A virtual capture framework for assembly tasks

Damien Petit[1*] Ixchel G. Ramirez-Alpizar[1]
Qiming He[1] Kensuke Harada[1,2]

***Abstract*— Nowadays more robots are used in manufacturing to accomplish more complex tasks, especially in assembly. Programming by demonstration method allows to teach robots some specific tasks, but it requires a lot of data. However in assembly, wearable motion capture device (like data-glove) are often difficult to set-up, calibrate and use with other objects. To solve this issue we propose a new capture method of assembly tasks to be used without any wearable capture devices, in which real hands are used to assemble virtual objects to collect the poses and contact points between hands and objects along the assembly process. The final goal of this work is to develop a system to easily collect data of a human assembly demonstration, analyse them, and realize the assembly on a dual-arm robot.**

I. Introduction

Teaching robots how to do a task is indispensable for the automation of manufacturing processes. In recent years, robots are required to perform different tasks instead of repeating the same task over and over again. On the other hand, recently, a number of research has been done on motion planning algorithms to automate the motion generation of a robot. However, if the planned trajectory is not executable due to the existence of unknown obstacle, the robot will not move and a new trajectory has to be computed after precisely measuring the environment shape by using a 3D vision sensor [1].

Traditionally, to compute a joint trajectory of a robot manipulator tracking the desired end effector trajectory, a robot is programmed by using inverse kinematics [2]. However, in clutter environments, the problem of computing a motion trajectory is non-trivial, as the robot needs to avoid obstacles in the environment by itself. The workload of programming a robot for complex tasks becomes increasingly difficult, as most of the environment is not static [3]. For an assembly task, the environment is usually dynamic and involves several tasks with different assembly components. Which is why most of the assembly tasks remain done by humans.

In the method of programming by demonstration (PbD) [4], the motions of humans doing the desired task are recorded and used to program the robot's motion. Aleotti et al. [5] use this method to program a robot in a virtual environment. In our set-up shown in Fig. 1, we use real hands

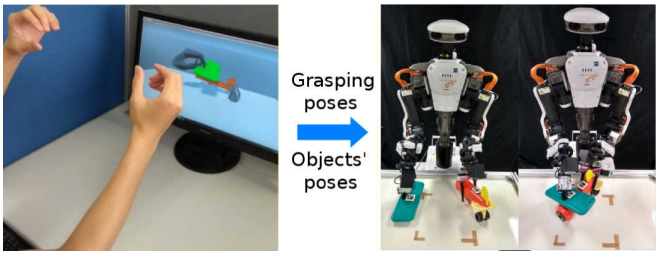


Fig. 1: Experimental environment and aim of the research

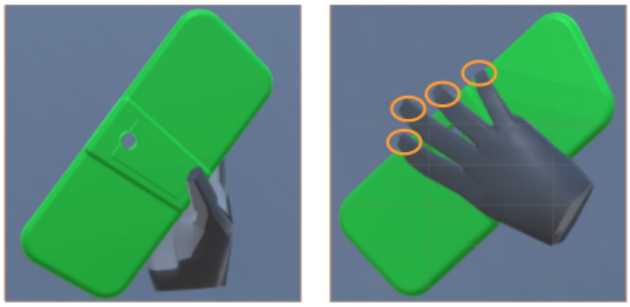


Fig. 2: The occlusion and object tracking problems are inexistant in the virtual environment. Moreover all the virtual poses (objects and hand joints) are known at each frames

to manipulate the virtual objects as to avoid the problem related to occlusions and object tracking [6] as seen in Fig. 2.

In this paper, we build up a 3D scene where human hands detected by a Leap Motion Controller (LMC) [a] are displayed and interact with the virtual objects to be assembled. When the hands are moved in the real world, they are also moved in the virtual world so that we can record accurate hands' poses [7]. The user can either put the LMC on the table below his hands or wears the LMC on his forehead. We record the fingers' positions and object parts' (in this study a toy airplane) poses when carrying out the assembly task. The recorded data will later be used to reproduce the same assembly task in a robot's simulator software to confirm the validity of the proposed method.

