applied sciences

MDPI

Article

Integrated Motion Planning for Assembly Task with Part Manipulation Using Re-Grasping

Ahmad Ali [1,2] and Ji Yeong Lee [1,3,*]

[1] Department of Mechatronics Engineering, ERICA Campus, Hanyang University, Ansan, Gyeonggi-do 425-791, Korea; ahmadali@hanyang.ac.kr
[2] Department of Mechatronics Engineering, University of Engineering & Technology, Lahore (Faisalabad Campus), Faisalabad 38000, Pakistan
[3] Department of Robotics Engineering, ERICA Campus, Hanyang University, Ansan, Gyeonggi-do 425-791, Korea
* Correspondence: jiyeongl@hanyang.ac.kr

Received: 9 December 2019; Accepted: 14 January 2020; Published: 21 January 2020

Abstract: This paper presents an integrated planner based on rapidly exploring random tree (RRT) for an assembly task with possible re-grasping. Given multiple grasp poses for the part to assemble, the planner chooses candidate grasp poses considering the environment (including the partially finished assembly) in addition to the initial and final poses of the part. Orientation graph search based re-grasping approach is proposed for part manipulation which is needed when there is no feasible grasp solution for a part between its initial and final poses. Orientation graph search helps finding a series of the intermediate poses of the part needed between its initial and final poses so that robot can grasp and assemble it without interfering the pre-assembled parts. Then while extending the tree, the algorithm tries to connect the tree to a robot configuration with a chosen candidate grasp pose. Also, since the task space undergoes changes at each step of the assembly task, a node or edge in the tree can become in collision during the assembly of later parts, making the node in collision and its descendant nodes disconnected from the whole tree. To handle this, Two stage extended RRT strategy is proposed. The disconnected parts of the main tree are put into forest, and attempts are made to re-connect the tree in the forest to main tree while extending the main tree, thus making it possible to use the disconnected part again. The algorithm is implemented in Linux based system using C++. The proposed algorithm is demonstrated experimentally using UR5e robot manipulator by assembling the soma puzzle pieces in different 3D formations.

Keywords: assembly operation; RRT; part manipulation; orientation graph search

1. Introduction

The role of robots in the industry environment is growing with the advancement in the research tools in the robotic industry to improve the production efficiency. The robots are commonly used in assembly line in the industry to effectively perform repetitive assembly operations [1] and tele-manipulation in remote sites [2,3]. The combinatorial based approaches for assembly planning are widely addressed in literature [4,5] however, these are used only for assembly sequence planning for the parts to be assembled. Also, there is a recent research trend of using machine leaching based approached in robotic assembly such as learning by demonstration [6] but currently these can handle only simple operations such as inserting peg in a hole, fastening the screw etc. In order to make the robots capable to perform a task autonomously, the planning algorithms are developed (for review [7,8]). However, the assembling task is quite challenging for robots. One of the recent trends in the robotics research is to make the robots capable of planning and performing the assembly task autonomously. This multi-disciplinary

task includes part recognition, part pose estimation, assembly planning, grasp planning, motion planning of part, and robot etc. For example integrated robotic systems such as [9,10] were developed to perform manipulation however, the main focus was the development of software architecture. These platforms still lack autonomously handling the complex situations such as performing an assembly operation such as, in [9] the part is grasped by caging it but for assembly task, the planners need to choose a grasp carefully among possibly multiple feasible grasps which not only consider the pose and geometry of current part but also the pre-assembled parts in the assembly. This paper focuses on the integrated grasp and motion planning of robot to perform an assembly task.

Motion planning for a robot with high degrees of freedom in the presence of the obstacles in the environment is P-space hard [7]. For this kind of problems probabilistic approaches such as Probabilistic Road Map (PRM), rapidly exploring random tree (RRT), Expansive Spaces Tree (EST), Single Query Bi-directional Probabilistic Road Map (SBL) etc., are widely used to plan collision free path [7]. RRT based approaches are quite successful for last two decades due to their simplicity and reliability [11]. In [12], an RRT-based integrated approach is proposed to plan robot motion along with grasp planning as Grasp-RRT. This integrated approach searches a feasible grasp pose of hand while planning the robot motion. However, Grasp-RRT only considers the grasp problem for individual pieces but does not consider the assembly task constraints. Also, Grasp-RRT finds a grasp in a trial-and error manner. Instead, in this paper, we assume that a set of feasible grasps is already given, and then the algorithm first evaluates these grasps to check whether they can be used at the given assembly step. This results in a subset of the initially given grasps (but still possibly containing more than one grasp). Later on, while planning the robot motion, the planner tries to reach to these grasps using a method similar to Grasp-RRT. Note that the algorithm does not choose a specific grasp prior to planning, but the choice is made during the planning of the robot manipulator.

Part manipulation and re-grasping has been addressed in literature for Single and dual-arm robot [13–16]. In general, the part manipulation is performed in configuration space (*C-Space*) jointly defined for the robot, and the object to be manipulated. The C-space search based part manipulation involves high dimensional C-space search which is quite computationally expensive. On the other hand, we propose a part manipulation approach based on the orientation graph search in the task space. If there is no feasible grasp for the part for its given initial, and final poses and assembly constraints then we can find intermediate poses of the part such that we can have at least one feasible grasp from initial pose of part to its intermediate pose and also from intermediate pose to final pose. A finite set of stable poses of a part can be evaluated based on the geometry of the part to build an orientation graph. Since the proposed approach considers a finite set of orientations of the part, it is computationally less costly than the C-space search based approaches.

Integrated assembly sequence and motion planning is presented in [17,18] using combinatorial approach for assembly planning. But the motion planning for the robot in [17] was variant of RRT [19] which grows a tree in the configuration space from start at every stage of the assembly process. But it would be more efficient if we can use a single tree for the whole assembly process. Because while assembling multiple parts sequentially, the task space is changing at every step of the assembly operation. In this situation, the robot configuration which is collision free at an earlier stage of assembly operation may not be collision free in a later stage. In [20], lazy collision checking was introduced for single query probabilistic planner which performs the collision checking only along the searched path. In [21], multi-partite RRT (MP-RRT) was introduced for re-planning in a dynamic environment. The MP-RRT removes the failed nodes and edges and stack the collision free broken branches in the forest data structure. On the contrary, we propose a semi-lazy collision checking strategy for part assembling task. While growing tree, the new node added to tree is checked for collision according to the current state of the task space. But collision checking along the edges is only performed for those which are in the searched path. In case of collision along the edge (between nodes a and b), the edge is removed while the node b (which is the root of a tree disconnected from the main tree) is included in the set of forest roots. While growing tree, the algorithm tries to connect the disconnected node to main tree

using what we term two stage extend-RRT function. While extending main tree, it adds a new node q_1 in main tree in first stage and extend the q_1 to nearest forest root in second stage. This strategy helps to re-connect the forest roots to the main tree.

1.1. Contribution

The key contributions of this article are the following:

(a) Integrated part manipulation and robot motion planner is proposed to perform an assembly task.
(b) Orientation graph search based approach is proposed as re-grasping strategy to manipulate a part, if necessary, while performing the assembly task.
(c) Collision checking strategy is proposed for changing task space. Since the task space for assembly task is not static that is why the node (robot configuration) which is valid in current state of task space but may be invalid in next stage of assembly. Because of this we used lazy collision checking based approach so that we can use same tree while doing assembly rather than creating new tree at every stage of the assembly task.

1.2. Organization of Paper

The paper is organized as follows, Section 2 presents an overview of problem statement, and the proposed algorithm. The integrated planning algorithm is presented in Section 3. Section 3.1 describes the idea of feasible grasp to perform the assembly task. The part manipulation approach based on orientation graph search is presented in Section 3.2. In Section 3.3, the planning algorithm is presented. The experimental demonstration is presented in Section 4. Finally, Section 5 presents the conclusion and future work.

