applied sciences

Article

High-Speed Autonomous Robotic Assembly Using In-Hand Manipulation and Re-Grasping

Taewoong Kang, Jae-Bong Yi, Dongwoon Song and Seung-Joon Yi *

Department of Electrical Engineering, Pusan National University, Busan 46241, Korea; touy1@pusan.ac.kr (T.K.); niteofhunter@pusan.ac.kr (J.-B.Y.); vehddnsv@pusan.ac.kr (D.S.)
* Correspondence: seungjoon.yi@pusan.ac.kr; Tel.: +82-51-510-7917

Abstract: This paper presents an autonomous robotic assembly system for Soma cube blocks, which, after observing the individual blocks and their assembled shape, quickly plans and executes the assembly motion sequence that picks up each block and incrementally build the target shape. A multi stage planner is used to find the suitable assembly solutions, assembly sequences and grip sequences considering various constraints, and re-grasping is used when the block target pose is not directly realizable or the block pose is ambiguous. The suggested system is implemented for a commercial UR5e robotic arm and a novel two degrees of freedom (DOF) gripper capable of in-hand manipulation, which further speeds up the manipulation speed. It was experimentally validated through a public competitive demonstration, where the suggested system completed all assembly tasks reliably with outstanding performance.

Keywords: robotic assembly; re-grasping; in-hand manipulation

check for updates

Citation: Kang, T.; Yi, J.-B.; Song, D.; Yi, S.-J. High-Speed Autonomous Robotic Assembly Using In-Hand Manipulation and Re-Grasping. *Appl. Sci.* **2021**, *11*, 37. http://dx.doi.org/10.3390/app11010037

Received: 2 November 2020
Accepted: 15 December 2020
Published: 23 December 2020

1. Introduction

Although industrial robots have long been used for various manufacturing tasks, their role has mostly been limited to repetitive high-volume tasks as robot deployment has typically required task-specific fixtures, end effectors and hand-programmed motion sequences [1,2]. The introduction of human-safe, quickly programmable collaborative robots has broadened the robot deployment to high-mix, low-volume works. Thanks to the recent advances of sensor technology and major breakthrough in the machine learning field, we are now expecting a widespread adoption of intelligent robots that can perceive the environment and make decisions by itself.

In this work, we present an integrated system, which, given individual components and their assembled shape, perceives their positions, orientations and assembled configuration using sensors; generates the optimal assembly plan to build the target shape from components while avoiding a collision; and rapidly executes the assembly motions using a robotic manipulator and gripper. The suggested system was validated in both simulated and real environments using multiple combinations of sensor, manipulator and gripper systems and was successfully used to rapidly solve a number of assembly tasks autonomously in a public competitive demonstration with outstanding results.

The remainder of the paper proceeds as follows. Section 2 introduces the related works and highlights how the suggested work differs from each of them. Section 3 presents the overview of the hardware and software used for the system. Section 4 explains how the robot system processes the sensor data to perceive the environment. Section 5 presents how the system finds the optimal motion plan that rapidly assembles the components while avoiding a collision. Section 6 describes the simulation setup we used to test the system with different hardware configurations. Section 7 shows the experimental results acquired from the laboratory experiment and public demonstration. Finally, we conclude with a discussion of potential future directions arising from this work.

2. Related Work

In this work, we present an integrated robotic assembly system that can perceive the environment, plan for the assembly sequence, feasible grasps and manipulator motion and control the hardware in real time. The system is deployed and tested for reliability and performance beyond controlled laboratory environment. As such a system requires multiple components to work together reliably, previous works mostly focus on individual components rather than an integrated system. In [3], the authors provided a summary of recent robot manipulation research topics, which include robotic hand design, perception for manipulation, grasping, manipulation and approaches to utilize machine learning for manipulation, but most of the works presented are focused on individual components. For example, in [4], the authors focused on assembly sequence and grasp planning for assembly tasks, and, in [5,6], the authors used deep learning based approach to learn grasp planning strategy from data acquired by repeated trials.

There have been a few works that address an integrated system that can perceive, plan and control its manipulators for assembly tasks. In [7], the authors presented an integrated system for autonomous robotic manipulation that integrates 3D perception, motion planning and real-time motion control. In [8], the authors presented a robot system that has 2D stereo vision camera to detect target objects and uses two 7 DOF arms for a simple assembly task. In [9], the authors presented an integrated robotic system for autonomous bin picking task, which integrates the 3D perception, grasp and arm motion planning and motion control of the hardware. The system has been validated in public competition, the Amazon Robotics Challenge, and has shown exceptional performance in terms of reliability, accuracy and speed.

Recently, there have been a number of works using the Soma cube assembly task as a demonstration [10–14]. The Soma cube assembly task is a good benchmark task for robotic assembly as it has nontrivial number of possible assembly solutions and grasp plans, yet most of the assembly can be realized using a single robotic manipulator. In [14], the authors focused on finding robust grasping strategies by 2-D form closure with cylindrical fingers for arbitrarily shaped objects. The suggested algorithm has been demonstrated by assembling Soma cube blocks into a cube.

In [10], the authors proposed an integrated planning system that plans for the assembly sequence and motion to stack a Soma cube block on another. Although the suggested work is close to this work as it integrates the assembly sequence planning, motion planning and re-grasping required for the Soma cube assembly task, the suggested system is largely limited as it relies upon full information of the blocks and the assembly is limited for only two blocks. The authors extended the suggested system in [11], which allows for stacking up to five blocks. In [13], the authors proposed a planner that plans the optimal assembly sequence of Soma cube blocks for a dual armed robot. Given the mesh model of objects and the final assembly configuration, the suggested algorithm finds an optimal assembly sequence for dual armed robot considering the stability, graspability and assemblability. Finally, recently, in [12], the authors proposed the single-arm motion planner for Soma cube block assembly, which, given the initial and final pose of a Soma cube block, automatically generates the collision free arm motion plan with optional re-grasping to assemble the blocks incrementally.

3. System Overview

Figure 1 shows the overall system architecture. The only inputs to the system are the target shape provided as blocks assembled by human operator and individual building blocks needed for the assembly. The robot observes the target shape and individual building blocks, calculates all possible block configurations for the target shape and plans the assembly sequences and grasp strategy for moving each block while satisfying multiple constraints. If there exists no manipulation sequence that can move each block to the target pose in a single manipulation, the planner performs additional re-grasp planning to move the block in multiple steps. The suggested system is modular and platform agonistic, and it can accept different sensors, manipulators and gripper hardware with little effort.

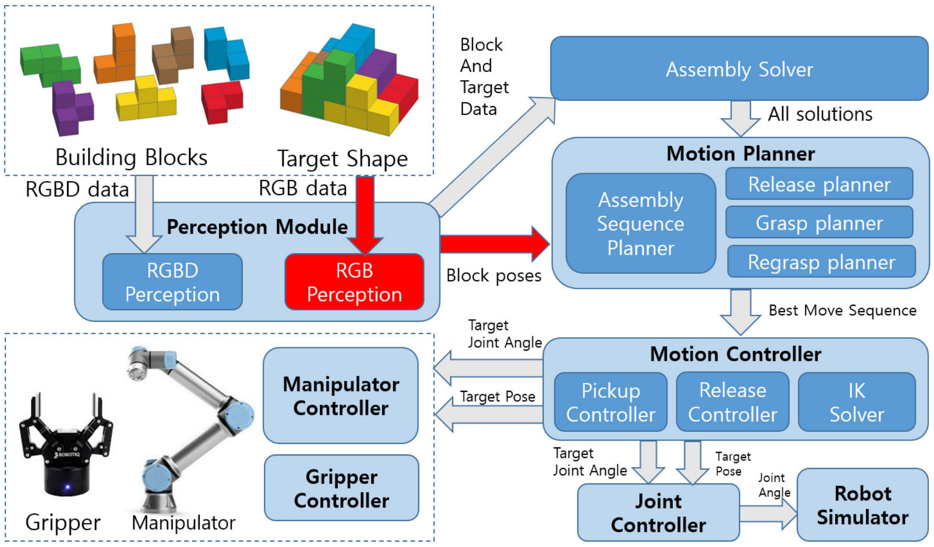


Figure 1. Overall system architecture.

As the system is designed primarily to handle the Soma cube assembly task, which is described in Section 7 in detail, we use following assumptions while designing the system:

- Seven standard Soma blocks are used as building components.
- Each block is separated from others with distance larger than the unit block size.
- The assembly sequence does not require peg-in-hole assembly or screwing.
- The assembly sequence can be realized using a single robotic manipulator.

The first two assumptions allow simplifying the perception modules, while the last two allow using simple placement to assemble the blocks. However, the suggested system can be easily extended for more general tasks by adding new software and hardware modules, which we discuss more in detail in Section 7.

3.1. Manipulator and Gripper

We used a commercial UR5e 6 DOF manipulator for the task. To manipulate the blocks, we iterated through multiple different gripper systems during the development. First, we tried using a commercial Robotiq 2F-85 adaptive gripper without modification, and found its long, center located fingers are not ideal for lateral block release. We replaced its fingers with a pair of short, off-center custom made fingers, as shown in Figure 2a. Based on the experience from using the commercial gripper for the task, we developed a custom designed parallel gripper with an additional degree of freedom at the fingers, which is shown in Figure 2b. This gripper allows the pickup and placement motions to be simplified thanks to its short finger length, parallel actuation and the ability of rotating the grasped object without a wrist movement. We present the Soma cube assembly process using various manipulator and gripper setups in the following sections.