Two of the difficulties and the main contributions of this work were the development of a custom collider and virtual assembly conditions used to render realistically the assembly of the virtual objects between themselves and their interaction with the virtual hands. As it will be shown later in the paper, the usual colliders (convex, cubic or concave) were unable to be used for this purpose and a new collider

[1] Department of Systems Innovation, Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama-cho, Toyonaka, 560-8531, Japan.
[2] Manipulation Research Group, Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology (AIST), 1-1-1 Umezono, Tsukuba, 305-8560, Japan.
* Corresponding author e-mail: damien.petit@hlab.sys.es.osaka-u.ac.jp

[a] https://www.leapmotion.com/

had to be developed.

This paper is organized as follows. In section II the modules of the framework are presented. Next, in section III the results of the assembly task recorded via the framework are presented. Finally, in section IV we discuss the results of the assembly framework and future work.

II. Framework

In this section we present the different modules of the framework used to capture an assembly task to be able to play it back on a dual-arm robot. In this paper we focus on the assembly tasks of a toy airplane but the implementation of the framework allows the import and assembly of any object model.

A. Hand tracking

There are many ways to track human's hands [8] [9] [10]. We chose the tracking sensor named 'Leap Motion', which can track hands so that people do not need to wear motion capture devices on their hands, as shown in Fig.1. The LMC has two cameras and three infrared LEDs to track hands visually.

We use the Interaction Engine API provided by the LMC and import the toy airplane 3D models to build the virtual environment. The assembled parts are shown in Fig.3. When the hands are tracked, you can interact with the virtual objects. The user is able to see his virtual hands and the object on the screen while performing the virtual assembly as seen in Fig.1 It is possible to grasp an object similarly as grasping it in the real world.

B. Assembly conditions

In order to explain the assembly conditions that we defined, we take as an example the assembly task of the top wing on the main body of the airplane as shown in Fig. 3. At first, both of the objects can be moved separately, as shown in Fig. 3 namely Disassembled state. When the two objects are moved to the position and orientation which respect the assembly conditions, they will be assembled. Then the assembled objects behave as a single object, i.e. all parts of the object are moved together.

The conditions for assembling the body and the wing are the following (Each condition can be satisfied in any order):

1) The angle β between the two objects' Y axes condition is: $0 \leq \beta \leq \theta_1$.
   Initially, the body and the wing are placed randomly as shown in Fig.4. We define the angle between two objects' Y axes as β. When the value of β is nonnegative and less than θ_1, we regard this condition as satisfied.

2) The angle γ between the two objects' Z axes condition is: $0 \leq \gamma \leq \theta_2$. To ensure that the two objects are in the desired orientation, we also set a condition on the angle γ, which represents the angle between each objects' Z axes, as shown in Fig. 5. When the value of γ is non-negative and less than θ_2, we regard this condition as satisfied. In addition to the first condition, the two objects are in the same orientation.

3) The vector from the desired position to the current position is within a sphere radius around the desired position center.
   Besides the orientation, we also have to consider the relative position between the objects. Assuming that one object (the airplane body) is fixed, then the other object's (the airplane wing's) position for which two objects can be assembled is called 'desired position' when it is at 'current position'. We set the length of vector P_B from the desired position center O_{Bd} to the current position center O_{Br} (Fig.6), to be less than the radius of the sphere around O_{Bd}.

When these three conditions are satisfied, the two objects will be assembled. The angle θ is a value that can be adjusted. During the experiment their values were set as $\theta_1 = \theta_2 = 10$ deg. The assembled airplane body and wing are shown in Fig.3.

C. Assembly order

Assembly tasks usually involve more than two objects. Planning every possible assembly order is difficult because of the large workload. In this research, different assembly orders are expected to be recorded. And/Or graph can express possible assembly steps. Fig.7 shows that an And/Or graph for 4 components (from left to right: cockpit, wing top, body upper, body lower) in this experiment. There are 10 assembly steps in total. Each step means the assembly of two components.

To express all possible assembly orders in a relatively easy way, a tree structure shown in Fig. 8 is presented. In this tree, every node is a component of the assembly and all of the assembled components are included. The root node can be any single component. Here we use the 'body upper' as the root. After an assembly between two objects, a parent-child relationship is generated. The object with arrow tail becomes the parent of the object with arrow head. When the whole assembly is finished, the parent-child relationship between objects is same as the tree structure. Generally, only two objects that are connected can be assembled.

