Wan and Harada *Robot. Biomim.* (2016) 3:18
DOI 10.1186/s40638-016-0050-2

Robotics and Biomimetics

RESEARCH Open Access

Integrated assembly and motion planning using regrasp graphs

Weiwei Wan[1*] and Kensuke Harada[1,2]

Abstract

This paper presents an integrated assembly and motion planning system to recursively find the assembly sequence and motions to assemble two objects with the help of a horizontal surface as the supporting fixture. The system is implemented in both assembly level and motion level. In the assembly level, the system checks all combinations of the assembly sequences and gets a set of candidates. Then, for each candidate assembly sequence, the system incrementally builds regrasp graphs and performs recursive search to find a pick-and-place motion in the motion level to manipulate the base object as well as to assemble the other object to the base. The system integrates the candidate assembly sequences computed in the assembly level incrementally and recursively with graph searching and motion planning in the motion level and plans the assembly sequences and motions integratedly for assembly tasks. Both simulation and real-world experiments are performed to demonstrate the efficacy of the integrated planning system.

Keywords: Assembly planning, Grasp and regrasp, Motion planning

Background

Introduction

This paper studies the integrated assembly and manipulation planning using regrasp graphs [1] and a horizontal surface [2]. Given two parts and their relative positions in an assembled structure, our integrated planning system decides (1) which object is used as the base, (2) how to place the base, and (3) how to assemble the second part to the base. The results are the integration of assembly sequences and robot motions.

In state-of-the-art robotic assembly systems, the assembly sequences and robot motions are pre-defined manually, which significantly impairs the automation of next-generation manufacturing. Take Fig. 1a for example. To use robots to assemble two objects, the traditional way is that technicians figure out the assembly plan and program the robots using teach pads: They make the robot to pick up an object as the base, place it down on a fixture using some pre-defined position and orientation, pick up the second object, and assemble the second object to the base following some given geometric relationship. The assembly sequence is decided by the intelligence of the technicians: There are several possible solutions shown in Fig. 1b, but the technicians manually select one of them for the robot using their experience.

In this paper, we automate the manual process performed by the technicians. We propose an integrated assembly and motion planning system which is done in both assembly level and motion level. In the assembly level, the system checks all combinations of the assembly sequences and gets a set of candidates. In the motion level, the system performs motion planning recursively for each candidate assembly sequence and finds the motions to manipulate the base object as well as to assemble the other object to the base.

Lots of studies have been devoted to individual problems like assembly planning [3], picking-up and placing-down objects [4, 5], regrasping [6–8], as well as motion planning [9]. But to our best knowledge, few studied the integration, which requires considering not only the assembly orders and assembly directions, but also the accessible grasps of each objects and the goal pose of the assembled structure. The later part is an important problem which was never discussed by the assembly planning, grasp planning, and motion planning literature. To

*Correspondence: wan-weiwei@aist.go.jp
[1] Intelligent System Research Institute, Artificial Intelligence Research Center, National Institute of AIST, Tsukuba, Japan
Full list of author information is available at the end of the article

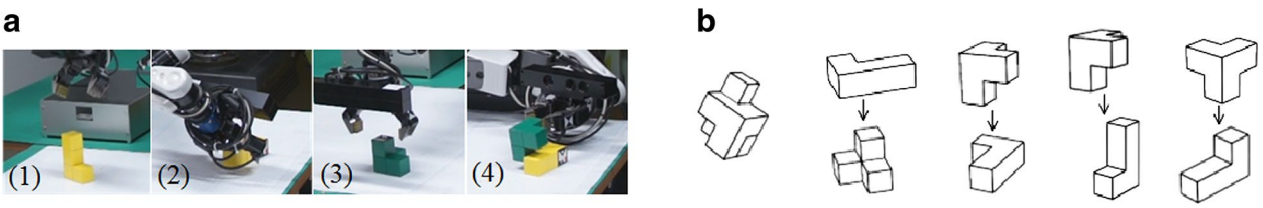

Fig. 1 The assembly sequences and motions in state-of-the-art robotic assembly system are manually programmed by technicians using teach pads: Human technicians figure out the assembly plan and program the robot to pick up and place down the base, and assemble the second object to the base (**a**). There are several candidate assembly sequences shown in (**b**), but the technicians manually select one of them using their experience. This paper proposes a novel system to do this automatically using integrated assembly and manipulation planning. **a** The snapshots of assembling two objects. In subfigures (1–2), the robot picks up the *yellow* object and places down it on the table as the base. In subfigures (3–4), the robot picks up the *green* object and assembles it to the base. **b** The possible assembly sequences. The left plot shows the goal. The other plots show different ways of selecting the base and different ways of attaching the second object

this extent, our paper is the initial work that does integrated assembly and motion planning. Specifically, we develop an integrated planning system which considers: (1) Which object should be treated as the base and be manipulated first, (2) which position and orientation should the base be, and (3) how to assemble the second object to the base. Our system plans assembly sequences and motions automatically and can be used to replace some manual work of system integrators. The efficacy of our system is demonstrated using both simulations and real-world executions in the experimental section. The work is expected to be a precedent planner for object assembly using force sensors [10, 11].

Related work

In respective fields, assembly planning and motion planning are well studied. In assembly planning, early work like [12] and [3] were devoted to symbolic planning. Mello [12] presented the AND/OR graph approach to analyze assembly structures. Wilson and Latombe [3] presented the seminal work that uses non-directional blocking graph (NDBG) to generate assembly sequences. Bozma and Koditschek [13] presented a sphere assembly method which is essentially a path planning problem. More recent work like [14] and [15] concentrate on searching the feasible grasp and manipulation motions with respect to a pre-defined geometric relationships. In particular, [15] does assembly planning from the view point of multiple robot cooperation. Instead of assembly sequences of the objects, [15] plans the assembly sequences of multiple robots. The poses and sequences of objects in the work are pre-defined. Other work like [16] converts assembly planning to a semi-automatic process and learns the assembly sequence from human beings. The adopted method avoids the constraints from the robot bodies and the other parts of the assembled structure, instead of using them. Comparing with our work, it did not integrate the assembly and motion. In motion planning, [17, 18] are the leading study that compared the workspace and joint space approaches. Kavraki et al. [19], Simeon et al. [20] and Lavalle and Kuffner [9] presented the probabilistic approaches to find collision-free motion in the joint space. Vahrenkamp et al. [21], Berenson et al. [22], Phillips and Likhachev [23] represent the more recent studies that use historic data to improve system performance.