2. Overview of Algorithm

The objective of this work is to develop an integrated planning algorithm which can simultaneously plan the feasible grasp, part manipulation and collision free C-space path for robot to assemble multiple parts. The algorithm is based on RRT, which grows a tree of robot configurations that can cover the C-space. Let K be the total number of parts to be assembled. The parts are assumed to be represented as polytopes. A body frame is attached to each part to define its position and orientation with reference to the inertial frame. Let $P = (p, R)$ be the pose of a part where $p \in \mathbb{R}^3$ be the position and $R \in \mathbb{SO}(3)$ be the rotation matrix representing the orientation of the part. Let $P_{i,init}$, and $P_{i,final}$ be the initial, and final poses of parts, respectively where $i = \{1, \ldots, K\}$. It is assumed that the initial, and final poses of the parts along with the part sequence for the assembly are known. Given the set of grasps G, which are the hand poses to grasp a part, the planner will be able to choose a feasible grasp such that robot could grasp the part to be assemble it without interfering the other parts in sub-assembly. A grasp can be represented by the position and orientation of the hand frame in grasping pose and the contact points of hand on grasped object with respect to hand frame. If it is not possible to find a feasible grasp for a part in its initial pose, the planner will find the solution that how to manipulate or re-grasp the part to find a feasible grasp solution.

The outline of the algorithm is as follows: First the sequence of poses for each part are identified using orientation graph search. Then a set of feasible grasp poses of hand are identified to pick and place the part without having collision with environment. Later, the motion of the manipulator is planned using an RRT based algorithm. By taking inspiration from Grasp-RRT [12], while growing the tree using RRT, attempts are made to connect the tree to configurations whose hand poses are the chosen feasible grasp. After successfully grasping a part, robot motion is planned to move the part to is final pose. The detail of different parts of the algorithm are presented in next section.

3. Feasible Grasp, Part Manipulation, and Integrated Planning Algorithm

- This section presents an integrated planning algorithm which plans the path for the robot to perform an assembly task. The algorithm is composed of following three major parts; Feasible grasp selection
- Path planning of robot with attempt to grasp and re-grasp a part, if necessary, using orientation graph search.
- Path planning for the robot to move the grasped part to its final assembly pose.

Following sections presents the main parts of the algorithm in details.

3.1. Feasible Grasps for an Assembly Task

In order to perform an assembly task using robotic manipulator, the grasp planning is of key importance. Finding a stable grasp for objects with arbitrary shape is an active field of research [22–25], and as mentioned Grasp-RRT tries to find a grasp in a trial-and-error based way. In general, a grasp is deemed stable or feasible considering only the robot hand (or gripper) and the object, but the environment is not considered. However, for assembly, the environment must be considered and the environment also must include the partially assembled parts which change as the assembly process progresses. In this paper, we do not focus on finding stable grasps for the given object, but how to choose a feasible grasp among the given set of grasps, which make assembly possible, and how to select intermediate grasp if re-grasping is necessary. Thus, we assume that a set of grasps (G) is given and we will find a set of feasible grasps $FG \subset G$ that can be used for assembly. While selecting the set of feasible grasps, only the hand but not the whole robot body, is checked for collision with environment as well as the partial assembly at initial and final poses. Note that we keep not just one feasible grasp but a *set* of feasible grasps since some feasible grasp maybe actually unusable due to other constraints such as collision of the robot body environment or joint limit of the robot etc. Later, during motion planning phase, the algorithm tries to find a collision free path such that the robot can grasp the part using any one of the chosen feasible grasps. Even without considering the robot body, it may not be possible to move the part from its initial pose to its final assembly pose, using a single grasp as shown in Figure 1. In Figure 1a, the initial pose of the part to be assembled with the sub-assembly is shown, while Figure 1b shows the desired assembly formation of the parts. It can be seen that it is not possible to assemble the part in single pick, and place cycle. That is why, one or more re-grasping of a part may be needed to assemble it. Part manipulation approach using re-grasping is presented in next sub-section.

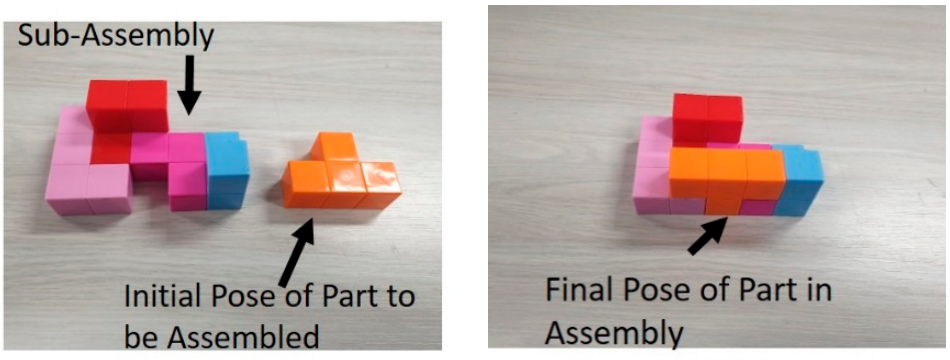
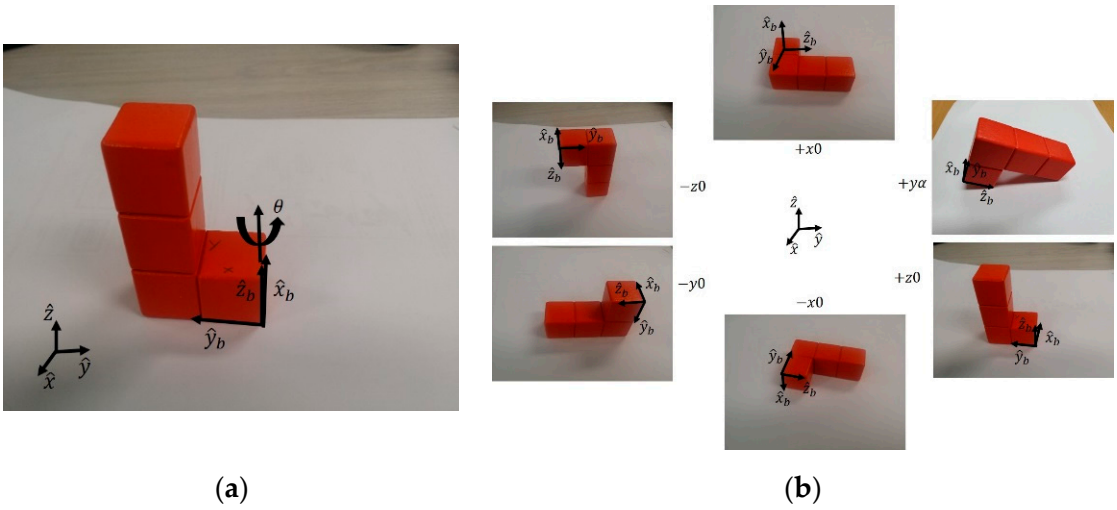


Figure 1. (**a**) The initial pose of the part (to be assembled) and sub-assembly are shown. (**b**) The final pose of the part in the assembly is shown. It can be seen that there is no feasible grasp to pick the part from its initial pose and move to its final pose without part manipulation.

3.2. Re-Grasping Approach

To perform the assembly task, there must be at least one feasible grasp for each pair of initial and final poses of a part. In case, if there does not exist any feasible grasp then part has to undergo an intermediate pose(s) before going to it final pose. One possibility can be in-hand manipulation such as used in [16] but it is computationally costly. We propose a re-grasping approach for part manipulation using "orientation graph search" which is explained below.

We assume that at the intermediate pose, the parts are placed on a flat horizontal surface. A pose of a part is stable if part can sustain its pose without external support. At a stable pose, the part may be resting on one of its face or on edges or on vertices as shown in Figure 2b. Given a stable pose, any pose obtained by rotating the part along the vertical axis, without losing contact with ground, is also stable as shown in Figure 2a. We call the collection of poses obtained by rotating a given stable pose a, along the vertical axis an "orientation family" $\hat{a}$. We assume that for any part, there are a finite number of orientation families.



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

Figure 2. Part Orientation family example. (**a**) The local frame attached to part is represented with unit vectors $\{\hat{x}_b,\ \hat{y}_b, \hat{z}_b\}$. The orientation of part is represented with respect to the inertial frame $\{\hat{x}, \hat{y}, \hat{z}\}$. If the part is rotated about the $\hat{z}_b$ axis by any angle θ, the $\hat{z}_b$ will have same orientation with respect to $\hat{z}$ axis and the resulting pose will also be stable. (**b**) Different stable orientation families are represented. In each orientation family, one of the body frame axis has constant angle with $\hat{z}$ of inertial frame.