For example, considering the assembly of body upper, body lower, wing top and cockpit (all of them are marked blue in Fig. 8). At first, only assemblies between body upper and body lower, body upper and wing top or wing top and cockpit are allowed. By the step 8 shown in Fig. 7, the cockpit and wing top are assembled, then the assembly between cockpit and body upper (step 5) is allowed because the cockpit is connected with the wing top that can be assembled with body upper. After step 5, body upper and body lower can be assembled and the assembly of these 4 objects is finished.

D. Visual guidance for assembly task

To realize the assembly in virtual environment, the user is able to receive some visual cues to help him understand how to assemble the objects. This function rely on the knowledge of the And/Or graph. In most cases, the user feel uneasy to

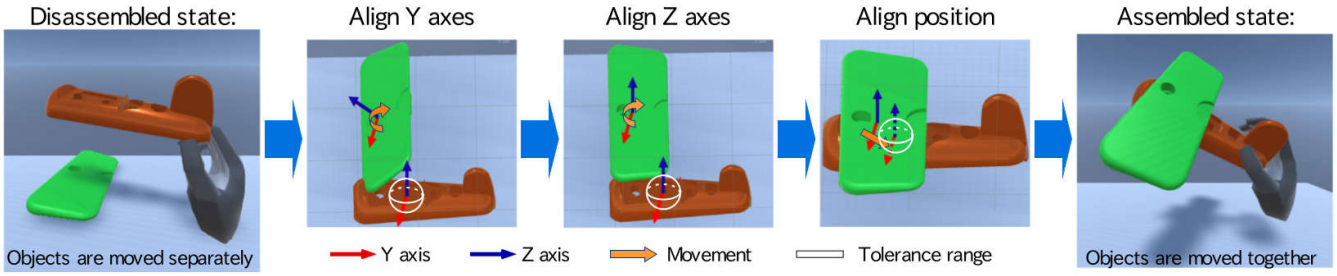


Fig. 3: The different assembly conditions

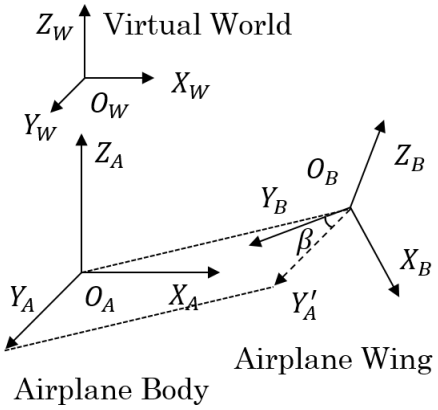


Fig. 4: Condition 1

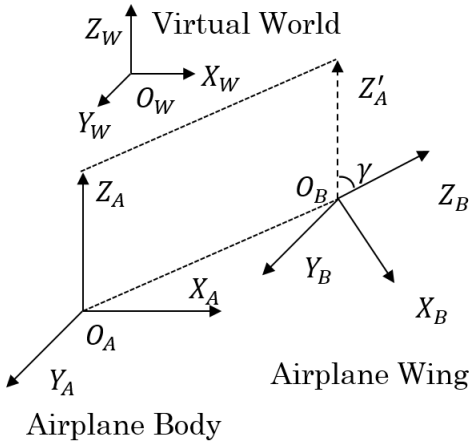


Fig. 5: Condition 2

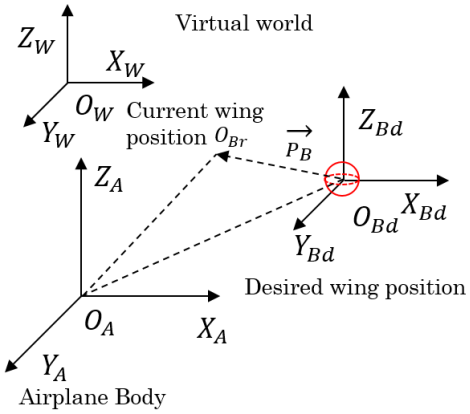


Fig. 6: Condition 3

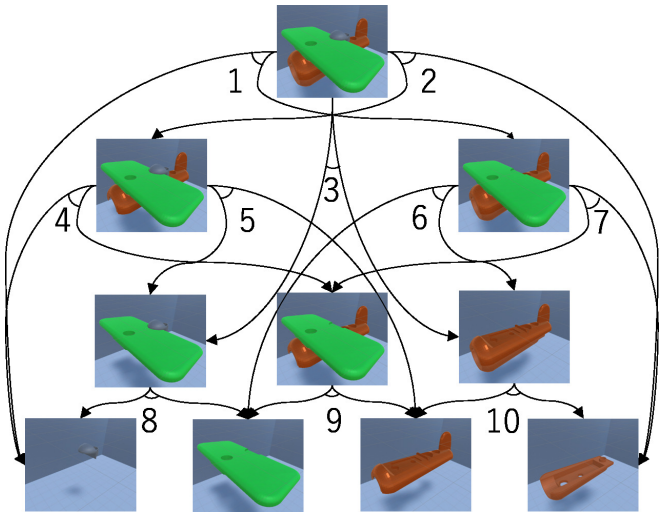


Fig. 7: And/Or Graph

the assembly components at first (due to the lack of haptic feedback), so the guidance for the assembly can be necessary.

We designed two different guidance methods. One is to show the video of the assembly and the other one is to show arrows between the two objects as seen in Fig.9.

We add a screen in the background to show the assembly tutorial video in the scene (Only if necessary). At the beginning, there is no screen until the user grasps two objects. Then the screen shows the assembly task of these two specific parts. So that the user is able to understand how to assemble those two parts.

The arrow is shown as soon as an object is grasped. The arrow is half transparent and its color is set according to the object's color. We take the RGB value of the object mesh into consideration.

We consider that the object's RGB values are r, g, b; then the arrow's RGB value will be r', g', b' as defined bellow:

- If r $\leq$ 127, r' = r + 128, else r' = r - 128.
- If g $\leq$ 127, g' = g + 128, else g' = g - 128.
- If b $\leq$ 127, b' = b + 128, else b' = b - 128.

Besides the color, we also set the tail point and the direction of the arrow. In the toy airplane case, the arrow tail point of