For the integration, although there are few studies about integrated assembly and motion planning, there are lots of contemporary research focused on integrated task and motion planning. [24–31] are examples where the planning is done in both task level and motion level. In the task level, these planners employ meta-primitives to divide and conquer tasks. In the motion level, these planners plan motions to implement the primitives generated in the task level. The task-level planning is usually done incrementally and recursively along with the motion-level planning. For example, [28] uses geometric backtracking in task level to decompose and plan the motion in the motion level. Krontiris and Bekris [30] use randomly sampled subgoals as a guide to help motion planners to perform cylinder rearrangements. Dantam et al. [31] present an incremental solution for stacking and rearranging tasks which are effective to exploded combinatorics. The incremental subtasks are alternatives to subgoals in [30] and [32]. Integrate task and motion planning share the same principle with our work, except that we solve the specific task of assembly, and our constraints are from assembly sequences and 6D configurations. These constraints are more complex than picking and moving and require robot to do regrasp [6]. In the assembly level, we check all combinations of the assembly and get a set of candidate assembly sequences. For each sequence, we perform motion planning in the motion level to manipulate the base object and to assemble the other object to the base. The planning in the assembly

level is integrated with the planning in the motion level and is done incrementally and recursively by searching regrasp graphs and invoking motion planning algorithms. If the motion planning algorithms cannot find a path in the motion level, we roll back to the assembly level and try another candidate assembly sequence or try a different base until a solution is found or failure is reported.

Methods

Overview of the Integrated System

In our work, the integrated assembly and motion planning problem are solved incrementally and recursively by searching regrasp graphs and invoking motion planning algorithms. The overview of the integrated system is shown in Fig. 2. In the first component shown in the upper part of the figure, the system selects a base, computes its placements on the table, and checks the possible sequences of assembling the second object to the placements of the base. The output of the first component is a set of candidate assembly sequences. In the second component shown in the lower part, the system checks each assembly sequence incrementally using regrasp graphs. Each object has an initial state and a goal state. The system computes all the stable placements of the object, connects them with the initial and goal states, and builds a regrasp graph. It searches the regrasp graph to see whether there is a direct motion or a sequence of regrasp motions that the robot can use to pick up the object from the initial state and place it down to the goal state. The regrasp graph is repeatedly built and searched for each object. If the search fails, the system starts over to choose a different candidate sequence or to select a different base.

The details of the two components will be explained in sections "Overview of the integrated system" and "The assembly planning component." Before that, we list some essential symbols to facilitate readers.

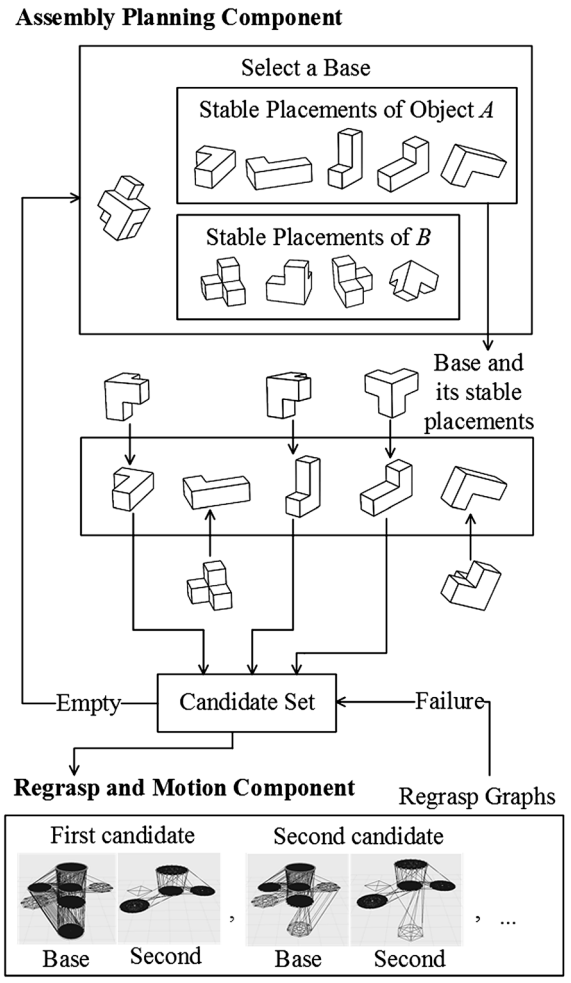

Fig. 2 Overview of the integrated assembly and motion planning system

$\boldsymbol{p}_X{}^s$ The position of Object X on a horizontal surface. We use A and B to denote the two objects, and consequently, $\boldsymbol{p}_A{}^s$ and $\boldsymbol{p}_B{}^s$ are used to denote the positions of the two objects where letter $\boldsymbol{p}$ indicates "position," and the letter s indicates that the object is on a horizontal surface. The notation could denote the initial placements or the positions of intermediate placements on a horizontal surface, depending on the context.

$\mathbf{R}_X{}^s$ The orientation of Object X on a horizontal surface. The letter $\boldsymbol{p}$ indicates "rotation." Like $\boldsymbol{p}_X{}^s$, X is to be replaced by A or B. $\mathbf{R}_X{}^s$ could denote the initial orientations or the orientations of intermediate placements on a horizontal surface, depending on the context.

$\boldsymbol{p}_X{}^a$ The position of Object X at its assembled state in the assembled structure's local coordinate system. The letter a indicates that the object is at its assembled state.

$\mathbf{R}_X{}^a$ The orientation of Object X at its assembled state in the assembled structure's local coordinate system.

$\boldsymbol{p}_X{}^{a(g)}$ The position of Object X at its assembled state in global coordinate system. The letter g indicates the value is described in global coordinate system.

$\mathbf{R}_X{}^{a(g)}$ The orientation of Object X at its assembled state in global coordinate system.

$\mathbf{g}_X{}^f$ The force-closure grasps of Object X. The letter f indicates the object is in a free area without any obstacles. It is neither in an assembled structure nor laying on any

surface. X's position and orientation are ([0,0,0], diag(1, 1, 1)).

$\mathbf{g}_X{}^{s'}$ The force-closure grasps of Object X on a horizontal surface. Object X is at a placement on the horizontal surface. Its position and orientation are ($\boldsymbol{p}_X{}^s$, $\mathbf{R}_X{}^s$). The letter s indicates the grasps are associated with an object laying on a horizontal surface. The symbol f' indicates the grasps are raw, not necessarily collision-free and IK-feasible (IK = inverse kinematics).

$\mathbf{g}_X{}^s$ The collision-free and IK-feasible grasps of Object X on a horizontal surface. Object X is at a placement on the horizontal surface. Its position and orientation are also ($\boldsymbol{p}_X{}^s$, $\mathbf{R}_X{}^s$). The collision-free and IK-feasible grasps are named accessible grasps.

$\mathbf{g}_X{}^{a(g)'}$ The force-closure grasps of Object X at its assembled state. The grasps are described in global coordinate system. X's position and orientation are ($\boldsymbol{p}_X{}^{a(g)}$, $\mathbf{R}_X{}^{a(g)}$). The symbol $'$ indicates the grasps are raw, not necessarily collision-free and IK-feasible.