Using all orientation families of a part based on its geometry, we build an orientation graph as shown in Figure 3. Each node of the orientation graph represents an orientation family of the part. Two nodes in orientation graph are connected bi-directionally by an edge if both nodes represent stable orientation families and there exists at least one feasible grasp between them. While building an orientation graph, we consider the collision between environment and hand, but not the robot body. Also, at this point, we do not consider the initial or final pose of the part in assembly. The orientation graph for a part is shown in Figure 3. Note that this graph can be computed given just the geometry of a part and the hand. The final pose, the geometry of other parts, and assembly sequences are not needed. The example of the part manipulation using re-grasping approach is presented in Figure 4.

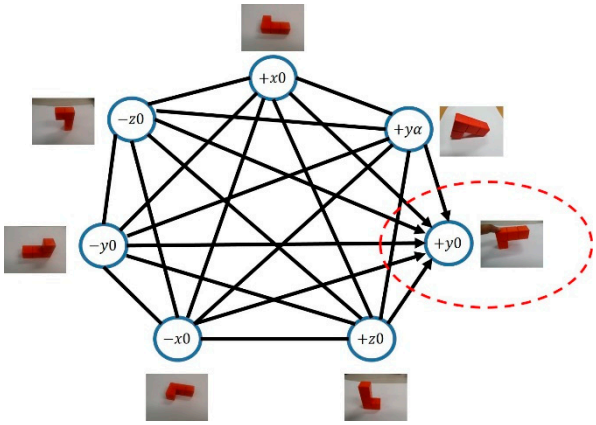


Figure 3. Orientation graph. Orientation families are represented as nodes of the graph and manipulation between two nodes is represented as an edge. An unstable orientation which can be a final pose of the part (encircled in figure with dotted line) in assembly is connected with stable orientations with only incoming edges. But the stable orientations lying on the flat surface of part geometry can have un-directed edges.

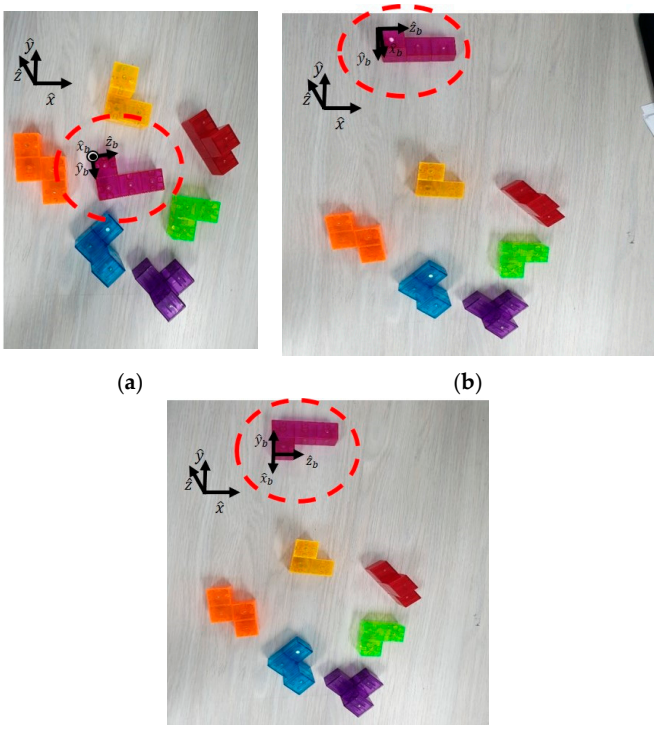


Figure 4. Part manipulation for L-shaped part (encircled in (**a**–**c**)) using orientation graph search. (**a**) Initial poses of the parts. (**b**) Intermediate pose of L-Shaped part after picking from its initial pose. (**c**) Final pose of L-shaped part. In the initial pose of the part, it can only be picked with the vertical approach direction of hand due to other parts in the neighborhood. In first manipulation, part is moved from orientation $+x0$ (in (**a**)) to $-y0$ (in (**b**)). In second manipulation, part is moved from $-y0$ (in (**b**)) to $-x0$ (in (**c**)).

In the planning phase, we can perform re-grasp planning (if necessary) using the orientation graph search. First, at the beginning of the planning for a given piece, if the final pose of the part does not belong to any family in orientation graph (such as an inherently unstable pose if placed on flat surface, but that can be supported by other part in the assembly) then we add it as an extra node

to the graph. This node is connected to other nodes in the graph using directed edges from stable poses using the same method above. For a part to be assembled, given its initial, and final poses and pre-assembled parts before it, if initial and final poses are connected by an edge in the graph, and there exists a feasible grasp in presence of assembly constraints that means the part can be grasped and moved to its assembled pose without re-grasping. Otherwise re-grasping is needed, and intermediate poses can be found by searching a path from initial to final pose in the orientation graph other than a direct edge between them. The position of an intermediate pose can be any position in the dexterous workspace of the manipulator other than the initial or final position of the part. The next sub-section describes the main parts of the planning algorithm.

3.3. Integrated Part Manipulation and Robot Motion Planner

This section presents the RRT-based integrated motion planning algorithm to do grasp planning, motion planning for robot to move part to target pose and part manipulation to perform an assembly task. The outer loop of the algorithm is given in Algorithm 1. Let K be the number of the parts to be assembled. Knowing the geometries of the parts and their initial and final poses, we can find the sequence of poses for ith part as $\overline{P}_i = \left\{\overline{P}_{i,1}, \overline{P}_{i,2}, \ldots \overline{P}_{i,a}\right\}$ which are needed to grasp a part from its initial pose and move it to its final pose through intermediate poses, if any. If ith part has to undergo a number of poses during assembly process then $\overline{P}_{i,1}\left(=\overline{P}_{i,init}\right)$ is its start pose and $\overline{P}_{i,a}\left(=\overline{P}_{i,final}\right)$ is its final pose. The set of poses $\overline{P}_i$ for ith part can be found by searching the orientation graph OG_i for given $\overline{P}_{i,init}$ and $\overline{P}_{i,final}$ using the part manipulation approach presented in Section 3.2. In a one pick and place cycle for a part from jth pose to $(j+1)$th pose then $\overline{P}_{i,j}$ will be named as the start pose and $\overline{P}_{i,j+1}$ will be named as target pose. Let $\overline{P} = \left\{\overline{P}_1, \ldots, \overline{P}_K\right\}$ be the set containing the sequences of poses of all part to be assembled. According to the sequence of the part assembly, parts are grasped from their start pose using Grasp_Part algorithm which is presented in Section 3.3.3 and moved to target pose using Move_Part algorithm presented in Section 3.3.4 until the part reaches its final pose. Since the proposed algorithm is based on RRT algorithm which is a sampling based motion planning algorithm. Recall that an RRT based algorithm is basically a single query algorithm [11]. On contrary, the proposed algorithm is for a multi-stage problem because it plans sequentially for pairs of start and target poses for each part to be assembled. The collision check strategy, which is presented in Section 3.3.1, considers the changing task space at each stage of the assembly task. Due to changing conditions of task space while performing an assembly task, the robot state that it is collision free or not, may vary from one stage of assembly to next. To handle this issue, the algorithm maintains a set of failed nodes (that is, the nodes which are in collision with environment in later stages) as forest roots (FR) which, at later stages, are attempted to be connected to main tree while growing tree in *Two_Stage_Extend_RRT* which is presented in Section 3.3.2.

Algorithm 1. Motion Planner for Assembly Task

1. *For* $i = 1 : K$
2. OG_i = *Build Orientation graph for ith part*
3. $\overline{P}_i$ = *Graph_Search* (OG_i, $P_{i,init}$, $P_{i,final}$)
4. *End For*
5. *For* $i = 1 : K$ // Part Index
6. *For* $j = 1 : Size\left(\overline{P}_i\right) - 1$ // Pose Index
7. *Grasp_Part*$(\overline{P},\ i,\ j)$
8. *Move_Part*$\left(\overline{P},\ i,\ j\right)$
9. *End For*
10. *End For*

3.3.1. Collision Checking in Changing Task Space

During the robotic assembly operation, objects in the task space are not continuously changing their position and orientation but only the grasped part moves with robot hand. Since at each step of the assembly process, one part is moved from start pose to target pose, we consider the task space as quasi-dynamic space. Due to change in task space at every step of assembly task, the robot configuration which was collision free in previous steps may not be necessarily collision free in current step of assembly as shown in Figure 5. That is why at every stage we need to make sure that nodes in the searched path are collision free with respect to current state of the task space. We are using semi-lazy collision checking which is as follows:

- When a node is generated, it is checked that robot is not in self-collision as well as not in collision with static environment obstacles which includes floor and other static equipment in the task space. Though the collision checking for the nodes is performed again after searching the path, collision checking for nodes at earlier stage minimizes the chances of failure at later stage. This is because a configuration node which is collision free while adding to tree, later on it may only be in collision with some movable obstacles.
- Collision along the edges of the tree are not checked while growing the tree because it consumes a lot of time. In order to avoid the un-necessary computations, the collision checking along the edge is performed only for those which are included in the searched path.
- If an edge along the path is found to be in collision, this edge is deleted and the child node is stored in the forest roots (FR). That is, if an edge (a,b) is found to be in collision, and the node a is the parent node, the node b, the edge (a,b) is deleted and the node b is put into FR. Note that the node b is the root of a tree which is disconnected from the main tree.