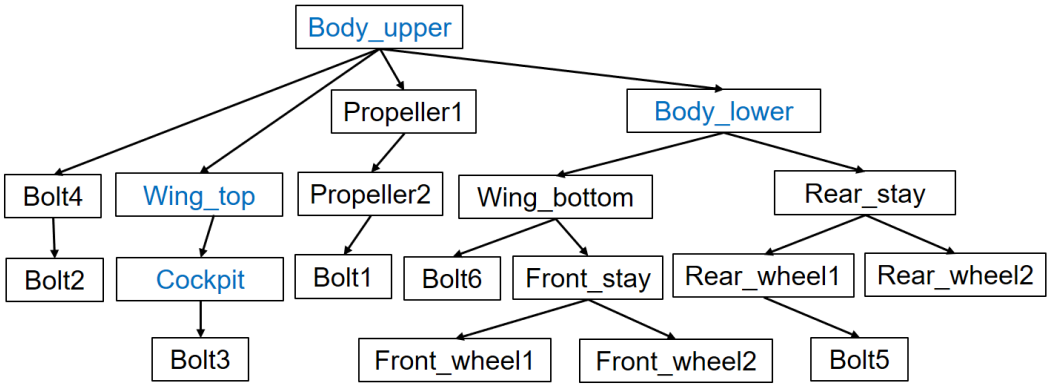


Fig. 8: Tree Structure

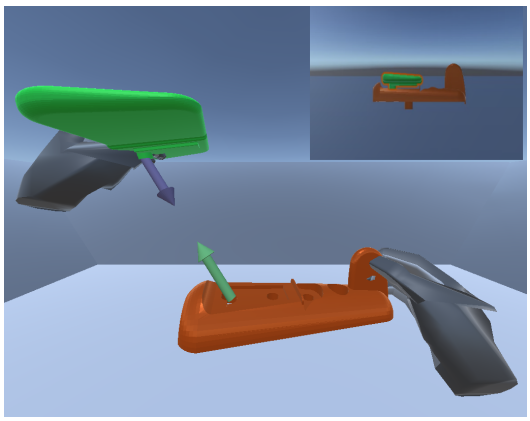


Fig. 9: Visual assembly guidance

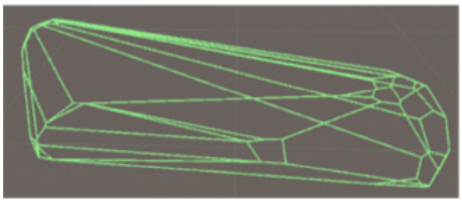


Fig. 10: Convex collider

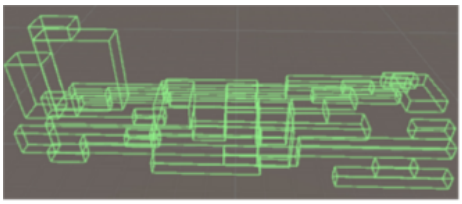


Fig. 11: Cubic collider

the body upper is the airplane body's center of the assembly. The direction is pointed to the arrow tail attached on the target assembly object (wing top) as seen in Fig. 9.

E. Collider

In order to detect the contacts between the fingers and the objects as well as the contacts between objects it is necessary to use colliders. After empirically testing three different types of colliders (convex shown in Fig. 10, cubic shown in Fig. 11, concave shown in Fig. 12), we came to the following conclusion. We need the collider to have a high accuracy and a low computing cost. However the three colliders we tried, did not satisfied our demands for object-fingers contacts and object-object contacts. Only the convex collider satisfied the demand of the object-fingers contact (but with poor spacial accuracy as seen in Fig. 10 which lead to a gap during the assembly as seen in Fig. 14) nonetheless it did not satisfy the demand of the object-object contact. On the other hand the convex collider did satisfy our demand of object-object contact, however due to its complexity (as seen in Fig. 12) its interaction behavior with the fingers is quite slower than the other colliders, and thus not feasible for continuous motions like assembly tasks. For these reasons we developed a point collider Fig. 13.

The point collider is composed of basic spherical colliders while respecting faithfully the shape of the object. Compared to the concave collider, the point collider is simpler, so the computational cost is lower. Compared to the cubic and convex colliders, the point collider has higher accuracy. With the point collider, we are able to get a higher accuracy with a relatively low computing cost. The point collider is generated with the following steps:

1) Use Poisson-disk algorithm sampling points V_i from the object mesh.
