connect1}$, $p_{connect2}$←ConnectNode ($T_{explore}$, T_{prior}, p_{new});
E_{new}.AddEdge ($p_{connect1}$, $p_{connect2}$);
p_{new}←PathBacktrack ($T_{explore}$, T_{prior}, $p_{connect1}$, $p_{connect2}$)
PS_{prior}.AddPath (p_{new});
}
}
}
P_{min}←MinCostPath (PS_{prior});
return P_{min};

A path planned using the S-RRT* algorithm is shown in Figure 5. In this figure, a new assembly path is established, the part of the a priori path from $p_{connect1}$ to p_{end} is reused, and the path from p_{start} through $p_{connect1}$ to p_{end} is the final path generated by the S-RRT* algorithm. Compared with the RRT* algorithm, the S-RRT* algorithm generates a path with fewer sampling times by reusing the a priori path, which helps to improve the efficiency and success ratio of APP for complex products.

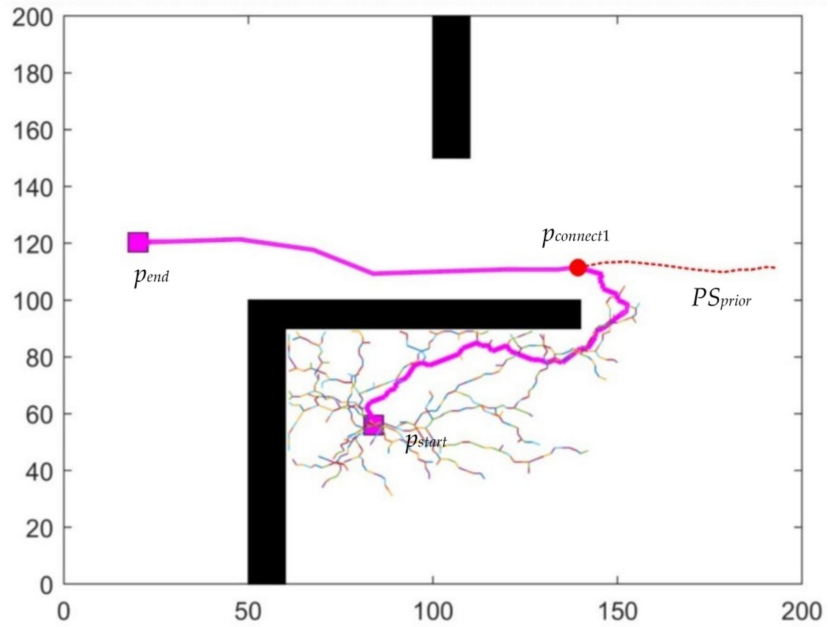


Figure 5. A path planned using S-RRT* for static reuse of a priori paths.

3. D-RRT* algorithm for dynamic reuse of a priori paths

When assembling a new part, if the bounding sphere of the assembled part is smaller than that of the new part, its configuration space cannot be used for path planning of the new part. This is because the expansion of the new part's obstacles is more extensive than that of the assembled part. The more formidable obstacles may interfere with the a priori path and lead to the failure of path planning using the S-RRT* algorithm. Therefore, a D-RRT* algorithm (an improved RRT* algorithm dynamically reusing the a priori path) is proposed to solve this problem. The main improvement of the D-RRT* algorithm is the utilization of the dynamic window approach [32] and further extension of the exploration tree to avoid interference between the a priori path and the obstacles after the connection node pair is found. The a priori path is reused non-continuously through dynamic backtracking to obtain the new path. The main steps of the dynamic backtracking of the a priori path are as follows:

Step 1: Two connection nodes $p_{connect1}$ and $p_{connect2}$ are selected from the a priori tree T_{prior} and the exploration tree $T_{explore}$, respectively, as for the S-RRT* algorithm.

Step 2: $p_{connect1}$ and its parent node p_{parent} are connected with a line to check whether the line interferes with obstacles. If there is no interference, p_{parent} is added to the new path, and the algorithm goes to Step 4; otherwise, the algorithm proceeds to Step 3.

Step 3: The local path from $p_{connect1}$ to p_{parent} is planned and added to the new path.

Step 4: $p_{connect1}$ and p_{parent} are updated with p_{parent} and its parent node, respectively, which expands the exploration tree along the a priori path.