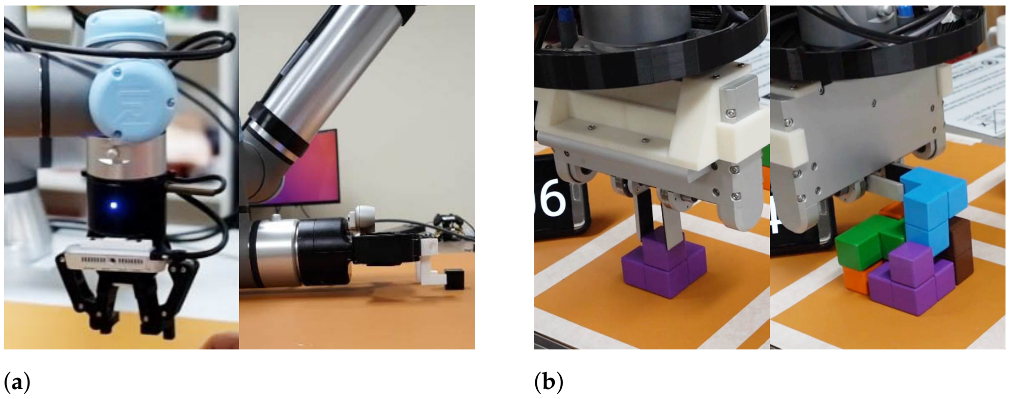


Figure 2. Two gripper systems used for the system: (a) 1 DOF gripper with short off-center fingers; and (b) 2 DOF gripper with tilting fingers.

3.2. Sensors

For perception, we prepared a dual-sensor setup that consists of a ceiling-mounted high-resolution RGB camera and a wrist-mounted RGBD camera. To achieve the very high pose accuracy required for the assembly task, we used a Sony A7R Mark II commercial mirrorless camera paired with Zeiss Sonnar T* 55 mm 1.8 lens as the RGB camera, which can output 7952 × 5304 resolution image with low noise even under indoor lighting. The camera uses stock firmware and is connected to control PC via USB-C cable and remotely controlled by `libgphoto2` library. For the wrist RGBD camera, we tested Intel Realsense D435 and SR305 cameras and decided to use the SR305 camera, which we found to be better optimized for close-range subjects. Two sensors setups are shown in Figure 3.

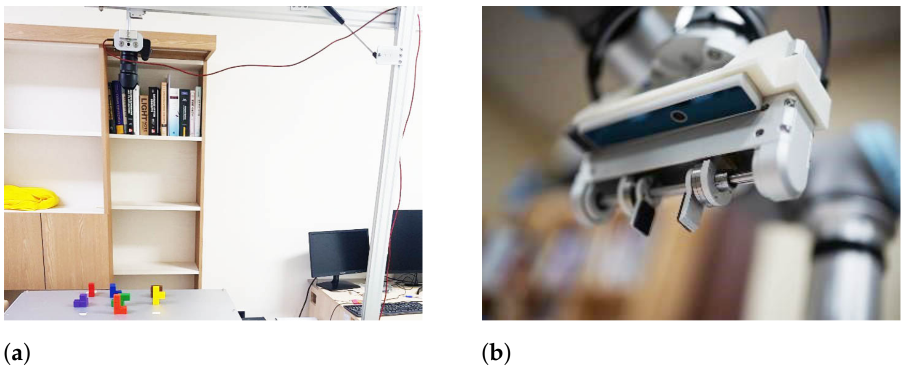


Figure 3. Dual sensor setup for the system: (a) ceiling RGB camera; and (b) wrist RGBD camera.

3.3. Computation

As our software modules are fairly lightweight and do not require a powerful computer for processing, we used a single laptop with core i7 CPU and GTX 1080 GPU to process all CPU and GPU bound computational tasks, which includes the RGB and depth based perception, assembly and grasp planning and finally real-time control of the robotic arm and gripper. We did not use the preemptive real-time kernel on the control PC as the UR5E manipulator has low external control rate of 500 Hz and has internal motion planner and controller that ensure smooth arm motion.

4. Perception

To assemble arbitrarily positioned blocks into the target shape, the system needs to perceive the type, position and orientation of each block first. Standard seven colored Soma cube blocks are used, which are shown in Figure 4, and we label them Blocks 1–7 according to their shapes. To make planning easier, we define the pose and the orientation of blocks separately. As there are 24 3D orientations resulting from multiple applications of 90° rotations about X-axis, Y-axis and Z-axis, we enumerate all 24 orientations and label each block's orientation as a number from 1 to 24. We define the pose of a block as the position and orientation of the bounding cuboid of the block. For a general block in 3D space, we denote its pose by a tuple $(x, y, z, \phi, \theta, \psi)$ where (x, y, z) is the 3D center position of the bounding cuboid and (ϕ, θ, ψ) is its Euler angle. To remove the duplicate cases, we limit Euler angles as $0 \leq \phi < \pi/2, 0 \leq \theta < \pi/2, 0 \leq \psi < \pi/2$. For blocks that lie flat on the surface, we use a simpler 2D pose (x, y, ϕ) where (x, y) is the center position of the bounding box and $0 \leq \phi < \pi/2$ is the yaw angle. Examples of 2D and 3D block pose, as well as blocks with the same pose but different orientations, are shown in Figure 5.

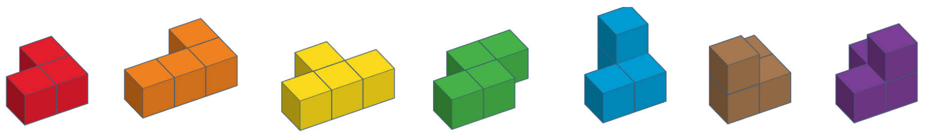


Figure 4. Seven Soma cube blocks types and their default colors.

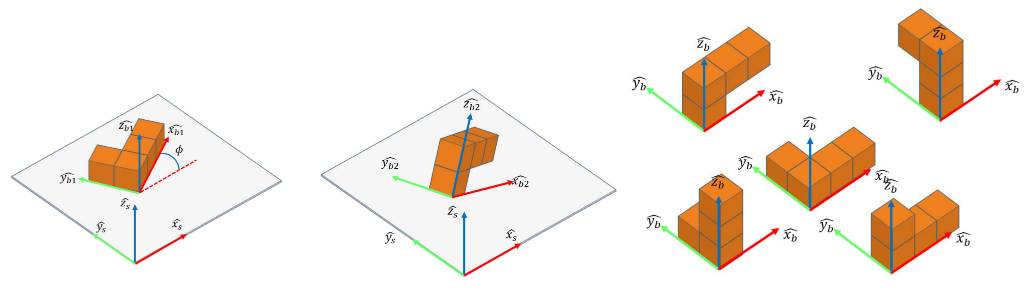
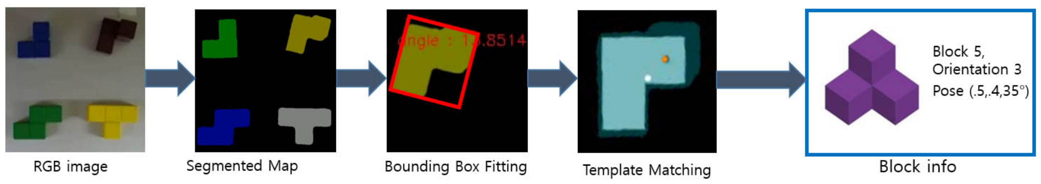

(**a**) (**b**) (**c**)

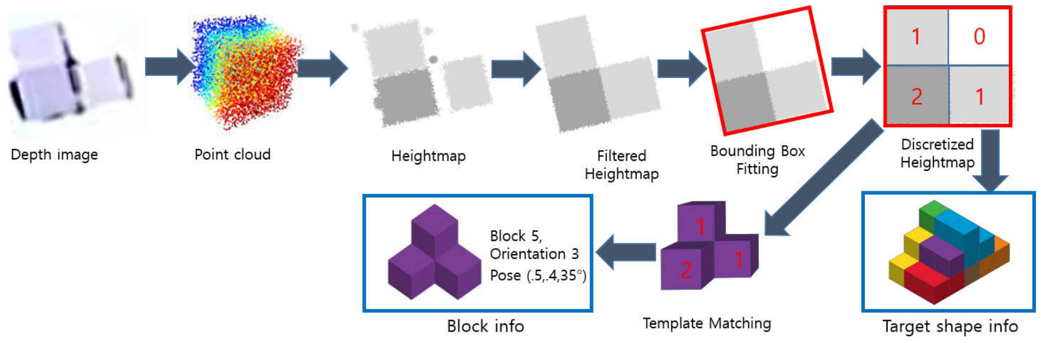
Figure 5. Block pose and orientation: (**a**) 2D pose; (**b**) 3D pose; and (**c**) different block orientations.