2) Compute the normal N_i for each point V_i.
3) Translate V_i along the direction of N_i as: $V_i' = V_i - N_i * r$, where r is the radius of the sphere colliders.

With step 1, we generate points on the surface of the mesh. Since we will add sphere colliders at each point position, the collider would be bigger than the mesh (by the size of the radius of the sphere collider). To make the size of the point collider the same as the mesh size, we translate the points' positions of the sphere colliders along their normal. For the assembly task discussed in this work, the radius of the sphere colliders is set to 2 mm.

Even if the point collider's computing cost is low for finger-object contact, the computational cost for the object-object contact is still considerably high due to the large number of spheres on each model. In order to improve this,

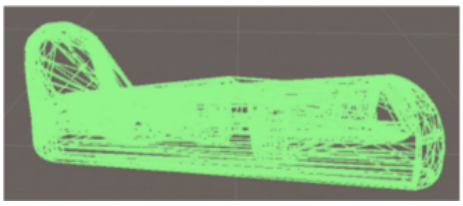


Fig. 12: Concave collider

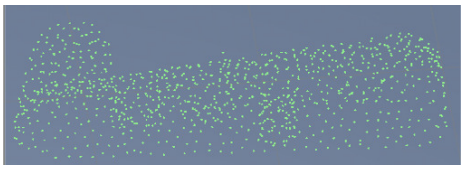


Fig. 13: Point Collider

we use 2 point colliders with a different number of spheres. The model with a high number (1000 spheres) is used for finger-object contact, and the model with a low number (200 spheres) is used for object-object contact.

Depending on whether the manipulated object is grasped or not, we exchange the colliders. Before the object is grasped, the collider attached is the "accurate" point collider (high number of spheres), so we can get accurate grasping points. After the object is grasped, the collider is changed to the "fast" collider (low number of spheres) so that the movement of the object is smoothly and the assembly gap is smaller than with the convex collider.

We exchange colliders when the grasp state changes. However, when the hand is trying to grasp the object, the grasp state is usually unstable and the collider changes every sampling frame. To ensure the stability of the grasp state, we regard the object as grasped if the object is computed as grasped continuously for 5 frames in a row. We also regard the object as not being grasped if it is computed as not grasped continuously for 5 frames.