Step 5: $p_{connect2}$ is checked to determine whether it is the end node. If the end node is identified, a new path P_{new} is planned by backtracking from the end node p_{end} to the start node p_{start}; otherwise, the algorithm returns to Step 2.

The pseudo-code of the dynamic backtracking of the a priori path is shown in Algorithm 2.

Algorithm 2 Dynamic backtracking of the a priori path.

Input: T_{prior}, $T_{explore}$, $p_{connect1}$, $p_{connect2}$, $C_{obstacle}$, S
Result: A path P_{new} without collision from $p_{connect1}$ to $p_{connect2}$
DynamicBacktracking(T_{prior}, $T_{explore}$, $p_{connect1}$, $p_{connect2}$, $C_{obstacle}$, S) {
$p_{current} \leftarrow p_{connect1}$;
$p_{parent} \leftarrow p_{current}$.parentNodeInPriorTree ();
for j = 1 to N do {
if CollisionFree ($C_{reusable}$, $C_{obstacle}$, $p_{current}$, p_{parent}) then{
P_{prior}.addNode ($p_{current}$, p_{parent});
P_{prior}.addEdge (Edge ($p_{current}$, p_{parent}));
}
else{
$P_{local} \leftarrow$LocalPath ($p_{current}$, p_{parent});
$P_{prior} \leftarrow$addLocalPath (P_{local});
}
$p_{current} \leftarrow p_{parent}$;
$p_{parent} \leftarrow p_{parent}$. parentNodeInPriorTree ();
if p_{parent} = $p_{connect2}$ then {
return P_{prior};
}
}
}

A path planned using the D-RRT* algorithm is shown in Figure 6. There is an obstacle in Figure 6 that did not appear in Figure 5, which occurs in the part of the a priori path from $p_{connect1}$ to p_{end}, and the D-RRT* algorithm can establish new assembly paths in this situation. Thus, a new assembly path was generated by D-RRT* from p_{start} through $p_{connect1}$ to p_{end}. The D-RRT* algorithm avoids obstacles by using dynamic backtracking and local path planning, which helps efficiently obtain an optimal assembly path without additional interference.

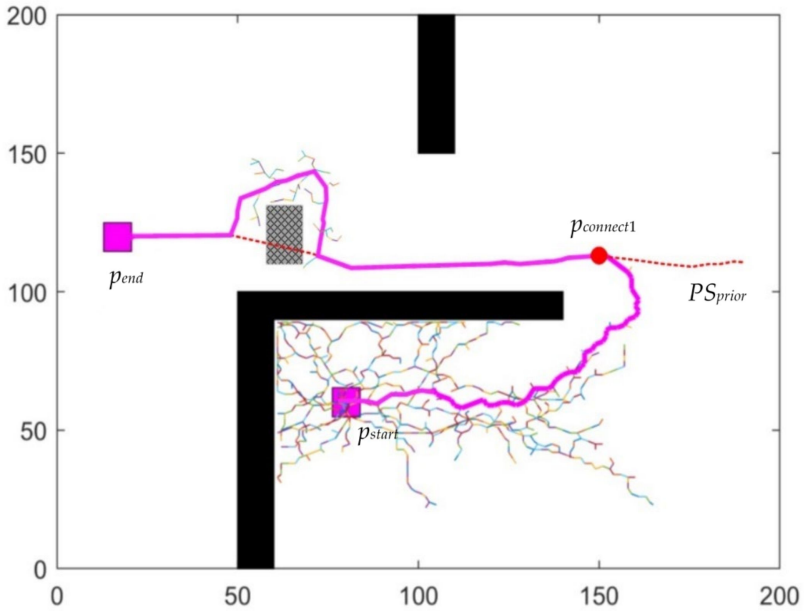


Figure 6. A path planned using dynamic RRT* (D-RRT*) for dynamic reuse of the a priori path.

4. Hybrid process of the three algorithms

Using the three abovementioned algorithms, a hybrid process based on a priori path reuse was designed for assembly path planning, as presented in Figure 7.