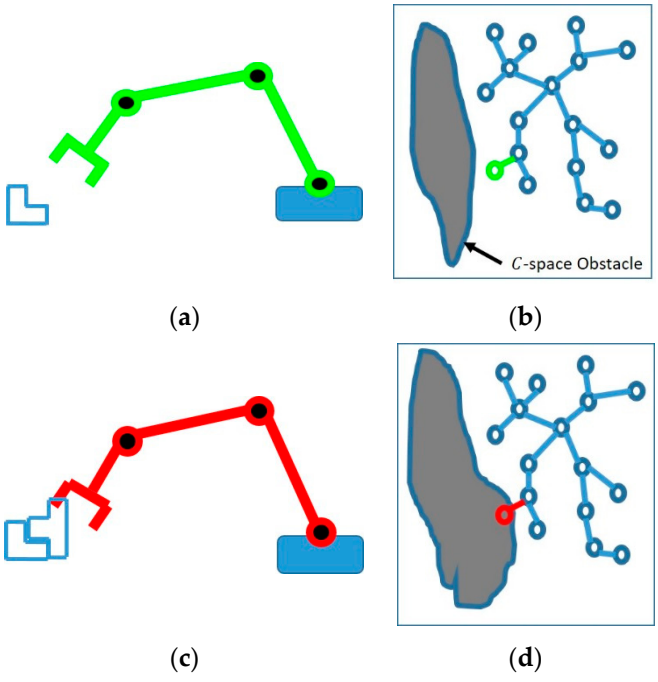


Figure 5. Collision status of a robot state in changing task space. (**a**,**b**) shows that the robot configuration is collision free in initial state of task space. (**c**,**d**) shows the robot state, which was collision free in initial stage of assembly task, is now in collision with environment due to change in task space.

The failed nodes in FR are tried to be connected to the main tree while extending the tree. The two stage extend tree is presented in the next sub-section.

3.3.2. Extending Tree in Two Steps

Due to the multi-query nature of the motion planning problem under consideration and the collision checking strategy presented in Section 3.3.1, the conventional approach to extend the tree needs to be modified as Two_Stage_Extend_RRT. The pseudo code is presented in the Algorithm 2. Considering a set of the forest roots (FR) containing failed nodes which were part of the tree but were disconnected from tree due to the edge found in collision while finding a collision free path. The main idea of the two stage extend tree is following,

- In the 1st stage, a new collision free node q_{new} is added to the main tree in a conventional way using *Extend_Tree* function but without checking collision along the edge connecting new node to the tree.
- In the 2nd stage, the algorithm tries to connect the main tree and one of the disconnected trees using q_{new}. First it finds a closest node q_{cl} in FR, which is closest to q_{new}. Then we extend the main tree from q_{new} towards q_{cl}. If they are connected, it means the tree having q_{cl} as the root is reconnected to the main tree. If so, q_{cl} is removed from FR. If the fails, the node during the connecting attempt is returned.
- For each q_{new}, either in Stage 1 or 2, it is associated to one of the grasps for each part yet to be assembled including the part manipulation sequences. The Node association to a grasp point is described below.
- Similar to the grasp-RRT, the newly added node q_{new} is associated to one of the grasps for each part yet to be assembled including the part manipulation sequences (but not to the surface of an abstract object like, grasp sphere in grasp-RRT). The details are described below.

Algorithm 2. *Two_Stage_Extend_RRT*$\left(\mathcal{T},\ \overline{P}\ ,\ i, j\right)$

1. *For* $N = 1:2$
2. $\quad if\ (N = 1)$
3. $\quad\quad q_{new} = Extend\_Tree(\mathcal{T})$
4. $\quad else$
5. $\quad\quad q_{new} = Extend\_to\_Forest(\mathcal{T},\ q_{new},\ FR)$
6. $\quad End\ if$
7. $\quad For\ m = i:K$ // *Part index*
8. $\quad\quad\quad if(m = i)\ \overline{o} = j$
9. $\quad\quad\quad else\ \ \overline{o} = 1$
10. $\quad\quad\quad End\ if$
11. $\quad\quad For\ o = \overline{o} : Size\left(\overline{P}_m\right) - 1$ // *Pose index for ith part*
12. $\quad\quad\quad \overline{fg} = \underset{fg_k \in FG}{argmax}(m_k)\ and\ (d > d_{safe})$
13. $\quad\quad\quad \overline{fg}.Asc\_node = add\left(\overline{fg},\ q_{new}, d\right)$
14. $\quad\quad\quad End\ For$
15. $\quad End\ For$
16. $\quad End\ For$

The node association with one of the feasible grasps for a part is elaborated in Figure 6. Let q_{new} be a configuration of the robot as a new node added to tree. Let gcp_{new} be the grasp center point of hand in robot configuration q_{new}, defined at the center of palm of the robot hand and $\hat{z}_{hand,new}$ be the z-axis of hand frame at q_{new}, defined as normal to the palm of the hand. Recall that FG be the set of feasible grasps for ith part in jth pose. Let $fg \in FG$ then $fg.\hat{z}_{hand}$ be the $z-axis$ of the hand frame at fg and $fg.gcp$ be the position of grasp center point of hand at fg. Let d be the component of the distance from gcp_{new} to $fg.gcp$ along $-(fg.\hat{z}_{hand})$ and d_{safe} be a safe distance margin for hand from a feasible grasp which is defined as distance of tip of a finger from $fg.gcp$ when the finger a fully extended along $fg.\hat{z}_{hand}$. While trying to approach the part from q_{new}, the hand may need to re-orient itself before

reaching the grasp pose $P_{i,grasp}$. That is why the distance d should be at least greater than d_{safe}. Each candidate of the feasible grasp is evaluated according to metric m which is as follows: Let,

- α be the dot product of the $(fg.\hat{z}_{hand})$, and $\hat{z}_{hand,new}$.
- β be the dot product of the $-(fg.\hat{z}_{hand})$, and unit vector along line jointing $fg.gcp$ to gcp_{new}.

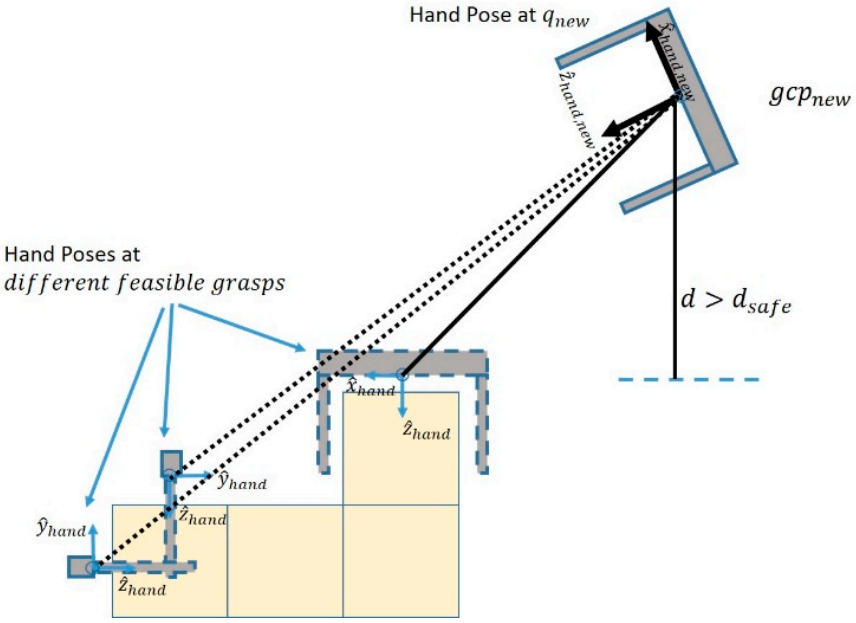


Figure 6. The idea of node association to one of the feasible grasp for a part is presented. When a q_{new} be a new node added to the tree in $C-space$, d be the distance of gcp_{new} from feasible grasp of the part. If the component of the d along the $-\hat{z}_{hand}$ of the feasible grasp is greater than d_{safe} (minimum safe distance of hand from part to change orientation of hand), then the new node will be associated to the feasible grasp which has normal more closely aligned to the line joining the grasp center point and center of the square.