$\mathbf{g}_X{}^{a(g)}$ The collision-free and IK-feasible grasps of Object X at its assembled state. The grasps are described in global coordinate system. X's position and orientation are ($\boldsymbol{p}_X{}^{a(g)}$, $\mathbf{R}_X{}^{a(g)}$). The collision-free and IK-feasible grasps are named accessible grasps.

The assembly planning component

The goal of the assembly planning component is to find a set of candidate assembly sequences. The given input to the assembly planning component includes: (1) the relative poses of the two parts, (2) their geometric models, and (3) the position on the supporting table to do assembly. Besides the given input, we assume that the second object is assembled to the base from inverse normal direction of the horizontal surface.

The input (1) means the values of $\boldsymbol{p}_B{}^a$-$\boldsymbol{p}_A{}^a$ and $\mathbf{R}_B{}^a \cdot (\mathbf{R}_A{}^a)'$ are known. The input (3) means the position of the assembled base (x_a, y_a) on the table is known. By setting $\boldsymbol{p}_A{}^a$ as the original point and setting $\mathbf{R}_A{}^a$ as the orientation of the assembled structure's local coordinate system, the variables $\boldsymbol{p}_X{}^a$ and $\mathbf{R}_X{}^a$ can be computed as follows

$$\boldsymbol{p}_B{}^a = \boldsymbol{p}_B{}^a - \boldsymbol{p}_A{}^a, \quad \mathbf{R}_B{}^a = \mathbf{R}_B{}^a \cdot (\mathbf{R}_A{}^a)' \tag{1}$$

$$\boldsymbol{p}_A{}^a = [0,0,0]', \quad \mathbf{R}_A{}^a = \mathbf{I} \tag{2}$$

The assembly planning is done based on the placements of the base object. There are two phases in the assembly planning component. The first phase is done in the assembly level. In the first phase, the assembly planner selects a base object (Object A for instance), computes all its stable placements on the table, attaches the second object to the base, and computes the collision-free poses of the second objects. We use a set $\{(\boldsymbol{p}_{Ap}, \mathbf{R}_{Ap})\}$ to denote the stable placements. For each ($\boldsymbol{p}_{Ap}$, $\mathbf{R}_{Ap}$), the planner computes an assembly candidate $\mathbf{C}_i$ using:

$$\mathbf{C}_i = \{(\boldsymbol{p}_A{}^{a(g)}, \mathbf{R}_A{}^{a(g)}), (\boldsymbol{p}_B{}^{a(g)}, \quad \mathbf{R}_B{}^{a(g)}), \tag{3}$$

$$([\boldsymbol{p}_B{}^{a(g)} - k \cdot \boldsymbol{n}_t], \quad \mathbf{R}_B{}^{a(g)})\} \tag{4}$$

where

$$\boldsymbol{p}_A{}^{a(g)} = [x_a, y_a, \boldsymbol{p}_{Ap}.z + h_t], \quad \mathbf{R}_A{}^{a(g)} = \mathbf{R}_{Ap} \tag{5}$$

$$\boldsymbol{p}_B{}^{a(g)} = \boldsymbol{p}_A{}^{a(g)} + \boldsymbol{p}_B{}^a, \quad \mathbf{R}_B{}^{a(g)} = \mathbf{R}_A{}^{a(g)} \cdot \mathbf{R}_B{}^a \tag{6}$$

and

$$\mathsf{S}(\mathsf{A}(\boldsymbol{p}_A{}^{a(g)}, \mathbf{R}_A{}^{a(g)}), \mathsf{B}(\boldsymbol{p}_B{}^{a(g)}, \mathbf{R}_B{}^{a(g)}) \text{ is } \mathsf{TRUE} \tag{7}$$

$$\mathsf{C}(\mathsf{B}(\boldsymbol{p}_B{}^{a(g)}, \mathbf{R}_B{}^{a(g)}), \text{obstacles}) \text{ is } \mathsf{FALSE} \tag{8}$$

$$\mathsf{C}(\mathsf{V}(\mathsf{B}(\boldsymbol{p}_B{}^{a(g)}, \mathbf{R}_B{}^{a(g)}), k \cdot \boldsymbol{n}_t), \text{obstacles}) \text{ is } \mathsf{FALSE} \tag{9}$$

$\mathbf{C}_i$ is a triple where the first two elements indicate the pose of the base object and the pose of the second object assembled to it, respectively. The third element is the retraction of the second object from its assembled state along the normal of the supporting horizontal surface. The second object is ensured to be stable in the structure, not colliding with the environment, and assemblable along inverse the horizontal surface normal using expression (7, 8, 9).

In all, the first phase will output a set of candidate assembly sequences:

$$\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\} \tag{10}$$

where n is equal or smaller than the number of stable placements of the base, depending on the collisions between the second object and the horizontal surface.

The meaning of the unexplained symbols in these equations is as follows. h_t is the height of the horizontal surface. It is a constant value. $\boldsymbol{n}_t$ is the normal of the horizontal surface. (x_a, y_a) is the position on the horizontal surface to do assembly. Object A is treated as the base object. ($\boldsymbol{p}_B{}^{a(g)}$, $\mathbf{R}_B{}^{a(g)}$) is the configuration of object B in its assembled state in the global coordinate system. ($[\boldsymbol{p}_B{}^{a(g)} - k \cdot \boldsymbol{n}_t]$, $\mathbf{R}_B{}^{a(g)}$) is the configuration of Object B after being disassembled along the normal of the horizontal surface with a step length k. $\mathsf{S}(\mathsf{A}(\boldsymbol{p}, \mathbf{R}), \mathsf{B}(\boldsymbol{p}, \mathbf{R})$ is TRUE is a logical expression that ensures the structure composed by the two

objects at the given poses is stable (S indicates stable). $\mathsf{C}(\mathsf{X}(\boldsymbol{p},\mathbf{R}),\text{obstacles})$ is FALSE is a logical expression that ensures the Object X at position $\boldsymbol{p}$ and orientation $\mathbf{R}$ does not collide with the *obstacles* (C indicates collision). $\mathsf{C}(\mathsf{V}(\mathsf{X}(\boldsymbol{p},\mathbf{R}),k\cdot\boldsymbol{n}_t),\text{obstacles})$ is FALSE ensures the swept volume of Object X moving along $\boldsymbol{n}_t$ with a step length k.

Take the two objects in Fig. 2 for example. The Object A has five ($\boldsymbol{p}_{Ap}$, $\mathbf{R}_{Ap}$) which are plotted in the first row of Fig. 3 using yellow color. The states after attaching Object B to the placements of A are shown in the second row of blow each element of the first row. The third case in the second row is not stable. It is detected by expression (7). The unstable object is marked using light green. The fifth case collides with the surface, which is detected by expression (8). The collided object is marked using light pink. The remaining assemblies are stable and collision-free (objects are plotted in dark green). The stable and collision-free candidates are further checked using expression (9) in the third row. All three cases in the third row passed the check and are outputted as **C**.