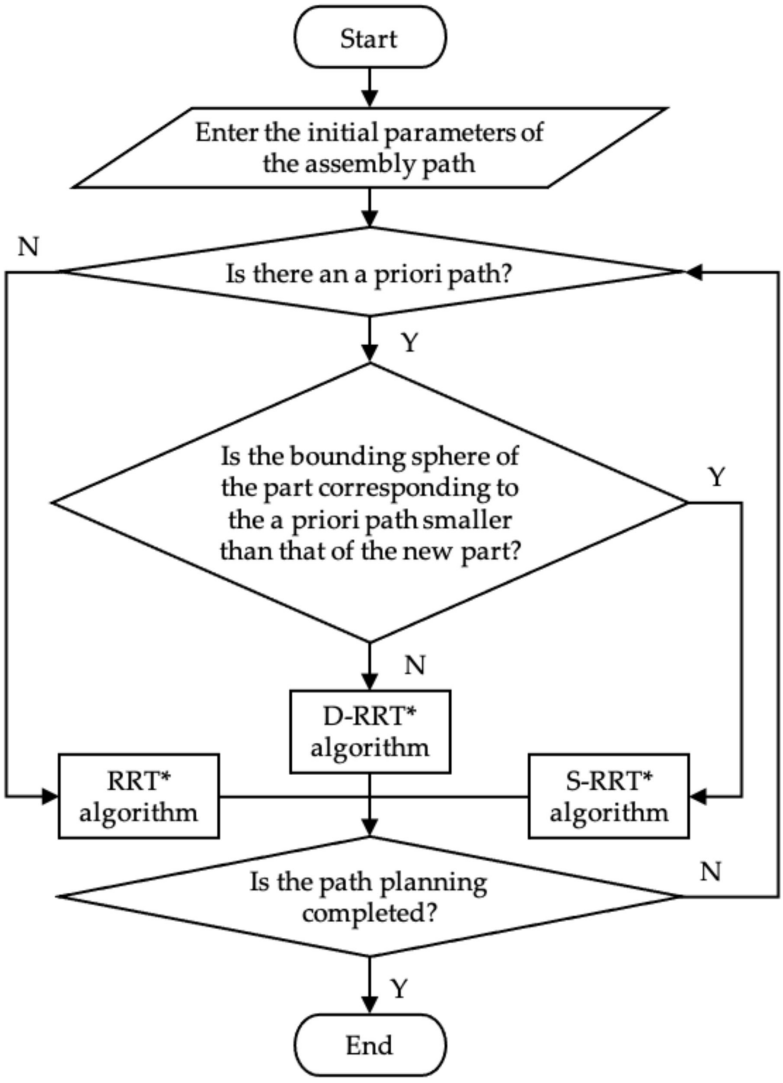
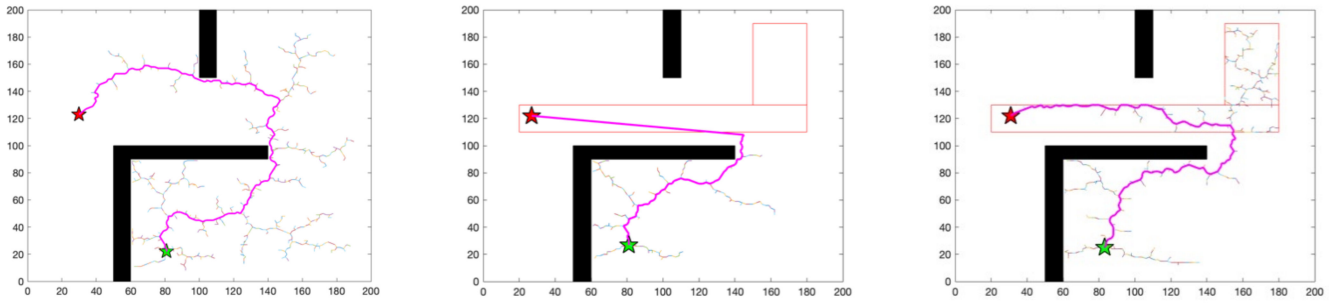


Figure 7. Hybrid process for assembly path planning.

4. Comparison and Analysis

To analyze the performance of the RRT*, S-RRT*, and D-RRT* algorithms in the same case, a comparison is provided in this section. D-RRT* can also be applied in a situation in which there is no obstacle in the prior path, since it is an algorithm based on S-RRT* which can establish new path plans in the presence of obstacles, illustrated in detail in Section 3. Thus, a suitable case must be prepared to test the performance and make comparisons for the three algorithms. Assuming some a priori paths in the configuration space, the three algorithms mentioned above were used to plan the assembly path from the same starting point to the same endpoint, and one of the results is shown in Figure 8. Information relating to Figure 8 is presented in Table 1.