The metric m is a weighted sum of α and β. The node q_{new} is associated to the feasible grasp fg among the candidate feasible grasps which has maximum value of m and $d > d_{safe}$. This node association with a grasp point helps in choosing a grasp point during grasp planning which is presented in next sub-section.

3.3.3. Grasp Planning

The grasp planning part of the proposed algorithm is a modified version of Grasp-RRT [12]. Recall that we choose a set of feasible grasps (not just one grasp) considering the hand geometry and the assembly so that the hand and the part/pre-assembled parts are collision free. But the whole robot geometry is not considered then. Similar to grasp RRT, the algorithm will try to grasp current object while expanding the tree, but only using the feasible grasps computed earlier. If a collision free path is found from a node in a tree to a node whose hand pose is the one of the feasible grasp, it means the part can be grasped using that grasp. Note that a grasp being feasible does not mean that it can actually grasp the part; for example, there is no configuration for that grasp, or even if such a configuration exists, it may not be connected to the initial (e.g., home) configuration of the robot. That is why we keep a set of grasps (not just one grasp), and try to connect them while growing tree. The pseudo code for grasp planning algorithm is presented in Algorithm 3.

Algorithm 3. *Grasp_Part*$\left(\overline{P},\ i, j\right)$

1. *while*(*Grasp attempt failed*)
2. *Two_Stage_Extend_RRT* ($\mathcal{T}$, $\overline{P}$, *i*, *j*)
3. *if* (*rand* < *pr*)
4. $A = set\ of\ feasible\ grasp\ of\ ith\ part\ in\ jth\ pose$
5. $fg = \underset{fg_j \in A}{argmax}\left(size\left(fg_j.Asc\_node\right)\right)$
6. $q_{cand} = \underset{q \in fg.Asc\_node}{argmin}\ (q.d)$
7. *Find_Path*($\mathcal{T}_{root}, q_{cand}$)
8. *if* (*failed*)
9. *Remove*($fg.Asc\_node$, q_{cand})
10. *Grasp attempt failed*
11. *End if*
12. $P_{pregrasp,hand}$, $P_{grasp,hand} \leftarrow Compute\ Hand\ Pose\ for\ fg$
13. *while* $\left(Grasp\ attempt\ is\ not\ failed\ and\ P_{pregrasp}\ is\ not\ reached\right)$ *do*
14. *Approach*$\left(P_{pregrasp}, q_{cand}\right)$
15. *if*(*failed*)
16. *Remove*($fg.Asc\_node$, q_{cand})
17. *Grasp attempt failed*
18. *break*
19. *End if*
20. *End while*
21. *while* $\left(Grasp\ attempt\ is\ not\ failed\ and\ P_{grasp}\ is\ not\ reached\right)$ *do*
22. *Approach*$\left(P_{grasp}, P_{pregrasp}\right)$
23. *if* (*failed*)
24. *Remove*($fg.Asc\_node$, q_{cand})
25. *Grasp attempt failed*
26. *break*
27. *End if*
28. *End while*
29. *End if*
30. *End while*
31. *Close_fingers*()

The grasp planning algorithm continues to grow the tree using the Two_Stage_Extend_RRT and attempts to grasp the part from given start pose with the probability *pr*. From the set of feasible grasps *FG* for $\overline{P}_{i,j}$ pose for *ith* part in *j*th pose, choose a feasible grasp $fg \in FG$ with maximum number of associated nodes $fg.Asc\_node$. Afterwards, a candidate node q_{cand} is chosen from $fg.Asc\_node$ such that gcp_{cand} (grasp center point of hand in configuration q_{cand}) has minimum distance from the $fg.gcp$. This is because while approaching the part from nearest candidate node the grasp attempt will have less chances of failure due to violation of joint limits and collision. Let P_{grasp} be the pose the hand grasping part at $\overline{P}_{i,j}$. Knowing the fg, we can define a grasp pose P_{grasp} and a pre-grasp pose $P_{pregrasp}$. In the next step, the algorithm tries to approach the P_{grasp} via $P_{pregrasp}$ in task space from gcp_{cand} using *inverse jacobian approach* and incrementally adding new node to the tree. While approaching the part using inverse jacobian approach, if a new node is found to be in collision or violating the joint limits of the robot then, the algorithm stops to reach fg from q_{cand} and q_{cand} is removed from $fg.Asc\_node$. If the grasp attempt failed, the grasp planning algorithm re-starts again until the part is grasped successfully. Once the grasp pose is reached, the fingers are closed to grasp the part.

3.3.4. Move Part

After grasping the ith part at pose $\overline{P}_{i,j}$, it has to be moved to target pose $\overline{P}_{i,j+1}$. Algorithm 4 presentes the pseudo code for the move part planning. It starts with finding the set of feasible approach direction to reach the target pose of the part. In [26], an approach based on "non-directional blocking graph" is presented to dis-assemble a part from an assembly by evaluating the geometry of parts in it. The main idea of non-directional blocking graph was that if we want to remove a part x from an assembly using translational motion then how the other parts in assembly block the motion of x. Using non-directional blocking graph, we can identify a set of directions of translational motions to remove a part from a given assembly. Using the same approach in reverse, given a sub-assembly of parts (including the ground surface) we can identify a set of feasible approach directions to place a part in given target pose $\overline{P}_{i,j+1}$. However, for an intermediate target pose, there is no subassembly constraining the approach direction to place part on ground, except the ground itself. Let w be the set of feasible approach directions to place the grasped part in target pose. After finding w, one of the feasible approach directions $\hat{a} \in w$ is randomly selected in order to attempt to place the part in $\overline{P}_{i,target}$.

Since parts are assumed to have flat faces that is why placing the part in $\overline{P}_{i,j+1}$ may become a narrow gap problem for motion planning due to presence of already assembled parts. Though the narrow gap problem is solvable using RRT, however it takes relatively longer time to find a solution [27]. Let $\overline{P}_{i,j+1}$ be the target pose $\overline{P}_{i,target}$ while moving the part. To find a solution faster, a pre-target pose $P_{preTarget}$ is defined from where $\overline{P}_{i,target}$ can be approached along a straight line. $P_{preTarget}$ has same orientation as that of in $\overline{P}_{i,target}$, however it is positioned at a safe distance d_{preasm} from $fg.c$ along $-\hat{a}$. The d_{preasm} can be chosen as multiple of finger length. The node $q_{preTarget}$ for $P_{preTarget}$ is computed using inverse kinematics. Basic-RRT is used to connect q_{grasp} to $q_{preTarget}$. Once $q_{preTarget}$ is reached, the inverse jacobian is used to approach $\overline{P}_{i,target}$ from $P_{preTarget}$. If the assembly of current part is failed due to collision or violation of joint limits, then the currently chosen assembly approach direction will be deleted from set w. The algorithm will re-start finding path by again randomly choosing assembly approach direction from remaining set of assembly approach directions. It continues until it finds the solution.

The experimental evaluation of the proposed planning algorithm is presented in next section.