4.1. RGB-Based Block Detection

We implemented a RGB-based block detection module to get accurate 2D poses of building blocks required for the assembly task. When a RGB image is requested, the ceiling camera is triggered and the recorded high-resolution `jpeg` image is transferred to control PC and rectified using the calibration parameters. Then, the Deeplab-v3 [15] deep learning-based semantic segmentation algorithm is used to segment the image it into separate blocks, which we found to be better at preserving outlines of blocks than Mask R-CNN [16] algorithm we also tested. Once each block is segmented from the image, their type is classified using the average RGB value, and rotated bounding boxes are fitted to each block. Then, the segmented block images are rotated to be aligned with the X-Y axis of the image space. Finally, they are compared to the standard Soma cube blocks to determine their orientation. have found that the resulting RGB-based block detecting algorithm is reliable against lighting changes, takes less than 200 ms for processing on the core i7 laptop with GTX 1080 GPU and has less than 1 mm positional error and 1.5 degree of orientation error in the worst case compared to the ground truth data. The RGB-based block detection pipeline is shown in Figure 6a.



(**a**)



(**b**)

Figure 6. RGB and depth-based block detection: (**a**) RGB block detection pipeline; and (**b**) depth block detection pipeline.

4.2. Depth-Based Block and Target Shape Detection

In addition to the RGB-based block detection, we also implemented a perception module based on the wrist-mounted RGBD camera. Recently, several studies have focused on detecting and estimating 6D pose of known objects from RGBD data [17]. Although such general RGBD-based object detection algorithms can robustly detect and estimate poses of complex-shaped objects even under heavy occlusion, we decided not to use them as the Soma cube assembly task requires a very high positional accuracy to reliably assemble the blocks using a single robotic manipulator, 1 mm or less for Soma cube blocks with 25 mm unit size we used. Instead, we devised a custom cube perception pipeline based on the 2D heightmap generated from the depth image, exploiting the specific shapes of Soma cube blocks.

Using the camera transform calculated from manipulator forward kinematics, the depth image is projected into 3D space to form the point cloud of the blocks. Then, a 2D heightmap is generated based on the distance of each point in point cloud from the supporting surface, and a 2D detection pipeline similar to the RGB-based detection is used afterward. As we can arbitrarily move the RGBD camera using the robotic manipulator, we use multiple different camera positions, which includes the whole block observation position, individual block observation positions and target shape observation position, to overcome the limitation of lower resolution of the RGBD camera compared to the high-resolution ceiling camera. We found that the RGBD camera used at a close range provides accuracy excelling that of high-resolution ceiling camera, while having two issues: (a) close camera position makes some area of the blocks occluded with no registered point on it, and as a result single block tends to show up as separate blocks instead of a single connected one in heightmap image; and (b) depth readings have noise at the edge of the blocks, which shows up as spike-shaped artifacts in the point cloud. We used multi-stage filtering to handle those issues, which includes k-nearest neighbors filtering in the depth space and erosion/dilation of the heightmap image. After filtering, each connected block is segmented and a rotated bounding box is fitted for each block. Then, each segmented heightmap is rotated to be aligned with the X-Y axis of the image space and discretized into a small 2D height array. This discretized target array can be directly used for target shape or further matched with individual building blocks with all possible orientations to detect each block type, orientation and pose. We found this depth based block detection algorithm, while not using the GPU, takes less than 20 ms for processing on the core i7 laptop with GTX 1080 GPU, is lighting independent and has superior accuracy to the RGB-based detection algorithm. The depth-based block detection pipeline is shown in Figure 6b.

5. Motion Planning

Once the individual blocks and the final assembled shape are known, we need to determine how the final shape can be assembled using the blocks and how each block can be grasped using the robotic gripper and moved to target pose without collision. We describe how each step is implemented in the following subsections.

5.1. Assembly Solving

There often exist a number of different block configurations that fills the 3D target shape without overlapping, which we call the assembly solutions. Although finding the solution in general is a hard combinational optimization problem, all solutions of a small-sized problem can be found quickly by a simple recursive search. For example, a $3 \times 3 \times 3$ cube can be assembled using the standard Soma cube blocks shown in Figure 4 in 240 distinctive ways excluding rotations and reflection, some of which are shown in Figure 7. The pseudo-code of the recursive search algorithm is shown in Algorithm 1. We start with the empty target volume as the root node and recursively expand each node by testing the fitness of all possible block positions and orientations. To remove a family of symmetric solutions derived from one unique solution, we fix the orientation of L shaped block and start expanding the tree from that block. We found that it takes less than 5 ms to find all the solutions of $3 \times 3 \times 3$ cube using the core i7 laptop with GTX 1080 GPU that we used for the system.

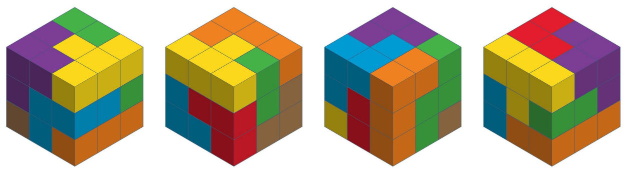


Figure 7. Example of multiple assembly solutions for $3 \times 3 \times 3$ cube.

Algorithm 1: Find_assembly_solutions(*blocks*, *space*)

```
if blocks is not empty then
    for ori ← 1; ori<num_total_orientations do
        for pos ← 1; pos<num_total_positions do
            block_candidate ← ( blocks[0], pos, ori);
            if can_pack( space, block_candidate) then
                blocks_new ← blocks.remove(blocks[0]) ;
                space_new ← space.remove(block_candidate) ;
                solutions.append(block_candidate,
                    Find_assembly_solutions(blocks_new, space_new) );
        end
    end
```

5.2. Assembly Sequence Planning

For each assembly solution, which contains the position and orientation of each block in the assembled shape, there can exist multiple assembly sequences to incrementally build the final shape from blocks, as shown in Figure 8. As we use a single manipulator without any fixture for assembly, we only consider the cases where each block can be moved to the final pose vertically downwards, otherwise a secondary manipulator or fixture would be needed for additional support. To find all the possible assembly sequences for a solution, we use a recursive algorithm whose pseudo-code is shown in Algorithm 2. The algorithm starts with an empty list of possible sequences and a 3D array that represents blocks fully assembled according to the target shape. Then, it finds all blocks that can be picked up vertically without colliding with other blocks and recursively solves its subproblems after removing one of the possible blocks. Finally, the algorithm ends with a list of all possible assembly sequences for the solution.

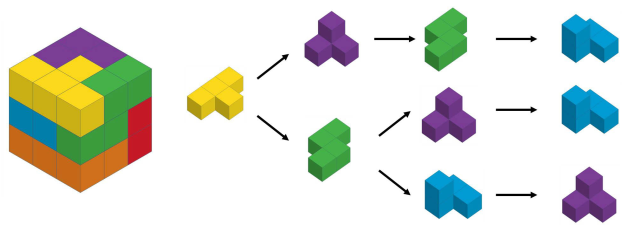


Figure 8. Example of assembly sequence planning.

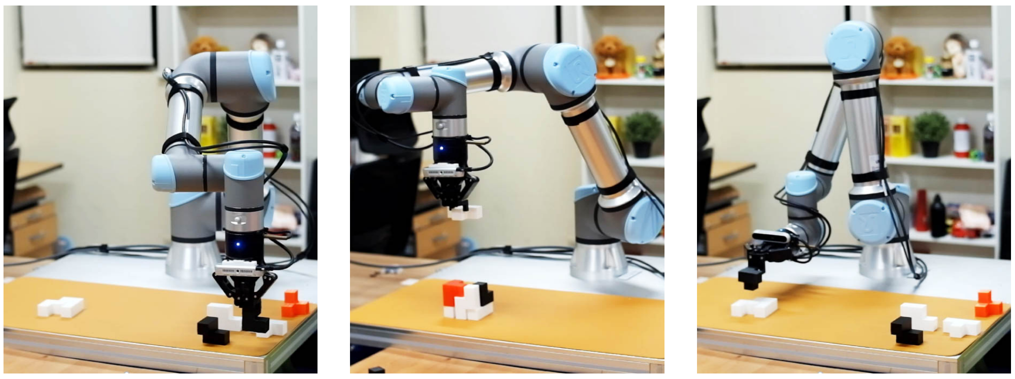
Algorithm 2: Find_assembly_sequence(*sequence*, *solution*)

```
if solution is not empty then
    top_blocks ← all blocks in solution;
    for i ← 1; i<top_blocks.size do
        if top_blocks[i] is blocked by other blocks in solution then
            top_blocks ← top_blocks.remove(top_blocks[i]);
    end
    for i ← 1; i<top_blocks.size do
        solution_new ← solution.remove(top_blocks[i]);
        sequence_new ← sequence.append(top_blocks[i]);
        Find_assembly_sequence(sequence_new,solution_new);
    end
else
    total_assembly_sequences.append(sequence);
```

5.3. Grasp Planning

Once an assembly sequence is found for an assembly solution, we can iteratively pick up each block from its initial position, rotate it to its final orientation and finally move it to its final position to assemble the final shape from blocks. To accomplish such actions using robotic manipulator and gripper, we need to find a stable grasp for the block and plan the manipulator motion sequence that moves the block from the initial pose to the final pose without a collision. As finding a stable grasp for arbitrary shaped object is a hard problem under active research [18–20], we use the following additional constraints to make the search space tractable:

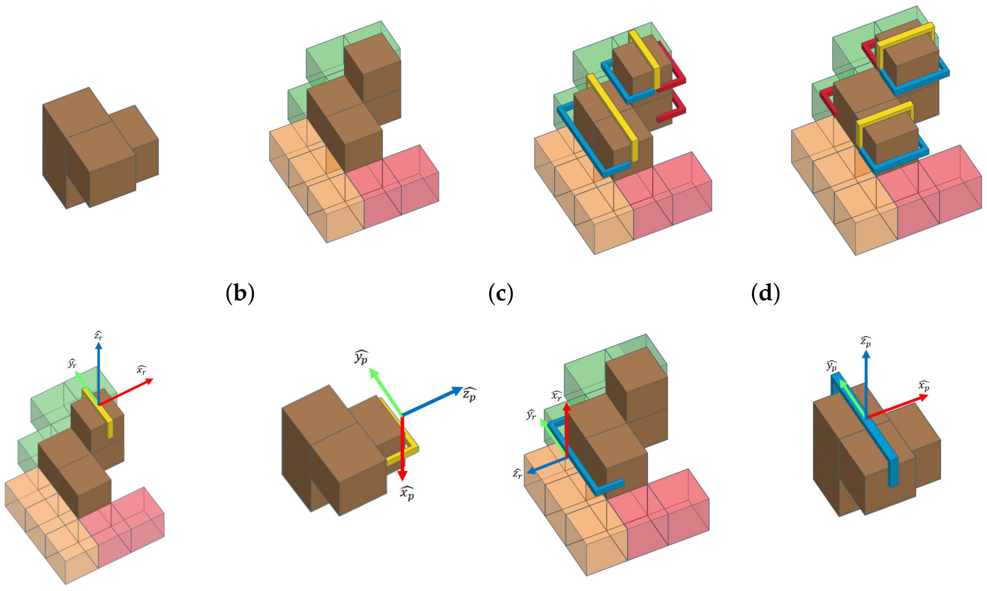
- The Z-axis of the pickup grasp pose should always be parallel to the Z-axis of the block pose (vertical pickup constraint, as shown in Figure 9a).
- The Z-axis of the release grasp pose should be either parallel to the Z-axis of the build surface (vertical release case, shown in Figure 9b) or perpendicular to the Z-axis of the build surface (horizontal release case, shown in Figure 9c).
- The Y-axis of the pickup and release grasp pose is either parallel to the X-axis of the block (X-axis grasp) or the Y-axis of the block (Y-axis grasp).
- The tip positions of the gripper fingers should be aligned with the centers of the unit building cubes each block is composed of.



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

Figure 9. Pickup and release constraints: (**a**) vertical pickup; (**b**) vertical release; and (**c**) lateral release.

As grasp pose relative to the block is fixed during the manipulation, we only need to determine the release grasp pose, and then the pickup grasp pose can be automatically found. Figure 10c,d shows all collision-free *Y*-axis and *X*-axis release gripper pose candidates for target block pose shown in Figure 10b. However, not all of them satisfy the vertical pickup constraint and are directly realizable without re-grasping. For example, the release grasp pose shown in Figure 10e results in the lateral pickup grasp pose in Figure 10f, which should be avoided due to the high risk of collision. On the other hand, the release grasp pose shown in Figure 10g results in the vertical pickup grasp pose shown in Figure 10h, which means we can move the block from initial pose to target pose without re-grasping.



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

(**e**) (**f**) (**g**) (**h**)

Figure 10. Grasp Planning process: (**a**) initial block pose; (**b**) target block pose; (**c**) *Y*-axis grasp candidates; (**d**) *X*-axis grasp candidates; (**e**) infeasible release grasp; (**f**) infeasible pickup grasp; (**g**) feasible release grasp; and (**h**) feasible pickup grasp.

To find feasible grasp satisfying all the constraints, we use the grasp planning algorithm shown in Algorithm 3, For each block, we enumerate all collision free release grasp pose candidates in X-axis and Y-axis first, and then calculate the pickup grasp poses for each release grasp poses using the relative transform of the block orientations. Due to symmetry, some blocks can have identical block configurations for multiple different orientations.We pre-calculate the transform results of all 24 each orientations for each block and put together orientations that result in the same configuration. Based on this information, we build the transform list, which contains all possible transforms for each block that transforms from one orientation to another orientation. Then, the transform list is used to calculate all possible pickup grasp poses. Finally, we calculate the Euler angles of pickup grasp pose and calculate the cost of each pickup–release grasp pose pair according to the pickup and release grasp pose pitch angle, relative yaw angle between pickup and release grasp poses and the grasp width. This process is iterated for each block movement in the sequence, and the grasp plan with the lowest cost is selected for each move to form the best move sequence for the solution. If all pickup grasp poses in the best move sequence have zero pitch angle, the move sequence satisfies the vertical pickup constraint and can be directly realizable without re-grasping. Otherwise, the specific solution requires re-grasping, which we explain in more detail in the next subsection.

Algorithm 3: Find_grasp_plan(*block_type*, *block_initial_orientation*, *solution*)

```
1  Release_grasp_candidates ← {};
2  for x ← 1; x<solution.xsize do
3      Release_grasp_candidates.add(Find_Y_axis_release_grasp(block_type,solution,x,0) );
4      Release_grasp_candidates.add(Find_Y_axis_release_grasp(block_type,solution,x,$\pi/2$) );
5      Release_grasp_candidates.add(Find_Y_axis_release_grasp(block_type,solution,x,$-\pi/2$) );
6  end
7  for y ← 1; y<solution.ysize do
8      Release_grasp_candidates.add(Find_X_axis_release_grasp(block_type,solution,y,0) );
9      Release_grasp_candidates.add(Find_X_axis_release_grasp(block_type,solution,y,$\pi/2$) );
10     Release_grasp_candidates.add(Find_X_axis_release_grasp(block_type,solution,y,$-\pi/2$) );
11 end
12 Target_orientation ← Get_block_orientation(block_type,solution);
13 Transform_list ← Get_block_transforms (block_type,
                          block_initial_orientation, Target_orientation);
14 for i ← 1; i< Release_Grasp_candidates.size do
15     for j ← 1; j<Transform_list.size do
16         Release_grasp ← Release_grasp_candidates[i].pose ;
17         Release_grasp_RPY ← Transform_to_euler(Release_grasp);
18         Pickup_grasp ← Release_grasp* Transform_list[j]$^{-1}$;
19         Pickup_grasp_RPY ← Transform_to_euler(Pickup_grasp);
20         Release_Grasp_candidates[i].cost ← $k_{width}$*Release_grasp_candidates[i].width
               + $k_{pickup\_pitch}$* Pickup_grasp_RPY.pitch + $k_{release\_pitch}$* Release_grasp_RPY.pitch
               + $k_{yaw}$* abs(Release_grasp_RPY.yaw-Pickup_grasp_RPY.yaw) ;
21     end
22 end
23 return Release_grasp_candidates with smallest cost ;
```

5.4. Re-Grasp Planning

Some block orientations, such as two orientation families shown in Figure 11a,f, cannot be directly moved to the other ones with a single movement and requires two movements with an intermediate orientation. To find the best move sequence with re-grasping, we start with the best move sequence found by grasp planner first, and then divide the move that violates the vertical pickup constraint into two separate moves, which move the initial block to the intermediate pose and then to the final pose. To determine the best intermediate block orientation, we first find the stable block orientations for each block by checking the center of gravity and support polygon and make two consecutive moves for each intermediate pose, one from the initial pose to the intermediate pose and the other from the intermediate pose to the final pose. We use the grasp planning algorithm shown in Algorithm 3 to find the best grasp for both moves and add their costs to get the cost of selecting the intermediate pose as the re-grasping pose. Finally, we evaluate all stable orientations for the block and find the best intermediate orientation that results in the lowest cost moves of the block. The re-grasp planning process is shown in Algorithm 4.

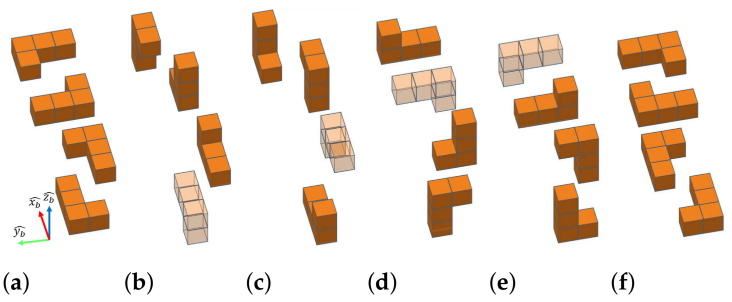


Figure 11. Orientation families and re-grasping. Unstable configuration is shown as transparent blocks: (**a**) 4 initial block orientations rotationally symmetric about *Z*-axis; (**b**) orientations acquired by a 90° rotation about *X*-axis; (**c**) orientations acquired by a −90° rotation about *X*-axis; (**d**) orientations acquired by a 90° rotation about *Y*-axis; (**e**) orientations acquired by a −90° rotation about *Y*-axis; and (**f**) target block orientations which cannot be directly reached by a 90 ° or −90° rotation about *X*-axis or *Y*-axis.