(**a**) (**b**) (**c**)

Figure 8. Results for the three path planning algorithms: (**a**) RRT* algorithm; (**b**) S-RRT* algorithm; (**c**) D-RRT* algorithm.

Table 1. Information relating to Figure 8.

	Figure 8a	Figure 8b	Figure 8c
Max number of iterations	2000	2000	2000
Distance to the target threshold	5	5	5
Goal of probability	0.05	0.05	0.05
Step size	2	2	2
Path distance	314	128	286
Number of iterations	1104	330	1805
Number of tree branches	730	206	486

The randomness of sampling points in the three algorithms leads to large differences in the results of path planning, even with the same start and end points. Therefore, each algorithm was executed 100 times, and the averages of the minimum number of sampling points, the path length, and the running time of the three algorithms were compared, as illustrated in Table 2.

Table 2. Comparison of the three algorithms in path planning with the same starting point.

Algorithm	Minimum Number of Sampling Points	Path Length	Running Time (s)
RRT*	1104	384	1.26
S-RRT*	330	248	0.356
D-RRT*	407	256	0.432

Table 2 demonstrates that the minimum numbers of sampling points for the S-RRT* algorithm and the D-RRT* algorithm were only 1/4 and 1/3 that for the RRT* algorithm, respectively, and their path length and running time were also less than those of the RRT* algorithm. The performance of the D-RRT* algorithm means that it is not as useful as the S-RRT* algorithm due to the additional sampling strategy.

Considering that the start position of each part of a complex product is different in the assembly space, 50 starting points were randomly generated in the configuration space for the path planning simulation. The RRT*, S-RRT*, and D-RRT* algorithms were used to plan the assembly path for each starting point with different maximum numbers of sampling points. The number of successfully planned paths was used to calculate the path planning success ratio for multiple starting positions with the specified maximum number of sampling points, as shown in Table 3.

Table 3. Comparison of the success ratios of path planning with the three algorithms.

Maximum Number of Sampling Points	Success Ratio of S-RRT* Algorithm	Success Ratio of D-RRT* Algorithm	Success Ratio of RRT* Algorithm
0	0.00	0.00	0.00
250	0.50	0.20	0.05
500	0.70	0.34	0.05
750	0.78	0.46	0.10
1000	0.82	0.60	0.15
1250	0.86	0.68	0.30
1500	0.90	0.74	0.35
1750	0.94	0.80	0.40
2000	0.96	0.85	0.42
2250	0.98	0.90	0.45
2500	1.00	0.93	0.50
2750	1.00	0.95	0.52
3000	1.00	0.97	0.58
3250	1.00	0.98	0.62
3500	1.00	1.00	0.66
3750	1.00	1.00	0.76
4000	1.00	1.00	0.80

Table 3 confirms that as the maximum number of sampling points increased, the success ratios of the three algorithms also increased. When the maximum number of sampling points was small, the success ratios of the S-RRT* algorithm and the D-RRT* algorithm were considerably greater than those of the RRT* algorithm. The success ratios of these three algorithms are also shown in Figure 9.