The output of the first phase is the output of the assembly planning component and is performed offline. The second phase is part of the assembly planning component, but it does not directly relate to the output. The algorithms in the second phase work in the motion level and are used online with graph building and searching in the regrasp and motion component. In the second phase, the assembly planer interacts with a grasp planner, loops through all elements in **C**, and finds IK-feasible and collision-free grasps that the robot can use move the objects during assembly. The output of the second phase is a subset of **C** where the objects are at collision-free states and graspable.

Beforehand, the planner sets the two objects at free space and computes the force-closure and collision-free grasps. Each grasp is represented using $\boldsymbol{g}_X{}^f=\{\boldsymbol{p}_0,\boldsymbol{p}_1,\mathbf{R}\}$ where $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ are the contact positions of the finger tips, and $\mathbf{R}$ is the orientation of the palm. The whole set is represented by $\mathbf{g}_X{}^f$, which includes many $\boldsymbol{g}_X{}^f$. Namely, $\mathbf{g}_X{}^f=\{\boldsymbol{g}_X{}^f\}$.

Given a triple $\mathbf{C}_i$, the IK-feasible and collision-free grasps that the robot can use to assemble it are computed as follows

$$\mathbf{g}_X{}^{a(g)}=\mathsf{IK}\left(\mathbf{g}_X{}^{a(g)'}\right)\cap\mathsf{CD}\left(\mathbf{g}_X{}^{a(g)'},\text{obstacles}\right) \quad (11)$$

where

$$\mathbf{g}_X{}^{a(g)'}=\mathbf{R}_X{}^{a(g)}\cdot\mathbf{g}_X{}^f+\boldsymbol{p}_X{}^{a(g)} \quad (12)$$

If none of the $\mathbf{g}_X{}^{a(g)}$ computed by the three $\boldsymbol{p}_X{}^{a(g)}$ and $\mathbf{R}_X{}^{a(g)}$ in the triple is empty, the triple will be saved as a candidate assembly sequence and will be used to build the regrasp graph and do motion planning.

Figure 4 shows some details of the second phase. The left two plots of the first row show the $\mathbf{g}_A{}^f$ and $\mathbf{g}_B{}^f$ associated with the two objects in Fig. 2. The two objects are at free space in these two plots, and no obstacles are nearby. The grasps are plotted as segments to make them easy to recognize. The remaining three plots of the first row show the collision-free grasps when the two objects are assembled on a horizontal surface. The removed grasps either collide with the table surface or collide with the other object in the assembled structure. They are the results of applying $\mathsf{CD}\left(\mathbf{g}_X{}^{a(g)'},\text{obstacles}\right)$ to $\mathbf{C}_{i(i=1,2,\ldots)}$. The results of further applying $\mathsf{IK}\left(\mathbf{g}_X{}^{a(g)'}\right)$ are shown in

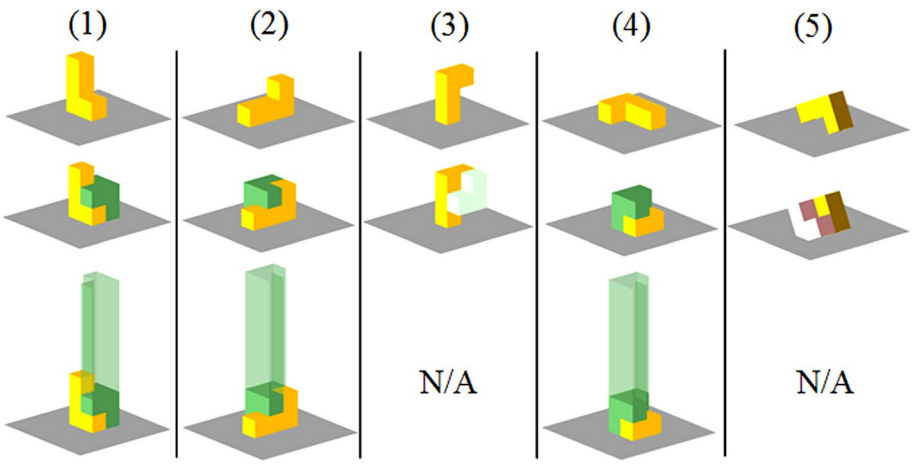

Fig. 3 The first phase of the assembly planning component. The *first row* shows the placements of the Object *A* in Fig. 2. The *second row* shows the states after attaching Object *B* to the placements. The *third row* shows the feasibility along the pre-defined assembly directions. Three cases in the *third row* are feasible and outputted as **C** by the first phase

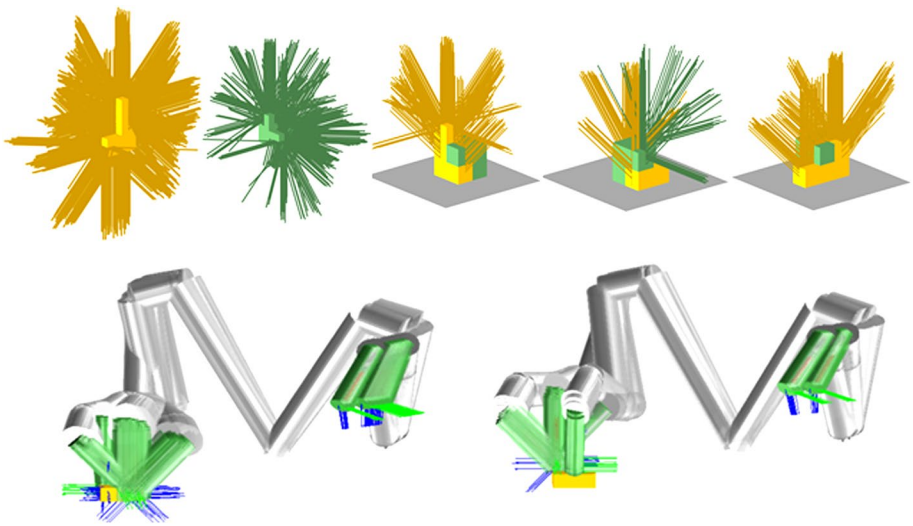

Fig. 4 The left two plots of the *first row* show the $\mathbf{g}_A{}^f$ and $\mathbf{g}_B{}^f$ associated with the two objects in Fig. 2 when they are posed at free space. The remaining three plots of the *first row* show the collision-free grasps when the two objects are assembled on a horizontal surface. The two plots in the *second row* show the IK-feasible grasps of the two objects. The kinematic model is from the Kawada Nextage robot

the second row of Fig. 4. Robot kinematics are considered in this case (Kawada Nextage[1]).