F. Screwing motion

Not all of the objects can be assembled by placing them in the desired position. In section II-B, the assembly between the airplane body and the airplane wing is described, but there are some objects with thread (like shown in Fig. 16) that do not follow the same conditions. This kind of objects need to be screwed to be assembled and this can be achieved by hands or tools. Indeed, the bolt shown in Fig. 16 is almost fully inserted into the hole of the object so it is difficult to be assembled by hands. In this case supporting tools to assemble the bolt are needed. The screw driver shown in Fig. 16 is used for the assembly of bolts. When the blue part is inserted into the concave part of the bolt, the bolt is able to rotate with the screw driver so that the bolt can be assembled.

To assemble objects with a thread in the virtual world, we defined the following rule taking into account the pitch of the thread.

The definition of a bolt is shown in Fig. 16. When the bolt rotates θ deg, it will move forward for $Pitch \cdot \frac{\theta}{360}$. The bolt cannot be assembled until it has moved forward for TL.

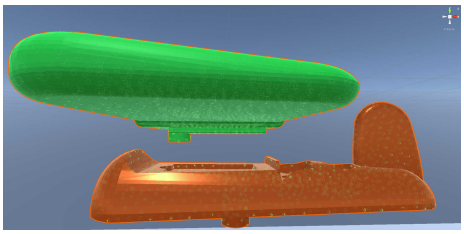


Fig. 14: Assembly using the convex collider resulting with a large gap between the wing and body

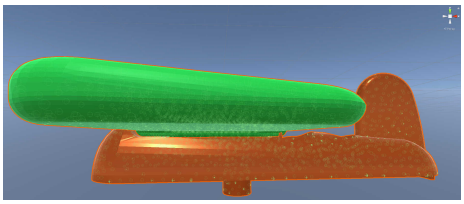
:w 

Fig. 15: Assembly using the point colliders resulting without a gap

The screwing motion is difficult to operate in the virtual world because there is no haptic feedback and the object to screw is usually smaller than a finger. To make the screw motion work properly, the target object (that the bolt will be insert in) is set to be in a fixed position and orientation. After the bolt is assembled with the target object, the target object becomes free in position and orientation.

G. Data recording

Finally, we record the data that will be used to control the robot and save it in a CSV format file. The data is recorded at a frame rate of 50 fps. The necessary data are:

1) Time stamp.
2) Palms' poses.
3) Fingers' poses.
4) Fingers' contact value (whether the finger is colliding with any objects or not).
5) Objects' poses.
6) Objects' grasped value (whether the object is grasped by left or right hand).

III. Results

When we direct the wing (from above) toward the airplane body in the defined contact pose, the wing and the body will be assembled into one object. Then, we can move the body and the wing together by grasping any part of the assembled object, even if we only grasp the wing or the body. We can then assemble the rest of the airplane following the same procedure. During all the procedure, the data is recorded in the CSV file.

To make the assembly more realistic, different assembly orders are allowed. A tree structure is built to represent the relationship between each component. The conditions for the assembly of objects with thread is also considered. To assemble bolts, a virtual screw driver is used, as shown in Fig. 17.