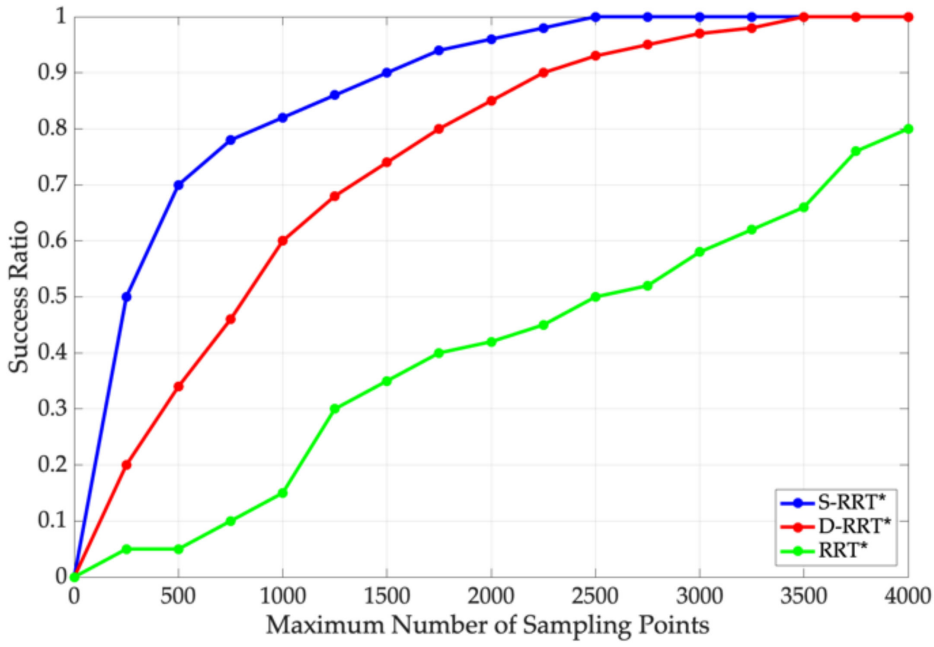


Figure 9. Success ratios of RRT*, S-RRT*, and D-RRT*.

The minimum numbers of sampling points and the total running times of the three algorithms are specified in Table 4 for the aforementioned path plans.

Table 4. Comparison of the three algorithms in path planning with different starting points.

Algorithm	Minimum Number of Sampling Points	Running Time (s)
RRT*	12,000	137.56
S-RRT*	2500	24.95
D-RRT*	3500	28.84

Table 4 shows that the minimum numbers of sampling points and the total running times of the S-RRT* and D-RRT* algorithms in APP for batch parts are much smaller than those of the RRT* algorithm.

Thus, the results of the figures and tables above can explain the advantages and disadvantages of RRT*, S-RRT*, and D-RRT*. The RRT* algorithm is employed to establish an assembly path without any a priori data, which requires many more sampling points and much more computing time than S-RRT* and D-RRT*. The S-RRT* algorithm is the fastest and the most efficient when compared to RRT* and D-RRT*, but it strictly demands a priori paths and stability in the sizes of new parts. The D-RRT* algorithm can establish assembly paths when there are new obstacles interfering with the a priori paths. However, D-RRT* still needs more sampling data and calculating time than S-RRT*, and a priori paths are required as well.

5. Conclusions

APP for complex products is tedious and repetitive work. A hybrid planning method that reuses a priori paths was proposed in this paper to improve the efficiency and success ratio of path planning. In this method, the assembly path is divided into motion segments and assembly segments to improve the reusability of the path, and the feasibility of the motion segment is analyzed. An a priori path set is established according to the planned assembly paths; on this basis, an a priori tree is established. An a priori space is created by expanding the a priori tree according to the bounding sphere of the part to be assembled, which establishes a foundation for path reuse.

Three algorithms improving upon the RRT algorithm were examined for path planning based on a priori path reuse. The RRT* algorithm establishes the exploration tree of the new path in the early planning stage when there is no a priori path to reuse. The S-RRT* and D-RRT* algorithms are utilized after the exploration tree of the new path is extended to the a priori space according to a comparison of bounding spheres between the part to be assembled and the part corresponding to the a priori path. If the former's bounding sphere is not greater than the latter's, the S-RRT* algorithm finds a pair of connection points to establish a connection between the exploration tree and the a priori tree and obtain a new path through direct backtracking from the endpoint to the starting point. Otherwise, the D-RRT* algorithm extends the exploration tree via the dynamic window approach after finding the connection point pair to avoid interference between the a priori path and the obstacles, and a new path is obtained through dynamic and non-continuous backtracking.

The RRT*, S-RRT*, and D-RRT* algorithms were compared in terms of their minimum number of sampling points, path length, and running time in single-start assembly path planning. The success ratio, minimum number of sampling points, and running time in multistart assembly path planning were also compared. The results show that the S-RRT* and D-RRT* algorithms are significantly better than the RRT* algorithm due to the reuse of the a priori paths. Therefore, hybrid path planning combining the three algorithms is helpful to improving the efficiency and success ratio of the assembly path planning of complex products.