The regrasp and motion component

The input to the regrasp and motion component includes: (1) the initial states of the objects, which are obtained using vision systems, and (2) the goal states of the two objects, which are the assembled states and are included in the candidate set $\mathbf{C}$ produced by the assembly planning component. The regrasp and motion component incrementally builds regrasp graphs using the initial state, the goal state, and the placements of each candidate in $\mathbf{C}$. It recursively searches the regrasp graphs to find sequences of grasps and plans the motion between the grasps using motion planning algorithms.

Given the initial states or the placements of an object on the horizontal surface, say $\boldsymbol{p}_X{}^s$ and $\mathbf{R}_X{}^s$, the IK-feasible and collision-free grasps that the robot can use to pick up the object at these states are computed as follows

$$\mathbf{g}_X{}^s = \mathsf{IK}(\mathbf{g}_X{}^{s'}) \cap \mathsf{CD}(\mathbf{g}_X{}^{s'}, \text{ obstacles}) \tag{13}$$

where

$$\mathbf{g}_X{}^{s'} = \mathbf{R}_X{}^s \cdot \mathbf{g}_X{}^f + \boldsymbol{p}_X{}^s \tag{14}$$

Here Eqs. (13, 14) are similar to the ones in (11) and (12) except they are performed on the initial states.

Then, the regrasp and motion component builds a graph using the elements in $\mathbf{g}_X{}^s$ and $\mathbf{g}_X{}^{a(g)}$. Figure 5 shows the flow using the Object B in Fig. 2. The initial pose of the object on the planery surface, namely $\boldsymbol{p}_B{}^s$ and $\mathbf{R}_B{}^s$, is shown in the upper-left plot. When assembled in the structure, its pose $\boldsymbol{p}_B{}^{a(g)}$ and $\mathbf{R}_B{}^{a(g)}$ is shown in the bottom-left plot. The grasps $\mathbf{g}_X{}^s$ and $\mathbf{g}_X{}^{a(g)}$ associated with them are shown in the plots besides the two poses. They are rendered using colored segments where green means the grasp is both collision-free and IK-feasible. Blue means the grasp is collision-free, but IK-infeasible. The collided grasps are not shown. A regrasp graph is built based on the common grasps, the initial grasps, goal grasps, and some intermediate placements. The planner recursively searches the regrasp graph, finds a sequence of grasps, and employs transition-RRT to find a motion between the grasps. Note that the planning is done between the grasp associated with the initial states and the third element of the triple in $\mathbf{C}_i$. Directly planning to the grasps associated with the first element is a narrow passage problem [33, 34] and should be avoided.

More details about the regrasp graph are shown in Fig. 6. It is basically the same as [29], except that we put the initial and goal states and their accessible grasps in the top and bottom layers, respectively. The top layer has only one circle which encodes the initial state of the object and its accessible grasps. The bottom layer also has only one circle which encodes the goal state of the object and its accessible grasps. The red arrows in the upper part of Fig. 5 show the relationship between the initial and goal state, their accessible grasps (collision-free and IK-feasible grasps, plotted as green segments in the figure), and the top and bottom layers of the regrasp graph. The middle layers are composed of several circles where each of them encodes a possible placement on a horizontal surface and

[1] http://nextage.kawada.jp/en.

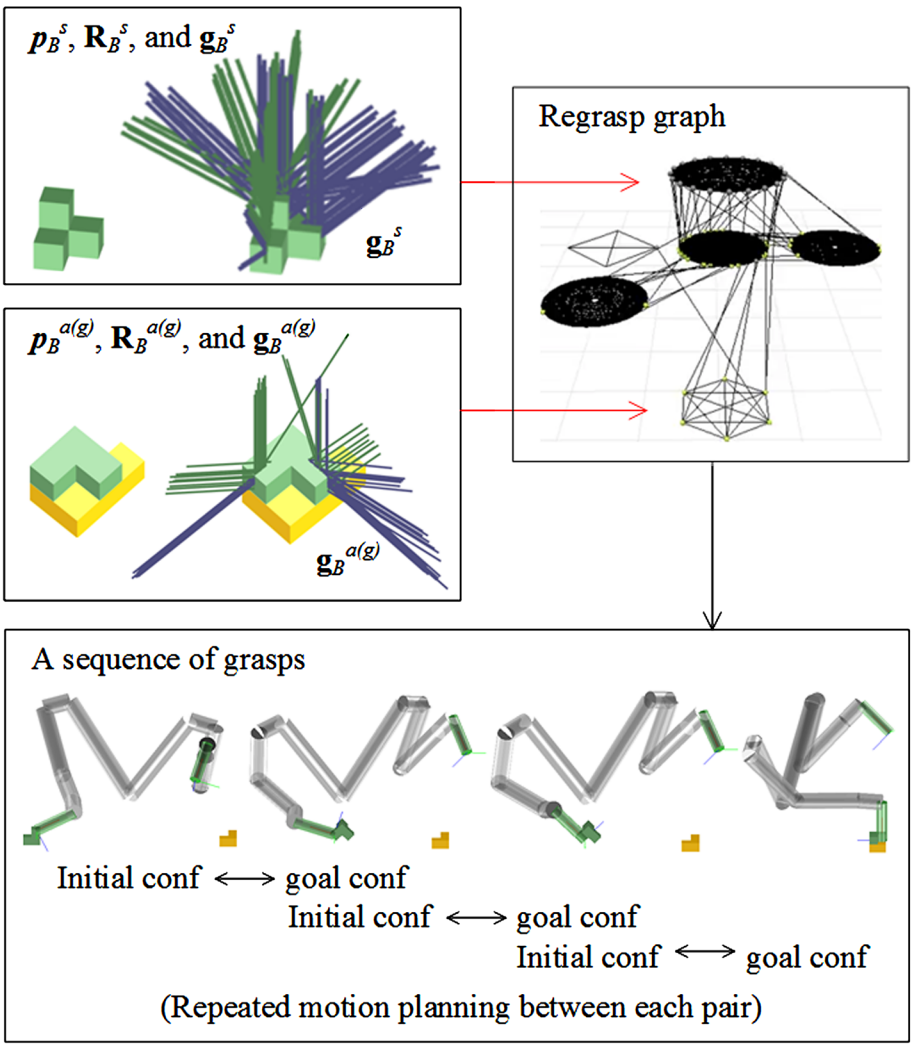

Fig. 5 The flow of the regrasp and motion component. Given the initial and goal states of an object (in this case the Object *B* in Fig.2), we compute its accessible (collision-free and IK-feasible) initial and goal grasps, and use these grasps together with the grasps associated with the placements of the object on a horizontal surface to build regrasp graphs. These two steps are shown in the upper part of the flowchart. The *green* segments in the plots denote accessible grasps. The *blue* ones denote the IK-infeasible grasps. We recursively search the regrasp graphs to find a sequence of grasps and do motion planning repeatedly between the grasps to find a solution to the desired task. The robot configurations in the *lower part* of the flowchart show as a sequence of grasps obtained by searching the regrasp graph

its accessible grasps. The arrows in Fig. 6 show the correspondence between the placements of Object *B*, their accessible grasps, and the circles in the middle layer.