Algorithm 4: Find_regrasp(*block_type*, *block_initial_orientation*, *solution*)

```
for intermediate_orientation ← 1; intermediate_orientation<24 do
    if is_stable(block_type, intermediate_orientation) then
        temporary_solution ← {};
        temporary_solution.add_block(block_type,intermediate_orientation) ;
        grasp_plan1 ← Find_grasp_plan(block_type, block_initial_orientation,
          temporary_solution) ;
        grasp_plan2 ← Find_grasp_plan(block_type, intermediate_orientation,
          solution) ;
        if grasp_plan1.is_feasible() and grasp_plan2.is_feasible() then
            return grasp_plan1, grasp_plan2 ;
        end
end
```

5.5. Arm and Gripper Motion Planning

After we get the optimal move sequence which consists of pickup grasp pose and release grasp pose for each block, we need to plan the motions of the robotic manipulator and gripper to realize the movement. For a physical robot, the motion control is handled by the provided motion API, which takes the Cartesian and joint space waypoints and maximum speed and acceleration parameters to plan the arm motion. For simulated robot which does not have such API available, we use an additional control layer that uses analytical inverse kinematics of the robot to move the simulated robotic arm in Cartesian and joint spaces. For arm motion planning, the most important thing is to make sure that the arm trajectory does not make the arm collide with the environment or itself. Motion planning for collision avoidance is an active area of research [21,22], especially an additional sensor can detect obstacles in the workspace [23–26]. Instead of using a full-blown path planning to avoid possible collision, we separate the task space into two regions and use different control strategy for each region to achieve collision-free arm motion without lengthy optimization.

- When the end effector is close to the support surface, for example during picking up and releasing a block, we only use vertical linear motions in Cartesian space, which guarantees that the gripper fingers do not touch other blocks thanks to the vertical pickup constraint we use for grasp planning. To make sure that the manipulator does not encounter singularity during the Cartesian space motion, we use the simulated environment to test all possible pickup and grasp motions at the designated areas.
- When the end effector has enough clearance from the support surface, we use the joint level movement using the analytic inverse kinematics of the manipulator to quickly move the arm to the target pose. To change the end effector orientation among perception, pickup and release poses, we use a set of arm joint angles where the 5th joint of the manipulator can be freely rotated without collision, which we call the safe joint angles. The 2nd, 3rd, 4th and 6th joints are first moved to the safe joint angles, and then the 5th joint is moved to the reach the safe joint angles without self-collision. The joint movements are done in reverse to reach target joint angles from safe joint angles. This process is shown in Figure 12.
- When a 2 DOF gripper is used, the additional DOF can be used for the pitch control, and we can keep the manipulator end-effector pitch angle to be zero. In that case, the direct joint space motion between two poses is guaranteed to have no self-collision for the workspace we assume and we skip the motion going through the safe joint angles to save time.

Once we generate motion plans in the Cartesian and joint space, we integrate multiple motions into one continuous motion with multiple waypoints, so that the manipulator executes consecutive motions without stopping in between. Figure 13 shows snapshots of such a continuous motion which includes picking up the block, moving and rotating the block to the target pose and placing the block to the final pose.

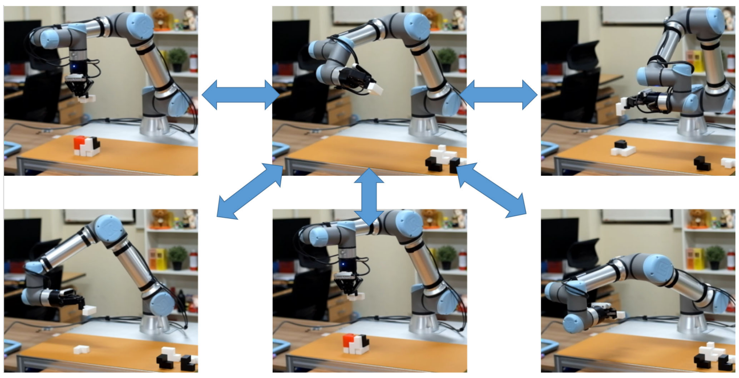
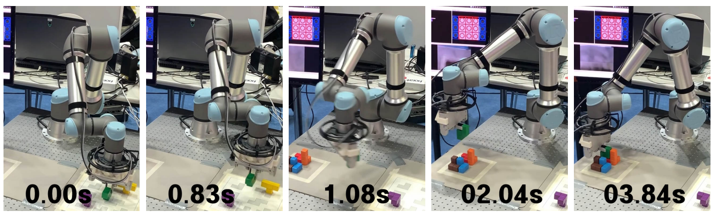


Figure 12. Collision free arm motion planning using the safe joint angles.



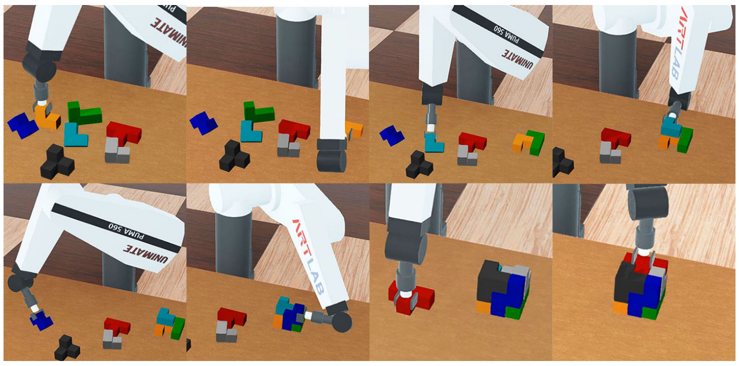
(**a**) (**b**) (**c**) (**d**) (**e**)

Figure 13. Arm motion plan for UR5e arm and 2 DOF gripper: (**a**) pickup start; (**b**) pickup end; (**c**) move and rotation to the release pose; (**d**) release start; and (**e**) release end.

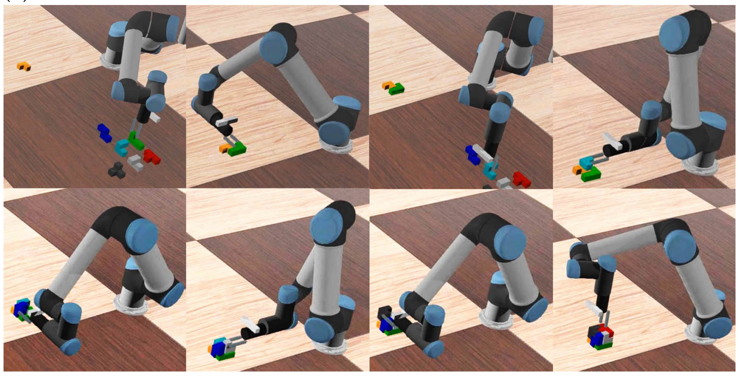
6. Simulation Results

We set up a simulation environment using the Webots open-source robotic simulator [27] to test the feasibility of the assembly task using different combinations of sensors, manipulators and grippers. We first used the PUMA 560 manipulator model with three wrist joint axes all intersecting at a common point, which eliminates the need for wrist collision-avoidance steps. The block poses were provided by the ground truth data from the simulator, and the 3D pose representation was used. Figure 14a shows the assembly process under such a simulation setup, where the motion planning algorithm successfully finds proper move sequence, grasp poses and arm motions to assemble the given blocks from blocks with arbitrary poses.

Figure 14b shows a more realistic simulation setup, where the UR5e manipulator model is used and block information is acquired using the depth-based perception module from a simulated depth camera mounted at the wrist of the manipulator. Collision-avoiding arm motion planning is used to prevent the self collision of the wrist actuators. We found that the depth-based block detection and the collision avoiding motion planning algorithm successfully let the system assemble target shape from individual blocks.



(**a**)



(**b**)

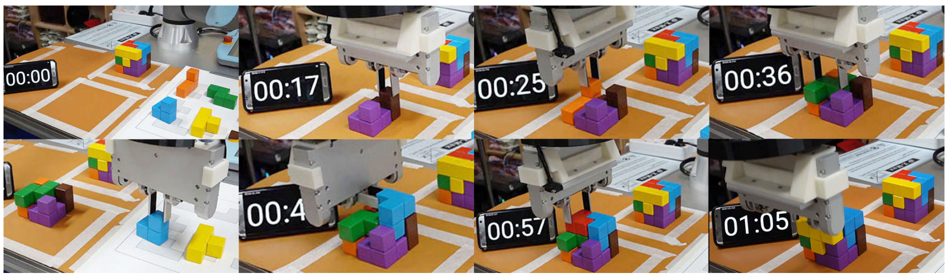
Figure 14. Soma cube assembly experiments in simulated environments: (**a**) simulation with PUMA 560 manipulator mode and ground truth block information; and (**b**) simulation with UR5e manipulator model and depth based perception.

7. Experimental Results

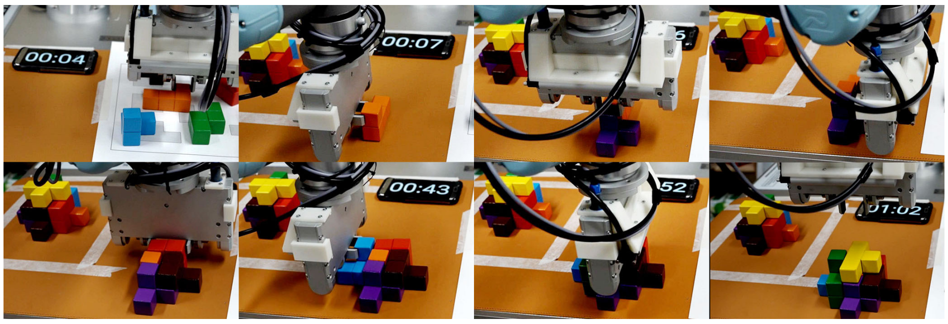
The suggested software system was implemented for physical UR5e robotic manipulator and custom 2 DOF gripper and tested with a number of block poses and target shapes in both lab environment and public demonstration.