Author Contributions: Methodology, G.Y. and Y.C.; investigation, G.Y. and H.H.; software and validation, C.Z. and H.H.; writing—original draft preparation, C.Z. and H.H.; writing—review and editing, G.Y. and Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China, grant number 2018YFB1701300, the National Natural Science Foundation of China, grant number 51875515, the Key Research and Development Program of Zhejiang Province, grant number 2020C0409, and the Science and Technology Project of Beijing, grant number Z191100001419014.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Demoly, F.; Yan, X.-T.; Eynard, B.; Rivest, L.; Gomes, S. An assembly oriented design framework for product structure engineering and assembly sequence planning. *Robot. Comput. -Integr. Manuf.* **2011**, *27*, 33–46. [CrossRef]
2. Demoly, F.; Toussaint, L.; Eynard, B.; Kiritsis, D.; Gomes, S. Geometric skeleton computation enabling concurrent product engineering and assembly sequence planning. *Comput. -Aided Des.* **2011**, *43*, 1654–1673. [CrossRef]
3. Tsutsumi, D.; Gyulai, D.; Kovács, A.; Tipary, B.; Ueno, Y.; Nonaka, Y.; Fujita, K. Joint optimization of product tolerance design, process plan, and production plan in high-precision multi-product assembly. *J. Manuf. Syst.* **2020**, *54*, 336–347. [CrossRef]
4. Wallis, R.; Erohin, O.; Klinkenberg, R.; Deuse, J.; Stromberger, F. Data Mining-supported Generation of Assembly Process Plans. *Procedia CIRP* **2014**, *23*, 178–183. [CrossRef]
5. Matei, O.; Contraş, D.; Pop, P.; Vălean, H. Design and comparison of two evolutionary approaches for automated product design. *Soft Comput.* **2016**, *20*, 4257–4269. [CrossRef]
6. Sierla, S.; Kyrki, V.; Aarnio, P.; Vyatkin, V. Automatic assembly planning based on digital product descriptions. *Comput. Ind.* **2018**, *97*, 34–46. [CrossRef]
7. Yang, Q.; Wu, D.L.; Zhu, H.M.; Bao, J.S.; Wei, Z.H. Assembly operation process planning by mapping a virtual assembly simulation to real operation. *Comput. Ind.* **2013**, *64*, 869–879. [CrossRef]
8. Qiu, C.; Zhou, S.; Liu, Z.; Gao, Q.; Tan, J. Digital assembly technology based on augmented reality and digital twins: A review. *Virtual Real. Intell. Hardw.* **2019**, *1*, 597–610. [CrossRef]
9. Yoon, J. Assembly simulations in virtual environments with optimized haptic path and sequence. *Robot. Comput. -Integr. Manuf.* **2011**, *27*, 306–317. [CrossRef]
10. Li, T.; Lockett, H.; Lawson, C. Using requirement-functional-logical-physical models to support early assembly process planning for complex aircraft systems integration. *J. Manuf. Syst.* **2020**, *54*, 242–257. [CrossRef]
11. Hui, W.; Dong, X.; Guanghong, D.; Linxuan, Z. Assembly planning based on semantic modeling approach. *Comput. Ind.* **2007**, *58*, 227–239. [CrossRef]
12. Kardos, C.; Váncza, J. Application of Generic CAD Models for Supporting Feature Based Assembly Process Planning. *Procedia CIRP* **2018**, *67*, 446–451. [CrossRef]
13. Gruhier, E.; Demoly, F.; Gomes, S. A spatiotemporal information management framework for product design and assembly process planning reconciliation. *Comput. Ind.* **2017**, *90*, 17–41. [CrossRef]
14. Ghandi, S.; Masehian, E. Review and taxonomies of assembly and disassembly path planning problems and approaches. *Comput. -Aided Des.* **2015**, *67–68*, 58–86. [CrossRef]
15. Chen, W.-C.; Hsu, Y.-Y.; Hsieh, L.-F.; Tai, P.-H. A systematic optimization approach for assembly sequence planning using Taguchi method, DOE, and BPNN. *Expert Syst. Appl.* **2010**, *37*, 716–726. [CrossRef]
16. Kardos, C.; Kovacs, A.; Vancza, J. A constraint model for assembly planning. *J. Manuf. Syst.* **2020**, *54*, 196–203. [CrossRef]
17. Ghandi, S.; Masehian, E. A breakout local search (BLS) method for solving the assembly sequence planning problem. *Eng. Appl. Artif. Intell.* **2015**, *39*, 245–266. [CrossRef]
18. Wang, H.; Rong, Y.; Xiang, D. Mechanical assembly planning using ant colony optimization. *Comput. -Aided Des.* **2014**, *47*, 59–71. [CrossRef]
19. Chen, R.-S.; Lu, K.-Y.; Tai, P.-H. Optimizing assembly planning through a three-stage integrated approach. *Int. J. Prod. Econ.* **2004**, *88*, 243–256. [CrossRef]
20. Masehian, E.; Ghandi, S. ASPPR: A New Assembly Sequence and Path Planner/Replanner for Monotone and Nonmonotone Assembly Planning. *Comput. -Aided Des.* **2020**, *123*, 102828. [CrossRef]
21. Ladeveze, N.; Fourquet, J.-Y.; Puel, B. Interactive path planning for haptic assistance in assembly tasks. *Comput. Graph.* **2010**, *34*, 17–25. [CrossRef]
22. Yuan, C.; Liu, G.; Zhang, W.; Pan, X. An efficient RRT cache method in dynamic environments for path planning. *Robot. Auton. Syst.* **2020**, *131*, 103595. [CrossRef]