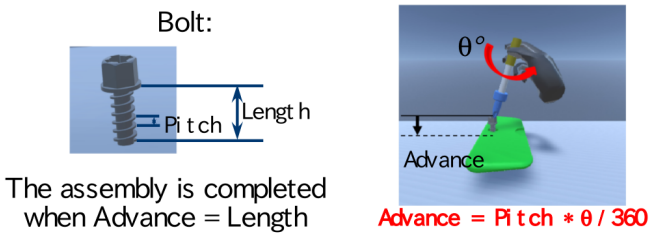


Fig. 16: Screwing motion for a bolt

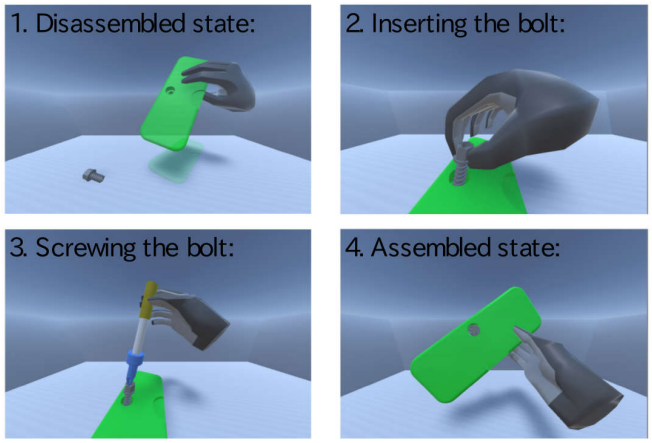


Fig. 17: Screwing assembly steps

IV. Discussion and Future Work

Assembly Conditions

In the current implementation the assembly conditions are pre-setted. As future work, we plan to get all possible assembly conditions automatically by analyzing the parts' models to determine which part can be assembled with an specific part. This will allow our framework to be used for different types of assembly tasks.

Virtual Workspace

Currently we use a monitor to render the virtual world, which limits the perspective of the user. Our next step is to use an HMD to increase the depth perception of the user and make the assembly tasks more realistic, enabling the user to execute the tasks more naturally.

Robot simulation

We are planing to use this framework with several assembly tasks and different users to have enough data to construct a motion plan for the robotic simulation on a dual arm robot [11]. This will allow the user to easily teach the robot how to assemble new tasks.

V. Conclusion

To reproduce assembly tasks, robots need to know when and how to move. In this work, we proposed a framework to easily collect data of a human assembly demonstration in order to teach a robot how to assemble objects. The proposed framework was developed in a virtual scene where we can track our hands' movements and the virtual objects motion all the time. We defined assembly conditions in order to merge two or more objects and considered them as one. We introduced a visual guidance for the user to know how to assemble the object it is holding to compensate for the absence of haptic feedback. We also developed a point collider to balance the contact points' position accuracy with the computational cost needed to obtain these points. Finally, we also showed how to assemble bolts using a screwdriver in the virtual scene.

In the future we would like to expand the proposed framework for other types of tasks, like manipulation of daily life objects, etc.

Acknowledgment

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

References

[1] R. C. Luo, C.-W. Kuo, and Y.-T. Chung, "Model-based 3d object recognition and fetching by a 7-dof robot with online obstacle avoidance for factory automation," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 2647–2652, IEEE, 2015.

[2] R. M. Murray, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[3] M. Mahdavian, M. Shariat-Panahi, A. Yousefi-Koma, and A. Ghasemi-Toudeshki, "Optimal trajectory generation for energy consumption minimization and moving obstacle avoidance of a 4dof robot arm," in *Robotics and Mechatronics (ICROM), 2015 3rd RSI International Conference on*, pp. 353–358, IEEE, 2015.

[4] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer handbook of robotics*, pp. 1371–1394, Springer, 2008.

[5] J. Aleotti, S. Caselli, and M. Reggiani, "Toward programming of assembly tasks by demonstration in virtual environments," in *Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. The 12th IEEE International Workshop on*, pp. 309–314, IEEE, 2003.

[6] S. Sridhar, F. Mueller, M. Zollhöfer, D. Casas, A. Oulasvirta, and C. Theobalt, "Real-time joint tracking of a hand manipulating an object from rgb-d input," in *European Conference on Computer Vision*, pp. 294–310, Springer, 2016.

[7] F. Hernoux, R. Béarée, and O. Gibaru, "Investigation of dynamic 3d hand motion reproduction by a robot using a leap motion," in *Proceedings of the 2015 Virtual Reality International Conference*, p. 24, ACM, 2015.

[8] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt, "Real-time hand tracking under occlusion from an egocentric rgb-d sensor," in *Proceedings of International Conference on Computer Vision (ICCV)*, vol. 10, 2017.

[9] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, *et al.*, "Accurate, robust, and flexible real-time hand tracking," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 3633–3642, ACM, 2015.

[10] A. Tagliasacchi, M. Schröder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly, "Robust articulated-icp for real-time hand tracking," in *Computer Graphics Forum*, vol. 34, pp. 101–114, Wiley Online Library, 2015.

[11] D. Bassily, C. Georgoulas, J. Guettler, T. Linner, and T. Bock, "Intuitive and adaptive robotic arm manipulation using the leap motion controller," in *ISR/robotik 2014; 41st international symposium on robotics; proceedings of*, pp. 1–7, VDE, 2014.