Each node on the circles represents a grasp: The ones in circle of the upper layer are from $\mathbf{g}_X{}^s$, and the ones in the circle of the bottom layer are from $\mathbf{g}_X{}^{a(g)}$. The ones from the circles in the middle layer are the grasps associated with the placements. The orientations of the placements are evenly sampled online. Their positions are fixed to a pre-defined position in front of the robot.[2] If the circles share some grasps (grasps with the same $\boldsymbol{p}_0$, $\boldsymbol{p}_1$, $\mathbf{R}$ values in the object's local coordinate system), we connect them at the correspondent nodes.

The regrasp and motion component searches the graph to find a sequence of grasps that manipulates the object from its initial state to the goal. For each adjacent pair in the grasp sequence, the regrasp and motion component repeatedly performs motion planning to find feasible motions. If no feasible motion was found, the component roll back to try a different sequence, a different triple in $\mathbf{C}$, or use a different based.

The regrasp graph is repeatedly built for each object and is recursively searched for motion planning. The details

[2] The position can be optimized by computing the robot's manipulability [21, 35].

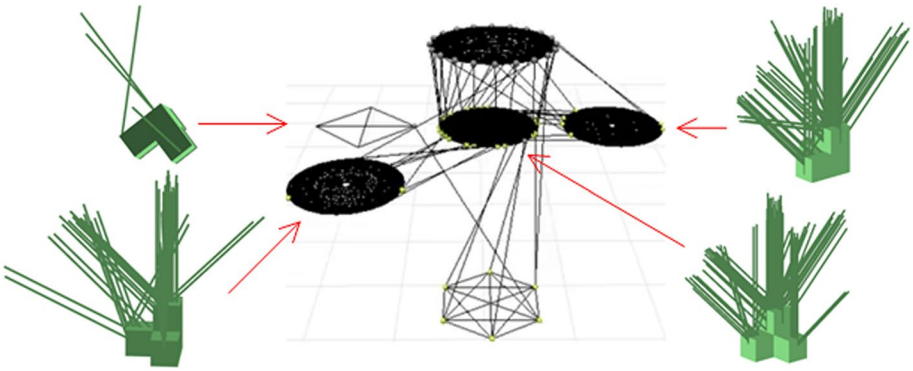

Fig. 6 More details about the regrasp graph. The regrasp graph has three layers where the *top layer* and the *bottom layer* have *one circle* , respectively. They encode the initial and goal states and their accessible grasps. The correspondence has been shown by the *red arrows* in Fig. 5. The *middle layer* encodes the placements on horizontal surfaces and their accessible grasps. The four placements and their accessible grasps of Object *B* are plotted in the figure. Their correspondent *circles* in the *middle layer* are denoted using *red arrows*. By searching this regrasp graph, the planner finds a sequence of grasps. The sequence may include some intermediate nodes like the one shown in the *lower part* of Fig. 5. It may also connect directly from the initial state to the goal

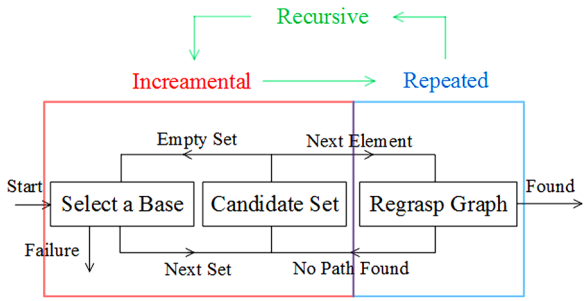

Fig. 7 Details of the repeatedly graph building and the recursive search. The algorithm in the component incrementally uses each element in the candidate set to do motion planning. If fails, it tries a different element or a different candidate until all possibility is tested

of the repetition and recursion are summarized in Fig. 7. Incrementally for each element in the candidate set, the component builds the regrasp graph and performs motion planning. If the planning succeeds, the component reports "found," and otherwise, it tries a different element from the candidate set. If all elements in the candidate set are visited, the component tries a different base. Exemplary results will be shown in the experimental section. A disadvantage of the incremental process is the time cost that may approach infinity. To avoid that, we also employed a time limit in implementation. When planning time goes over the limit, our system reports failure.

Results and discussion

We perform both simulation and real-world experiments to demonstrate the efficacy of the integrated assembly and motion planning system. The computer used to compute the grasps and motions is a Dell T7910 workstation (Processor: Intel Xeon E5-2630 v3 with 8CHT, 20MB Cache, and 2.4GHz Clock, Memory: 32G 2133MHz DDR4). The robot used to perform real-world experiments is Kawada Nextage Open. The relative poses between the two objects are obtained in advance using a teaching system.[3] The repeatedly invoked motion planning algorithm is transition-RRT [37].

Figure 8 shows two failure cases of the repeated graph building and the recursive searching. In the upper row of these figures, the system selects the Object *A* as the base and tries the assembly sequence shown in the left. It is the first element of the candidate assembly sequence **C**. The system builds the regrasp graph and searches the graph to do motion planning in (1–6). In details, the system checks the accessible grasps (the green plots) associated with the initial and goal states, and uses them to build the regrasp graph in (1). Then, the system searches the graph and does motion planning to transfer the Object *A* from the initial state to the goal in (2–4). In (5–6), the system checks the accessible grasps associated with the second object's initial and goal states. Since there is no accessible grasps (no green plots) associated with the goal state, the system cannot build the third layer of the regrasp graph and cannot find a path. It reports failure and rolls back to use a different assembly sequence.

In the lower row, the system uses the same base but tries a different assembly sequence in **C**. It builds and searches the graph and performs motion planning in (7–12). Like (1), the system checks the accessible grasps

[3] The basic principle is human beings teach the relative poses of the two objects in front of a camera using AR markers. The details can be found in [36]. Practitioners may also use CAD software to pre-define the relative poses.