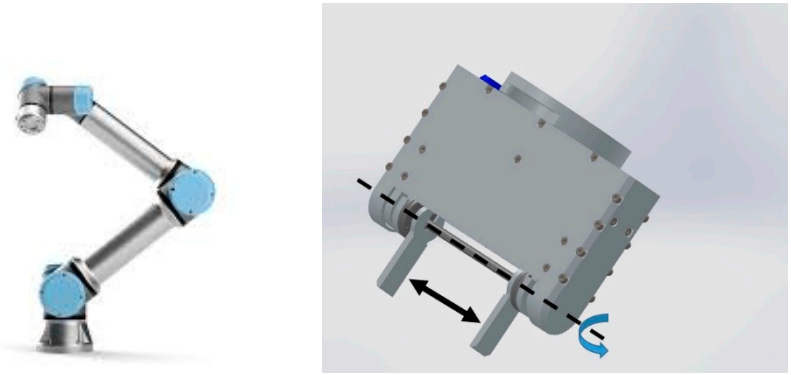
Algorithm 4. *Move_Part*$\left(\overline{P},\ i\ ,\ j\right)$

1. $w = Set\ of\ approach\ directions\ for\ P_{i,j+1}$
2. *while*$\left(\overline{P}_{i,j+1}\ is\ not\ reached\right)$
3. $\hat{a} = randomly\ choose\ an\ element\ from\ w$
4. $P_{preTarget},\ P_{Target} \leftarrow \left(\overline{P}_{i,j+1}, \hat{a}\right)$
5. *while* $\left(q_{preTarget}\ is\ not\ connected\ in\ \mathcal{T}\right)$ *do*
6. *Two_Stage_Extend_RRT* $(\mathcal{T},\ \overline{P},\ i,\ j)$
7. *if* $(rand < pr)$
8. *RRT_Connect*$\left(\mathcal{T}_{root}, q_{preTarget}\right)$
9. *End if*
10. *End while*
11. *while* $\left(\overline{P}_{i,\ target}\ is\ not\ reached\right)$ *do*
12. *Approach*$\left(\overline{P}_{i,\ target},\ P_{preTarget}\right)$
13. *if* $(failed)$
14. *delete* $\hat{a}$ *from* w
15. *break*
16. *End if*
17. *End while*
18. *End while*
19. *Open_fingers*

4. Experiment Results

This section presents the experimental evaluation of the proposed planning algorithm by assembling the soma puzzle pieces in different formations. We used Universal Robot UR5e which is an industrial robot with 6 articulating degrees of freedom as shown in Figure 7a. The Denavit-Hartenberg (DH) parameters of UR5e are given Table 1. We used a parallel gripper shown in Figure 7b which has two degrees of freedoms for fingers

- Linear motion to open and close the fingers. Both fingers move simultaneously.
- Rotation of fingers by $\pm\frac{\pi}{2}$ about the axis of linear motion of fingers.



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

Figure 7. (**a**) UR5e Robot manipulator (**b**) Parallel gripper with 2-dof of fingers. Both fingers of the parallel gripper can simultaneously rotate about the axis of rotation shown as dotted line. Both fingers can also translate simultaneously to grasp and release the part.

Table 1. DH parameters of UR5e robot manipulator.

α_i	a_i (In mm)	d_i (In mm)	θ_i
$\frac{\pi}{2}$	0	162.5	θ_1
0	−425	0	θ_2
0	−392.2	0	θ_3
$\frac{\pi}{2}$	0	133.3	θ_4
$-\frac{\pi}{2}$	0	99.7	θ_5
0	0	99.6	θ_6

Linear motion is to grasp and release the part while the fingers rotation is to manipulate the part.

Soma puzzle consisted of seven pieces with different shapes and it could be assembled in different 3D formations. All pieces had faces as flat surfaces. The surface normals of two consecutive surfaces were orthogonal to each other. A set of feasible grasps for a pair of start and target poses could be found by evaluating the geometry of part and gripper. A feasible grasp considering parallel gripper for a given pair of start and target poses of a part was considered to be one for which there was a combination of three exposed surfaces s_1, s_2 and s_3 as follows.

- An exposed surface s_1 with surface normal anti-parallel to hand approach axis while grasping the part
- A pair of exposed surfaces s_2, and s_3 for which the surface normals were anti-parallel to each other but both should have been orthogonal to surface normal of s_1.
- While placing grasped part in final pose using s_1, s_2 and s_3 as mentioned above, the hand should not have interfered with environment and pre-assembled parts.

A feasible grasp using the parallel gripper is presented in Figure 8.

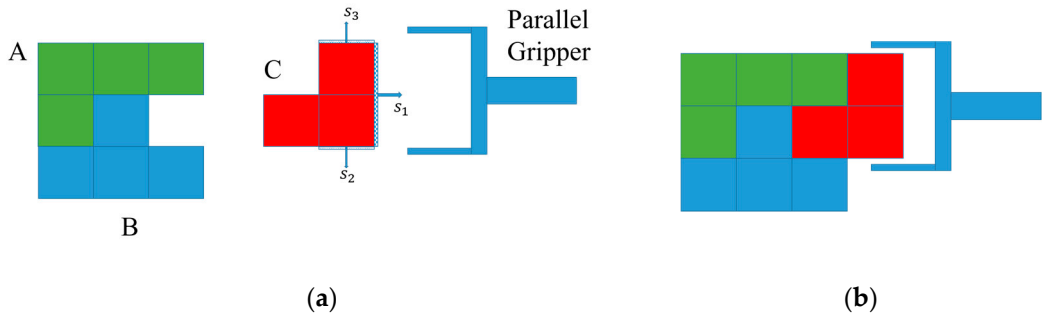
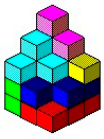
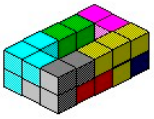
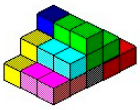
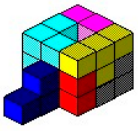
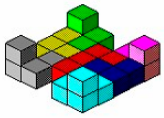


Figure 8. The idea of the feasible grasp using parallel gripper is presented using 2D example. (**a**) Part A (Green), and part B (Blue) are already assembled and part C (Red) is to be assembled. Part C contains three surfaces s_1, s_2, and s_3 which define a feasible grasp for parallel grasping as defined in Section 3.1. The surfaces for feasible grasp are exposed in initial pose (in (**a**)) as well as final pose (in (**b**)) pose of the part C.

The algorithm is implemented in C++ in Linux based system. The algorithm is executed on PC with Core i7 CPU with 1.8 GHz and 16 Gigabyte RAM. The proposed algorithm is implemented using open source Open Motion Planning Library (OMPL) [28] and it is demonstrated by assembling the soma puzzle pieces in different formations as shown in Figures 9–11 and Table 2. Figure 9 shows the step by step picking and placing poses of the hand and part in the assembly process. The Figure 9i–l show the part manipulation by re-grasping and Figure 9s shows the final assembly formation. Figures 10 and 11 show the assembly poses of the robot hand while assembling parts. The computation time in seconds of the three trials of five different formations is shown in Table 2. Additionally, a supplementary video has been added to show the execution of the assembly for a 4-piece formation of the soma puzzle using URe5 robot and the parallel gripper mentioned above.

Table 2. Computation time in seconds of planning for three different assembly formations.

Trials	Formation 1	Formation 2	Formation 3	Formation 4	Formation 5
1	129.3	121.6	81.8	99.7	99.2
2	116.4	133.1	104.3	118	84.8
3	136.5	121.7	86.1	110.2	94.1
Mean	127.4	125.5	90.7	109.3	92.7
FormationShape					