7.1. Lab Environment

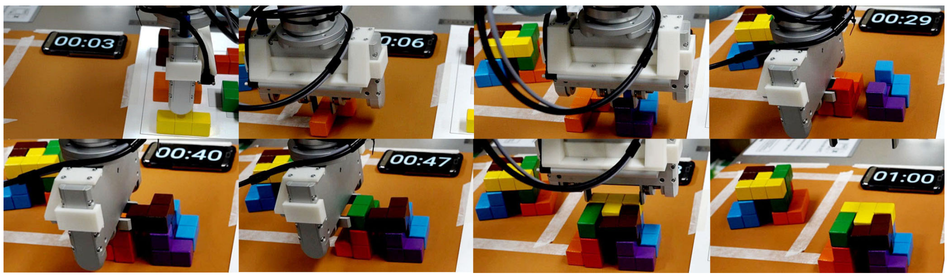
We first evaluated the system performance in the lab environment using multiple combinations of different initial block poses and target shapes. Figure 15 shows the building sequences for some of the standard shapes we tested and Table 1 shows the analysis of the result. For all target shapes we tested, the assembly solutions can be found very quickly, taking 1.64 ms on average for four target shapes tested. Checking feasible grasp for all the solutions takes significantly longer, 400 ms on average. As many solutions can have feasible grasps for all its blocks, we tried the faster but suboptimal planning strategy that stops at the first feasible solution and immediately executes the best move sequence found for that solution. We found that, at least for the three shapes that have feasible solutions, this strategy succeeds to find optimal solutions while reducing the planning time by more than a factor of 10. Thanks to an exhaustive grasp pose search considering block symmetries, only one of four target shapes has no feasible solution and requires re-grasping, which adds 14 s to the total time for additional perception, pickup and release movements. Overall, we found that assembly takes approximately 1 min on average.



(**a**)

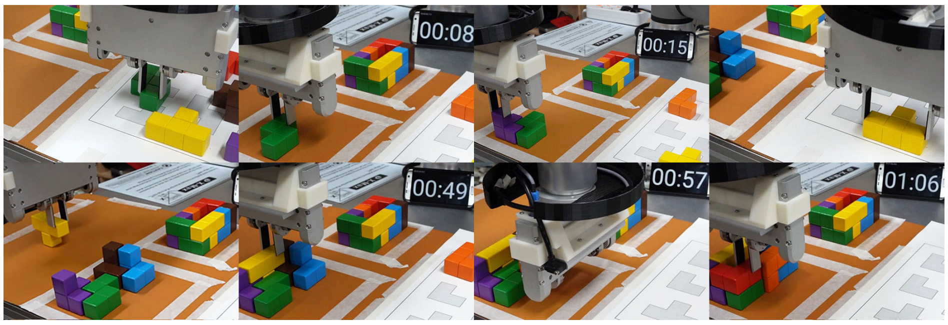


(**b**)



(**c**)

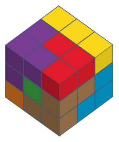
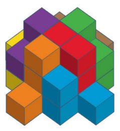
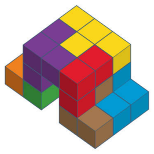
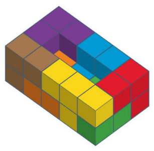
Figure 15. *Cont.*



(**d**)

Figure 15. Automatic assembly experiments in the lab setup: (**a**) 3 × 3 × 3 cube; (**b**) pyramid shape; (**c**) bridge shape; and (**d**) bathtub shape.

Table 1. Summary of lab experiment results.

Target Shape				
Total number of assembly solutions	240	39	13	79
Time to find all the solutions (ms)	3.5	1.0	0.47	1.58
Time for checking all the solutions (ms)	908.9	557.6	56.69	78.57
Number of non-re-grasp solutions	3	3	1	0
Cost of the first non-re-grasp solutions	620	620	590	-
Time to find the first non-re-grasp solution (ms)	40.2	18.6	56.65	-
Best cost of non-re-grasp solutions	620	620	590	-
Average cost of non-grasp solutions	768	780	590	-
Time spent for re-grasp searching (ms)	-	-	-	4.04
Total planning time (ms)	**43.7**	**19.6**	**57.12**	**82.54**
Average time spent for grasping blocks (s)	1.92	1.87	1.84	1.97
Average time spent for moving blocks (s)	3.59	3.47	2.94	3.26
Average time spent for returning (s)	2.98	2.39	2.80	2.54
Extra time spent for re-grasping (s)	-	-	-	14
Total assembly time (s)	**64**	**62**	**59**	**66**
Total path length (m)	**17.17**	**16.36**	**16.94**	**18.43**

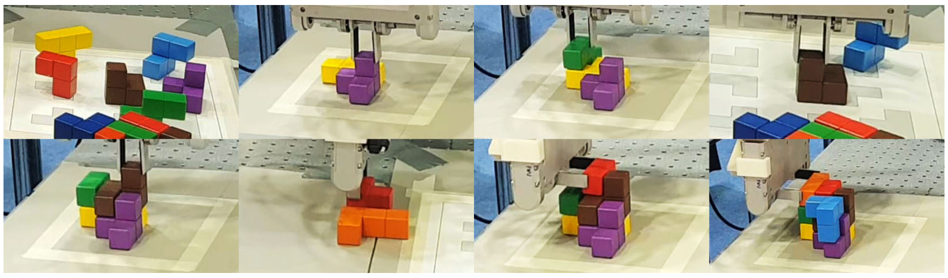
7.2. Public Demonstration

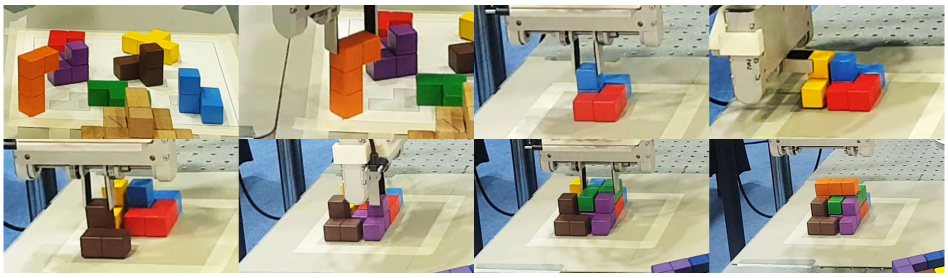
The suggested system was validated by a competitive public demonstration held at COEX, Seoul, Korea in December 2019. The public demonstration was held as the one of the selection processes of competitive research funding provided by the Korean Ministry of Science, ICT and Future Planning. Out of four teams that submitted the proposals, two teams were selected for the first phase, and one team was selected after the public demonstration and funded for the second phase of the research.

The demonstration was divided into three stages with different difficulties, and at each stage the referees decided the target shape and initial block formation. Once the timer started, each team began assembling the blocks autonomously, and the total score was determined by whether the system provided correct assembly plans on display, how many blocks were successfully picked up and placed correctly, if the target shape was successfully assembled and the total time spent for the assembly. In the case of failure, each team could request the referee reset the block formation for restart, with a time penalty. Each stage used two different sets of target shapes and initial block formations, and the best score of the two runs was recorded as the stage score.

To make the system as reliable as possible, we used lower maximum acceleration and velocity parameters for the manipulator motion control, and added more delay for block pickup and perception transitions by 0.5 s. In addition, we added the pre-positioning moves, which moved blocks with ambiguous orientation and tall blocks that could collide with gripper during pickup motion to the re-grasp position before motion planning. Finally, to handle the worst cases where the initial block formation was extremely close to the assembled target shape, we added an additional "plowing" motion that moved away the assembled blocks away from the blocks to be picked up. Even with those conservative settings, we achieved the official assembly time of 1 min 38 s at the third stage, which is more than three times faster than the other team's third stage assembly time exceeding 5 min.

Figure 16 shows a number of block assembly sequences at the public demonstration setup for hard assembly cases with upright blocks and Table 2 shows the detailed analysis of them. One of the four shapes, the tower shape, had a very large number of assembly solutions, which made the full grasp search take a fairly long 6497.59 ms to complete. Using the first feasible solution decreased the search time by more than a factor of 100 and made the total motion planning time less than 100 ms. In addition, due to the tricky initial block placements, all four target shapes required the initial repositioning of blocks, which was major cause of increasing the total assembly time compared to the lab experiments without one. Figure 17 shows a detailed assembly sequence with multiple re-grasping of blocks.