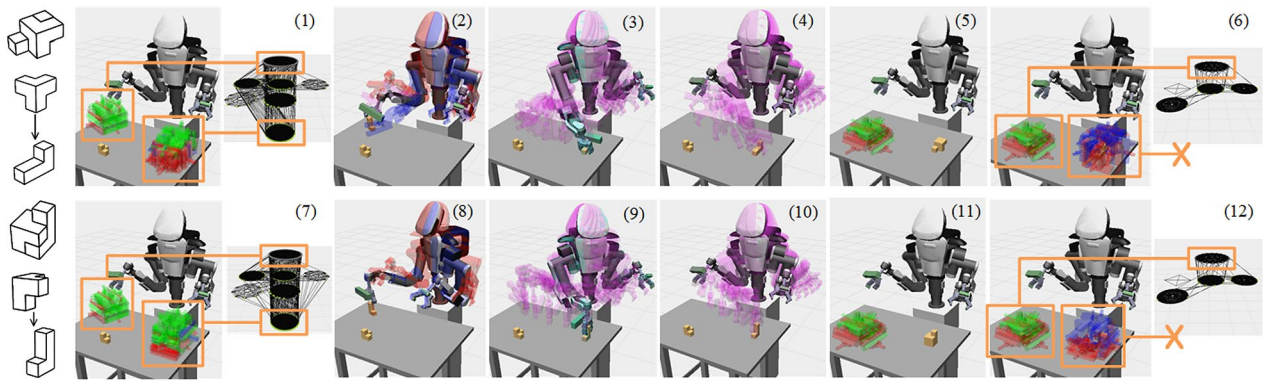

Fig. 8 Two failure cases during the repeated graph building and recursive searching. These two cases use the *first* and *second elements* in the candidate assembly sequence of the two objects shown in Fig. 2 to build the regrasp graphs and do motion planning. Both of them failed since there are no accessible grasps to place down the second object to its goal states. After each failure, the planner incrementally switches to different candidates

of the base object's initial and goal states, and builds a graph in (7). Then, it searches the graph, performs motion planning, and successfully finds a way to transfer the base object to the goal. In (11–12), the system checks the grasps of the second object and finds all grasps at the goal are not accessible. It reports failure and rolls back to another assembly sequence again. The searching and rolling back process continues incrementally until a solution is found or all bases and sequences are tried.

Figure 9 shows a successful case. Here, the system tries the third assembly order in **C**. Like the flow in Fig. 8, the system computes the grasps associated with the initial state and goal state of the base object in (1). The accessible grasps are rendered in green. They correspond to the top and bottom layer of the grasp shown in (1'). In (2), the system chooses one candidate from the accessible grasps and does motion planning to pick up the object. The selected grasp corresponds to one node in the top layer of the graph, which is marked with red color in (2'). In (3), the robot picks up Object *A* and transfers it to the goal state using a second motion planning. This corresponds to an edge in (3') which connects the node in one circle to the node in another. The edge directly connects to the goal without any intermediate placements. After that, the robot moves its arm back at (4), which corresponds to a node in the bottom layer of the graph shown in (4'). In (5), the system computes the grasps associated with the initial and goal states of the second object. The accessible grasps are rendered in green and correspond to the nodes in the top and bottom layer of the regrasp graph shown in (5'). Both initial and goal states have accessible grasps, and it is possible to build the regrasp graph for the second object this time. In (6), the system chooses one feasible grasp and does motion planning to pick up the object. The selected grasp corresponds to the marked node in (6'). In (7), the robot picks up Object *B* and assembles it to its goal state using a second motion planning which corresponds to an edge in (7'). Finally, the robot moves its arm back at (8) and (8').

The subfigures (2"–4") and (6"–8") in the third row show how the robot executes the planned motion. They correspond to (2–4), (6–8) and (2'–4'), (6'–8') in the first two rows. The whole process is divided at (4/4'/4") and (5/5'/6") where (1/1'/2"–4/4'/4") are picking-up and placing-down the base object and (5/5'/6"–8/8'/8") are assembling the second object to the base. A video clip that shows both the simulation and real-world execution is available online at: https://www.youtube.com/watch?v=MAT6ucmTRnE.

Conclusions

This paper presented an integrated assembly and motion planning system which recursively find how to assemble two objects with the help of a horizontal surface as the supporting fixture. The system decides (1) which object is used as the base, (2) how to place the base, and (3) how to assemble the second object to the base. It can find a feasible solution to assemble two objects with completeness. The proposed system is expected to help robot perform assembly tasks automatically, and take the place of the technicians who manually specify the assembly sequences using their experience.

Currently, the assembly is limited to two objects. Increasing the number of objects would lead to exploded combinatorics and is computationally infeasible. It is therefore a challenging open problem to plan the integrated assembly of more than two objects. In our most recent work [38], we studied a simplified version of this open problem and solved the assembly sequences of 3, 4, and 5 objects with a fixed final assembly pose. Interested

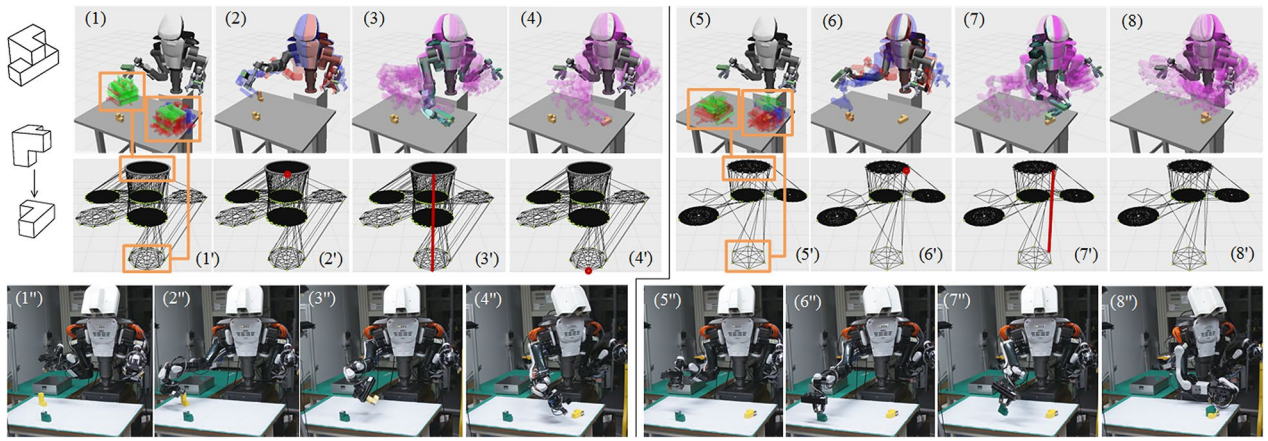

Fig. 9 The successful plan. The *third element* in the candidate assembly sequence of the two objects shown in Fig.2 leads to a successful plan. The simulated motion sequence is shown in (1–8). The correspondent regrasp graphs and the paths on the graphs are shown in (1′–8′). (1″–8″) show the correspondent robot execution. The searching process took about 30 s on the Xeon E5-2630 CPU

readers are recommended to read the paper. Another limitation of the work is that assembly motion is limited to translation. Assembly with rotation is a difficult problem and remains unsolved. The difficulties include planning a specific twist and detecting and taking advantages of contacts. This paper cannot tackle assembly with rotation and is limited to assembly with translational motion. Also, the current system is kinematic. We will add force control to the system in the future and use it to challenge real-world assembly tasks.