23. Li, Y.; Wei, W.; Gao, Y.; Wang, D.; Fan, Z. PQ-RRT*: An improved path planning algorithm for mobile robots. *Expert Syst. Appl.* **2020**, *152*, 113425. [CrossRef]
24. Zhong, M.; Yang, Y.; Dessouky, Y.; Postolache, O. Multi-AGV scheduling for conflict-free path planning in automated container terminals. *Comput. Ind. Eng.* **2020**, *142*, 106371. [CrossRef]
25. Fransen, K.J.C.; van Eekelen, J.A.W.M.; Pogromsky, A.; Boon, M.A.A.; Adan, I.J.B.F. A dynamic path planning approach for dense, large, grid-based automated guided vehicle systems. *Comput. Oper. Res.* **2020**, *123*, 105046. [CrossRef]
26. Singh, Y.; Sharma, S.; Sutton, R.; Hatton, D.; Khan, A. A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents. *Ocean Eng.* **2018**, *169*, 187–201. [CrossRef]
27. Zhao, Y.; Zheng, Z.; Liu, Y. Survey on computational-intelligence-based UAV path planning. *Knowl. -Based Syst.* **2018**, *158*, 54–64. [CrossRef]
28. Hassan, S.; Yoon, J. Haptic assisted aircraft optimal assembly path planning scheme based on swarming and artificial potential field approach. *Adv. Eng. Softw.* **2014**, *69*, 18–25. [CrossRef]
29. Wei, H.X.; Mao, Q.; Guan, Y.; Li, Y.D. A centroidal Voronoi tessellation based intelligent control algorithm for the self-assembly path planning of swarm robots. *Expert Syst. Appl.* **2017**, *85*, 261–269. [CrossRef]
30. Morato, C.; Kaipa, K.N.; Gupta, S.K. Improving assembly precedence constraint generation by utilizing motion planning and part interaction clusters. *Comput. -Aided Des.* **2013**, *45*, 1349–1364. [CrossRef]
31. Wang, W.; Deng, H.; Wu, X. Path planning of loaded pin-jointed bar mechanisms using Rapidly-exploring Random Tree method. *Comput. Struct.* **2018**, *209*, 65–73. [CrossRef]
32. Chao, N.; Liu, Y.-K.; Xia, H.; Peng, M.-J.; Ayodeji, A. DL-RRT* algorithm for least dose path Re-planning in dynamic radioactive environments. *Nucl. Eng. Technol.* **2019**, *51*, 825–836. [CrossRef]
33. Yan, Y.; Poirson, E.; Bennis, F. An interactive motion planning framework that can learn from experience. *Comput. -Aided Des.* **2015**, *59*, 23–38. [CrossRef]
34. Gaisbauer, F.; Lehwald, J.; Agethen, P.; Otto, M.; Rukzio, E. A Motion Reuse Framework for Accelerated Simulation of Manual Assembly Processes. *Procedia CIRP* **2018**, *72*, 398–403. [CrossRef]