(**a**)


(**b**)

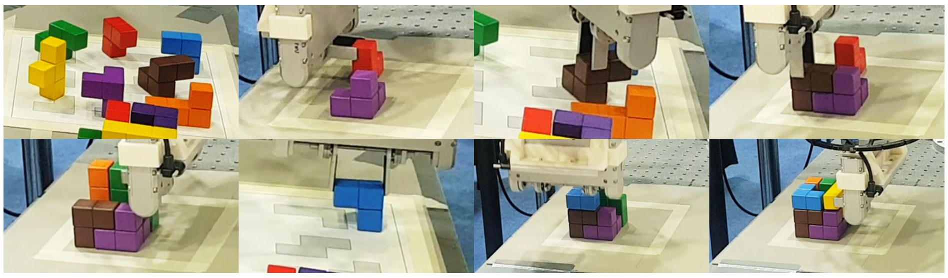

(**c**)

Figure 16. *Cont.*

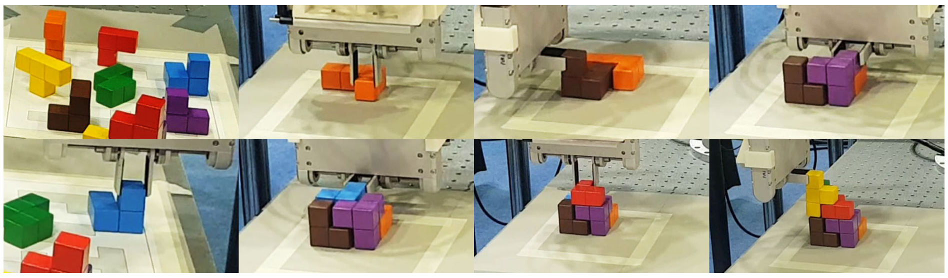

(**d**)

Figure 16. Automatic assembly runs during the public demonstration: (**a**) apartment shape; (**b**) stair shape; (**c**) well shape; and (**d**) tower shape.

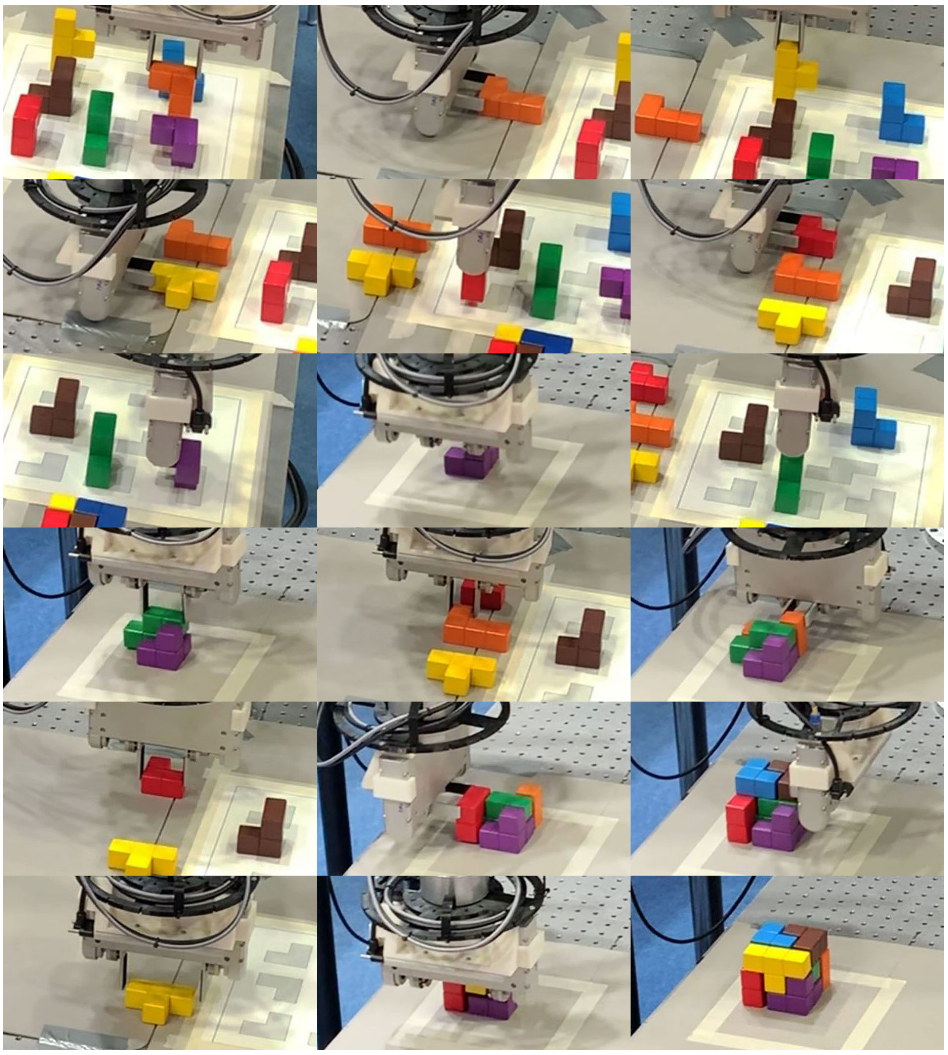
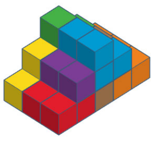
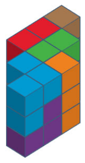
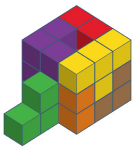
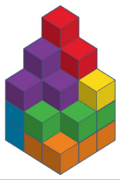


Figure 17. Detailed assembly steps for a 3 × 3 × 3 cube.

Table 2. Summary of Public Demonstration Results.

Target Shape				
Total number of assembly solutions	82	122	147	1400
Time to find all the solutions (ms)	1.86	2.4	1.42	14.74
Time for checking all the solutions (ms)	994.5	1478.1	786.03	6497.59
Number of non-re-grasp solutions	10	2	5	117
Cost of the first non-re-grasp solutions	500	480	750	720
Time to find the first non-re-grasp solution (ms)	52.0	463.32	43.66	55.84
Best cost of non-re-grasp solutions	450	480	360	420
Average cost of non-grasp solutions	598	490	633	686
Time spent for re-grasp searching (ms)	-	-	-	-
Total planning time (ms)	**53.86**	**465.72**	**45.08**	**70.58**
Average time spent for grasping blocks (s)	2.97	2.34	2.54	2.42
Average time spent for moving blocks (s)	4.30	4.13	4.24	4.38
Average time spent for returning (s)	3.35	3.45	3.39	3.75
Number of blocks initially repositioned	1	2	2	2
Extra time spent for initial repositioning (s)	12.54	23.83	24.12	24.15
Total assembly time (s)	**104**	**110**	**113**	**115**
Total path length (m)	**17.04**	**16.37**	**17.97**	**16.97**

7.3. Discussions

7.3.1. Time Analysis

Figure 18 shows the average time decompositions from the data in Tables 1 and 2. We found that the average time spent for motion planning is very short, occupying 0.11% of the total assembly time on average and 0.42% for the worst case. Although the perception itself is done very quickly, taking less than 20 ms, we added some delay to stabilize the camera before perceiving the block, which adds up to 14.9% of the total assembly time for the open demonstration case. We also used delays before and after the gripper actuation to reliably grasp and release blocks, which adds up to 13.8% of the total assembly time for the open demonstration case. Finally, the arm movement time takes more than 70% of the total assembly time, which differs by approximately 25% according to the acceleration and velocity parameters used. As each of these steps has room for improvement, we think the total assembly time can be further reduced in the future.

7.3.2. Effect of Additional DOF of Gripper

Although we used a custom 2 DOF gripper for the public demonstration, the suggested system does not rely upon one, and the system with 1 DOF gripper was tested in both simulation and real environments to have no workspace or self-collision issues. We did not measure the assembly time with 1 DOF gripper setup, but, even assuming the arm movement times triple due to the collision avoidance motion planning, we estimate the total assembly time will increase by approximately 60% based on the timing breakdown in Table 1.

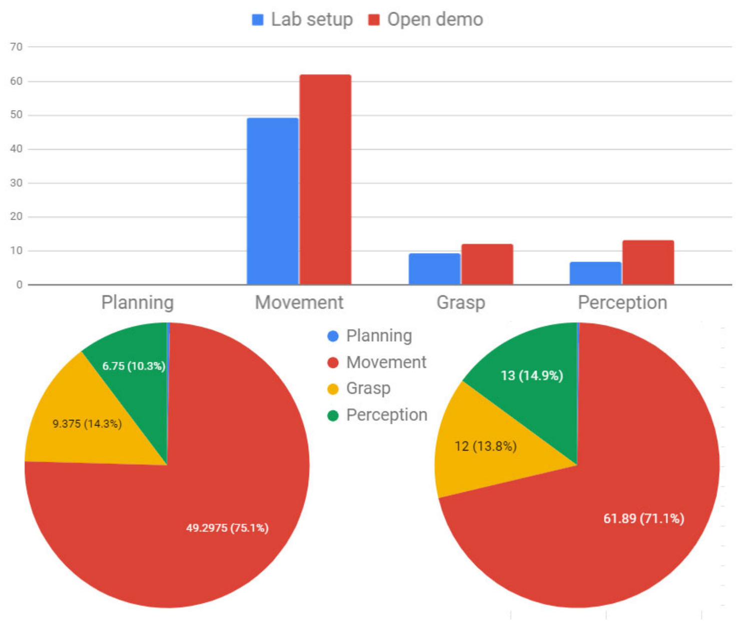


Figure 18. Time decomposition of lab experiment and open demonstration.