Authors' contributions

WW contributed to the ideas, the programming work, and the writing of the paper. KH contributed to the ideas and the software platform. Both authors read and approved the final manuscript.

Author details

[1] Intelligent System Research Institute, Artificial Intelligence Research Center, National Institute of AIST, Tsukuba, Japan. [2] Graduate School of Engineering Science, Osaka University, Osaka, Japan.

Acknowlegements

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

Competing interests

The authors declare that they have no competing interests.

Received: 15 July 2016 Accepted: 27 October 2016
Published online: 08 November 2016

References

1. Wan W, Harada K. Achieving high success rate in dual-arm handover using large number of candidate grasps. Adv Robot. 2016;30:1111–25.
2. Wan W, Mason MT, Fukui R, Kuniyoshi Y. Improving regrasp algorithms to analyze the utility of work surfaces in a workcell. In: Proceedings of IEEE international conference on robotics and automation; 2015.
3. Wilson R, Latombe J-C. Geometric reasoning about mechanical assembly. Artif Intell. 1994;71:371–96.
4. Bohg J, Morales A, Asfour T, Kragic D. Data-driven grasp synthesis: a survey. IEEE Trans Robot. 2013;30:289–309.
5. Holladay A, Barry J, Kaelbling L, Lozano-Perez T. Object placement as inverse motion planning. In: Proceedings of IEEE international conference on robotics and automation; 2013.
6. Tournassound P, Lozano-Perez T, Mazer E. Regrasping. In: Proceedings of IEEE international conference on robotics and automation; 1987.
7. Xue Z, et al. Planning regrasp operations for a multifingered robotic hand. In: Proceedings of IEEE international conference on automation science and engineering; 2008.
8. Lertkultanon P, Pham QC. A single-query manipulation planner. IEEE Robot Autom Lett. 2016;1:198–205.
9. Lavalle SM, Kuffner JJ. Rapidly-exploring random trees: progress and prospects. In: Proceedings of international workshop on the algorithmic foundations of robotics; 2000.
10. Rojas J, Harada K, Onda H. A relative-change-based hierarchical taxonomy for cantilever-snap assembly verification. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems; 2012.
11. Ramirez-Aplizar IG, Harada K, Yoshida E. Motion planning for dual-arm assembly of ring-shaped elastic objects. In: Proceedings of IEEE international conference on humanoid robots; 2014.
12. Mello LH. AND/OR graph representation of assembly plans. IEEE Trans Robot Autom. 1990;6:188–9.
13. Bozma HI, Koditschek DE. Assembly as a noncooperative game of its pieces: analysis of 1D sphere assemblies. Robotica. 2001;19(01):93–108.
14. Knepper R, Layton T, Romanishin J, Rus D. IkeaBot: an autonomous multi-robot coordinated furniture assembly system. In: Proceedings of IEEE international conference on robotics and automation; 2013.
15. Dogar M, Spielberg A, Baker S, Rus D. Multi-robot grasp planning for sequential assembly operations. In: Proceedings of IEEE international conference on robotics and automation; 2015.
16. Lioutikov R, Neumann G, Maeda G, Peters J. Probabilistic segmentation applied to an assembly task. In: Proceedings of international conference on humanoid robots; 2015.
17. Koga Y, Latombe J.-C. Experiments in dual-arm manipulation planning. In: Proceedings of IEEE international conference on robotics and automation; 1992.
18. Koga Y, Latombe J.-C. On multi-arm manipulation planning. In: Proceedings of IEEE international conference on robotics and automation; 1994.
19. Kavraki L, Svestka P, Latombe JC, Overmars M. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom. 1996;12:566–80.
20. Simeon T, Laumond J-P, Cortes J, Shbani A. Manipulation planning with probabilistic roadmaps. Int J Robot Res. 2004;23:729–46.

21. Vahrenkamp N, Berenson D, Asfour T, Kuffner J, Dillmann R. Humanoid motion planning for dual-arm manipulation and regrasp tasks. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems; 2009.
22. Berenson D, Abbeel P, Goldberg K. A robot path planning framework that learns from experience. In: Proceedings of IEEE international conference on robotics and automation; 2012.
23. Phillips M, Likhachev M. Speeding up heuristic computation in planning with experience graphs. In: Proceedings of IEEE international conference on robotics and automation; 2015.
24. Saut J-P, Gharbi M, Cortes J, Sidobre D, Simeon T. Planning pick-and-place tasks with two-hand regrasping. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems; 2010.
25. King J, Klingensmith M, Dellin C. Regrasp manipulation as trajectory optimization. In: Proceedings of robotics: science and systems; 2013.
26. Lozano-Perez T, Kaelbling LP. A Constraint-based method for solving sequential manipulation planning problems. In: Proceedings of IEEE/RSJ international conference on robots and systems; 2014.
27. Nedunuri S, Prabhu S, Moll M. SMT-based synthesis of integrated task and motion plans from plan outlines. In: Proceedings of IEEE international conference on robotics and automation; 2014.
28. Bidot J, Karlsson L, Lagriffoul F, Saffiotti A. Geometric backtracking for combined task and motion planning in robotic systems. Artif Intelli (In press) 2015. http://www.sciencedirect.com/science/article/pii/S000437021500051X.
29. Wan W, Harada K. Developing and comparing single-arm and dual-arm regrasp. IEEE Robot Autom Lett. 2016;1:243–50.
30. Krontiris A, Bekris K. Efficiently solving general rearrangement tasks: a fast extension primitive for an incremental sampling based planner. In: Proceedings of IEEE international conference on robotics and automation; 2016.
31. Dantam N, Kingston Z, Chauhuri S. Incremental task and motion planning: a constraint-based approach. In: Proceedings of robotics: science and systems; 2016.
32. Liu H, Wan W. A dynamic sub-goal path planner for unpredictable environments. In: Proceedings of international conference on robotics and automation; 2010.
33. Hsu D, Jiang T, Reif J, Sun Z. The bridge test for sampling narrow passages with probabilistic roadmap planners. In: Proceedings of IEEE international conference on robotics and automation; 2003.
34. Liu H, Ding D, Wan W. Predictive model for path planning using K-near dynamic bridge builder and inner Parzen window. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems; 2008.
35. Cao C, Wan W, Pan J, Harada K. Analyzing the utility of a support pin in sequential robotic manipulation. In: Proceedings of IEEE international conference on robotics and automation; 2016.
36. Wan W, Lu F, Wu Z, Harada K. Teaching robots to do object assembly using multi-modal 3D vision. 2016. arXiv:1601.06473
37. Jaillet L, Cortes J, Simeon T. Transition-based RRT for path planning in continuous cost spaces. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems; 2008.
38. Wan W, Harada K, Nagata K. Assembly sequence planning for motion planning. 2016. arXiv:1609.03108