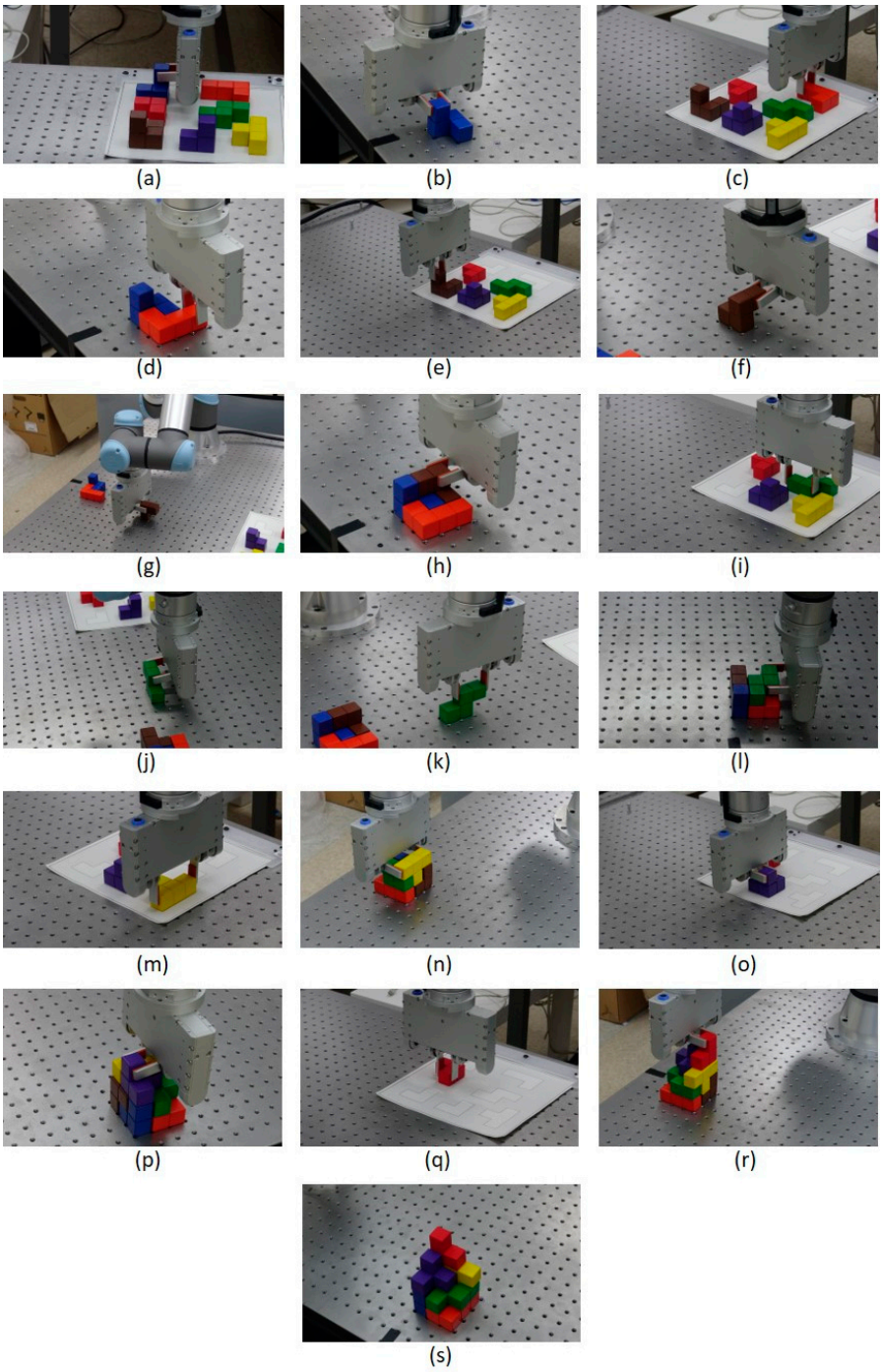


Figure 9. Soma puzzle assembly in formation 1 (shown in (**s**)). From (**a**–**r**) the figures show the snapshots of the grasping and assembly poses of the gripper and part during the assembly process. Figures (**e**–**h**) and (**i**–**l**) shows the part manipulation through intermediate poses of parts.

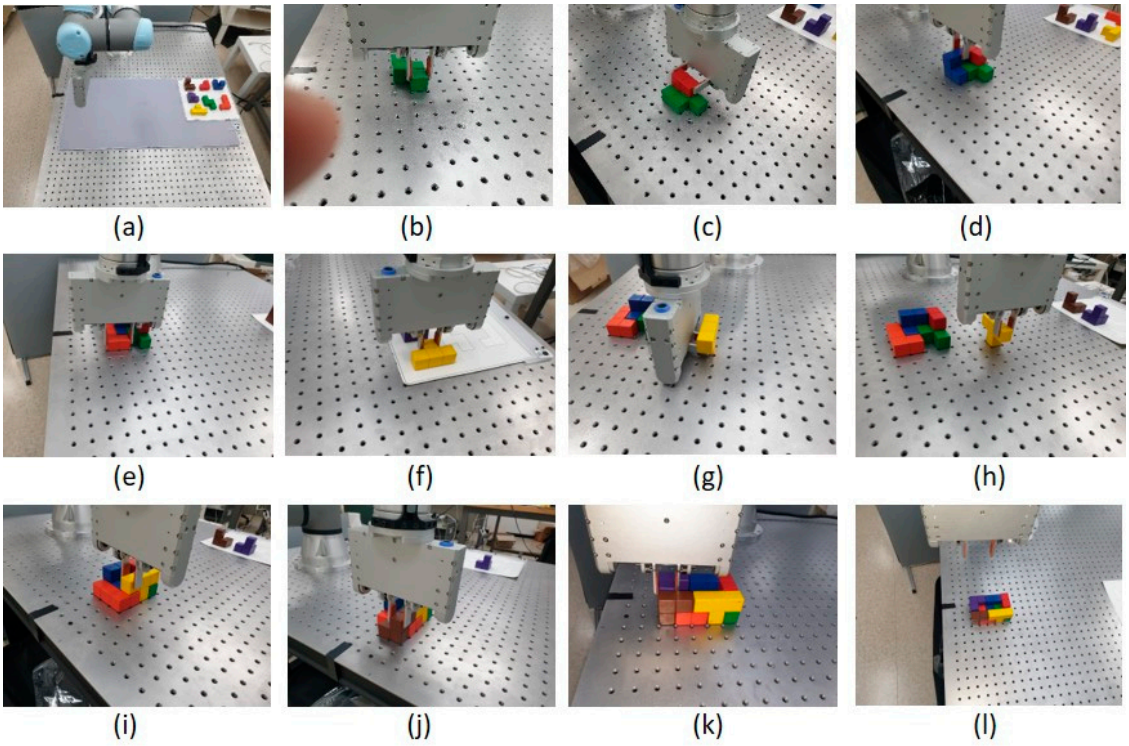


Figure 10. Soma puzzle assembly in formation 2 (shown in (**l**)). From (**a**–**k**) the figures show the snapshots of the assembly poses of the gripper and part during the assembly process.

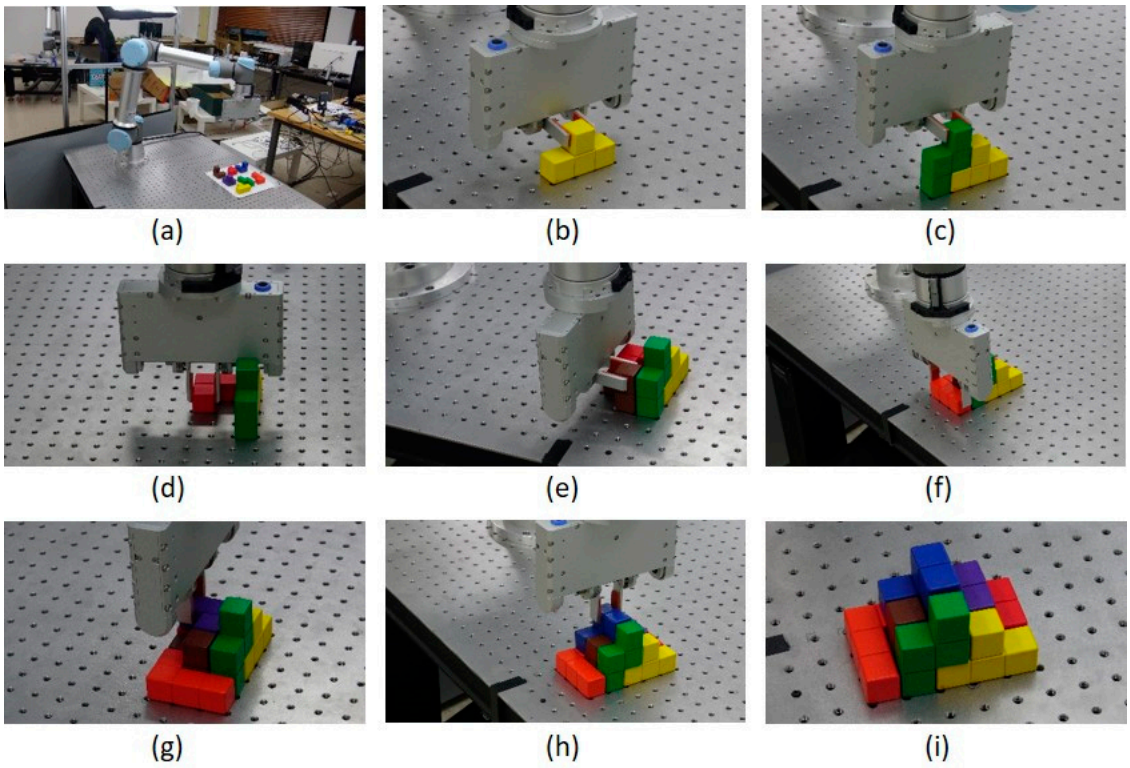


Figure 11. Soma puzzle assembly in formation 3 (shown in (**i**)). From (**a**–**h**) the figures show the snapshots of the assembly poses of the gripper and part during the assembly process.