7.3.3. Generalization to More Tasks

In Section 3, we list the assumptions that helped to keep the perception and motion control simpler for the Soma cube assembly problem. However, those assumptions can be removed to make the suggested system handle a wider range of tasks. For detection and 6D pose estimation of arbitrary building components, we can use deep learning-based approaches using RGB image [16,17] in addition to point cloud-based registration methods [28] to further improve the accuracy. Reliable detection and grasping of cluttered objects is a hard problem, where a scattering-based technique can be used to separate each object [29]. Finally, the assembly tasks that require peg-in-hole assembly or screwing can be realized by adding a second robotic arm for support during the assembly, as well as adding force or compliance based peg-in-hole motion controller (e.g., [30]).

We further think that the suggested system, especially with the help of the interactive human–robot collaboration, which we discuss below, can play a big role in future industry environments, which need to quickly adapt to a fast changing market demand with a high variety of products in small quantities. Compared to typical industrial robot setup that requires pre-scripted motion sequences and custom designed fixtures for assembly parts, the suggested system is much more flexible as it can autonomously plan for pickup sequences and assembly motions without human programming, and custom fixtures for new parts are not required as the perception subsystem can detect arbitrarily placed parts.

7.3.4. Extension to Human-Robot Collaborative Tasks

While the suggested system is only tested in a fully autonomous setup, where a human sets up the target shape and the robot does the rest, the system can be straightforwardly extended for a semi-autonomous setup where human operators interact with the robot to help and guide the assembly process using voice-, gesture- or touch-based communication. For example, the robot may ask human operator to disambiguate visual perception or move or hold parts that exceed the capability of the robotic manipulator. To ensure a safe operation, the current setup already uses a collaborative robotic manipulator with a safety stop feature, and we can make the system safer and more robust by adding additional sensors to detect the presence of human operator inside its workspace before movements.

7.3.5. Distinctive Contribution of the Paper

As presented in Section 2, the distinctive contributions of the software framework we suggest in this work are as follows: (a) full integration of various software modules required for accomplishing assembly task with full autonomy, which includes 3D perception, assembly planning, arm and gripper motion planning, re-grasp planning and real-time control; (b) platform agonistic modular software that accepts different sensors, manipulators and grippers with little effort; and (c) rigorously tested and proven to work reliably in a public competitive demonstration.

Compared to previous works addressing integrated robotic assembly system [7,8], the suggested system is applicable for much wider range of tasks as it includes explicit assembly sequence planning for complex tasks. Compared to recent works that use Soma cube assembly as a demonstration task [10–14], this work is arguably more practical as it does not require a priori information of the block states and is much faster overall due to a more optimized motion planning stage.

We are currently working on the public release of our software framework, which we hope will help other researchers in the field.

8. Conclusions

In this work, we present a robotic system that integrates a highly accurate 3D perception module, multi-layer motion planning module and real-time motion control module to autonomously perform a complex robotic assembly task without any human instruction. The suggested system is platform agonistic and was tested with multiple sensor, manipulator and gripper combinations in both simulated and real environments. The system was tested extensively in lab environment and public demonstration, where it showed excellent reliability and outstanding performance. Future work will include generalizing the framework to support a wider range of practical assembly tasks, by adding a precise insertion stage using the force and torque feedback.

Author Contributions: Methodology, S.-J.Y.; Software, T.K. and J.-B.Y.; and Hardware, T.K. and D.S. All authors have read and agree to the published version of the manuscript.

Funding: This work was partly supported by the Institute for Information & Communications Technology Promotion (2018-0-00622-RMI), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2018R1C1B5045389) and Pusan National University Research Grant, 2017.

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

Abbreviations

The following abbreviations are used in this manuscript:

DOF	Degree of Freedom
RRT	Rapidly-exploring Random Tree

References

1. Foumani, M.; Jenab, K. Cycle time analysis in reentrant robotic cells with swap ability. *Int. J. Prod. Res.* **2012**, *50*, 6372–6387. [CrossRef]
2. Foumani, M.; Jenab, K. Analysis of flexible robotic cells with improved pure cycle. *Int. J. Comput. Integr. Manuf.* **2013**, *26*, 201–215. [CrossRef]
3. Billard, A.; Kragic, D. Trends and challenges in robot manipulation. *Science* **2019**, *364*, eaat8414. [CrossRef] [PubMed]
4. Thomas, U.; Stouraitis, T.; Roa, M.A. Flexible assembly through integrated assembly sequence planning and grasp planning. In Proceedings of the 2015 IEEE International Conference on Automation Science and Engineering (CASE), Gothenburg, Sweden, 24–28 August 2015; pp. 586–592.
5. Tobin, J.; Biewald, L.; Duan, R.; Andrychowicz, M.; Handa, A.; Kumar, V.; McGrew, B.; Ray, A.; Schneider, J.; Welinder, P.; et al. Domain Randomization and Generative Models for Robotic Grasping. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 3482–3489.
6. Levine, S.; Pastor, P.; Krizhevsky, A.; Ibarz, J.; Quillen, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Robot. Res.* **2018**, *37*, 421–436. [CrossRef]

7. 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 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Algarve, Portugal, 7–12 October 2012; pp. 2955–2962.
8. Ahmad Shauri, R.L.; Nonami, K. Assembly manipulation of small objects by dual-arm manipulator. *Assem. Autom.* **2011**, *31*, 263–274. [CrossRef]
9. Morrison, D.; Tow, A.W.; McTaggart, M.; Smith, R.; Kelly-Boxall, N.; Wade-McCue, S.; Erskine, J.; Grinover, R.; Gurman, A.; Hunn, T.; et al. Cartman: The Low-Cost Cartesian Manipulator that Won the Amazon Robotics Challenge. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 7757–7764.
10. Wan, W.; Harada, K. Integrated assembly and motion planning using regrasp graphs. *Robot. Biomim.* **2016**, *3*, 1–11. [CrossRef] [PubMed]
11. Wan, W.; Harada, K.; Nagata, K. Assembly Sequence Planning for Motion Planning. *Assem. Autom.* **2018**, *38*, 195–206. [CrossRef]
12. Ali, A.; Lee, J.Y. Integrated Motion Planning for Assembly Task with Part Manipulation Using Re-Grasping. *Appl. Sci.* **2020**, *10*, 749. [CrossRef]
13. Chen, H.; Wan, W.; Koyama, K.; Harada, K. Planning to Build Soma Blocks Using a Dual-arm Robot. *arXiv* **2020**, arXiv:2003.00699.
14. Dobashi, H.; Hiraoka, J.; Fukao, T.; Yokokohji, Y.; Noda, A.; Nagano, H.; Nagatani, T.; Okuda, H.; Ichi Tanaka, K. Robust grasping strategy for assembling parts in various shapes. *Adv. Robot.* **2014**, *28*, 1005–1019. [CrossRef]
15. Chen, L.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 834–848. [CrossRef] [PubMed]
16. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.
17. He, Y.; Sun, W.; Huang, H.; Liu, J.; Fan, H.; Sun, J. PVN3D: A Deep Point-Wise 3D Keypoints Voting Network for 6DoF Pose Estimation. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 11629–11638.
18. Liu, S.; Carpin, S. Global grasp planning using triangular meshes. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 4904–4910.
19. 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]
20. Asif, U.; Bennamoun, M.; Sohel, F.A. RGB-D Object Recognition and Grasp Detection Using Hierarchical Cascaded Forests. *IEEE Trans. Robot.* **2017**, *33*, 547–564. [CrossRef]
21. Yershova, A.; LaValle, S. Motion Planning for Highly Constrained Spaces. *Lect. Notes Control Inf. Sci.* **2009**, *396*. [CrossRef]
22. You, J.; Kim, D.; Lim, S.; Kang, S.; Lee, J.Y.; Han, C. 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.
23. Novak, J.L.; Feddema, I.T. A capacitance-based proximity sensor for whole arm obstacle avoidance. In Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France, 12–14 May 1992; Volume 2, pp. 1307–1314.
24. 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, Alaska, 4–8 May 2010; pp. 831–838.
25. 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 2019 IEEE International Symposium on Robotic and Sensors Environments (ROSE), Ottawa, ON, Canada, 17–18 June 2019; pp. 1–6.
26. Conte, G.; Scaradozzi, D.; Rosettani, M. E-field Sensors and Sensor-based Control Strategies for M/M Safe Cooperation. *IFAC Proc. Vol.* **2011**, *44*, 8070–8075. [CrossRef]
27. Michel, O. Webots: Professional Mobile Robot Simulation. *J. Adv. Robot. Syst.* **2004**, *1*, 39–42.
28. Wang, Y.; Solomon, J.M. Deep Closest Point: Learning Representations for Point Cloud Registration. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27–28 October 2019.
29. Imran, A.; Kim, S.H.; Park, Y.B.; Suh, I.H.; Yi, B.J. Singulation of Objects in Cluttered Environment Using Dynamic Estimation of Physical Properties. *Appl. Sci.* **2019**, *9*, 3536. [CrossRef]
30. Park, H.; Park, J.; Lee, D.; Park, J.; Baeg, M.; Bae, J. Compliance-Based Robotic Peg-in-Hole Assembly Strategy Without Force Feedback. *IEEE Trans. Ind. Electron.* **2017**, *64*, 6299–6309. [CrossRef]