5. Conclusions and Future Work

This paper proposed an integrated planning algorithm to perform an assembly task using robotic manipulator. The algorithm takes into account the geometry of parts and their initial, and final poses to choose feasible grasps. Afterwards, it chooses a feasible grasp while planning motion of manipulator to perform assembly task. For this, each node in the tree is associated with one of the feasible grasp for each parts yet to be assembled. If the part manipulation is necessary to do an assembly operation, a re-grasping approach is proposed using orientation graph search (for each part). The algorithm finds the sequences of poses of a part between its initial and final poses so that the part can be assembled without interfering the obstacles in the environment. While assembly multiple parts, the task space undergoes changes at every step of assembly, making some previously collision-free nodes be in collision. To handle this and to be able to use the disconnected parts of the tree, a two-stage-extend-RRT method is proposed in which while growing a tree, attempts are made to reconnect the disconnected parts to the main tree.

The motion planning to perform an assembly operation is inherently a multi-query problem. While planning feasible configuration space paths for grasping a part and assembling it, often they are connected through the root node of the tree, resulting in an inefficient path. One can optimize the solution after finding the collision free paths of the assembly operation or reconnect the disconnect trees so that the path between the start and the target poses are in general shorter. In addition, the proposed planner considers a pre-computed set of grasps are given at the beginning. Later on, it chooses feasible grasp during the motion planning phase. In future work, the authors are intended to integrate the grasp planning in the algorithm. In future, the authors are intended to incorporate the safety issue while planning and executing the algorithm using proximity sensor such as [10,29].

Supplementary Materials: The following are available online at http://www.mdpi.com/2076-3417/10/3/749/s1.

Author Contributions: This research work was conducted jointly by A.A. and J.Y.L. This research idea was proposed and supervised by J.Y.L. The I implementation and experiments were conducted by A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Technology Innovation Program (or Industrial Strategic Technology Development Program) (20001856, Development of robotic work control technology capable of grasping and manipulating various objects in everyday life environment based on multimodal recognition and using tools) funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Owen, T. *Assembly with Robots*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
2. Conte, G.; Scaradozzi, D.; Casalino, G.; Simetti, E.; Sperindè, A.; Torelli, S. A robotic platform for underwater assisted manipulation. In Proceedings of the International Offshore and Polar Engineering Conference, Rhodes, Greece, 26 June–2 July 2016; pp. 491–496.
3. Barbieri, L.; Bruno, F.; Gallo, A.; Muzzupappa, M.; Russo, M.L. Design, prototyping and testing of a modular small-sized underwater robotic arm controlled through a master-slave approach. *Ocean Eng.* **2018**, *158*, 253–262. [CrossRef]
4. Ghandi, S.; Masehian, E. Review and taxonomies of assembly and disassembly path planning problems and approaches. *Comput.-Aided Des.* **2015**, *67*, 58–86. [CrossRef]
5. Jiménez, P. Survey on assembly sequencing: A combinatorial and geometrical perspective. *J. Intell. Manuf.* **2013**, *24*, 235–250. [CrossRef]
6. Zhu, Z.; Hu, H. Robot learning from demonstration in robotic assembly: A survey. *Robotics* **2018**, *7*, 17. [CrossRef]
7. Choset, H.M.; Hutchinson, S.; Lynch, K.M.; Kantor, G.; Burgard, W.; Kavraki, L.E.; Thrun, S. *Principles of Robot Motion: Theory, Algorithms, and Implementation*; MIT Press: Cambridge, MA, USA, 2005.
8. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.

9. Bagnell, J.A.; Cavalcanti, F.; Cui, L.; Galluzzo, T.; Hebert, M.; Kazemi, M.; Klingensmith, M.; Libby, J.; Liu, T.Y.; Pollard, N.; et al. An integrated system for autonomous robotics manipulation. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, 7–12 October 2012; pp. 2955–2962.
10. Mayton, B.; LeGrand, L.; Smith, J.R. An electric field pretouch system for grasping and co-manipulation. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010; pp. 831–838.
11. Kuffner, J.J., Jr.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000.
12. Vahrenkamp, N.; Asfour, T.; Dillmann, R. Simultaneous grasp and motion planning: Humanoid robot armar-iii. *IEEE Robot. Autom. Mag.* **2012**, *19*, 43–57. [CrossRef]
13. Lozano-Pérez, T.; Jones, J.; Mazer, E.; O'Donnell, P.; Grimson, W.; Tournassoud, P.; Lanusse, A. Handey: A robot system that recognizes, plans, and manipulates. In Proceedings of the 1987 IEEE International Conference on Robotics and Automation, Raleigh, NC, USA, 31 March–3 April 1987; pp. 843–849.
14. Sánchez, D.; Wan, W.; Harada, K.; Kanchiro, F. Regrasp planning considering bipedal stability constraints. In Proceedings of the 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), Beijing, China, 6–9 Novermber 2018; pp. 271–277.
15. Siméon, T.; Laumond, J.P.; Cortés, J.; Sahbani, A. Manipulation planning with probabilistic roadmaps. *Int. J. Robot. Res.* **2004**, *23*, 729–746. [CrossRef]
16. You, J.-S.; Kim, D.-H.; Lim, S.-J.; Kang, S.-P.; Lee, J.Y.; Han, C.-S. Development of manipulation planning algorithm for a dual-arm robot assembly task. In Proceedings of the 2012 IEEE International Conference on Automation Science and Engineering (CASE), Seoul, Korea, 20–24 August 2012; pp. 1061–1066.
17. Wan, W.; Harada, K. Integrated single-arm assembly and manipulation planning using dynamic regrasp graphs. In Proceedings of the 2016 IEEE International Conference on Real-Time Computing and Robotics, RCAR 2016, Angkor Wat, Cambodia, 6–10 June 2016; pp. 174–179.
18. Wan, W.; Harada, K.; Nagata, K. Assembly sequence planning for motion planning. *Assem. Autom.* **2017**, *38*, 195–206. [CrossRef]
19. Jaillet, L.; Cortés, J.; Siméon, T. Transition-based RRT for path planning in continuous cost spaces. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Nice, France, 22–26 September 2008; pp. 2145–2150.
20. Sánchez, G.; Latombe, J.-C. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Robotics Research*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 403–417.
21. Zucker, M.; Kuffner, J.J.; Branicky, M.S. Multipartite RRTs for rapid replanning in dynamic environments. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 1603–1609.
22. Hang, K.; Stork, J.A.; Pollard, N.S.; Kragic, D. A framework for optimal grasp contact planning. *IEEE Robot. Autom. Lett.* **2017**, *2*, 704–711. [CrossRef]
23. Liu, S.; Carpin, S. Global grasp planning using triangular meshes. In Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 26–30 May 2015; pp. 4904–4910.
24. Mahler, J.; Liang, J.; Niyaz, S.; Laskey, M.; Doan, R.; Liu, X.; Ojea, J.A.; Goldberg, K. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv* **2017**, arXiv:1703.09312.
25. Mahler, J.; Pokorny, F.T.; Hou, B.; Roderick, M.; Laskey, M.; Aubry, M.; Kohlhoff, K.; Kroger, T.; Kuffner, J.; Goldberg, K. Dex-net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In Proceedings of the IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 1957–1964.
26. Wilson, R.H.; Latombe, J.C. Geometric reasoning about mechanical assembly. *Artif. Intell.* **1994**, *71*, 371–396. [CrossRef]
27. Yershova, A.; LaValle, S.M. Motion planning for highly constrained spaces. In *Lecture Notes in Control and Information Sciences*; Springer: Berlin/Heidelberg, Germany, 2009; Volume 396, pp. 297–306.

28. Şucan, I.A.; Moll, M.; Kavraki, L. The open motion planning library. *IEEE Robot. Autom. Mag.* **2012**, *19*, 72–82. [CrossRef]
29. De Leo, A.; Scaradozzi, D.; Genovesi, R.; Cerri, G.; Conte, G.; Perdon, A.M.; Omerdic, E. Preliminary study of a novel magnetic sensor for safety in industrial robotics. In Proceedings of the IEEE International Symposium on Robotic and Sensors Environments, Ottawa, ON, Canada, 17–18 June 2019.